The breakpoint distance for signed sequences (extended abstract)

Guillaume Blin, Guillaume Fertin
LINA FRE CNRS 2729, Université de Nantes

Cedric Chauve
LaCIM, Université du Québec à Montréal

CompBioNets 2004: Algorithms and Computational Methods for Biochemical and Evolutionary Networks, volume 3 of *Text in Algorithms*, KCL Publications (London), pages 3–16, 2004.

The breakpoint distance for signed sequences (extended abstract)

Guillaume Blin[1], Cedric Chauve[2] and Guillaume Fertin[1]

[1] LINA FRE 2729, Université de Nantes
2 rue de la Houssinière, BP 92208 - F44322 Nantes Cedex 3, France
blin,fertin@lina.univ-nantes.fr
[2] LaCIM et Département d'Informatique, Université du Québec À Montréal
CP 8888, Succ. Centre-Ville, H3C 3P8, Montréal (QC), Canada
chauve@lacim.uqam.ca

Abstract. We consider the problem of estimating the rearrangement distance in terms of reversals, insertion and deletion between two genomes, G and H with possibly multiple genes from the same gene family. We define a notion of breakpoint distance for this problem, based on matching genes from the same family between G and H. We show that this distance is a good approximation of the edit distance, but **NP**-hard to compute, even when just one family of genes is non-trivial. We also propose a branch-and-cut exact algorithm for the computation of the breakpoint distance.

1 Introduction

Evaluating the evolutionary distance between genomes in terms of gene order rearrangements has been intensively studied, from the algorithmic point of view, since the early 90's, due, among other reasons, to its important applications in comparative genomics [2]. From an algorithmic point, it can roughly be defined as follows: given a set $\mathcal{A}$ of *gene families*, two *genomes* G and H, represented as sequences of signed elements (genes) from $\mathcal{A}$, and a set of evolutionary operations that operate on segments of genes (like reversals, transpositions, insertions, duplications, deletions for example), what is the minimum number of operations needed to transform G into H ? This number is often called the *edit distance*.

Until recently, most of the algorithms developed to evaluate the rearrangement distance between two genomes relied on the assumption that there is exactly one copy of each gene in each genome: each element of $\mathcal{A}$ appears exactly once in G and once in H. In this framework, the comparison of genomes reduces to the comparison of signed permutations (see the book [6] that contains several surveys on this topic). In this framework, when only reversals are considered, computing the distance between two genomes can be achieved in linear time [1]. However, this "non-duplication" assumption induces a strong limitation, and prevents to perform whole-genome analysis, at least with gene orders (in [2], Bourque et *al.* consider genomic segments containing possibly many genes): in the general case, sequences of signed integers need to be considered instead of signed permutations, as well as operations like insertions and deletion of gene segments.

Extending the so-called Hannenhalli-Pevzner theory for sorting signed permutations by reversals [8], El-Mabrouk proposed some algorithms allowing to consider some restricted cases of insertions and deletions [4]. Based on El-Mabrouk's work, Marron, Swenson and Moret proposed approximations algorithms both in the case where one of the genomes considered is a signed permutation [10] and in the case where the two genomes are sequences [12]. Their method relies on a matching between genes of the two genomes that allows to reduce the problem to the comparison of two signed permutations instead of two signed sequences. Such an idea, to reduce the problem to the case of permutations, based on the matchings between duplicated genes was also used, in the median problem, by Tang and Moret [13], who try all possible matchings and choose the best one. Sankoff [11] proposed another way to reduce the problem to signed permutations, the *exemplar strategy*, that consists in deleting all but one gene for each gene family (see also Bryant [3] for a proof of the **NP**-completeness of the problem).

Here, we extend the breakpoint distance model in order to consider insertions and deletions of genes segments, and we relate this model to the case of sorting by reversals, insertions and deletions (Section 2). Our main theoretical results state that computing the corresponding breakpoint distance between two signed sequences is **NP**-complete (Section 3). In Section 4, we provide a branch-and-cut exact algorithm.

2 Breakpoint distance and edit distance for signed sequences

Genomes, gene families and gene. Given an alphabet $\mathcal{A}$ of elements called *gene families*, a *genome* G on $\mathcal{A}$ is a sequence of signed ($+$ or $-$) elements of $\mathcal{A}$. Each occurrence of an element of $\mathcal{A}$ in G is called a *gene*. For two genomes G and H and a gene family $\mathbf{a}$, the number of occurrences of $\mathbf{a}$ in G and H is the cardinality of this family $\mathbf{a}$. A gene family $\mathbf{a}$ is said to be *trivial* if either $\mathbf{a}$ appears in exactly one genome or $\mathbf{a}$ has cardinality exactly 2. Otherwise, $\mathbf{a}$ is said to be *non-trivial*. A gene belonging to a trivial (resp. non-trivial) family is said to be trivial (resp. non-trivial). A segment (subsequence) of G that contains only non-trivial genes is called a *non-trivial segment*.

We say that two genomes G and H are *balanced* if, for any gene family $\mathbf{a}$, there are as many occurrences of $\mathbf{a}$ in G than in H.

Gene matchings. Let $G = g_1, \ldots, g_n$ and $H = h_1, \ldots, h_m$ be two genomes on $\mathcal{A}$. A *gene matching* $\mathcal{M}$ between G and H is a maximal matching between the genes of G and the genes of H such that, for every pair $(g_i, h_j) \in \mathcal{M}$, g_i and h_j belong to the same family ($|g_i| = |h_j|$). By maximal matching, we mean that for any gene family $\mathbf{a}$, it is forbidden to have at the same time an occurrence of $\mathbf{a}$ in G and one in H that do not belong to $\mathcal{M}$.

It follows from the maximality condition of matchings that in any matching $\mathcal{M}$ between balanced G and H, every gene of G is matched to a gene of H and conversely.

Breakpoint distance and the minimum breakpoint matching. Given G, H and a gene matching $\mathcal{M}$, a *breakpoint* between G and H with respect to $\mathcal{M}$ is

- either a pair of consecutive genes $g_i g_{i+1}$ in G (resp. $h_j h_{j+1}$ in H) such that exactly one of them does not belong to $\mathcal{M}$ (these breakpoints are called *deletion breakpoints* (resp. *insertion breakpoints*)),
- or a pair of consecutive genes $g_i g_{i+1}$ in G such that both genes belong to $\mathcal{M}$ (say $\mathcal{M}(g_i) = h_j$ and $\mathcal{M}(g_{i+1}) = h_k$) but neither $h_j = g_i$, $h_k = g_{i+1}$ and $k = j+1$, nor $h_j = -g_{i+1}$, $h_k = -g_i$ and $k = j-1$ (these breakpoints are called *rearrangement breakpoints*).

One denotes by $D_b(G, H, \mathcal{M})$ the number of breakpoints between G and H with respect to $\mathcal{M}$, and by $D_b(G, H)$ the breakpoint distance between G and H, that is the minimum of $D_b(G, H, \mathcal{M})$ among all matchings $\mathcal{M}$.

The minimum breakpoint matching problem can be stated as follows: given $G = g_1, \ldots, g_n$ and $H = h_1, \ldots, h_m$ two genomes, find the gene matching $\mathcal{M}$ between G and H such that $D_b(G, H, \mathcal{M}) = D_b(G, H)$. In the rest of this paper, we call such a matching a *minimum breakpoint matching.*

Remark 1. The notion of breakpoint was originally introduced for signed permutations (see [9] for example), as a good approximation for the reversal distance, and here we defined breakpoints for signed sequences. However, it is immediate to see that in the case where G and H are signed permutations (every gene occurs exactly once in G and once in H), there is only one minimum gene matching between G and H and $D_b(G, H)$ is the classical breakpoint distance defined for signed permutations.

Rearrangements and edit distance. In the case of signed permutations, it was shown [9] that there is a close link between the breakpoint distance and the edit distance when only reversals are allowed. We extend here this result to the case of signed sequences, and reversals, insertions and deletions of segments of genes. A *reversal* of a segment $S = x_1, \ldots, x_k$ of genes replaces this segment by its reverse $\overline{S} = -x_k, \ldots, -x_1$ at the same position in the genome. An *insertion* inserts a new segment of genes into a genome, while a *deletion* removes a segment of genes from a genome. We impose that, for any gene family $\mathbf{a}$, and a sequence of rearrangements that transform G into H, there can not be an insertion and a deletion of segments containing occurrences of $\mathbf{a}$. In other words, if G has more (resp. less) occurrences of $\mathbf{a}$ than H, then no segment containing an $\mathbf{a}$ will be inserted (resp. deleted)[3]. The *edit distance* between G and H is the minimum number of rearrangements that transform G into H, denoted by $D_e(G, H)$.

Theorem 1. *For two genomes G and H, $D_e(G, H)/2 \leq D_b(G, H) \leq 2D_e(G, H)$.*

Proof. Let $\sigma_1, \ldots, \sigma_k$ be an edit sequence (reversals, insertions and deletions) that transforms G into H. Every reversal creates at most two rearrangement breakpoints. Similarly, every deletion of a genes segment creates at most two deletion breakpoints, while every insertion of a segment creates at most two insertion breakpoints. Hence, $D_b(G, H) \leq 2D_e(G, H)$.

Let $\mathcal{M}$ be a gene matching between G and H, inducing k_1 rearrangement breakpoints, k_2 insertion breakpoints and k_3 deletion breakpoints. Note that k_2 and k_3 are even numbers. One can delete all the non-trivial segments of G in at most $k_3/2$ deletions, that will remove the k_3 deletion breakpoints and create at most $k_3/2$ rearrangement breakpoints. The resulting signed sequence is then such that all its genes belong to $\mathcal{M}$. Now one can remove all rearrangement breakpoints (there are $k_1 + k_3/2$ such breakpoints) with at most $2k_1 + k_3$ reversals, which gives a signed sequence that is equal to H where all the non-trivial segments are removed. It then suffices to perform $k_2/2$ insertions (the non-trivial segments of H) to obtain H. The above edit sequence contains exactly $2k_1 + k_2/2 + (3/2)k_3$, and then $D_e(G, H)/2 \leq D_b(G, H)$. □

Remark 2. In [10], Marron et *al.* showed that when H is a signed permutation and $D_e(G, H) = k$, their algorithm (that is quadratic in the length of the genomes) produces a an edit sequence of length at most $10k + 4$. In the general case [12], they did not prove any theoretical result on the quality of their algorithm, but in both cases, experimental results on simulated data were very convincing. However, the close link showed by Theorem 1 between the breakpoint distance and the edit distance, coupled to a good approximation algorithm for the breakpoint distance could improve the theoretical approximation bound for the edit distance.

3 Complexity results with exactly one non-trivial family

In this section, we are interested in the case where just one gene family is non-trivial. We first prove that, even with just one non-trivial gene family, computing the breakpoint distance is **NP**-complete

[3] This restriction, that is limiting from the biological point of view, is necessary from the algorithmic point of view and was followed, although it is not stated explicitely, in [10, 12]

(Theorem 2). Note that Theorem 2 induces that in the general case (more than one non-trivial gene family), the problem is also **NP**-complete. This result also implies that the MINIMUM COVER problem introduced by Swenson et *al.* [12] is also **NP**-complete (Theorem 3). We then give an approximation algorithm for the breakpoint distance whose approximation ratio depends on the maximal length of a non-trivial segment (Theorem 4).

3.1 NP-hardness results

Theorem 2. *For two genomes G and H, computing $D_b(G,H)$ is* **NP***-complete, even when the number of non-trivial gene families is equal to* 1*.*

Our proof relies on a reduction of the classical decision problem associated to MINIMUM BIN PACKING, that is well known to be **NP**-complete [5], to a decision problem related to the minimum breakpoint matching problem. For the sake of clarity, we now state formally the two decision problems we consider, MINIMUM BIN PACKING and MBM(L,F).

> MINIMUM BIN PACKING
> INSTANCE: *A finite set $U = \{u_1, u_2, \ldots, u_n\}$, a size $s(u) \in \mathbb{Z}^+$ for each $u \in U$ and two positive integers $\mathcal{K}$ and $\mathcal{C}$.*
> QUESTION: *Is there a partition of U into $\mathcal{K}$ disjoint sets $U_1, U_2, \ldots, U_{\mathcal{K}}$ such that the sum of the sizes of the elements in each U_i is at most $\mathcal{C}$?*

> MBM(L,F)
> INSTANCE: *An integer $\mathcal{P}$ and two genomes G and H on a set of gene families such that there is exactly* F *non-trivial gene families and no non-trivial segment (either in G or H) contains more than* L *genes.*
> QUESTION: *$D_b(G,H) \leq \mathcal{P}$?*

It is easily seen that MBM(L,1) is in **NP** since given an integer $\mathcal{P}$ and a set of matchings between two genomes we can polynomially compute the breakpoint distance and thus check if this distance is less than or equal to $\mathcal{P}$. The remainder of the section is devoted to proving that MBM(L,1) is **NP**-complete, which implies Theorem 2. For this, we reduce the MINIMUM BIN PACKING problem to the problem MBM(L,1), *i.e.* we show that from any instance $(U,\mathcal{K},\mathcal{C})$ of MINIMUM BIN PACKING we are able to construct in polynomial time an instance $(G,H,\mathcal{P})$ of MBM(L,1) such that $D_b(G,H) \leq \mathcal{P}$ if and only if there is a minimum bin packing for $(U,\mathcal{K},\mathcal{C})$. We denote $\mathcal{N} = \mathcal{K}.\mathcal{C} - \sum_{i=1}^{n} s(u_i)$.

Let us first detail the construction from of the genomes G and H from a minimum bin packing instance $(U,\mathcal{K},\mathcal{C})$. The gene families are $\{\alpha, \beta, \mathbf{x}, A_1, A_2 \ldots, A_{n+\mathcal{N}}, B_1, B_2, \ldots, B_{\mathcal{K}+1}\}$. On the whole, there are $\mathcal{K}+\mathcal{N}+n+4$ families of genes, and $\mathbf{x}$ is the unique non-trivial family. For $1 \leq i \leq n$ (resp. $n+1 \leq i \leq n+\mathcal{N}$), we denote by $\mathbf{u'_i}$ a sequence of $s(u_i)$ consecutive genes $\mathbf{x}$ (resp. one gene $\mathbf{x}$). For $1 \leq j \leq k$, $\mathbf{U'_j}$ represents a sequence of $\mathcal{C}$ consecutive genes $\mathbf{x}$. G and H are defined as follows:

$$G = \alpha\ \mathbf{u'_1}\ A_1\ \mathbf{u'_2}\ A_2\ \ldots\ \mathbf{u'_{n+\mathcal{N}}}\ A_{n+\mathcal{N}}\ B_1\ B_2\ \ldots\ B_{\mathcal{K}+1}\ \beta$$
$$H = \alpha\ B_1\ \mathbf{U'_1}\ B_2\ \mathbf{U'_2}\ \ldots\ B_{\mathcal{K}}\ \mathbf{U'_{\mathcal{K}}}\ B_{\mathcal{K}+1}\ A_1\ A_2\ \ldots\ A_{n+\mathcal{N}}\ \beta$$

An illustration of such a construction, that can obviously be achieved in polynomial time, is given in Figure 1. To complete the construction of the instance of MBM(L,1), it remains to define $\mathcal{P}$: $\mathcal{P} = 2(n+\mathcal{N}+1)+\mathcal{K}$.

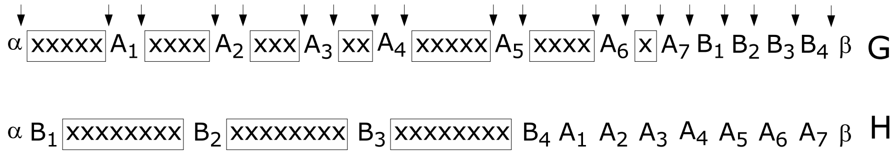

Fig. 1. Instance of MBM(L,1) associated to the minimum bin packing instance where $\mathcal{K} = 3$, $\mathcal{C} = 8$ and $U = \{u_1, \ldots, u_6\}$ with $s(u_1) = s(u_5) = 5$, $s(u_2) = s(u_6) = 4$, $s(u_3) = 3$ and $s(u_4) = 2$. Black arrows indicate breakpoints that are not dependant of a particular gene matching.

In the next two lemmas, that establish that MBM(L,1) is **NP**-complete, we consider an instance $(U, \mathcal{K}, \mathcal{C})$ of Minimum Bin Packing and the corresponding, according to the above construction, instance $(G, H, \mathcal{P})$ of MBM(L,1).

Lemma 1. *$D_b(G, H) \geq \mathcal{P}$, and, in any minimum matching between G and H, at least $\mathcal{P}$ breakpoints involve at least one trivial gene.*

Proof. Let us count the number of breakpoints in G that occur between two adjacent genes, of which at least one is non-trivial. For a gene g in a genome G, let $L^G(g)$ (resp. $R^G(g)$) denotes the left (resp. right) neighbor of the gene g in the genome G.

For any gene A_i with $1 \leq i \leq n + \mathcal{N}$, $L^G(A_i) \neq L^H(A_i)$ and $R^G(A_i) \neq R^H(A_i)$. Thus, each gene A_i, with $1 \leq i \leq n + \mathcal{N}$, induces two breakpoints (one with both its left and right neighbor). Similarly, for any gene B_j with $1 \leq j \leq \mathcal{K}$, $R^G(B_j) \neq R^H(B_j)$. Thus, each gene B_j induces a breakpoint. By now, we have $2n + 2\mathcal{N} + \mathcal{K}$ breakpoints in G. Finally, $R^G(\alpha) \neq R^H(\alpha)$ and $L^G(\beta) \neq L^H(\beta)$. Thus, on the whole the number of breakpoints in G is at least $2(n+\mathcal{N}+1)+\mathcal{K} = \mathcal{P}$, and the $\mathcal{P}$ breakpoints we described above all involve at least one trivial gene. $\square$

Lemma 2. *There is a partition of U into $\mathcal{K}$ disjoint sets $U_1, U_2, \ldots, U_{\mathcal{K}}$ such that the sum of the sizes of the elements in each U_i is at most $\mathcal{C}$ if and only if $D_b(G, H) \leq \mathcal{P}$.*

Proof. ($\Leftarrow$) Suppose that $D_b(G, H) \leq \mathcal{P}$. By Lemma 1, we know that $D_b(G, H) = \mathcal{P}$, and all the breakpoints with respect to a minimum matching between G and H occur between two adjacent genes, of which at least one is trivial. In other words, any couple of adjacent genes $\mathbf{x}$ in G is matched with a couple of adjacent genes $\mathbf{x}$ in H. Consequently, any non-trivial segment u'_i, with $1 \leq i \leq n$, in G is matched with a sequence of consecutive genes $\mathbf{x}$ in H. Precisely, for any $1 \leq i \leq n$, there is a given $1 \leq j \leq \mathcal{K}$ such that the sequence u'_i is totally matched with a contiguous subsequence of U'_j as illustrated in Figure 2.

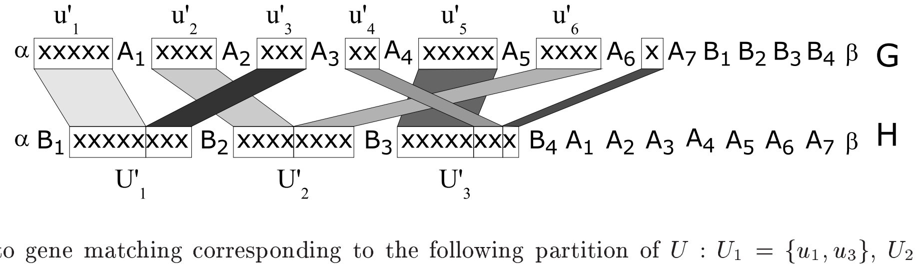

Fig. 2. Gene to gene matching corresponding to the following partition of U : $U_1 = \{u_1, u_3\}$, $U_2 = \{u_2, u_6\}$ and $U_3 = \{u_4, u_5\}$

Therefore, such a matching induces a partition of the set of sequences $\{u'_1, u'_2, \ldots, u'_n\}$ into at most $\mathcal{K}$ disjoint sequences $U'_1, U'_2, \ldots, U'_{\mathcal{K}}$. As, by construction, $|U'_i| = \mathcal{C}$ for $1 \leq i \leq \mathcal{K}$, this partition corresponds to an answer to the corresponding MINIMUM BIN PACKING instance.

($\Rightarrow$) Suppose we have a partition of U into disjoint sets $U_1, U_2, \ldots, U_{\mathcal{K}}$ each of cardinality at most $\mathcal{C}$. We compute a gene matching $\mathcal{M}$ between G and H as follows:

- each trivial gene in G is matched with its corresponding gene in H,
- for $1 \leq j \leq \mathcal{K}$, for each u_i, if $u_i \in U_j$, then the sequence of genes $\mathbf{x}$ of u'_i in G is matched gene to gene with the first free (i.e. not already matched) sequence of genes $\mathbf{x}$ of U'_j in H.

Since $\mathcal{M}$ is built according to the partition of U, we claim that each non-trivial segment u'_i, with $1 \leq i \leq n$, is matched to a contiguous sequence of genes $\mathbf{x}$ in H. Thus, all the breakpoints induced by this matching occur between two adjacent genes, at least one of them being trivial. By Lemma 1, this leads to $D_b(G, H, \mathcal{M}) \leq \mathcal{P}$, and so to $D_b(G, H) \leq \mathcal{P}$. □

Using the result of Theorem 2, we can answer a question introduced in [12] by Swenson et *al.* They defined a *cover* as a mapping that relates substrings of genes in a target genome to the same substring of genes (possibly reverted) in a source genome. Swenson et *al.* also defined the notion of *Minimum Cover* as a cover that maps the subject to the target genome with the fewest substrings. They introduced this notion as a way to obtain a good approximation of the edit distance (in terms of reversals, insertions and deletions) between two genomes. In the following, we prove that finding a *Minimum Cover* between two genomes is **NP**-complete. Our proof relies on a reduction of the MBM(L,F) problem over balanced genomes. We now state formally the decision problem we consider: MINIMUM COVER .

MINIMUM COVER INSTANCE: *Two genomes G and H and an integer $\mathcal{Q}$.* QUESTION: *Is there a cover that maps G to H with at most $\mathcal{Q}$ substrings?*

Theorem 3. *For two genomes G and H, computing the minimum cover of (G, H) is* **NP***-complete.*

Proof. (Sketch) It is easily seen that MINIMUM COVER is in **NP** since given an integer $\mathcal{Q}$ and a set of matchings between two genomes we can polynomially compute the number of substrings and thus check if this number is less than or equal to $\mathcal{Q}$.

We propose hereafter an intuitive reduction from MBM(L,F) over balanced genomes, with $\mathcal{Q} = \mathcal{P} + 1$. Indeed, MBM(L,F) over balanced genomes is **NP**-complete since the reduction used in the proof of Theorem 2 uses balanced genomes. Given an instance $(G, H, \mathcal{P})$ of MBM(L,F) where G and H are balanced, we take the same instance (i.e. the genomes are identical) $(G, H, \mathcal{P} + 1)$ of *Minimum Cover* problem. If there is a gene matching between G and H inducing at most k breakpoints then it follows from the definition of a breakpoint that this matching is a cover that maps G to H with at most $k + 1$ substrings. Moreover, given a minimum cover involving at most k substrings then the corresponding mapping is also a gene matching inducing at most $k - 1$ breakpoints. Therefore, there is a minimum cover of $(G, H, \mathcal{P} + 1)$ if and only if $D_b(G, H) \leq \mathcal{P}$. □

3.2 An approximation result for balanced instances

Theorem 4. *Let G and H be a balanced instance for the minimum breakpoint matching problem, with exactly one non-trivial gene family and every non-trivial segment of length at most* L*. Computing $D_b(G, H)$ is* (L + 1)*-approximable.*

To prove this result, we introduce the notion of *duplicated* non-trivial segment: a segment $\mathbf{a}\ S\ \mathbf{b}$ of G, where $\mathbf{a}$ and $\mathbf{b}$ are trivial genes and S is a non-trivial segment, is duplicated if H contains either $\mathbf{a}\ S\ \mathbf{b}$ or $\overline{\mathbf{b}}\ \overline{S}\ \overline{\mathbf{a}}$ (this segment of H is called the copy of $\mathbf{a}\ S\ \mathbf{b}$ in H). In other words, matching $\mathbf{a}\ S\ \mathbf{b}$ to its copy in H does not induce any rearrangement breakpoint involving a gene of S.

Lemma 3. *Let G and H be a balanced instance for the minimum breakpoint matching problem, with exactly one non-trivial gene family. There is an optimal matching between G and H that matches any duplicated segment of G to its copy in H.*

Proof. Let $\mathcal{M}$ be a matching between G and H, and $\mathbf{a}\ S\ \mathbf{b}$ a duplicated segment of G that is not perfectly matched to its copy. W.l.o.g we suppose that the copy of $\mathbf{a}\ S\ \mathbf{b}$ in H is $\mathbf{a}\ S\ \mathbf{b}$. Let g_i be the first gene of $\mathbf{a}\ S\ \mathbf{b}$ that is not matched to its copy h_j in H, say $(g_i, h_k) \in \mathcal{M}$ and $(g_\ell, h_j) \in \mathcal{M}$. There are two breakpoints, one to the left of g_i and one to left of g_ℓ. It is immediate to see that if one replaces (g_i, h_k) and (g_ℓ, h_j) by (g_i, h_j) and (g_ℓ, h_k), then one does not increase the number of breakpoints, which proves the lemma. An illustration is given in Figure 3. □

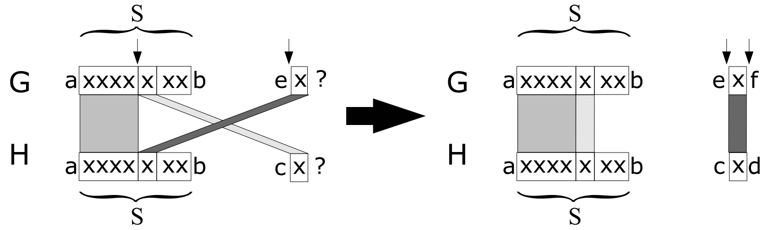

Fig. 3. Matching a segment of G to its copy in H does not increase the number of breakpoints.

The following lemma follows obviously from the definition of a duplicated segment.

Lemma 4. *Let G and H be a balanced instance for the minimum breakpoint matching problem, with exactly one non-trivial gene family, and S a non-trivial segment of G that is not part of a duplicated segment. In every optimal matching between G and H, there is at least one breakpoint involving a gene in S.*

Proof of Theorem 4. Consider the following algorithm that computes a gene matching between G and H.

```
SimpleAlgorithm(G,H)
```

1. Perfectly match each duplicated segment of G to its copy in H.
2. Match each remaining trivial gene in G with its unique copy gene in H
3. Randomly match each remaining non-trivial gene in G with a non-trivial gene in H

Let $\mathcal{M}$ be the matching produced by the above algorithm and $\mathcal{M}_{\text{opt}}$ be an optimal matching that matches every duplicated segment of G to its copy in H (by Lemma 3, such a matching exists). Hence, in $\mathcal{M}$ as well as in $\mathcal{M}_{\text{opt}}$, there is no breakpoint involving two genes of a same duplicated segment. By definition of a matching, in the result of the above algorithm, the number of breakpoints involving two trivial genes is the same in $\mathcal{M}$ and in $\mathcal{M}_{\text{opt}}$. So the only breakpoints in $\mathcal{M}$ that differ from $\mathcal{M}_{\text{opt}}$ involve pairs of consecutive genes that do not belong both to the same duplicated segment and where at least one is non-trivial. As every non-trivial segment is of length at most L, in $\mathcal{M}$ there are at most $\text{L}+1$ breakpoints induced by each non-trivial segment, when, by Lemma 4, in $\mathcal{M}_{\text{opt}}$ there are at least one breakpoint induced by each non-trivial segment. The result follows immediately. □

4 An exact algorithm

In this section we propose an exact algorithm for computing the breakpoint distance between two genomes G and H, based on the following intuition: long segments that match (up to a complete reversal) in G and H are likely to belong to a minimum gene matching. This idea was introduced in the context of computing the edit distance between genomes by Swenson et *al.* [12]. (They proposed an approximation algorithm, with no theoretical bound, based on this intuition and the notion of minimum cover, that performs well on simulated data).

The algorithm focuses only on non-trivial segments. Indeed, if a gene appears in just one genome, it does not belong to any gene matching, and if a gene appears exactly once in each genome, then these two occurrences have to be matched together. Hence, the main principle of this algorithm is to match parts of non-trivial segments that are common to G and H[4].

Our algorithm follows a branch-and-cut strategy. In principle we try all the possible matchings between G and H, starting with an empty matching and trying all possibilities to extend it. But as soon that the current matching already induced more breakpoints than the best matching previously computed, we do not try to extend it. We use the algorithm by Swenson et *al.* [12] to compute an initial matching that gives an upper bound for the breakpoint distance.

As we are interested segments that appear both in G and H, we will use a suffix-tree that will allow to find quickly the occurrences of a segment of G that appears in H (for a reference on suffix-trees, see [7]). More precisely, let $\mathcal{S} = \{S_1, S_2 \ldots S_m, \overline{S_1}, \overline{S_2} \ldots \overline{S_m}\}$ be the set of non-trivial segments of H (the S_i's), and of the reversed non-trivial segments of H (the $\overline{S_i}$'s), and $\mathcal{T}(H)$ the generalized suffix-tree of this set of segments. For a node N of $\mathcal{T}(H)$, one denotes by label(N) the concatenation of the labels of the edges from the root of $\mathcal{T}(H)$ to N. We assume that each node N of $\mathcal{T}(H)$ is provided with a list, possibly empty, of triples (b, h, p), such that the suffix of S_h (resp. $\overline{S_h}$) if $b = 0$ (resp. $b = 1$) starting at position p in H is equal to label(N).

Let $\mathcal{R} = \{R_1, R_2 \ldots R_n\}$ be the set of non-trivial segments of G and $\mathcal{M}'$ a partial matching between G and H involving only genes of $R_1, \ldots, R_k[1..i-1]$ ($R_k[1..i-1]$ is the prefix of length $i-1$ of R_k): some genes of $R_1, \ldots, R_k[1..i-1]$ are matched to genes of H, while the ones that are not are considered as belonging to segments that will induce deletion breakpoints (they will not be matched in any extension of $\mathcal{M}'$). Without loss of generality, we suppose that R_k is of length ℓ greater than $i-1$. The principle of the algorithm is to try all possibilities to extend the partial matching $\mathcal{M}'$, either by deciding to leave the gene $R_k[i]$ unmatched (it will belong to a deleted segment) or by matching a prefix of $R_k[i..\ell]$ that is common to G and H and is unmatched. Following the intuition that long matches are better, we try to match prefixes of $R_k[i..\ell]$ with segments of H greedily, that is in decreasing order of their length.

Before presenting the algorithm, we need to precise a few more details. We use a global variable B_m (initially computed on the matching produced by the algorithm of [12]) that records the breakpoint distance given by the best matching generated so far, and a global variable B_c that records the number of breakpoints induced by the current partial matching. Moreover, we suppose that for each gene family $\mathbf{x}$, we know the number of genes of this family that will have to be left unmatched (this depends only on the number of occurrences of $\mathbf{x}$ in G and H and on $\mathcal{M}'$), denoted $d_{\mathbf{x}}$.

[4] In this section, by *common* segment, we mean a segment that appears, up to a complete reversal, both in G and H.

The pseudo-code of the recursive procedure that tries to extend a partial matching is presented in Figure 4.

```
ExactAlgorithmRec($R_k$,$i$)

Let $\mathbf{x} = R_k[i]$ and $\ell$ be the length of $R_k$.
// Comment: first one tries to find a match for $R_k[i..\ell]$
For $j = \ell$ down to $i$ do
  If there is a node $N$ of $\mathcal{T}(H)$ such that label($N$) = $R_k[i..j]$ Then
    For each node $N'$ of the subtree rooted at $N$ and each triple $(b, h, p)$ of the node $N'$ do
      If all the genes of the prefix of $S_j[p..p+j-i]$ are unmatched Then
        Match $R_k[i..j]$ and $S_j[p..p+j-i]$
        If $B_m > B_c$ Then
            $B := B_c$ and update of the variables $B_c$ and $B_m$
            If $(j = \ell)$ Then ExactAlgorithmRec($R_{k+1}$, 1) Else ExactAlgorithmRec($R_k$, $j+1$)
            $B_c := B$
        Unmatch $R_k[i..j]$ and $S_j[p..p+j-i]$
// Comment: next one tries to consider $R_k[i]$ as unmatched.
If $d_{\mathbf{x}} > 0$ Then
  If $B_m > B_c$ Then
    $B := B_c$ and update of the variables $B_c$ and $B_m$
    If $(i = \ell)$ Then ExactAlgorithmRec($R_{k+1}$, 1) Else ExactAlgorithmRec($R_k$, $i+1$)
    $B_c := B$
```

Fig. 4. Recursive procedure extending a partial matching

The complexity of this algorithm does not improve the complexity of the brute-force algorithm (see [13] where the brute-force strategy is used for the median problem, in the context of edit distance), that tries all possible gene matchings. However, the combination of an upper bound for the breakpoint distance together with a branch-and-cut strategy seems to improve dramatically the computation time. We are currently running tests in order to quantify this speed-up.

There are several ways to improve this algorithm. For example, we build the matching by reading the non-trivial segments of G in the order they appear in the genome. In the context of a branch-and-cut strategy, one can think that improving the order in which the segments of G are considered is of primary importance. This was shown to be effective, for the edit distance, by Swenson et *al.* that used a greedy strategy to build a minimum cover [12]. One can also notice that when all unmatched segments left are of length 1, then there is no need to use explore the space of all matchings to find the best extension of the current partial matching. Indeed, when $L = 1$ it is immediate to see that the problem of computing a minimum gene matching reduces to the computation of a minimum weigthed bipartite matcing, that can be solved in polynomial time.

5 Conclusion

In this note, we extended the breakpoint distance to the case of signed sequences, and we showed how it can be related to the edit distance. We proved that, contrary to signed permutations, computing the breakpoint distance is **NP**-complete, even in the case of simple instances, with one non-trivial gene family and no insertion and deletion breakpoints. As far as we know, aside from the exemplar distance, these are the first complexity results on these problems, for which some algorithms that perform well in practice exist [10, 12].

This leaves many theoretical questions. For example, it is not known at which value of L (length of non-trivial segments) the problem of computing the breakpoint distance becomes **NP**-hard: if $L = 1$, this problem reduces easily to a minimum weight bipartite matching problem, but nothing is known for greater values of L. Also, following Remark 2, it would be interesting to design an approximation algorithm for the general MBM(L,F) problem with a good (*e.g.* constant) ratio, that could translate into an approximation algorithm for the edit distance. Moreover, in the context of phylogenetic reconstruction, it would be interesting to design a good heuristic for computing the median of three signed sequences.

References

1. D.A. Bader, B.M.E Moret, and M. Yan. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comp. Biol.*, 8(5):483–491, 2001.
2. G. Bourque, P.A. Pevzner, and G. Tesler. Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse and rat genomes. *Genome Res.*, 14(4):507–516, 2004.
3. D. Bryant. The complexity of calculating exemplar distances. In D. Sankoff and J. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. Kluwer Acad. Pub., 2000.
4. N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Combinatorial Pattern Matching (CPM'00)*, volume 1848 of *Lecture Notes in Comput. Sci.*, pages 222–234. Springer, 2000.
5. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co, 1979.
6. O. Gascuel, editor. *Mathematics of Evolution and Phylogeny.* Oxford Univ. Press, 2004. To appear.
7. D. Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge Univ. Press, 1997.
8. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46(1):1–27, 1999.
9. J.D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13(1-2):180–210, 1995.
10. M. Marron, K.M. Swenson, and B.M.E. Moret. Genomic distances under deletions and inversions. In *Computing and Combinatorics (COCOON'03)*, volume 2697 of *Lecture Notes in Comput. Sci.*, page 537547. Springer, 2003.
11. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.
12. K.M. Swenson, M. Marron, J.E Earnest-DeYoung, and B.M.E. Moret. Approximating the true evolutionary distance between two genomes. Technical Report TR-CS2004-15, Department of Computer Science, University of New Mexico, 2004.
13. J. Tang and B.M.E. Moret. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. In *Workshop on Algorithms and Data Structures (WADS'03)*, volume 2748 of *Lecture Notes in Comput. Sci.*, pages 37–46. Springer, 2003.