Analysis of Tree Edit Distance Algorithms

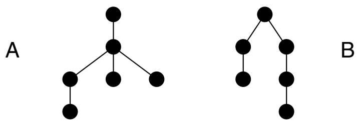
Serge Dulucq[1] and Hélène Touzet[2]

[1] LaBRI – Université Bordeaux I
33 405 Talence cedex, France
Serge.Dulucq@labri.fr
[2] LIFL – Université Lille 1
59 655 Villeneuve d'Ascq cedex, France
Helene.Touzet@lifl.fr

Abstract. In this article, we study the behaviour of dynamic programming methods for the tree edit distance problem, such as [4] and [2]. We show that those two algorithms may be described in a more general framework of *cover strategies*. This analysis allows us to define a new tree edit distance algorithm, that is optimal for cover strategies.

1 Introduction

One way of comparing two ordered trees is by measuring their edit distance: the minimum cost to transform one tree into another using elementary operations. This problem has several application areas. Examples include comparison of hierarchically structured data [1], such as XML documents, or alignment of RNA secondary structures in computational biology [3]. There are currently two main algorithms for solving the tree edit distance: Zhang-Shasha [4] and Klein [2]. Both algorithms use the dynamic programming paradigm, and they may be seen as an extension of the basic string edit distance algorithm. The difference lies in the set of subforests that are involved in the decomposition. This leads to different complexities: Zhang-Shasha is in n^4 in worst case, as Klein is in $n^3 \log(n)$. However, this does not mean that Klein's approach is strictly better than Zhang-Shasha's. The performance of each algorithm depends on the shape of the two trees to be compared. For example, for the pair of trees (A, B) below Klein's approach needs 84 distinct recursive calls, whereas the Zhang-Shasha's approach needs only 72 distinct recursive calls.



The purpose of this paper is to present a general analysis of dynamic programming for tree edit distance algorithms. For that, we consider a class of tree decompositions, which involves Zhang-Shasha and Klein algorithms. We study the complexity of those decompositions by counting the exact number of distinct

R. Baeza-Yates et al. (Eds.): CPM 2003, LNCS 2676, pp. 83–95, 2003.

recursive calls for the underlying corresponding algorithm. As a corollary, this gives the number of recursive calls for Zhang-Shasha, that was a known result [4], and for Klein, that was not known. In the last section, we take advantage of this analysis to define a new edit distance algorithm for trees, which improves Zhang-Shasha and Klein algorithms with respect to the number of recursive calls.

2 Edit Distance for Trees and Forests

Definition 1 (Trees and forests). *A* tree *is a node (called the root) connected to an ordered sequence of disjoint trees. Such a sequence is called a* forest*. We write $l(A_1 \circ \cdots \circ A_n)$ for the tree composed of the node l connected to the sequence of trees $A_1, \ldots, A_n$.*

This definition assumes that trees are ordered trees, and that the nodes are labeled. Trees with labeled edges can by handled similarly. In the sequel, we may use the word *forest* for denoting both forests and trees, a tree being a sequence reduced to a single element.

Notation 1 *Let F be a forest.*

- *$|F|$ denotes the number of nodes of the forest F,*
- *$\mathcal{SF}(F)$ is the set of all subforests of F,*
- *$F(i)$, i is a node of F, denotes the subtree of F rooted at i,*
- *$deg(i)$ is the degree of i, that is the number of children of i.*

As usual, the edit distance relies on three elementary edit operations: the *substitution*, which consists in replacing a label of a node by another label, the *insertion* of a node, and the *deletion* of a node. Each edit operation is assigned a cost: c_s, c_i and c_d denote the costs for (respectively) substituting a label, inserting and deleting a node.

Definition 2 (Edit distance). *Let F and G be two forests. The* edit distance *between F and G, denoted $d(F, G)$, is the minimal cost of edit operations needed to transform F into G.*

Before investigating algorithms for trees, we recall some basic facts about edit distance for strings. The usual well-known dynamic programming algorithm proceeds by solving the more general problem of measuring the distance for all pairs of suffixes of the two strings. This is based on the following recursive relationship:

$$d(xu, yv) = \min \begin{cases} c_d(x) + d(u, yv) \\ c_i(y) + d(xu, v) \\ c_s(x, y) + d(u, v) \end{cases}$$

where u and v are strings, x and y are alphabet symbols. It is likely to develop an alternative algorithm by constructing the distance for all pairs of prefixes.

$$d(ux, vy) = \min \begin{cases} c_d(x) + d(u, vy) \\ c_i(y) + d(ux, v) \\ c_s(x, y) + d(u, v) \end{cases}$$

For strings, these two approaches are of course equivalent, since the number of pairs of prefixes equals the number of pairs of suffixes. Let us now come to our main concern, edit distance for trees. As for strings, the distance can be computed with dynamic programming techniques. The first decomposition applies necessarily to the roots of the trees:

$$d(l(F), l'(F')) = \min \begin{cases} c_d(l) + d(F, l'(F')) \\ c_i(l') + d(l(F), F') \\ c_s(l, l') + d(F, F') \end{cases}$$

We now have to solve the problem for forests. Since a forest is a sequence, two directions of decomposition are allowed. This gives the following recursive relationships.

$$d(l(F) \circ T, l'(F') \circ T') = \min \begin{cases} c_d(l) + d(F \circ T, l'(F') \circ T') \\ c_i(l') + d(l(F) \circ T, F' \circ T') \\ d(l(F), l'(F')) + d(T, T') \end{cases} \tag{1}$$

or alternatively

$$d(T \circ l(F), T' \circ l'(F')) = \min \begin{cases} c_d(l) + d(T \circ F, T' \circ l'(F')) \\ c_i(l') + d(T \circ l(F), T' \circ F') \\ d(l(F), l'(F')) + d(T, T') \end{cases} \tag{2}$$

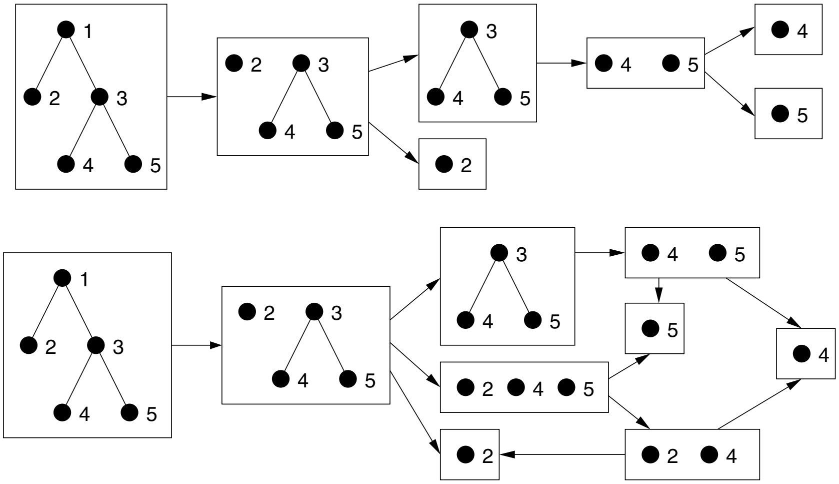


Fig. 1. These two graphics show two strategies of decomposition for a five-node tree: in the first case, all decompositions are left decompositions (according to Equation 1), and in the second case, all decompositions are right decompositions (according to Equation 2). This gives respectively 7 and 9 subforests.

We call decomposition according to Equation 1 a *left* decomposition and decomposition according to Equation 2 a *right* decomposition. In the sequel, we use the word *direction* to indicate that the decomposition is left or right. The difference with string edit distance is that the choice between 1 and 2 may lead to different numbers of recursive calls. Figure 1 shows such an example. We will see in this article that it is advantageous to alternate both schemes. An alternation of left and right decompositions gives rise to a *strategy.*

Definition 3 (Strategy). *Let F and G be two forests. A* strategy *is a mapping from $\mathcal{SF}(F) \times \mathcal{SF}(G)$ to $\{left, right\}$.*

Each strategy is associated with a specific set of recursive calls. We name *relevant forests* the forests that are involved in these recursive calls. This terminology is adapted from [2].

Definition 4 (Relevant Forests). *Let (F, F') be a pair of forests provided with a strategy ϕ. The set $\mathcal{RF}_\phi(F, F')$ of* relevant *forests is defined as the least subset of $\mathcal{SF}(F) \times \mathcal{SF}(F')$ such that if the decomposition of (F, F') meets the pair (G, G'), then (G, G') belongs to $\mathcal{RF}_\phi(F, F')$.*

Proposition 1. *Let (F, F') be a pair of forests provided with a strategy ϕ. The set $\mathcal{RF}_\phi(F, F')$ of* relevant forests *satisfies:*

- *if F and F' are empty forests, then $\mathcal{RF}_\phi(F, F') = \emptyset$,*
- *if $\phi(F, F') = left$, $F = l(G) \circ T$ and F' is empty, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(G \circ T, F')$,*
- *if $\phi(F, F') = right$, $F = T \circ l(G)$ and F' is empty, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(T \circ G, F')$,*
- *if $\phi(F, F') = left$, $F' = l'(G') \circ T'$ and F is empty, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(F, G' \circ T')$,*
- *if $\phi(F, F') = right$, $F' = T' \circ l'(G')$ and F is empty, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(F, T' \circ G')$,*
- *if $\phi(F, F') = left$, $F = l(G) \circ T$ and $F' = l'(G') \circ T'$, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(G \circ T, F') \cup \mathcal{RF}_\phi(F, G' \circ T') \cup \mathcal{RF}_\phi(l(G), l'(G')) \cup \mathcal{RF}_\phi(T, T')$,*
- *if $\phi(F, F') = right$, $F = T \circ l(G)$, and $F' = T' \circ l'(G')$, then $\mathcal{RF}_\phi(F, F')$ is $\{(F, F')\} \cup \mathcal{RF}_\phi(T \circ G, F') \cup \mathcal{RF}_\phi(F, T' \circ G') \cup \mathcal{RF}_\phi(l(G), l'(G')) \cup \mathcal{RF}_\phi(T, T')$.*

Proof. By induction on $|F| + |F'|$, using equations 1 and 2. □

We write $\mathcal{RF}_\phi(F)$ and $\mathcal{RF}_\phi(F')$ to denote the projection of $\mathcal{RF}_\phi(F, F')$ on $\mathcal{SF}(F)$ and $\mathcal{SF}(F')$ respectively. The index ϕ in $\mathcal{RF}_\phi$ will be omitted when it is clear from the context. Finally, we denote `#relevant` the number of relevant forests.

Lemma 1. *Given a tree $A = l(A_1 \circ \cdots \circ A_n)$, for any strategy we have*

$$\texttt{\#relevant}(A) \geq |A| - |A_i| + \texttt{\#relevant}(A_1) + \cdots + \texttt{\#relevant}(A_n)$$

where $i \in [1..n]$ is such that the size of A_i is maximal.

Proof. Let $F = A_1 \circ \cdots \circ A_n$. By Proposition 1, we have $\mathcal{RF}(A) = \{A\} \cup \mathcal{RF}(F)$. So it is sufficient to prove that $\texttt{\#relevant}(F) \geq |F| - |A_i| + \texttt{\#relevant}(A_1) + \cdots + \texttt{\#relevant}(A_n)$. The proof is by induction on n. If $n = 1$, then the result is direct. If $n > 1$, assume that a left operation is applied to F. Let l, F_1, T such that $A_1 = l(F_1)$ and $T = A_2 \circ \cdots \circ A_n$. We have $\mathcal{RF}(F) = \{F\} \cup \mathcal{RF}(A_1) \cup \mathcal{RF}(T) \cup \mathcal{RF}(F_1 \circ T)$. It is possible to prove by induction on $|F_1| + |T|$ that the number of relevant forests of $F_1 \circ T$ containing both nodes of F_1 and nodes of T is greater than $\min\{|F_1|, |T|\}$. Therefore $\texttt{\#relevant}(F) \geq 1 + \texttt{\#relevant}(A_1) + \texttt{\#relevant}(T) + \min\{|F_1|, |T|\}$. Let $j \in [2..n]$ such that A_j has the maximal size among $\{A_2, \ldots, A_n\}$. By induction hypothesis for T, $\texttt{\#relevant}(F)$ is greater than

$$1 + \texttt{\#relevant}(A_1) + \cdots + \texttt{\#relevant}(A_n) + |T| - |A_j| + \min\{|F_1|, |T|\}.$$

It remains to verify that

$$1 + |T| - |A_j| + \min\{|F_1|, |T|\} \geq |F| - |A_i|.$$

If $|F_1| \leq |T|$, then $1 + |T| + \min\{|F_1|, |T|\} = |F|$. Since $|A_j| \leq |A_i|$, it follows that $1 + |T| + \min\{|F_1|, |T|\} - |A_j| \geq |F| - |A_i|$. If $|T| < |F_1|$, then $i = 1$ and $|F| - |A_i| = |T|$, that implies $1 + |T| - |A_j| + \min\{|F_1|, |T|\} \geq |F| - |A_i|$. □

This Lemma yields a lower bound for the number of relevant forests.

Lemma 2. *For every natural number n, there exists a tree A of size n such that for any strategy, $\texttt{\#relevant}(A)$ has a lower bound in $O(n \log(n))$.*

Proof. For each complete balanced binary tree T_n of size n, we prove by induction on n that $\texttt{\#relevant}(T_n) \geq (n+1)\log_2(n+1)/2$ using Lemma 1. □

This gives a $O(n^2 \log^2(n)$ bound for the total number of pairs of relevant forests.

3 Cover Strategies

In this section, we define a main family of strategies that we call *cover strategies*. The idea comes from the following remark. Assume that $l(F) \circ T$ is a relevant forest for a given strategy. Suppose that the direction of $l(F) \circ T$ is left. This decomposition generates three relevant forests: $l(F)$, T and $F \circ T$. The forest T is a subforest of $F \circ T$. An opportune point of view is then to eliminate in priority nodes of F in $F \circ T$, so that $F \circ T$ and T share relevant forests as most as possible. We make this intuitive property more formal by defining *covers*.

Definition 5 (Cover). *Let F be a forest. A* cover *r of F is a mapping from F to $F \cup \{right, left\}$ satisfying for each node i in F*

- *if $deg(i) = 0$ or $deg(i) = 1$, then $r(i) \in \{right, left\}$,*
- *if $deg(i) > 1$, then $r(i)$ is a child of i.*

In the first case, $r(i)$ is called the direction *of i, and in the latter case, $r(i)$ is called the* favorite child *of i.*

Definition 6 (Cover Strategy). *Given a pair of trees (A, B) and a cover r for A, we associate a unique strategy ϕ as follows.*

- *if $deg(i) = 0$ or $deg(i) = 1$, then $\phi(A(i), G) = r(i)$, for each forest G of B.*
- *if $A(i)$ is of the form $l(A_1 \circ \ldots \circ A_n)$ with $n > 1$, then let $p \in \{1, \ldots, n\}$ such that the favorite child $r(i)$ is the root of A_p. For each forest G of B, we define*

$$\begin{aligned} \phi(A(i), G) &= \textit{right whenever } p = 1, \textit{ left otherwise,} \\ \phi(T \circ A_p \circ \cdots \circ A_n, G) &= \textit{left, for each forest } T \textit{ of } A_1 \circ \cdots \circ A_{p-1}, \\ \phi(A_p \circ T, G) &= \textit{right, for each forest } T \textit{ of } A_{p+1} \circ \cdots \circ A_n. \end{aligned}$$

The tree A is called the cover tree*. A strategy is a* cover strategy *if there exists a cover associated to it.*

The family of cover strategies includes Zhang-Shasha and Klein algorithms. The Zhang-Shasha algorithm is the decomposition strategy uniquely based on Equation 1. It corresponds to the cover strategy

- for any node of degree 0 or 1, the direction is *left*,
- for any other node, the favorite child is the rightmost child.

Klein algorithm may be described as the cover strategy

- for any node of degree 0 or 1, the direction is *left*,
- for any other node, the favorite child is the root of the heaviest subtree.

The aim is now to study the number of relevant forests for a cover strategy. This task is divided into two steps. First, we compute the number of strategy for the cover tree alone. This intermediate result will indirectly contribute to our final objective.

Lemma 3. *Let F and G be two forests. For any strategy ϕ, for any nodes i of F and j of G, $(F(i), G(j))$ is an element of $\mathcal{RF}_\phi(F, G)$.*

Proof. By induction on the depth of i in F and of j in G. □

Lemma 4. *Let F be a forest, T a non empty forest and l a labeled node.*

- *Let ϕ be a cover strategy for $l(F) \circ T$ such that $\phi(l(F) \circ T) = left$, let $k = |F|$. We write $F_1, \ldots, F_k$ for denoting the k subforests of F corresponding to the successive left decompositions of F: F_1 is F, and each F_{i+1} is obtained from F_i by a left deletion.*

$$\mathcal{RF}(l(F) \circ T) = \{l(F) \circ T, F_1 \circ T, \ldots, F_k \circ T\} \cup \mathcal{RF}(l(F)) \cup \mathcal{RF}(T).$$

- *Let ϕ be a cover strategy for $T \circ l(F)$ such that $\phi(T \circ l(F)) =$ right, let $k = |F|$. We write $G_1, \dots, G_k$ for denoting the k subforests of F corresponding to the successive right decompositions of F: G_1 is F, and each G_{i+1} is obtained from G_i by a right deletion.*

$$\mathcal{RF}(T \circ l(F)) = \{T \circ l(F), T \circ G_1, \dots, T \circ G_k\} \cup \mathcal{RF}(l(F)) \cup \mathcal{RF}(T).$$

Proof. We show the first claim, the proof of the other one being symmetrical. By Proposition 1,

$$\mathcal{RF}(l(F) \circ T) = \{l(F) \circ T\} \cup \mathcal{RF}(l(F)) \cup \mathcal{RF}(T) \cup \mathcal{RF}(F \circ T).$$

Let's have a closer look at $F \circ T$. We establish that

$$\mathcal{RF}(F \circ T) = \{F_1 \circ T\} \cup \dots \cup \{F_k \circ t\} \cup_{i \in F} \mathcal{RF}(F(i)) \cup \mathcal{RF}(T)$$

The proof is by recurrence on k. If $k = 1$, then $F = F_1$ and $\mathcal{RF}(F \circ T) = \{F_1 \circ T\} \cup \mathcal{RF}(T)$. If $k > 1$, let l', F' and T' such that $F = l'(F') \circ T'$. We have

$$\mathcal{RF}(F \circ T) = \{F \circ T\} \cup \mathcal{RF}(l'(F')) \cup \mathcal{RF}(F \circ T' \circ T) \cup \mathcal{RF}(T' \circ T).$$

On the other hand $F = F_1$, $F' \circ T' = F_2$ and $T' = F_{|l'(F')|+1}$. Since ϕ is a cover strategy, the direction for $F \circ T$ and for successive subforests containing T is left. We apply the induction hypothesis to T' and $F' \circ T'$ to get the expected result.

We come back to $\mathcal{RF}(l(F) \circ T)$. For $\mathcal{RF}(l(F) \circ T)$, we finally get

$$\{l(F) \circ T\} \cup \{F_1 \circ T\} \cup \dots \cup \{F_k \circ T\} \cup_{i \in F} \mathcal{RF}(F(i)) \cup \mathcal{RF}(l(F)) \cup \mathcal{RF}(T).$$

According to Lemma 3, for each node i in F, $\mathcal{RF}(Fi))$ is included in $\mathcal{RF}(l(F))$. It follows that

$$\mathcal{RF}(l(F) \circ T) = \{l(F) \circ T\} \cup \{F_1 \circ T\} \cup \dots \cup \{F_k \circ T\} \cup \mathcal{RF}(l(F)) \cup \mathcal{RF}(T).$$

□

The first consequence of this Lemma is that cover strategies can reach the $n \log(n)$ lower bound of Lemma 2. We give a criterion for that. Note that this criterion is fulfilled by Klein strategy.

Lemma 5. *Let F be a forest provided with a cover strategy ϕ, such that for each relevant forest $A \circ T \circ B$ (A and B are trees, T is a forest)*

- *if $|B| > |A \circ T|$, then $\phi(A \circ T \circ B) =$ left,*
- *if $|A| > |T \circ B|$, then $\phi(A \circ T \circ B) =$ right,*

then `#relevant`$(F) \leq |F| \log_2(|F| + 1)$.

Proof. The proof is by induction on $|F|$. If F is a tree, the result is direct. Otherwise assume $\phi(F) =$ left. Let $F = A \circ T \circ B$, $n = |A|$ and $m = |T \circ B|$. By Lemma 4, `#relevant`$(F) = n +$ `#relevant`$(A) +$ `#relevant`$(T \circ B)$. By induction hypothesis for A and $T \circ B$, it follows that `#relevant`$(F) \leq n + n \log_2(n+1) + m \log_2(m+1)$. Since $\phi(F) =$ left, $n \leq m$ and so `#relevant`$(F) \leq (n + m) \log_2(n + m + 1)$. □

Another application of Lemma 4 is that it is possible to know the exact number of relevant forests for the cover tree.

Lemma 6. *Let $A = l(A_1 \circ \ldots \circ A_n)$ be a cover tree such that $n = 1$ or the root of A_j is the favorite child.*

$$\texttt{\#relevant}(A) = |A| - |A_j| + \texttt{\#relevant}(A_1) + \cdots + \texttt{\#relevant}(A_n).$$

Proof. It is sufficient to establish that $\mathcal{RF}(A)$ equals

$$\{A\} \cup \{F_1 \circ A_j \circ \cdots \circ A_n, \ldots, F_k \circ A_j \circ \cdots \circ A_n\} \cup \{G_1, \ldots, G_h\} \cup_i \mathcal{RF}(A_i)$$

where k is the size of $A_1 \circ \cdots \circ A_{j-1}$, h is the size of $A_{j+1} \circ \cdots \circ A_n$, F_1 is $A_1 \circ \cdots \circ A_{j-1}$, and each F_{i+1} is obtained from F_i by a left deletion, G_1 is $A_{j+1} \circ \cdots \circ A_n$ and each G_{i+1} is obtained from G_i by a right deletion. The proof is by induction on the size of A. If $|A| = 1$, then $\mathcal{RF}(A)$ is $\{A\}$. If $|A| > 1$, by Proposition 1, $\mathcal{RF}(A)$ is $\{A\} \cup \mathcal{RF}(A_1 \circ \cdots \circ A_n)$. Repeated applications of Lemma 4 yield the expected result. □

As a corollary of Lemma 6 and Lemma 1, we know that Klein algorithm is a strategy that minimizes the number of relevant forests for the cover tree, and Lemma 5 implies that the number of relevant forests for the cover tree is in $n \log(n)$.

After the study of the number of relevant forests for the cover tree, we are now able to look for the total number of relevant forests for a pair of trees. Given a pair of trees (A, B) provided with a cover for A, it appears that all relevant forests of A fall within three categories:

(α) those that are compared with all *rightmost* forests of B,
(β) those that are compared with all *leftmost* forests of B,
(δ) those that are compared with all *special* forests of B.

Definition 7 (Special, Rightmost and Leftmost Forests). *Let A be a tree.*

- *The set of* special forests *of A is the least set containing all relevant forests of A: F is a special forest of A if there exists a strategy ϕ such that $F \in \mathcal{RF}_\phi(A)$;*
- *the set of* rightmost forests *is the set of relevant forests wrt the strategy Left (Zhang-Shasha);*
- *the set of* leftmost forests *is the set of relevant forests wrt the strategy Right.*

We write $\texttt{\#special}(A)$, $\texttt{\#right}(A)$ *and* $\texttt{\#left}(A)$ *for the number of special, rightmost and leftmost forests of A.*

The task is to assign the proper category (α), (β) or (δ) to each relevant forest of A. We have this immediate result.

Lemma 7. *Let A be a cover tree, and let F be a relevant forest of A.*

- *if the direction of F is right, then F is at least compared with all leftmost forests of B,*
- *if the direction of F is left, then F is at least compared with all rightmost forests of B.*

Proof. By induction on $|A| - |F|$. □

For subtrees whose roots are *free nodes*, the category is entirely determined by the direction.

Definition 8 (Free Node). *Let A be a cover tree. A node i is* free *if i is the root of A, or if its parent is of degree greater than one and i is not the favorite child.*

Lemma 8. *Let i be a free node of A.*

1. *if the direction of i is left, then $A(i)$ is (α),*
2. *if the direction of i is right, then $A(i)$ is (β).*

Proof. We establish the first claim. Assume there exists a free node i with direction left, such that $A(i)$ is not (α). Applying Lemma 7, it means that $A(i)$ is compared with a forest that is not a rightmost forest. It is then possible to consider G, the largest forest of B such that $(A(i), G)$ belongs to $\mathcal{RF}(A, B)$ and G is not a rightmost forest. G is not the whole tree B, since B is a particular rightmost forest. So there are four possibilities for the generation of $(A(i), G)$.

– If $(A(i), G)$ is generated by an insertion: since the direction of i is *left*, there exists a node l and two forests H and P such that $G = H \circ P$ and $(A(i), l(H) \circ P)$ is in $\mathcal{RF}(A, B)$. Since the size of $l(H) \circ P$ is greater than the size of G, $l(H) \circ P$ is a rightmost forest, that implies that $G = H \circ P$ is a also rightmost forest.

– If $(A(i), G)$ is generated by a deletion: there exists a node l such that either $l \circ A(i)$, $A(i) \circ l$ or $l(A(i))$ is a relevant forest. In the two first cases, this would imply that i is a favorite child, and in the third case, that the degree of the parent of i is 1. In all cases, this contradicts the hypothesis that i is a free node.

– If $(A(i), G)$ is generated by a substitution, being the matching part of the substitution: this would imply that G is a tree, that contradicts the hypothesis that G is not a rightmost forest.

– If $(A(i), G)$ is generated by a substitution, being the remaining part of the substitution: $(A(i), G)$ should be obtained from a relevant pair of the form $(A' \circ A(i), B' \circ G)$ or $(A(i) \circ A', G \circ B')$, where A' and B' are subtrees of A and B respectively. In both cases, this contradicts the hypothesis that i is not a favorite child. □

For nodes that are not free and for forests, the situation is more complex. It is then necessary to take into account the category of the parent too. The following lemma establishes that those nodes inherit the relevant forests of B from their parents.

Lemma 9. *Let F be a relevant forest of A that is not a tree. Let i be the lower common ancestor of the set of nodes of F and let j be the favorite child of i.*

1. *if F is a rightmost forest whose leftmost tree is not $A(j)$, then F has the same category as $A(i)$,*
2. *if F is a leftmost forest, then F has the same category as $A(i)$,*
3. *otherwise F is (δ).*

Proof. The proof of the two first cases is similar to the proof of Lemma 8. We give a detailed proof of the third case. By definition of cover strategies, the first tree of F is $A(j)$ and the direction of F is right. Assume there is a special forest G of B such that (F, G) is not a relevant pair. We suppose that G is of maximal size.

– If G is a rightmost forest. Since F is not a leftmost forest, j is not the leftmost child of i, that implies that the direction of $A(i)$ is *left*. Lemma 7 implies that $(A(i), G)$ is a relevant pair. The pair (F, G) is obtained from $(A(i), G)$ by successive left deletions until j and right deletion until F.

– If G is not a rightmost forest. There exists a node l such that $G \circ l$ is a special forest. By construction of G, $(F, G \circ l)$ is a relevant pair. Since the direction of F is right, a right deletion gives (F, G). □

Lemma 10. *Let A be a cover tree, i be a node of A that is not free, and j be the parent of i.*

- *if the direction of i is* left*, if i is the rightmost child of j and $A(j)$ is (α), then $A(i)$ is (α),*
- *if the direction of i is* right*, if i is the leftmost child of j and $A(j)$ is (β), then $A(i)$ is (β),*
- *otherwise $A(i)$ is (δ).*

Proof. Similar to proof of Lemma 9. □

We introduce notations for classifying nodes of A according to their inheritance.

Notation 2 *Let i be a node of A, let j be the parent of i (if i is not the root).*

$$\begin{aligned}
\mathit{Free}(A(i)) &: \textit{cardinality of } \mathcal{RF}(A,B) \cap (A(i), B) \textit{ if } i \textit{ is free}\\
\mathit{Right}(A(i)) &: \textit{cardinality of } \mathcal{RF}(A,B) \cap (A(i), B) \textit{ if } A(j) \textit{ is } (\alpha)\\
\mathit{Left}(A(i)) &: \textit{cardinality of } \mathcal{RF}(A,B) \cap (A(i), B) \textit{ if } A(j) \textit{ is } (\beta)\\
\mathit{All}(A(i)) &: \textit{cardinality of } \mathcal{RF}(A,B) \cap (A(i), B) \textit{ if } A(j) \textit{ is } (\delta).
\end{aligned}$$

With this notation, `#relevant`(A, B) equals *Free*(A). We are now able to formulate the main result of this section, which gives the total number of relevant forests for a cover strategy.

Theorem 1. *Let (A, B) be a pair of trees, A being a cover tree.*

1. If A is reduced to a single node whose direction is right

$$\begin{aligned} \mathit{Free}(A) = \mathit{Left}(A) &= \texttt{\#left}(B) \\ \mathit{All}(A) = \mathit{Right}(A) &= \texttt{\#special}(B) \end{aligned}$$

2. If A is reduced to a single node whose direction is left

$$\begin{aligned} \mathit{Free}(A) = \mathit{Right}(A) &= \texttt{\#right}(B) \\ \mathit{All}(A) = \mathit{Left}(A) &= \texttt{\#special}(B) \end{aligned}$$

3. If $A = l(A')$ and the direction of l is right

$$\begin{aligned} \mathit{Free}(A) = \mathit{Left}(A) &= \texttt{\#left}(B) + \mathit{Right}(A') \\ \mathit{All}(A) = \mathit{Right}(A) &= \texttt{\#special}(B) + \mathit{All}(A') \end{aligned}$$

4. If $A = l(A')$ and the direction of l is left

$$\begin{aligned} \mathit{Free}(A) = \mathit{Right}(A) &= \texttt{\#right}(B) + \mathit{Left}(A') \\ \mathit{All}(A) = \mathit{Left}(A) &= \texttt{\#special}(B) + \mathit{All}(A') \end{aligned}$$

5. If $A = l(A_1 \circ \cdots \circ A_n)$ and the favorite child is the leftmost child

$$\begin{aligned} \mathit{Free}(A) = \mathit{Left}(A) &= \textstyle\sum_{i>1} \mathit{Free}(A_i) + \mathit{Left}(A_1) + \texttt{\#left}(B)(|A| - |A_1|) \\ \mathit{All}(A) = \mathit{Right}(A) &= \textstyle\sum_{i>1} \mathit{Free}(A_i) + \mathit{All}(A_1) + \texttt{\#special}(B)(|A| - |A_1|) \end{aligned}$$

6. If $A = l(A_1 \circ \cdots \circ A_n)$ and the favorite child is the rightmost child

$$\begin{aligned} \mathit{Free}(A) = \mathit{Right}(A) &= \textstyle\sum_{i<n} \mathit{Free}(A_i) + \mathit{Right}(A_n) + \texttt{\#right}(B)(|A| - |A_n|) \\ \mathit{All}(A) = \mathit{Left}(A) &= \textstyle\sum_{i<n} \mathit{Free}(A_i) + \mathit{All}(A_n) + \texttt{\#special}(B)(|A| - |A_n|) \end{aligned}$$

7. If $A = l(A_1 \circ \cdots \circ A_n)$ and the favorite child is A_j with $1 < j < n$

$$\begin{aligned} \mathit{Free}(A) &= \textstyle\sum_{i \neq j} \mathit{Free}(A_i) + \mathit{All}(A_j) + \texttt{\#right}(B)(1 + |A_1 \circ \cdots \circ A_{j-1}|) \\ &\quad + \texttt{\#special}(B)|A_j \circ \cdots \circ A_n| \\ \mathit{Right}(A) &= \mathit{Free}(A) \\ \mathit{All}(A) &= \mathit{Left}(A) = \textstyle\sum_{i \neq j} \mathit{Free}(A_i) + \mathit{All}(A_j) + \texttt{\#special}(B)(|A| - |A_j|) \end{aligned}$$

Proof. By induction on the size of A, using Lemmas 6, 9, 8 and 10. □

Lemma 11. *For Zhang-Shasha algorithm,*

$$\texttt{\#relevant}(A, B) = \texttt{\#right}(A) * \texttt{\#right}(B).$$

Proof. In Theorem 1, it appears that the cases *2*, *4* and *6* are the only useful cases. Applying Lemma 6, it follows that $\texttt{\#relevant}(A, B) = \texttt{\#right}(A) * \texttt{\#right}(B)$ (by induction on the size of A). □

We get the symmetrical result for the symmetrical strategy: for nodes of degree 0 or 1, the direction is *right*, and for other nodes, the favorite child is the leftmost child. In this case, $\texttt{\#relevant}(A, B) = \texttt{\#left}(A) * \texttt{\#left}(B)$.

For Theorem 1 to be effective, it remains to evaluate the values of `#right`, `#left` and `#special`.

Lemma 12. *Let A be a tree.*

$$\begin{aligned}\texttt{\#right}(A) &= \sum(|A(i)|, i \in A) - \sum(|A(j)|, j \textit{ is a rightmost child})\\ \texttt{\#left}(A) &= \sum(|A(i)|, i \in A) - \sum(|A(j)|, j \textit{ is a leftmost child})\end{aligned}$$

Proof. Lemma 6 gives

$$\begin{aligned}\texttt{\#right}(A) = \texttt{\#left}(A) &= 1, \text{if the size of } A \text{ equals } 1\\ \texttt{\#right}(l(A_1, \dots, A_n)) &= \textstyle\sum_i \texttt{\#right}(A_i) + |A| - |A_n|, \text{otherwise.}\\ \texttt{\#left}(l(A_1, \dots, A_n)) &= \textstyle\sum_i \texttt{\#left}(A_i) + |A| - |A_1|, \text{otherwise.}\end{aligned}$$

Induction on the size of A concludes the proof. □

Lemma 13. *Let F be a forest.* $\texttt{\#special}(F) = \frac{|F|(|F|+3)}{2} - \sum_{i \in F} |F(i)|$.

Proof. The proof is by induction on the size of F. If $|F| = 0$, then $\texttt{\#special}(F) = 0$, which is consistent with the Lemma. If $F > 0$, then $F = l(T) \circ P$, where P and T are (possibly empty) subforest of F. There are two kinds of special subforests of F to be considered:

1. those containing the node l: there are $|P| + 1$ such subforests,
2. those not containing the node l: there are $\texttt{\#special}(T \circ P)$ such subforests.

It follows that $\texttt{\#special}(F) = |P| + 1 + \texttt{\#special}(T \circ P)$. The induction hypothesis applied to $T \circ P$, whose size is $|F| - 1$, ensures.

$$\begin{aligned}\texttt{\#special}(F) &= |P| + 1 + \tfrac{(n-1)(n+2)}{2} - \textstyle\sum_{i \in T \circ P} |t \circ P(i)|\\ &= n - |l(t)| + 1 + \tfrac{(n-1)(n+2)}{2} - \textstyle\sum_{i \in T \circ p} |T \circ P(i)|\\ &= n + 1 + \tfrac{(n-1)(n+2)}{2} - \textstyle\sum_{i \in F} |F(i)|\\ &= \tfrac{n(n+3)}{2} - \textstyle\sum_{i \in F} |F(i)|\end{aligned}$$

This concludes the proof. □

4 A New Optimal Cover Strategy

In this last section, we show that Theorem 1 makes it possible to design an optimal cover strategy. An optimal cover strategy is of course a strategy that minimizes the total number of relevant forests. The algorithm is as follows. For any pair of trees (A, B), define four dynamic programming structures Right, Left, Free, and All of size $|A|$. The definition of Right, Left, Free, and All is directly borrowed from Theorem 1. The only difference is that the favorite child is not known. So at each step, the favorite child is chosen to be the child that minimizes the number of relevant forests. For instance, if $A = l(A_1 \circ \cdots \circ A_n)$ then

$$\mathit{Free}(A) = \sum_{i \geq 1} \mathit{Free}(A_i) + \min \begin{cases} \mathit{Left}(A_1) - \mathit{Free}(A_1) + \texttt{\#left}(B) * (|A| - |A_1|)\\ \mathit{All}(A_j) - \mathit{Free}(A_j) + \texttt{\#special}(B) |A_j \circ \cdots \circ A_n|\\ + \texttt{\#right}(B)(1 + |A_1 \circ \cdots \circ A_{j-1}|), \quad 1 < j < n\\ \mathit{Right}(A_n) - \mathit{Free}(A_n) + \texttt{\#right}(B)(|A| - |A_n|)\end{cases}$$

The optimal cover is then built up using tracing back from *Free*(A). The size of each table is $|A|$. As for the time complexity, the time computation for a cell of a node i is proportional to the degree of i. So the overal time computation is in $O(\sum_{i \in A} deg(i))$, that is in $O(|A|)$. Figure 2 shows an example of optimal cover.

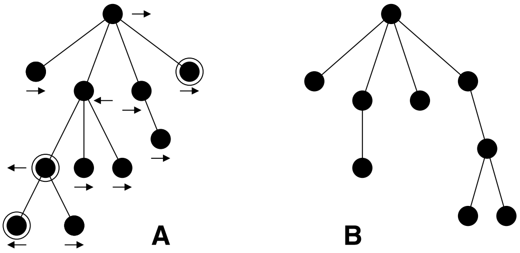


Fig. 2. This figure shows an optimal cover for A built up for the pair (A, B). For each node, the direction, left or right, is indicated with an arrow. For each node of arity greater than 1, the favorite child is the circled node. This cover yields 340 relevant forests. For the two Zhang-Shasha strategies, we get 405 and 350 relevant forests, and for the Klein strategy, we get 391 relevant forests.

By construction, this algorithm involves less relevant forests than Zhang-Shasha and Klein strategies. In particular it is in $n^3 \log(n)$. But we do not claim that this is the best algorithm, since we do not take into account the cost of the preprocessing in comparison with the expected gain of number of relevant forests. This is still an open question.

References

1. S. Chawathe, “Comparing hierarchical data in external memory” *Proceedings of the Twenty-fifth International Conference on Very Large Data Bases* (1999), Edinburgh, Scotland, p. 90–101.
2. P. Klein, “Computing the edit-distance between unrooted ordered trees” *Proceeding of 6th European Symposium on Algorithms* (1998), p. 91–102.
3. B. Shapiro and K. Zhang, “Comparing multiple RNA secondary structures using tree comparisons”, *Comput. Appl. Biosciences*, Vol.4, 3 (1988), p. 387–393.
4. K. Zhang and D. Shasha, “Simple fast algorithms for the editing distance between trees and related problems” *SIAM Journal of Computing*, Vol 18-6, (1989), p. 1245–1262.