Some Approximation Results for the Maximum Agreement Forest Problem*

E. M. Rodrigues[1] **, M-F. Sagot[2], and Y. Wakabayashi[1] ***

[1] Universidade de São Paulo, Brazil
{estela,yw}@ime.usp.br
[2] Institut Pasteur and Université de Marne-la-Vallée, France
sagot@pasteur.fr

Abstract. There are various techniques for reconstructing phylogenetic trees from data, and in this context the problem of determining how distant two such trees are from each other arises naturally. Various metrics (NNI, SPR, TBR) for measuring the distance between two phylogenies have been defined. Another way of comparing two trees $\mathcal{T}$ and $\mathcal{U}$ is to compute the so called *maximum agreement forest* of these trees. Informally, the number of components of an agreement forest tells how many edges need to be cut from each of $\mathcal{T}$ and $\mathcal{U}$ so that the resulting forests agree, after performing some forced edge contractions. This problem is known to be $\mathcal{NP}$-hard. It was introduced by Hein et al. [3], who presented an approximation algorithm for it, claimed to have approximation ratio 3. We present here a 3-approximation algorithm for this problem and show that the performance ratio of Hein's algorithm is 4.

1 Introduction

Phylogenetic trees or *phylogenies* are a standard model for representing evolutionary processes, mostly involving biological entities such as species or genes. By a phylogenetic tree we mean a rooted unordered tree whose leaves are uniquely labeled with elements of some set S, and whose internal nodes are unlabeled and have exactly two children. The elements of S stand for the contemporary taxa whose evolutionary relationships one intends to model. These taxa correspond to the leaves of the tree, whereas the ancestral taxa correspond to its internal nodes, so that for each ancestral taxon all of its nearest derived taxa are depicted as its children in the tree.

There are various techniques for reconstructing phylogenetic trees from data, and in this setting the problem of determining how distant two such trees are from each other arises naturally. Various metrics, such as NNI (nearest-neighbor

* This research is part of CAPES-COFECUB Project 272/99-II.

** Supported by CNPq Grant Proc. 142307/97-1. Also supported by CAPES Grant BEX 0650-99/4 during her visit to Institut Pasteur, where part of this research was done.

*** Partially supported by CNPq (Procs. 304527/89-0 and 464114/00-4) and by ProNEx Project 107/97 (Proc. CNPq 664107/97-4).

interchange), SPR (subtree prune and regraft) and TBR (tree bisection and reconnection) for measuring the distance between two phylogenies have been defined [5, 4, 2]. Many results relating these concepts are presented by Allen and Steel [1]. In particular, they show that the size of a maximum agreement forest of two trees is precisely the TBR-distance between them.

We are concerned here with the problem of finding the size of a maximum agreement forest of two trees, which is known to be $\mathcal{N}P$-hard [3, 1]. The formal definition of this problem is given in the next section. We present a 3-approximation algorithm for this problem and show that a previous algorithm by Hein et al. [3], claimed to have performance ratio 3, has performance ratio 4.

The algorithm we shall describe is simple, but the analysis of its approximation ratio is quite long and technical. To give an idea of the analysis we have to introduce many concepts that make the notation somewhat heavy. We hope once the formal definitions are given and the proof is outlined, the reader will be able to complete the details that we shall omit.

In Section 2 we present the basic concepts that will be needed to define the problem and to describe the algorithm and its proof, in Section 3 we outline our approximation algorithm, and in Section 4 we sketch the proof that its approximation ratio is 3 and give a family of instances for which Hein's algorithm attains approximation ratio 4.

2 Basic definitions

A *phylogenetic tree* consists of an unordered rooted tree, called its *topology*, such that each internal node has two children, and of a set of labels which are mapped one-to-one to the leaves of the tree. If $\mathcal{T}$ is a phylogenetic tree, then $T_\mathcal{T}$ denotes its topology, $L_\mathcal{T}$ its set of leaves, $S_\mathcal{T}$ its set of labels, $f_\mathcal{T}$ its one-to-one label-to-leaf mapping, and $r_\mathcal{T}$ its root. Since we consider phylogenetic trees as having rooted topologies, these are naturally oriented: we assume the arcs are oriented towards its root.

For each arc e in a phylogenetic tree $\mathcal{T}$, we denote by $l(e)$ its *lower endpoint* (the endpoint of e which is the farthest from the root) and by $u(e)$ its *upper endpoint* (the other endpoint of e). The *lowest common ancestor* of a set of $m \geq 1$ nodes $v_1, \ldots, v_m$ of $\mathcal{T}$, written $\text{lca}_\mathcal{T}(v_1, \ldots, v_m)$, is the farthest node from the root that is an ancestor of all v_i, $1 \leq i \leq m$. For each node u in $\mathcal{T}$, we denote by $D(u)$ the set of leaves in $\mathcal{T}$ that are descendants of u.

The concept of phylogenetic tree can be generalized so as to consider topologies with more than one component. Such phylogenies are called *phylogenetic forests*. Each component in the topology of a phylogenetic forest corresponds to a *component* of the forest, with its own topology, root, leaf set, label set and label-to-leaf mapping. The notation adopted for all these objects is the same as that for phylogenetic trees, except that we use $F_\mathcal{F}$ instead of $T_\mathcal{F}$ to denote the topology of a phylogenetic forest $\mathcal{F}$.

2.1 Restrictions and agreement forests

Two phylogenetic forests $\mathcal{G}$ and $\mathcal{H}$ are said to be *isomorphic* if $S_{\mathcal{G}} = S_{\mathcal{H}}$, their topologies $F_{\mathcal{G}}$ and $F_{\mathcal{H}}$ are isomorphic, and the isomorphism preserves both the roots of the components and the labels of the leaves. If S is a subset of $S_{\mathcal{V}}$ for some component $\mathcal{V}$ of $\mathcal{G}$, then the *simple restriction* of $\mathcal{G}$ to S, denoted by $\mathcal{G}|S$, is the phylogenetic tree $\mathcal{G}'$ such that $S_{\mathcal{G}'} = S$; $L_{\mathcal{G}'} = f_{\mathcal{G}}(S) := \{f_{\mathcal{G}}(a) : a \in S\}$; for each pair of nodes u, v in $\mathcal{G}'$ we have $\mathrm{lca}_{\mathcal{G}'}(u,v) = \mathrm{lca}_{\mathcal{G}}(u,v)$; and $r_{\mathcal{G}'} = \mathrm{lca}_{\mathcal{G}'}(L_{\mathcal{G}'})$. Let $\mathcal{G}[S]$ denote the minimal subtree of $F_{\mathcal{G}}$ which connects all leaves with labels in S.

Let $\mathcal{G}$ be a phylogenetic forest and $\mathcal{S} = \{S_i : 1 \leq i \leq m\}$ a family of $m \geq 1$ subsets of $S_{\mathcal{G}}$ such that for each S_i we have $S_i \subseteq S_{\mathcal{V}}$ for some component $\mathcal{V}$ of $\mathcal{G}$, and the trees $\mathcal{G}[S_i]$ are pairwise node-disjoint. Then the *restriction* of $\mathcal{G}$ to $\mathcal{S}$, written $\mathcal{G}|\mathcal{S}$, is the phylogenetic forest composed of the node-disjoint components $\mathcal{G}|S_i$. The *size* of $\mathcal{G}|\mathcal{S}$ is the cardinality of $\mathcal{S}$. If $\mathcal{G}' = \mathcal{G}|\mathcal{S}$, then we denote the size of $\mathcal{G}'$ by $|\mathcal{G}'|$. If $S_{\mathcal{G}|\mathcal{S}} = S_{\mathcal{G}}$, then $\mathcal{G}|\mathcal{S}$ is a *full restriction* of $\mathcal{G}$. An *agreement phylogenetic forest*, abbreviated to an *agreement forest*, of two given phylogenetic forests $\mathcal{G}$ and $\mathcal{H}$ is a phylogenetic forest $\mathcal{F}$ that is isomorphic to a full restriction of $\mathcal{G}$ and to a full restriction of $\mathcal{H}$. An *agreement forest* is said to be *maximum* if it has *minimum* size. The problem of computing a maximum agreement forest of two given phylogenetic trees $\mathcal{T}$ and $\mathcal{U}$ is the *Maximum Agreement Forest (MAF) problem*, abbreviated as the *MAF problem*.

We can prove the following simple claim from the above definitions:

Lemma 1. *If $\mathcal{V}_1$ and $\mathcal{V}_2$ are two simple restrictions of a phylogenetic forest $\mathcal{G}$ such that the trees $\mathcal{G}[S_{\mathcal{V}_1}]$ and $\mathcal{G}[S_{\mathcal{V}_2}]$ are node-disjoint, a is the label of a leaf in $\mathcal{V}_1$, b if the label of a leaf in $\mathcal{V}_2$, and $f_{\mathcal{G}}(a)$ and $f_{\mathcal{G}}(b)$ are sibling leaves in $\mathcal{G}$, then either $\mathcal{V}_1$ or $\mathcal{V}_2$ consists of a unique node.*

2.2 Links

Let $\mathcal{G}$ be a phylogenetic forest and $\mathcal{G}'$ be a full restriction of $\mathcal{G}$. We say an arc e of $\mathcal{G}$ is said to be a *link* of $\mathcal{G}$ with respect to $\mathcal{G}'$ if there are $m \geq 1$ components $\mathcal{V}_1, \ldots, \mathcal{V}_m$ of $\mathcal{G}'$ such that $\bigcup_{1 \leq i \leq m} L_{\mathcal{V}_i} = D(l(e))$ in $\mathcal{G}$.

The following lemma gives an important property of links:

Lemma 2. *Let $\mathcal{G}$ be a phylogenetic forest and $\mathcal{G}'$ be a full restriction of $\mathcal{G}$. Let e be an arc of $\mathcal{G}$. If e is not a link with respect to $\mathcal{G}'$, then:*

1. *there is a unique component $\mathcal{V}$ in $\mathcal{G}'$ such that $S_{\mathcal{V}} \cap D(l(e)) \neq \emptyset$ and $S_{\mathcal{V}} \setminus D(l(e)) \neq \emptyset$;*
2. *$\mathcal{V}[S_{\mathcal{V}} \cap D(l(e))]$ and $\mathcal{V}[S_{\mathcal{V}} \setminus D(l(e))]$ are node-disjoint.*

Observe that the definition of a link depends on the full restriction one is considering. Hereafter, full restrictions will be given by the context and therefore not explicitly written.

2.3 Eliminations

Let $\mathcal{W}$ be a phylogenetic tree, and e an arc of $\mathcal{W}$. In what follows, we define the *elimination* of an arc e, which is the basic operation used in the forthcoming algorithms.

The elimination of an arc e is the operation that yields the phylogenetic forest whose components are $\mathcal{W}|D(l(e))$ and $\mathcal{W}|(S_{\mathcal{W}} \setminus D(l(e)))$. This operation can be implemented by *removing* arc e from $\mathcal{W}$ (which splits $\mathcal{W}$ into $\mathcal{W}|D(l(e))$ and $\mathcal{W}[S_{\mathcal{W}} \setminus D(l(e))]$) and by performing a *forced contraction* of arc e' in $\mathcal{W}[S_{\mathcal{W}} \setminus D(l(e))]$ where $u(e') = u(e)$ (which yields $\mathcal{W}|(S_{\mathcal{W}} \setminus D(l(e)))$ from $\mathcal{W}[S_{\mathcal{W}} \setminus D(l(e))]$). Figure 1 illustrates this definition. This operation can be defined for phylogenetic forests as well, by letting all components that do not contain arc e remain the same and applying elimination on the component that contains e.

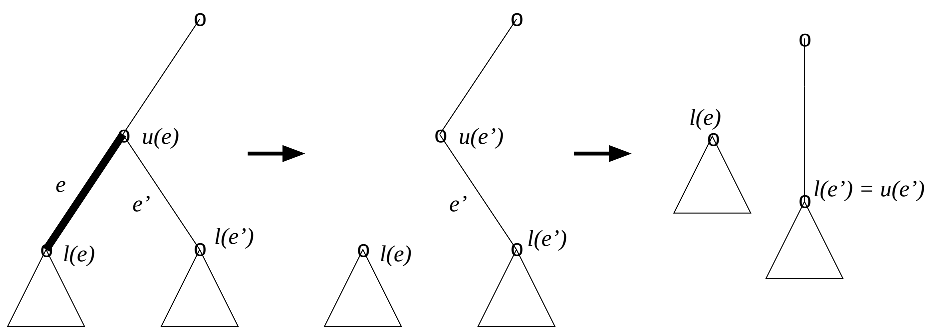

Fig. 1. Elimination of an arc e.

The lemma below exhibits the equivalence between a full restriction of a phylogenetic forest and a sequence of arc eliminations starting from $\mathcal{W}$.

Lemma 3. *Let $\mathcal{W}$ be a phylogenetic forest. Then performing a sequence of m eliminations starting from $\mathcal{W}$ yields a full restriction of $\mathcal{W}$ with size $|\mathcal{W}| + m$. Conversely, let $\mathcal{S}$ be a partition of $S_{\mathcal{W}}$ such that the full restriction $\mathcal{W}|\mathcal{S}$ is defined. Then $\mathcal{W}|\mathcal{S}$ can be obtained by performing a sequence of $|\mathcal{S}| - |\mathcal{W}|$ eliminations.*

3 Algorithms

The algorithm takes as input two phylogenetic trees $\mathcal{T}$ and $\mathcal{U}$ with $S_{\mathcal{T}} = S_{\mathcal{U}}$, and returns an agreement forest of $\mathcal{T}$ and $\mathcal{U}$ with size up to 3 times the size of a maximum agreement forest of $\mathcal{T}$ and $\mathcal{U}$. Our algorithm proceeds by cutting arcs and shrinking pairs of sibling leaves in $\mathcal{T}$ and $\mathcal{U}$ until two isomorphic full restrictions are obtained.

The algorithm is iterative: each iteration starts with two restrictions obtained from $\mathcal{T}$ and $\mathcal{U}$ by the sequence of operations performed up to that iteration by the algorithm. In each iteration, the algorithm searches for a pair of sibling leaves in one of the restrictions and, if it succeeds, it identifies the *case* that is satisfied by this pair in the other (see Figure 2). According to the case, the algorithm then selects and applies on both restrictions a predefined sequence of operations, which we call *transactions* (see Section 3.2), is selected and applied on both restrictions.

An earlier algorithm, designed by Hein, Jiang, Wang and Zhang [3], proceeds mostly within this same framework. In our paper, we refer to this algorithm as Algorithm 1, and ours shall be called Algorithm 2. Section 3.3 outlines both algorithms.

In what follows, we shall consider the iterations numbered $1, 2, \ldots$, and so on. In each iteration i, we label the operations, dividing them into two sequences $(x_i^1, \ldots, x_i^{g_i})$ and $(y_i^1, \ldots, y_i^{h_i})$, where g_i and h_i are the number of operations performed at iteration i on (a full restriction of) $\mathcal{T}$ and (a full restriction of) $\mathcal{U}$ respectively. Let $\mathcal{G}_1 := \mathcal{T}$ and $\mathcal{H}_1 := \mathcal{U}$, and for $i \geq 2$ let $\mathcal{G}_i$ be the full restriction we get from $\mathcal{G}_{i-1}$ after executing the g_{i-1} operations of iteration $i-1$. Let $\mathcal{G}_i^1 := \mathcal{G}_i$, and for $2 \leq j \leq g_i + 1$ let $\mathcal{G}_i^j$ be the full restriction we get by applying operation x_i^{j-1} on $\mathcal{G}_i^{j-1}$. Similarly, we define $\mathcal{H}_i$ and $\mathcal{H}_i^j$ for each $i \geq 1$ and $j \in \{1, \ldots, h_i + 1\}$.

For the analysis of the performance of the algorithm, we consider throughout the iterations two full restrictions, $\mathcal{T}'$ and $\mathcal{U}'$, of $\mathcal{T}$ and $\mathcal{U}$ respectively, which in the beginning of the first iteration are isomorphic to a maximum agreement forest $\mathcal{F}$ of $\mathcal{T}$ and $\mathcal{U}$. For each operation done on $\mathcal{T}$ (respectively $\mathcal{U}$), a "similar" operation is performed on $\mathcal{T}'$ (respectively $\mathcal{U}'$), in order to preserve the invariants of Algorithm 2 to be stated in the sequel. In the same way as we have done for $\mathcal{T}$ and $\mathcal{U}$, for each iteration i we also define for $\mathcal{T}'$ and $\mathcal{U}'$ the sequences of operations $({x'}_i^j : 1 \leq j \leq g_i)$, $({y'}_i^j : 1 \leq j \leq h_i)$, and the corresponding full restrictions $\mathcal{G}'_i$, $\mathcal{H}'_i$, ${\mathcal{G}'}_i^j$ and ${\mathcal{H}'}_i^j$.

Roughly speaking, the initial full restrictions $\mathcal{T}'$ and $\mathcal{U}'$ give a block configuration attained by an optimum agreement forest $\mathcal{F}$ of $\mathcal{T}$ and $\mathcal{U}$ (each block being a component of $\mathcal{F}$). This (fixed) "ideal" block configuration is used to derive some information of the size of the agreement forest returned by the algorithm. Whenever the algorithm performs a cut of an arc e, a "similar" operation is performed on $\mathcal{T}'$ and $\mathcal{U}'$: if e is not a link then this cut splits a block of the current configuration, if e is a link then no new block is derived. With this procedure, we can get hold of the number of the "additional" blocks our algorithm produces.

3.1 Operations

In this section we define the operations that can be applied on the phylogenetic trees $\mathcal{T}$ and $\mathcal{U}$.

Let $\mathcal{W}$ be a phylogenetic tree. The operation of *cutting an arc* e of $\mathcal{W}$ is the elimination of e defined in Section 2.3. The operation of *shrinking a pair of* *sibling leaves u and v to leaf u* corresponds to the elimination of the arc e such that $l(e) = v$, followed by the removal of the isolated node v. This operation is used whenever a pair of labels which yields a pair of sibling leaves both in $\mathcal{G}_i^j$ and $\mathcal{H}_i^j$ is located.

As Algorithm 2 proceeds performing cuts and shrinkings in $\mathcal{T}$ and $\mathcal{U}$, their full restrictions $\mathcal{T}'$ and $\mathcal{U}'$ must have some of their arcs eliminated as well, if we want to retain as invariants the following properties:

$$\mathcal{G'}_i^j \text{ is a full restriction of } \mathcal{G}_i^j \text{ and } \mathcal{H'}_i^j \text{ is a full restriction of } \mathcal{H}_i^j.$$

For each arc e eliminated, if e is a link, then the invariants are satisfied just by making $\mathcal{G'}_i^{j+1} := \mathcal{G'}_i^j$ and $\mathcal{H'}_i^{j+1} := \mathcal{H'}_i^j$. Otherwise, it can be proved from Lemmas 2 and 3 that for each arc e eliminated that is not a link, there exists an arc in $\mathcal{T}'$ or in $\mathcal{U}'$ respectively whose elimination preserves the invariants.

A similar result is valid for $\mathcal{U}$ and $\mathcal{U}'$. Henceforth, we shall take these eliminations in $\mathcal{T}'$ and $\mathcal{U}'$ for granted at each elimination in $\mathcal{T}$ and $\mathcal{U}$.

When a shrinking is performed and a one-node component is removed in $\mathcal{G}_{i+1}$ and $\mathcal{H}_{i+1}$, we must also remove the corresponding component in $\mathcal{G'}_{i+1}$ and $\mathcal{H'}_{i+1}$. Observe that after shrinkings, $\mathcal{G}_i$ and $\mathcal{H}_i$ are no longer full restrictions of $\mathcal{G}_1$ and $\mathcal{H}_1$ respectively.

3.2 Cases and transactions

Suppose that at the beginning of iteration i there is at least an arc in $\mathcal{G}_i$. Then $\mathcal{G}_i$ admits at least a pair of sibling leaves. Let a and b be its labels. We consider five cases, each of them defined by a different configuration yielded in $\mathcal{H}_i$ by the leaves $f_{\mathcal{H}_i}(a)$ and $f_{\mathcal{H}_i}(b)$.

If $f_{\mathcal{H}_i}(a)$ and $f_{\mathcal{H}_i}(b)$ belong to the same component in $\mathcal{H}_i$, then the *(a,b)-axis* in $\mathcal{H}_i$ is the (unique) path in $\mathcal{H}_i$ connecting $f_{\mathcal{H}_i}(a)$ and $f_{\mathcal{H}_i}(b)$, and the *stems* of the (a,b)-axis are the arcs e in $\mathcal{H}_i$ such that $u(e)$ belongs to the axis but e does not. The stems are labeled s_1 through s_k according to the order in which they appear along the axis, with stem s_1 being the nearest to $f_{\mathcal{H}_i}(a)$. The cases are shown in Figure 2. Case 1 differs from Case 2 in that there is a fixed limit on the number k of stems: in Case 1 we have $1 \leq k \leq 2$, and in Case 2 we have $k \geq 3$.

The transactions for each case are listed below. We denote by $\textsc{Cut}(\mathcal{G}, v)$ the operation "cut the arc in $\mathcal{G}$ whose lower endpoint is v", and by $\textsc{Srk}(\mathcal{G}, u, v)$ the operation "shrink pair u, v of sibling leaves to leaf u".

Case 1: For each $j \in \{1, \ldots, k\}$ do $\textsc{Cut}(\mathcal{H}_i^j, l(s_j))$.
Case 2: $\textsc{Cut}(\mathcal{G}_i^1, f(a))$; $\textsc{Cut}(\mathcal{G}_i^2, f(b))$; $\textsc{Cut}(\mathcal{H}_i^1, f(a))$; $\textsc{Cut}(\mathcal{H}_i^2, f(b))$.
Case 3: $\textsc{Cut}(\mathcal{G}_i^1, f(a))$; $\textsc{Cut}(\mathcal{G}_i^2, f(b))$; $\textsc{Cut}(\mathcal{H}_i^1, f(a))$; $\textsc{Cut}(\mathcal{H}_i^2, f(b))$.
Case 4: $\textsc{Cut}(\mathcal{G}_i^1, f(b))$.
Case 5: $\textsc{Srk}(\mathcal{G}_i^1, f(a), f(b))$; $\textsc{Srk}(\mathcal{H}_i^1, f(a), f(b))$.

We give below an important lemma, whose proof is straightforward from Lemmas 1 and 2.

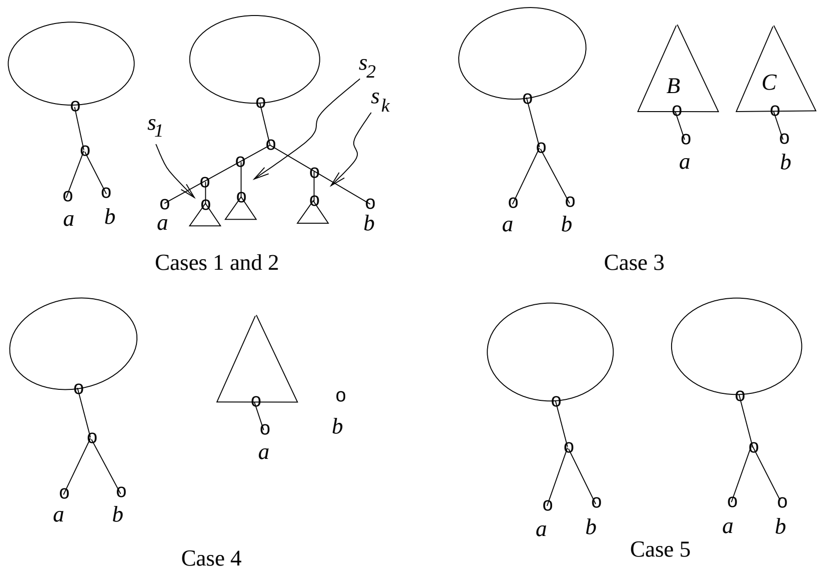

Fig. 2. Cases for Algorithm 2. In each case $\mathcal{G}_i$ is to the left and $\mathcal{H}_i$ is to the right.

Lemma 4. *Let $\mathcal{G}$ and $\mathcal{H}$ be two phylogenetic forests with $S_{\mathcal{G}} = S_{\mathcal{H}}$, and let $\mathcal{G}'$ and $\mathcal{H}'$ be two full restrictions of $\mathcal{G}$ and $\mathcal{H}$ respectively, such that $\mathcal{H}'$ is isomorphic to a full restriction of $\mathcal{G}'$. Suppose that $f_{\mathcal{G}}(a)$ and $f_{\mathcal{G}}(b)$ form a pair of sibling leaves in $\mathcal{G}$.*

1. *If $f_{\mathcal{G}'}(a)$ and $f_{\mathcal{G}'}(b)$ are separated in $\mathcal{G}'$, then at least one of the arcs incident to $f_{\mathcal{H}}(a)$ or $f_{\mathcal{H}}(b)$ is a link.*
2. *If $f_{\mathcal{H}'}(a)$ and $f_{\mathcal{H}'}(b)$ are in the same component of $\mathcal{H}'$, and the (a, b)-axis in $\mathcal{H}$ admits at least one stem, then all of its stems in $\mathcal{H}$ are links.*

3.3 Algorithms 1 and 2

We give below an outline of Algorithm 2. We remark that before the algorithm outputs $\mathcal{H}$ as a solution, those nodes which are removed from $\mathcal{H}$ at shrinkings must be reattached to it in reverse order.

```
G_1 := T;
H_1 := U;
i := 1;
While there is a pair of sibling leaves f_{G_i}(a) and f_{G_i}(b) in G_i
        Find out which case is satisfied by f_{H_i}(a) and f_{H_i}(b);
        Apply the corresponding transaction;
        i := i + 1;
Return H_i with all nodes that were removed at shrinkings reattached.
```

Algorithm 1 differs from Algorithm 2 by defining otherwise the limit on the number k of stems in Case 1. In Algorithm 1, if $k = 1$ we have Case 1 and if $k \geq 2$ we have Case 2. The transactions applied in each of these cases are the same as before.

It is not difficult to conclude that Algorithm 2 can be implemented to run in polynomial time in the size of the instance. In each iteration i, whenever a pair (a, b) of sibling leaves is considered in $\mathcal{G}_i$, all we need to find out the case satisfied by (a, b) is to find (if existent) the (a, b)-axis in $\mathcal{H}_i$ and check the number k of stems of this axis. According to the value of k we are in one among cases 1, 2 or 5. If no such path exists, then we are in Case 3 or 4. In all these cases, the transactions can be performed in time linear in the size of the corresponding tree. Thus, since the number of iterations is bounded by the number of leaves of the original trees, the claim follows. This rough analysis allows to say that if the original trees have m arcs, then Algorithm 2 can be implemented to run in $O(m^2)$. The same holds for Algorithm 1.

4 Approximation ratio

Our proof technique uses an accounting system, according to which an initial amount of *credits* is alloted to one of the trees given as input, and this tree (or rather a full restriction of it) is charged by means of *debits* recorded whenever an operation needs payment. The initial amount of credits is related to the size of a maximum agreement forest whereas the debits are related to the number of cuts made by the algorithm, and the goal in this proof technique is to show that the credits suffice to pay the debits recorded. This technique also follows the sketch given in [3], but here we improve the control over the charged elements in the trees. It turns out that some debits recorded can only be paid later; in such situations these debits are attached to special sets of arcs which we call *barriers*. These arc sets not only work as "bags" of postponed credits and as such are fit to be charged for the postponed debits, but they also keep these debits apart from each other, which avoids their clustering and the need for paying great amounts of them at once.

4.1 Barriers

Let B be a node set of a phylogenetic forest $\mathcal{G}$. We denote by $U_{\mathcal{G}}(B)$ the set of arcs whose upper endpoints are the lca's in $\mathcal{G}$ of at least two nodes in B.

Take some $\mathcal{H}_i^j$ with $(i, j) \neq (1, 1)$. Consider the arcs e and e' as shown in Figure 1 for the elimination performed to produce $\mathcal{H}_i^j$. A node set B in $\mathcal{H}_i^j$ with at least two nodes is a *barrier* if:

1. B is a *primary barrier*. That is, $j = 1$; iteration $i - 1$ satisfies Case 2, with $f_{\mathcal{H}'_{i-1}}(a)$ and $f_{\mathcal{H}'_{i-1}}(b)$ in a component of $\mathcal{H}'_{i-1}$ which has at least three leaves; $B = \{l(e) : e \text{ is a stem of the } (a, b)\text{-axis in } \mathcal{H}_{i-1}\}$; or

2. B admits an *antecessor barrier* B'. That is, $j \geq 2$ and there exists a barrier B' in $\mathcal{H}_i^{j-1}$ such that one of the following conditions holds:
   (a) $e, e' \in U_{\mathcal{H}_i^{j-1}}(B')$; $B = B' \cap D(l(e))$ or $B = B' \setminus D(l(e))$, the one with the largest cardinality (if there is a tie, choose any of them); or
   (b) $e, e' \notin U_{\mathcal{H}_i^{j-1}}(B')$; $B = B'$.

Below we list some properties of barriers.

Lemma 5. *Let $\mathcal{H}_i^j$ be such that $(i, j) \neq (1, 1)$, and let B be a barrier of $\mathcal{H}_i^j$. Then all arcs in $U_{\mathcal{H}_i^j}(B)$ are links.*

Lemma 6. *In some $\mathcal{H}_i^j$ with $(i, j) \neq (1, 1)$, consider two distinct barriers B and C. Let $t_B := \mathrm{lca}_{\mathcal{H}_i^j}(B)$ and $t_C := \mathrm{lca}_{\mathcal{H}_i^j}(C)$. Then $U_{\mathcal{H}_i^j}(B)$ and $U_{\mathcal{H}_i^j}(C)$ are arc-disjoint and if they have a node in common, then either $t_B \in C$ or $t_C \in B$.*

The notion of barrier is fundamental in our proof that Algorithm 2 yields approximation ratio 3 for the maximum agreement forest problem. In the next subsection, we introduce credits and debits and the rules that command them along the iterations of the algorithm, as well as their relation with barriers.

4.2 Credits and debits

At the beginning of the first iteration, we consider that a certain amount of *credits* is *placed* on the unique component of $\mathcal{U}$. As Algorithm 2 proceeds doing cuts and shrinkings, those operations which must be paid *record* debits equivalent to their due fees, and our proof aims to guarantee that at the end of the algorithm, each debit recorded is paid with a credit out of those placed on $\mathcal{U}$ at the beginning. However, as the algorithm proceeds, the credits are *kept* by the components of $\mathcal{U}$, and they must be *released* before they can be used to pay any recorded debit. Also, as long as they are kept, the credits must be *redistributed* among the components at each elimination performed on $\mathcal{U}$.

The precise rules for debit recording and credit placing, releasing and redistributing are as follows. Let $\mathcal{F}$ be the maximum agreement forest taken as reference in Section 3. At the beginning, we place $3(|\mathcal{F}| - 1)$ credits on $\mathcal{U}$. These credits are released and redistributed when eliminations are performed. We have already seen that each elimination splits a component $\mathcal{W}$ of its argument $\mathcal{H}_i^j$ into two new components, $\mathcal{W}|D(l(e))$ and $\mathcal{W}|(S_{\mathcal{W}} \setminus D(l(e)))$. Let m be the number of components $\mathcal{W}'$ of $\mathcal{H}'^j_i$ such that $S_{\mathcal{W}'} \subseteq S_{\mathcal{W}}$, $m^\cap$ be the number of components $\mathcal{W}'$ of $\mathcal{H}'^{j+1}_i$ such that $S_{\mathcal{W}'} \subseteq D(l(e))$, and $m^\setminus$ be the number of components $\mathcal{W}'$ of $\mathcal{H}'^{j+1}_i$ such that $S_{\mathcal{W}'} \subseteq S_{\mathcal{W}} \setminus D(l(e))$. It can easily be verified that the following rules preserve credits throughout the eliminations.

If we eliminate a link e of a component $\mathcal{W}$ of some $\mathcal{H}_i^j$, then from the $3(m-1)$ credits placed on $\mathcal{W}$, three credits are released, and each of the two resulting components keeps now respectively $3(m^\cap - 1)$ and $3(m^\setminus - 1)$ credits. Otherwise, if we eliminate an arc which is not a link, then no credit is released and each of the resulting components keeps respectively $3(m^\cap - 1)$ and $3(m^\setminus - 1)$ credits.

Among the operations used in Algorithm 2, only cuts record debits. Each cut operation records one debit, which either is paid with credits released in the same iteration at which the cut is performed, or (if the cut arc is not a link) is attached to a suitable barrier. In fact, at each iteration either a credit is released or a primary barrier is created. It turns out that each primary barrier has two debits attached to itself. As these debits are placed on the whole primary barrier rather than on any of its arcs, they are paid as soon as the algorithm begins to eliminate arcs of the barrier.

4.3 An upper bound for the approximation ratio of Algorithm 2

Since all arcs in a barrier are links (Lemma 5), barriers are suited for receiving postponed debits, that is those debits which could not be handled at the iteration in which they were recorded because no credit was released in that iteration. Also, since distinct barriers are arc-disjoint (Lemma 6), then postponed debits are kept apart, and thus the maximum amount of postponed debits that must be paid at each iteration is bounded. For Algorithm 2 this maximum is 1 since each primary barrier has at least three nodes. For Algorithm 1 we can have primary barriers with only two nodes, so we must guarantee that two postponed debits can be paid.

Our proof consists in verifying that at the end of Algorithm 2, all recorded debits are paid. Once this is done, we have that the number of cuts done by Algorithm 2 in $\mathcal{U}$ is $|\mathcal{H}_i| - 1$, where i denotes the last iteration, and that the number of cuts is equal to the number of recorded debits, which is less than or equal to the number of credits alloted to $\mathcal{U}$ in the beginning, that is $3(|\mathcal{F}| - 1)$; so $|\mathcal{H}_i| \leq 3|\mathcal{F}|$.

A slight modification in this proof technique yields a proof that Algorithm 1 has approximation ratio at most 4. It suffices to place $4(|\mathcal{F}| - 1)$ credits on $\mathcal{U}$ at the beginning of the algorithm and release four credits whenever links are eliminated.

4.4 A lower bound for the approximation ratio of Algorithm 1

In this section, we exhibit a family of instances which shows that the approximation ratio for Algorithm 1 gets close to 4.

Figure 3 shows an instance of this family. Tree $\mathcal{U}$ (*bottom*) has two parts, one comprising m Cases 1 with $k = 2$ piled in a stack, and the other composed by m single leaves (in the instance shown above we have $m = 9$). Tree $\mathcal{T}$ (*top*) is constructed so as to make Algorithm 1 handle all Cases 1 in the stack (cutting leaves 1–2–7–8– . . . –14 in $\mathcal{U}$ in the example) and then alternate cuts between the remaining leaves out of the stack (0– . . . –15) and the single leaves (28–. . . –36).

It turns out that for any $m \geq 1$, Algorithm 1 yields an agreement forest with $4m$ components, while there is an agreement forest with at most $m + 2$ components (in Figure 3 consider the agreement forest which contains leaves 1–2–7–. . . –14 together with leaves 28–29–30–. . . –36 as a single component and the other leaves as isolated nodes).

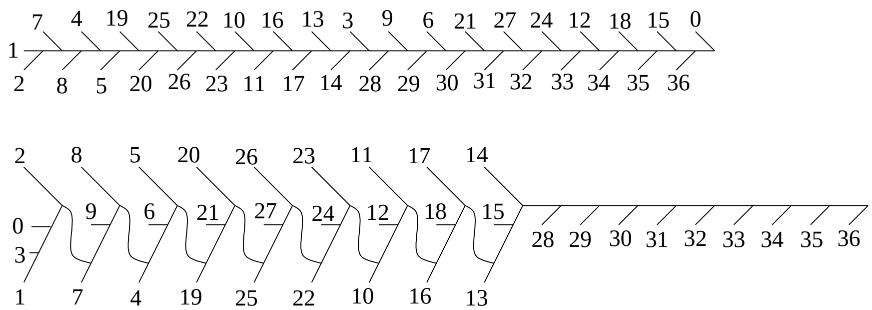

Fig. 3. An example showing that Algorithm 1 attains approximation ratio close to 4.

This family of instances can be modified to show that the approximation ratio of Algorithm 2 is at least 3. It suffices to substitute the Cases 1 with $k = 2$ by Cases 2 with $k = 3$.

5 Final remarks

The concept of barriers and the proof technique based on an accounting system were useful to prove the approximation ratio of Algorithm 1 and Algorithm 2. The barriers were introduced because of the transactions defined for Case 2. We note, however, that if we can provide a set of transactions (one for each case) with the property that each of them performs at most k cuts and eliminates at least p links, then an approximation ratio k/p can be guaranteed. These ideas will be further explored in a forthcoming paper. We hope that the concept of barrier, in spite of being complicated, may lead to some further improvement.

6 Acknowledgments

We thank the referee for the suggestions that improved the presentation of this paper, and Cristina G. Fernandes for the helpful discussions.

References

1. B. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Submitted to the Annals of Combinatorics*, 2001.
2. B. dasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On distances between phylogenetic trees. In *Proceedings of the 8th ACM-SIAM Symposium of Discrete Algorithms*, pages 427–436, 1997.
3. J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.
4. M. Li, J. Tromp, and L. Zhang. On the nearest neighbor interchange distance between evolutionary trees. *Journal on Theoretical Biology*, 182(4):463–467, 1996.

5. D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. H. Hillis. Phylogenetic inference. In D. Hillis, C. Moritz, and B. Mable, editors, *Molecular Systematics*, pages 407–513. Sinauer Associates, 1996.