An Unambiguous And Complete Dynamic Programming Algorithm For Tree Alignment

Cedric Chauve, Julien Courtiel, Yann Ponty

An Unambiguous And Complete Dynamic Programming Algorithm For Tree Alignment

Cedric Chauve[1], Julien Courtiel[1,3], and Yann Ponty[2,3]

[1] Department of Mathematics, Simon Fraser University
[2] CNRS-LIX, Ecole Polytechnique
[3] Pacific Institute for the Mathematical Sciences

Abstract. Pairwise ordered tree alignment is an important pattern matching problem, motivated by RNA secondary structure comparison for example, that can be solved efficiently using Dynamic Programming (DP). An inquiry into the multiplicity of optimal solutions, as well as the existence of potentially interesting sub-optimal solutions, naturally motivates the question of exploring the space of all, optimal and sub-optimal, tree alignments. There are well-known DP-based techniques that allow for such an exploration, but they require completeness and unambiguity of the DP scheme, i.e. the existence of a score-preserving bijection between the search space and the set of possible derivations of the DP scheme. In this paper, we present the first unambiguous and complete dynamic programming algorithm for the alignment of a pair of ordered rooted trees. Our algorithm optimally aligns two trees of size n_1 and n_2 in $\Theta(n_1\, n_2\, \max(n_1, n_2)^2)$ time in the worst-case scenario. Assuming uniformly-drawn random trees as input, it has average-case time and space complexities in $\Theta(n_1\, n_2)$.

1 Introduction

Comparing a pair of ordered labeled rooted trees[4] is a classical pattern matching problem, motivated among others by the comparison of RNA secondary structures (see [1, 2] for recent surveys on RNA structure comparison). The comparison of trees was introduced in an edit distance framework by Tai in [3] and has been widely studied since then. Currently, the best algorithm has a cubic worst-case time complexity [4]. However, unlike for sequences, tree alignment differs from tree edit distance: an optimal sequence of edit operations (substitutions, insertions, deletions) does not always induce an optimal alignment of the considered trees, and conversely (see [5] for a fundamental work that clarifies the link between edit distance and alignment). The first tree pairwise alignment algorithm, a Dynamic Programming (DP) algorithm, was introduced by Jiang *et al.* in [6]; its worst-case time complexity is quartic, but its average-case time complexity is only quadratic [7]. It has since then been extended in several ways (see [8] for a recent reference), but the time complexity has not been improved.

[4] From now on, unless otherwise specified, tree means ordered labeled rooted tree.

A fundamental, although often overlooked, issue with optimization algorithms lies in the multiplicity of – potentially diverse – optimal solutions, one of which is typically singled out and returned by the algorithm using some – usually arbitrary – criteria. Within a predictive setting, i.e. when the solution of an optimization problem is supposed to inform on the past/present/future state of reality, a single returned solution may then be a poor representative of the set of – equally likely – optimal solutions, and lead to distorted conclusions. More generally, it may be relevant, in many applications, to consider and analyze the set of all solutions (i.e. the *solution space*), including optimal and suboptimal ones. Instances of this general methodology are based on an enumeration of all optimal solutions [9], the random sampling of solutions, possibly including suboptimal ones under a probability distribution influenced by their cost [10, 11] (e.g. Boltzmann distribution [12]), or even the direct computation of the probabilities associated with the presence/absence of certain features of the solutions [13, 12, 14, 11]. For problems that can be solved using DP, these goals are sometimes achieved through algebraic substitutions and symbolic transforms of the optimization DP scheme, for which multiple theoretical [15, 11] and practical frameworks [16–18] have been proposed.

However, in order to benefit from those frameworks, especially for applications which hinge on a probabilistic model, the initial DP scheme must respect some notions of *completeness* and *unambiguity* [19, 20, 11]. Intuitively, these properties require that each element of the solution space, characterized *a priori*, is explored *exactly once* by the DP scheme. They are derived from eponymous notions defined on formal grammars, and their automated verification is a notoriously undecidable problem [21, Theorem 9.20, p.405-406]. Consequently, the task of designing a DP scheme amenable to a probabilistic analysis, and the proof of its unambiguity/completeness, are mainly manual, technical and, sometimes, highly non-trivial. As we will show, existing DP schemes [6, 5] for tree alignment are typically complete but ambiguous, thus preventing the implementation of probabilistic alternatives for tree alignments. Such a transposition is motivated by applications in bioinformatics, where a probabilistic analysis of the solution space has been successfully applied for decades in sequence analysis [22] and appears promising in the context of RNA structural alignment [23]. However, despite recent progress described in Schirmer thesis [24], an unambiguous DP scheme for tree alignment has so far remained elusive.

The main contribution of the present paper is the first unambiguous DP scheme for the alignment of two trees. In Section 2, we introduce the necessary background. We then present our unambiguous tree alignment algorithm and analyze its complexity. We also use it to compute precise expressions for the total number of expected tree alignments between two trees of given size.

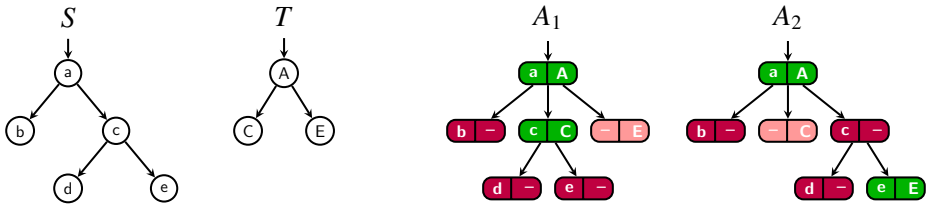

Fig. 1: Two alignments A_1 and A_2 of S and T which are optimal under a scoring scheme that maximizes the number of (x, X) matches. Note that (c, C) and (e, E) cannot be simultaneously matched without violating the ancestrality condition.

2 Background on tree alignment and DP schemes

Trees and forests. Let S and T be two rooted ordered trees, with labeled vertices; we denote by V_S and V_T the vertex sets of S and T. For two vertices x and y of a tree T, we write $x \prec y$ if y is an ancestor of x, and $x < y$ if x is visited before y in a preorder traversal of T. For a non-root vertex x of a tree T we denote by $p(x)$ its parent (closest ancestor). An *ordered forest* X is a sequence $(X_1, \ldots, X_k)$ of $k \geq 0$ trees. The number k of trees in X is also denoted by $|X|$. We write $a(Y)$ for a tree composed of a root a and subtrees forming an ordered forest Y. We denote by $X \circ Y$ the forest formed by the concatenation of two forests X and Y. Every non-empty ordered forest can be put under the form $X = a(Y) \circ X'$. Given a forest X and a tree $t \in X$ we denote by X/t the forest obtained by deleting t.

Tree alignment. Intuitively, an alignment between S and T is defined by a set of pairs of *matched vertices*, simply referred as *matches* in the following, consistent with the topologies of both S and T, in terms of ancestrality and order. We denote the set of all potential matches as $\Sigma_m = V_S \times V_T$ and, for any match $\sigma = (a, b) \in \Sigma_m$, we define shorthands $\sigma_S \equiv a$ and $\sigma_T \equiv b$.

Definition 1 (Tree alignment). An alignment of S and T is a set $A \subset \Sigma_m$ that satisfies the following properties:

1. For every $a \in V_S$, there exists at most one $\sigma \in A$ such that $\sigma_S = a$;
2. For every $b \in V_T$, there exists at most one $\sigma \in A$ such that $\sigma_T = b$;
3. [Ancestrality condition] For every $\sigma, \tau \in A$, $\sigma_S \prec \tau_S$ iff $\sigma_T \prec \tau_T$;
4. [Order condition] For every $\sigma, \tau \in A$, $\sigma_S < \tau_S$ iff $\sigma_T < \tau_T$.

Given an alignment A, any vertex of S (resp. T) that is not in a match is called a *insertion* (resp. *deletion*). To denote matches, insertions and deletions in a unified format, we extend Σ_m to $\Sigma = \Sigma_m \cup V_S \times \{-\} \cup \{-\} \times V_T$ and represent an insertion as $(a, -)$ and a deletion as $(-, b)$. Note that a set of matches uniquely defines an alignment, and implies a set of insertions and deletions. A subtree of S (resp. T) is said to be *matched* (resp. *inserted*, *deleted*) in an alignment A if

Initialization.

$$\mathsf{JS}\left[\varnothing, \varnothing\right] = 0 \tag{1}$$

Recursive steps.

$$\mathsf{JS}\left[a(u) \circ X, \varnothing\right] = \mathsf{Ins}\left(a\right) + \mathsf{JS}\left[u, \varnothing\right] + \mathsf{JS}\left[X, \varnothing\right] \tag{2}$$

$$\mathsf{JS}\left[\varnothing, b(v) \circ Y\right] = \mathsf{Del}\left(b\right) + \mathsf{JS}\left[\varnothing, v\right] + \mathsf{JS}\left[\varnothing, Y\right] \tag{3}$$

$$\mathsf{JS}\left[a(u) \circ X, b(v) \circ Y\right] = \min \begin{cases} \mathsf{Match}\left(a, b\right) + \mathsf{JS}\left[u, v\right] + \mathsf{JS}\left[X, Y\right] & \text{(4a)} \\ \min\limits_{Y' \circ Y'' = b(v) \circ Y} \mathsf{Ins}\left(a\right) + \mathsf{JS}\left[u, Y'\right] + \mathsf{JS}\left[X, Y''\right] & \text{(4b)} \\ \min\limits_{X' \circ X'' = a(u) \circ X} \mathsf{Del}\left(b\right) + \mathsf{JS}\left[X', v\right] + \mathsf{JS}\left[X'', Y\right] & \text{(4c)} \end{cases}$$

Fig. 2: A simple DP scheme for tree alignment [7, 5].

at least one (resp. all) of its vertices belong to a match (resp. insertion, deletion) in A (see Fig. 1 for an illustration). Given a *cost* function $c : \Sigma \to \mathbb{R}$ associating a bonus/penalty to (mis)match ($\mathsf{Match}\left(\star, \star\right)$), insertion ($\mathsf{Ins}\left(\star\right)$) and deletion ($\mathsf{Del}\left(\star\right)$) events, the total cost $c(A)$ of an alignment is defined as the sum of the costs associated to its events. An alignment is *optimal* for S and T if it has the lowest cost within the set of all alignments of S and T.

Remark 1. All definitions above can be naturally extended to the alignment of a pair of ordered forests if one transforms each forest into a single tree by grafting all its trees under a newly created root and assumes that both created roots of the two forests form a match.

The first DP algorithm for tree alignment was introduced, along with the problem itself, in a seminal paper by Jiang *et al* [6]. For the sake of simplicity, we focus in Fig. 2 on a simpler – equivalent in term of the explores search space – alternative [7, 5]. It relies on a single DP table, denoted JS, indexed by pairs (X, Y) of forests, where X (resp. Y) is a set of subtrees rooted at consecutive siblings in S (resp. T). The cell $\mathsf{JS}\left[X, Y\right]$ contains the minimal cost of aligning two input forests X and Y. This DP scheme considers three different cases: either the roots of the first trees are (mis)matched; the root of X_1 is inserted; or the root of Y_1 is deleted. This DP scheme largely resembles that of the classic Needleman and Wunsch algorithm for sequence alignment [25].

Unambiguity and completeness of a DP scheme. Given an instance I, an associated search space $\mathcal{S}_I$ and an objective function $f : \mathcal{S}_I \to \mathbb{R} \cup \{+\infty\}$, a classic combinatorial optimization problem consists in finding $s_I^\star := \mathrm{argmin}_{s \in \mathcal{S}_I} f(s)$. A DP algorithm solves such a problem by building on a *recurrence equation*, which expresses the optimal value $f^\star(I) := \min_{s \in \mathcal{S}_I} f(s)$ as a function of that a list of subproblems $f^\star(I_1), f^\star(I_2), \ldots$ occurring on the right-hand side terms of the equation. Once computed, e.g. efficiently using *memoization*, an optimal solution is reconstructed from $f^\star(I)$ using a *backtracking* procedure which determines which, out of a list of alternatives (right-hand side terms) in the

recurrence, contributes to the optimal value. The algorithm then proceeds recursively on associated subproblems, until the backtrack is fully developed, i.e. no further (non-trivial) subproblem is found to contribute.

A DP algorithm may be abstracted from its initial optimization goal, and the DP recurrence be reinterpreted as a – possibly partial and/or redundant – decomposition/generation strategy for the search space. One then defines the *derivations* $\mathcal{D}_I$ of a DP scheme as the *syntactical structures* describing the sequences/trees of local alternatives chosen during a fully developped backtrack. They include elements associated with suboptimal scores, but exclude elements having prohibitive values (e.g. $+\infty$ in a minimization setting) for the objective function. The notion of derivation can be further formalized, e.g. as parse trees of context-free grammars to [19, 20], or hyperpaths in directed hypergraphs [11]. Finally, a function $\Phi : \mathcal{D}_I \to \mathcal{S}_I$ associates a *semantics* $\Phi(d)$ – an element of the search space – to each derivation d in a problem-dependent manner.

A DP scheme is said to be *unambiguous* if and only if any pair of distinct derivations represents different elements of the search space, i.e. the mapping Φ is injective. It is *complete* if and only if every element of the search space is represented by at least one derivation, i.e. the mapping Φ is surjective. Unambiguous and complete DP algorithms are easily amenable to diverse extensions using trivial algebraic substitutions. For instance, one easily transforms such an optimization algorithm into an algorithm for enumerating the search space, by replacing the minimization algebra $(\min, +, \infty, 0)$ with a counting algebra $(+, \times, 0, 1)$. In the context of tree alignment, the *instance* is the pair of input trees S and T. The *search space* is the set of valid alignments of S and T, i.e. any sets of matches that are consistent with Definition 1. As illustrated by Fig. 3, a *derivation* is a tree whose nodes are labeled by cells in the DP table, and whose parent/children relationship correspond to left-hand-side/right-hand-side transitions in the DP recurrence. The semantics function Φ traverses the tree, and returns the set of matches induced by LHS/RHS transitions – associated with

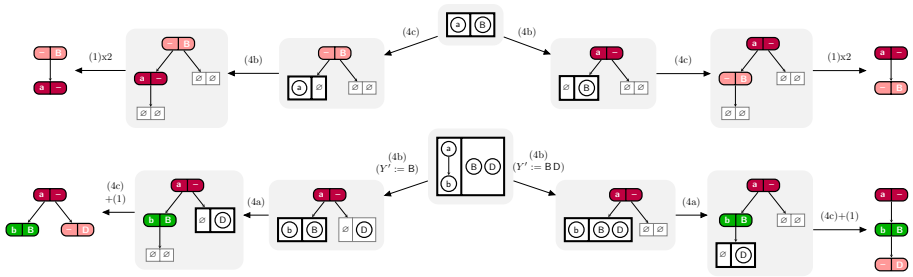

Fig. 3: Two types of ambiguity. Top: An empty alignment is obtained in multiple ways by arbitrarily interleaving insertions and deletions. Bottom: Two different partitions a forest of consecutive siblings may lead to the same final alignment.

occurrences of case (4a) in Fig. 2 and case (11b) in Fig. 5) – thus defining an alignment of S and T.

Property 1. The DP scheme defined in Fig. 2 is complete and ambiguous.

Proof (Sketch of). The proof of the above claim is straightforward and, due to space constraints, we only substantiate our ambiguity claim. As illustrated in Fig. 3, there are two main sources of ambiguity. First, if an alignment between two forests X and Y contains no match, then the insertions and deletions created by Equations (2), (3), (4b) and (4c) may appear in any order, while representing the same (empty) alignment. This phenomenon is similar to a well-known cause for ambiguity in established DP algorithms for sequence alignment [25]. A second cause of ambiguity, which is specific to tree alignments, arises from the operation of partitioning a forest into two subforests, occurring in Equations (4b) and (4c). As shown in Fig. 3, using Equation (4b), all the executions obtained from the recursive call in which the last tree t of Y' is not matched (i.e. it is deleted) can also be obtained from an alternative decomposition (i.e. recursive call) of $Y = Y_1 \circ Y_2$ in which t is the first tree of the second subforest Y_2. □

The two types of ambiguity can be combined repeatedly within a given derivation, leading to the generation of certain alignments in many different ways. This behavior is in fact typical, and we show in Sec. 4 that, on average, each alignment between two trees of size n_1 and n_2 is explored by $\Theta(1.412^{n_1+n_2})$ derivations in the DP scheme of Fig. 2.

3 An unambiguous dynamic programming scheme

We propose a new, unambiguous and complete, DP scheme for the alignment of two trees S and T. In comparison with the DP scheme of Fig. 2, our proposal is relatively complex, and requires several DP tables which will also be indexed by pairs (X, Y) of forests from S and T. Similarly to our presentation of the DP scheme defined by equations (1) to (4c), we focus only on the decomposition strategy of the search space, and defer the discussion of the nature of the indexing pairs of forests (X, Y) to Section 4.

In order to control the ambiguity of the DP scheme, we rely on three families of tables, called respectively tables of type V, VH and H. Each family of tables is characterized by invariants that describe the set of derivations contained in the cells. These invariants will be crucial to prove that the proposed DP scheme is indeed unambiguous and complete.

The DP tables of the family V are indexed by pairs of subtrees (i.e. subforests of size 1), and explore derivations where the two indexing trees end up being matched (i.e. cannot be fully inserted or deleted).

Specification 1 (V tables) Let $a(u)$ and $b(v)$ be subtrees of S and T.

a. Derivations initiating in $\mathsf{V}\big[a(u), b(v)\big]^{\beta}$, with $\beta \in \{\varnothing, \uparrow\}$, correspond to alignments of $a(u)$ and $b(v)$ such that $a(u)$ and $b(v)$ are *matched*.
b. Moreover, if $\beta = \uparrow$, then the vertex a forms a *match* (a, c) with some vertex c from $b(v)$.

The VH table addresses the more general case where the first indexing forest is not necessarily reduced to a single tree, and the root of the second forest is required to be matched.

Specification 2 (VH table) Let X be a forest of S and $b(v)$ be a subtree of T. Derivations initiating from $\mathsf{VH}\,[X, b(v)]$ correspond to alignments of X and $b(v)$ where $b(v)$ is *matched*.

The last family, H tables, corresponds to the most general case, where both indices are forests. The two important points related to H tables concern (1) the introduction of a "deletion" mode ($\alpha = \mathsf{D}$) that prevents the uncontrolled alternation of insertions and deletions, by preventing insertions to occur directly after deletions; and (2) the introduction of constraints on the partition of indexing forests, ensuring their uniquenesses.

Specification 3 (H tables) Let X (resp. Y) be a forest of S (resp. T) and (α, M, M') be a triplet from $\{\mathsf{I}|\mathsf{D}, \mathsf{D}\} \times \{\varnothing, \leftrightarrow, \rightarrow\}^2$. Derivations in $\mathsf{H}\big[X, Y\big]^{\alpha}_{M,M'}$ correspond to alignments of X and Y such that:

a. the first tree in X *is not inserted* if $\alpha = \mathsf{D}$;
b. the first and last (resp. only the last) tree(s) in X *are matched* if $M = \leftrightarrow$ (resp. $M = \rightarrow$);
c. the first and last (resp. only the last) tree(s) in Y *are matched* if $M' = \leftrightarrow$ (resp. $M' = \rightarrow$).

In particular, if $(M, M') \neq (\varnothing, \varnothing)$, any generated derivation contains a match.

Fig. 4 presents our DP scheme, which is illustrated in Fig. 5. The propositions below state the important properties of our DP scheme. Proofs are provided in appendix.

Proposition 1. *The equations of Fig. 4 define an unambiguous DP scheme: for any derivations E and E' arising from $\mathsf{H}\big[S, T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$, where S and T are two trees, $E \neq E' \Rightarrow \Phi(E) \neq \Phi(E')$.*

H – Initialization.

$$\mathsf{H}\big[\varnothing,\varnothing\big]^{\alpha}_{M,M'} = \begin{cases} 0 & \text{if } M,M' = \varnothing,\varnothing \quad (5a)\\ +\infty & \text{otherwise} \quad (5b)\end{cases}$$

H – Recursive steps.

$$\mathsf{H}\big[a(u)\circ X,\varnothing\big]^{\alpha}_{M,M'} = \begin{cases} \mathsf{Ins}\,(a) + \mathsf{H}\big[u,\varnothing\big]^{\alpha}_{M,M'} + \mathsf{H}\big[X,\varnothing\big]^{\alpha}_{M,M'} & \text{if } \alpha,M,M' = \mathsf{I}|\mathsf{D},\varnothing,\varnothing \quad (6a)\\ +\infty & \text{otherwise} \quad (6b)\end{cases}$$

$$\mathsf{H}\big[\varnothing,b(v)\circ Y\big]^{\alpha}_{M,M'} = \begin{cases} \mathsf{Del}\,(b) + \mathsf{H}\big[\varnothing,v\big]^{\alpha}_{M,M'} + \mathsf{H}\big[\varnothing,Y\big]^{\alpha}_{M,M'} & \text{if } M,M' = \varnothing,\varnothing \quad (7a)\\ +\infty & \text{otherwise} \quad (7b)\end{cases}$$

$$\mathsf{H}\big[a(u)\circ X,b(v)\circ Y\big]^{\alpha}_{M,M'} =$$

$$\min\begin{cases} \mathsf{Ins}\,(a) + \mathsf{H}\big[u,\varnothing\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} + \mathsf{H}\big[X,b(v)\circ Y\big]^{\mathsf{I}|\mathsf{D}}_{M,M'} & \text{if } \alpha = \mathsf{I}|\mathsf{D},\ M \neq \leftrightarrow \quad (8a)\\ \mathsf{Del}\,(b) + \mathsf{H}\big[\varnothing,v\big]^{\mathsf{D}}_{\varnothing,\varnothing} + \mathsf{H}\big[a(u)\circ X,Y\big]^{\mathsf{D}}_{M,M'} & \text{if } M' \neq \leftrightarrow \quad (8b)\\ \min\limits_{\substack{Y'\circ Y''=b(v)\circ Y\\ |Y'|\geq 2}} \mathsf{Ins}\,(a) + \mathsf{H}\big[u,Y'\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\leftrightarrow} + \mathsf{H}\big[X,Y''\big]^{\mathsf{I}|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y''}} & \quad (8c)\\ \min\limits_{\substack{X'\circ X''=a(u)\circ X\\ |X'|\geq 2}} \mathsf{Del}\,(b) + \mathsf{H}\big[X',v\big]^{\mathsf{I}|\mathsf{D}}_{\leftrightarrow,\varnothing} + \mathsf{H}\big[X'',Y\big]^{\mathsf{I}|\mathsf{D}}_{\delta_{M,X''},\delta_{M',Y}} & \quad (8d)\\ \mathsf{V}\big[a(u),b(v)\big]^{\varnothing} + \mathsf{H}\big[X,Y\big]^{\mathsf{I}|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y}} & \quad (8e)\end{cases}$$

where $\delta_{\varnothing,Z} = \varnothing$ and $\delta_{\leftrightarrow,Z} = \delta_{\rightarrow,Z} = \begin{cases}\varnothing & \text{if } |Z| = 0\\ \rightarrow & \text{otherwise}\end{cases}$

VH – Initialization.

$$\mathsf{VH}\,[\varnothing,b(v)] = +\infty \quad (9)$$

V & VH – Recursive steps.

$$\mathsf{V}\big[a(u),b(v)\big]^{\varnothing} = \min\begin{cases} \mathsf{Ins}\,(a) + \mathsf{VH}\,[u,b(v)] & (10a)\\ \mathsf{V}\big[a(u),b(v)\big]^{\uparrow} & (10b)\end{cases}$$

$$\mathsf{V}\big[a(u),b(v)\big]^{\uparrow} =$$

$$\min\begin{cases} \min\limits_{v=Y\circ c(w)\circ Y'} \mathsf{Del}\,(b) + \mathsf{H}\big[\varnothing,Y\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} + \mathsf{V}\big[a(u),c(w)\big]^{\uparrow} + \mathsf{H}\big[\varnothing,Y'\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} & (11a)\\ \mathsf{Match}\,(a,b) + \mathsf{H}\big[u,v\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} & (11b)\end{cases}$$

$$\mathsf{VH}\,[a(u)\circ X,b(v)] = \min\begin{cases} \mathsf{Ins}\,(a) + \mathsf{H}\big[u,\varnothing\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} + \mathsf{VH}\,[X,b(v)] & (12a)\\ \min\limits_{\substack{X'\circ X''=a(u)\circ X\\ |X'|\geq 2}} \mathsf{Del}\,(b) + \mathsf{H}\big[X',v\big]^{\mathsf{I}|\mathsf{D}}_{\leftrightarrow,\varnothing} + \mathsf{H}\big[X'',\varnothing\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} & (12b)\\ \mathsf{V}\big[a(u),b(v)\big]^{\varnothing} + \mathsf{H}\big[X,\varnothing\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing} & (12c)\end{cases}$$

Fig. 4: Unambiguous/complete DP scheme for aligning trees

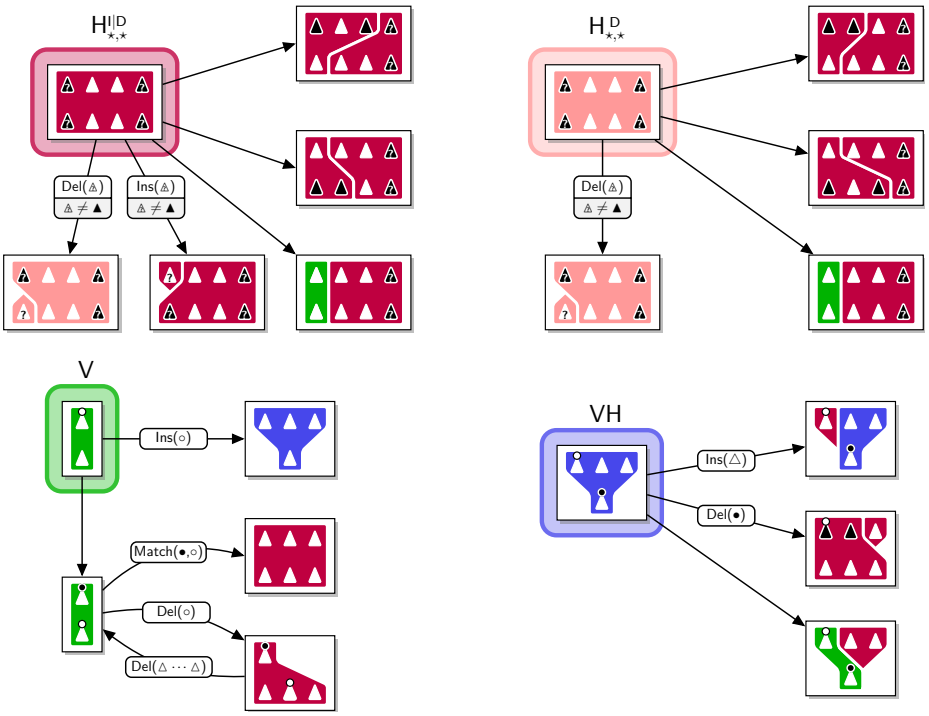

Fig. 5: Combinatorial illustration of our unambiguous tree alignment DP scheme. Black triangles indicate trees that must be matched in order to maintain unambiguity, and correspond to occurrences of $\rightarrow$ and $\leftrightarrow$ in the DP recurrences.

Proposition 2. *The equations of Fig. 4 define a complete DP scheme: for any alignment A between S and T, there exists a derivation E arising from $\mathsf{H}\big[S,T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ such that $A = \Phi(E)$.*

Corollary 1. *Let $c(S,T)$ be the content of the cell $\mathsf{H}\big[S,T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ computed by the optimization version of the DP scheme defined by the equations of Fig. 4. Then $c(S,T)$ is the cost of an optimal alignment of S and T.*

4 Complexity analysis and enumerative aspects

We now turn to the analysis of our algorithm, and of its underlying derivations/search space. We first analyze the time complexity of our unambiguous DP scheme, both in the worst and average case. Next, we consider the expected number of derivations for both the ambiguous and unambiguous DP schemes for an input of a given size. This analysis reveals that the ambiguous DP scheme [7, 5] sports a large average degree of ambiguity.

Time/space complexities of the unambiguous DP scheme. It is easy to observe that the time and space complexities of our algorithm, based on the DP scheme of Fig. 4, are dominated by that of the table-filling initial stage and, more precisely, by the computation of the cells of the DP table H. We first consider the question of describing the *indexing pairs* of H, i.e. pairs of forests that serve as indices to this table. An *infix forest* of a tree (either S or T) is a contiguous sequence of subtrees rooted at a given node, the *parent* of the forest. An infix forest is also a *suffix forest* if and only it contains the last child of its parent. The following proposition, describing indexing pairs, can be easily proven by induction.

Proposition 3. *Consider the DP scheme defined by Fig. 4, instantiated on two trees S and T. Then the indexing pairs (X, Y) of H obey one of the following conditions:*
(i) X is a suffix forest in S, and Y is an infix forest in T;
(ii) X is a infix forest in S, and Y is an suffix forest in T.

The complexity of a DP algorithm can be measured as the number of operations min and + that are required to compute the DP tables. Equations (8c), (8d), (11a) and (12b) involve loops that partition indexing forests, thus their computation requires a time complexity proportional to the size of the corresponding forests. Namely, computing a single DP cell indexed by (X, Y) can be done in $\mathcal{O}(|X| + |Y|)$ time, where $|\cdot|$ denotes the number of trees in the forest. Furthermore, the number of required DP tables is constant, thus we obtain from Prop. 3 the following proposition, whose proof is very similar to the proof of the worst-case time complexity of the tree alignment algorithm of Jiang *et al.* [6].

Proposition 4. *The complexity of the DP algorithm defined in Fig. 4, applied to trees S and T, is asymptotically proportional to $\sum_{X,Y}(|X| + |Y|)$, where the sum is being taken over all indexing pairs (X, Y) defined in Proposition 3.*

Corollary 2. *The worst-case time complexity of the DP algorithm defined in Fig. 4 on a pair of trees of respective sizes n_1 and n_2 is in $\Theta\left(n_1\, n_2\, \min(n_1, n_2)^2\right)$.*

We now focus on the *average-case time complexity*, i.e. the expected time taken by the algorithm, when executed on uniformly-drawn random trees of respective sizes n_1 and n_2. Let $\mathsf{comp}(S, T)$ be the complexity of the algorithm on inputs S and T, then the average-case complexity can be defined as

$$\sum_{\substack{S,T \text{ such that} \\ |S|=n_1 \text{ and } |T|=n_2}} \frac{\mathsf{comp}(S, T)}{\mathcal{C}_{n_1} \times \mathcal{C}_{n_2}}$$

where $\mathcal{C}_n$ denotes the number of trees of size n, *i.e.* the n-th Catalan number. Our approach to establish the average-case time complexity borrows heavily from [7], our main originality residing in the use of techniques introduced in the field of analytic combinatorics, allowing for a precise determination of the asymptotic terms. We refer the reader to Flajolet and Sedgewick's *magnum opus* [26] for further references.

Proposition 5. *The average-case time/space complexities of the DP algorithm defined in Fig. 4 on uniformly-drawn pairs of random trees of respective sizes n_1 and n_2 are in $\Theta(n_1\, n_2)$.*

The results above show that our unambiguous DP scheme, despite being more complex than the one introduced in Fig. 2, retains the same asymptotic worst and average-case complexity as its predecessors [7, 5].

The size of the search space. Our unambiguous/complete DP scheme can be interpreted as an enumeration scheme, allowing for a precise computation of the expected number of alignments between random pairs of trees of a given size. Indeed, each such alignment is an derivation of the DP scheme, and the DP equations also define counting recurrence equations.

Proposition 6. *The expected number of alignments between two random trees of total size n is asymptotically equivalent to*

$$\frac{\sqrt{2}\left(3-\sqrt{3}\right)}{6}\left(\frac{3}{2}\right)^n. \tag{13}$$

It is then interesting to compare this number with the expected number of derivations of the ambiguous DP scheme [7, 5], obtained by the same approach.

Proposition 7. *The expected number of derivations of the ambiguous DP scheme from Fig. 2 for two random trees of total size n is asymptotically equivalent to*

$$\sqrt{2}\left(2\sqrt{5-2\sqrt{5}}-2\right)\left(\sqrt{5}-2\right)\left(1+\frac{\sqrt{5}}{2}\right)^n. \tag{14}$$

The ratio (14) over (13) provides an asymptotic estimate for the *degree of ambiguity* of the DP scheme of Fig. 2, i.e. the expected number of derivations that represent (through Φ) a given tree alignment. This degree is approximately equal to 0.875×1.412^n. This discrepancy would challenge any attempt to adapt previous algorithms to produce unbiased estimates on alignments of a pair of trees, and confirms the rationale underlying our present contribution.

5 Conclusion

The main result of the present work is an unambiguous dynamic programming scheme for the tree alignment problem, having the same asymptotic complexity as previously proposed – ambiguous – DP schemes, both in the worst-case and in the average-case. Moreover, its underlying decomposition allows us to provide precise expressions for the expected number of alignments between two trees of a given size, and to quantify the level of ambiguity of the Jiang *et al.* algorithm,

using techniques of enumerative combinatorics. The natural extension of this result is to apply this algorithm in the context of tree alignments problems, such as RNA secondary structures comparison. Applications include the possibility to count and generate all optimal tree alignments, to compute probabilities, under a Boltzmann distribution, of matching bases, etc. We refer the reader to [11] for a work illustrating such applications for the problem of RNA folding.

References

1. Denise, A., Rinaudo, P.: Optimisation problems for pairwise RNA sequence and structure comparison: A brief survey. T. Computational Collective Intelligence **13** (2014) 70–82
2. Schirmer, S., Ponty, Y., Giegerich, R.: Introduction to RNA secondary structure comparison. In RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods. Vol. 1097 of Methods in Mol. Biol. Springer (2013) 247–73
3. Tai, K.C.: The tree-to-tree correction problem. J. ACM **26** (1979) 422–433
4. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. ACM Trans. Algorithms **6** (2009) 2:1–2:19
5. Blin, G., Denise, A., Dulucq, S., Herrbach, C., Touzet, H.: Alignments of RNA structures. IEEE/ACM Trans. Comput. Biology Bioinform. **7** (2010) 309–322
6. Jiang, T., Wang, L., Zhang, K.: Alignment of trees - an alternative to tree edit. Theor. Comput. Sci. **143** (1995) 137–148
7. Herrbach, C., Denise, A., Dulucq, S.: Average complexity of the Jiang-Wang-Zhang pairwise tree alignment algorithm and of a RNA secondary structure alignment algorithm. Theor. Comput. Sci. **411** (2010) 2423–2432
8. Schirmer, S., Giegerich, R.: Forest alignment with affine gaps and anchors, applied in RNA structure comparison. Theor. Comput. Sci. **483** (2013) 51–67
9. Bansal, M.S., Alm, E.J., Kellis, M.: Reconciliation revisited: handling multiple optima when reconciling with duplication, transfer, and loss. J. Comput. Biol. **20** (2013) 738–754
10. Ding, Y., Lawrence, C.E.: A statistical sampling algorithm for RNA secondary structure prediction. Nucleic Acids Res. **31** (2003) 7280–7301
11. Ponty, Y., Saule, C.: A combinatorial framework for designing (pseudoknotted) RNA algorithms. In WABI 2011. Vol. 6833 of LNCS, Springer (2011) 250–269
12. McCaskill, J.: The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers **29** (1990) 1105–1119
13. Younger, D.H.: Recognition and parsing of context-free languages in time n^3. Information and Control **10** (1967) 189 – 208
14. Reinharz, V., Ponty, Y., Waldispühl, J.: Using structural and evolutionary information to detect and correct pyrosequencing errors in noncoding rnas. J. Comput. Biol. **20** (2013) 905–919
15. Finkelstein, A.V., Roytberg, M.A.: Computation of biopolymers: a general approach to different problems. Biosystems **30** (1993) 1–19
16. Lefebvre, F.: A grammar-based unification of several alignment and folding algorithms. In ISMB 1996, AAAI Press (1996) 143–154
17. Höner zu Siederdissen, C.: Sneaking around concatmap: Efficient combinators for dynamic programming. In ICFP '12, ACM (2012) 215–226

18. Sauthoff, G., Möhl, M., Janssen, S., Giegerich, R.: Bellman's GAP – a language and compiler for dynamic programming in sequence analysis. Bioinformatics **29** (2013) 551–560
19. Giegerich, R.: Explaining and controlling ambiguity in dynamic programming. In CPM 2000, Vol. 1848 of LNCS, Springer (2000) 46–59
20. Reeder, J., Steffen, P., Giegerich, R.: Effective ambiguity checking in biosequence analysis. BMC Bioinformatics **6** (2005) 153
21. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. second edn. Addison-Wesley (2001)
22. Vingron, M., Argos, P.: Determination of reliable regions in protein sequence alignments. Protein Engineering **3** (1990) 565–569
23. Ferrè, F., Ponty, Y., Lorenz, W.A., Clote, P.: DIAL: a web server for the pairwise alignment of two RNA three-dimensional structures using nucleotide, dihedral angle and base-pairing similarities. Nucleic Acids Res. **35** (2007) W659–W668
24. Schirmer, S.: Comparing forests. PhD thesis, Bielefeld University (2011)
25. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol **48** (1970) 443–453
26. Flajolet, P., Sedgewick, R.: Analytic combinatorics. Cambridge Univ. Press (2009)

Appendix

DP schemes as enumeration schemes We now aim at proving the important properties that our DP scheme is complete (Proposition 2) and unambiguous (Proposition 1).

To do so, as these properties are related to the set of all derivations of the DP scheme, we turn the optimization DP scheme introduced in the main body of the paper into a derivation enumeration scheme, in which the content of a DP table cell is not any more a number representing the cost of an optimal alignment of the indexing forests, but the *set of all derivations of the DP scheme* for aligning the indexing forests.

The equivalent statement of Propositions 1 and 2 then becomes:

Proposition 1, reformulated. *The equations of Fig. 4 define an unambiguous DP scheme: for any E and E' in $\mathsf{H}\big[S,T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$, where S and T are two trees, $E \neq E' \Rightarrow \Phi(E) \neq \Phi(E')$.*

Proposition 2, reformulated. *The equations of Fig. 4 define a complete DP scheme: for any alignment A between S and T, there exists a derivation E in $\mathsf{H}\big[S,T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ such that $A = \Phi(E)$.*

Moreover, in the specific case of the tree alignment DP schemes we consider in the present paper, the notion of derivation can be simplified, which will be helpful for the forthcoming proofs. In Section 2, a derivation is defined as a tree whose nodes contain matches, insertions and deletions; such a derivation can also be seen as the sequence of matches, insertions and deletions obtained by a prefix traversal of the derivation-tree. In this context, the set of derivation-sequences associated to a DP cell can be a multi-set, if the considered DP scheme is ambiguous. It is important to note that the mapping Φ applies as well on derivations seen as sequences, as it simply amounts to extracting the matches from a derivation. From now, we consider derivations as such sequences.

This enumeration DP scheme, which is implicitly at the heart of the counting recurrences we used in Section 4, can be obtained from the optimization DP scheme by a change of algebra, from a $(\min, +, 0, +\infty)$ algebra to a $(\cup, \times, \epsilon, \varnothing)$ algebra, where ϵ represents the empty sequence, $\cup$ the (non-disjoint) union, $\times$ the set-theoretic Cartesian product, $\varnothing$ is absorbing for the Cartesian product, and where the cost of an edit operations is replaced by the actual corresponding match/insertion/deletion. For example, the DP scheme described in Fig. 2 can be turned into such a enumeration DP scheme as shown in Fig. 6.

From now on, we consider that we work with derivations seen a sequences of matches, insertions and deletions, and that a DP scheme is an enumeration DP scheme. For the sake of exposition, we do not rewrite our DP scheme in an enumeration algebra and we will refer to the DP recurrence equations described in Fig. 5, with the implicit understanding that they are seen as enumeration recurrence: min is replaced by *cup*, $+$ by $\times$, 0 by ϵ, $+\infty$ by $\varnothing$, $\mathsf{Match}\,(a,b)$ by (a,b), $\mathsf{Ins}\,(a)$ by $(a,-)$ and $\mathsf{Del}\,(b)$ by $(-,b)$.

Initialization.

$$\mathsf{JS}\left[\varnothing, \varnothing\right] = \epsilon \tag{15}$$

Recursive steps.

$$\mathsf{JS}\left[a(u) \circ X, \varnothing\right] = (a, -) \times \mathsf{JS}\left[u, \varnothing\right] \times \mathsf{JS}\left[X, \varnothing\right] \tag{16}$$

$$\mathsf{JS}\left[\varnothing, b(v) \circ Y\right] = (-, b) \times \mathsf{JS}\left[\varnothing, v\right] \times \mathsf{JS}\left[\varnothing, Y\right] \tag{17}$$

$$\mathsf{JS}\left[a(u) \circ X, b(v) \circ Y\right] = \bigcup \begin{cases} (a, b) \times \mathsf{JS}\left[u, v\right] \times \mathsf{JS}\left[X, Y\right] & \text{(18a)} \\ \bigcup\limits_{Y' \circ Y'' = b(v) \circ Y} (a, -) \times \mathsf{JS}\left[u, Y'\right] \times \mathsf{JS}\left[X, Y''\right] & \text{(18b)} \\ \bigcup\limits_{X' \circ X'' = a(u) \circ X} (-, b) \times \mathsf{JS}\left[X', v\right] \times \mathsf{JS}\left[X'', Y\right] & \text{(18c)} \end{cases}$$

Fig. 6: The enumeration version of the DP scheme of Fig. 2

Proving correctness We first prove that the DP equations we introduced indeed satisfy the required specifications, that will be used as invariants in the following proofs. Each cell of one of the DP tables (called a DP cell from now) we introduced contains a set of derivations, and we want to prove that if Specifications 1, 2 and 3 define constraint on a given table, then all derivations in an arbitrary cell of this table indeed satisfy these constraints. To do so, we apply an induction argument on the length of an derivation present in a given cell of one of the DP tables, where the length of an derivation is the number of match, insertion and deletion it contains, *i.e.* the length of the corresponding sequence on Σ.

Proposition 8. *The DP equations (5) to (12c) satisfy the constraints stated in Specifications 1, 2 and 3.*

Proof. First, we can notice that the constraints defined by Specifications 1, 2 and 3 concern only DP cells whose indexing pair of forests is composed of two non-empty forests and derivations that contain at least one match. So we consider here only derivations of length at least 1 containing at least one match.

Base case. The base case is composed of derivations of length exactly 1 composed of a single match, which implies the indexing pair is composed of two vertices (a, b). It is immediate to verify that Specifications 3, 1 and 2 are satisfied in such cases.

Induction hypothesis. We assume that, for any cell of the considered DP tables, derivations of length $k \geq 1$ encoded in a given DP cell satisfy the constraints defines by Specifications 3, 1 and 2.

Induction argument. We now consider an derivation of length $k + 1 \geq 2$.

(1). We first consider the case where it belongs to a DP cell $\mathsf{H}\big[a(u) \circ X, b(v) \circ Y\big]^{\mathsf{D}}_{M,M'}$ and prove that Specification 3.a is satisfied, i.e. that the tree $a(u)$ is not inserted.
We need to consider equations (8b) to (8e). By examining the right-hand of these equations and more precisely the DP cells whose index contains the tree $a(u)$, we observe that the corresponding constraints on the derivations in these DP cells imply that $a(u)$ will be matched in these derivations. This is immediate for equations (8b) to (8d) as derivations in DP tables cells from their right-hand side are of length at most k and we can thus apply induction.

For equation (8e), if $|X| = |Y| = 0$, the derivations in $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$ have length k and we cannot apply induction immediately; however, derivations in $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$ will contain either (i) an insertion $(a, -)$ (by equation (10a)) and induction can then be applied to $\mathsf{VH}\,[u, b(v)]$, or (ii) an insertion $(-, b)$ (by equation (10b) followed by (11a)) and induction can be applied to $\mathsf{V}\big[a(u), c(w)\big]^{\uparrow}$, or (iii) a match (a, b) (by equation (10b) followed by (11b)) which ensures Specification 3.a is satisfied.

(2). We now turn to Specification 3.b, with $M = \leftrightarrow$ and consider an derivation from $\mathsf{H}\big[a(u) \circ X, b(v) \circ Y\big]^{\alpha}_{\leftrightarrow,M'}$. We again need to consider equations (8b) to (8e) to prove that indeed the first and last tree of $a(u) \circ X$ are matched. First we can notice that $|X| > 0$ as otherwise $\mathsf{H}\big[a(u) \circ X, b(v) \circ Y\big]^{\alpha}_{\leftrightarrow,M'} = \varnothing$.

For (8b), induction on the right-hand side DP cell $\mathsf{H}\big[a(u) \circ X, Y\big]^{\mathsf{D}}_{M,M'}$ allows immediately to conclude.

For (8c), induction on $\mathsf{H}\big[u, Y'\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\leftrightarrow}$ implies that a vertex from u will be in a match and we need to show that the last tree of X is also matched. We consider the derivations in the DP cell $\mathsf{H}\big[X, Y''\big]^{\mathsf{I}|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y''}}$: $\delta_{M,X} = \rightarrow$ and so $|Y''| \neq 0$ (otherwise there will be no corresponding derivation by equation (6a)) and induction implies that the last tree of X is indeed matched.
A similar argument applies for equation (8d).

Finally for equation (8e) induction applied to the DP cell $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$ (which can be done immediately as $|X| > 0$) ensures that $a(u)$ is matched, while the fact that $\delta_{M,X} = \rightarrow$, together with induction applied to $\mathsf{H}\big[X, Y\big]^{\mathsf{I}|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y}}$ ensures the last tree of X is matched.

(3). Very similar arguments, based on careful case analysis, apply for Specification 3.b, with $M = rightarrow$ and Specification 3.c, and we omit them here.

We now address the case of derivations from a DP cell from a V table, $\mathsf{V}\big[a(u), b(v)\big]^{\beta}$ and prove that Specification 1 is satisfied.

(4). We first consider an derivation of $\mathsf{V}\big[a(u), b(v)\big]^{\uparrow}$. For equation (10b), again we cannot apply induction as the corresponding derivations have also length $k+1$. However, following the same approach than when dealing with equation (8e) we can consider equations (11a) and (11b) and apply induction to $\mathsf{V}\big[a(u), c(w)\big]^{\uparrow}$ in the former case and observe the match (a, b) in the latter to conclude that Specification 1.b is satisfied.

(5). We now consider an derivation of $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$. For equation (10a), induction applied to $\mathsf{VH}\,[u, b(v)]$ implies a match with a vertex of u and a vertex of $b(v)$. The case of equation (10b) was considered above, which lets us conclude that Specification 1.a is satisfied.

(6). We finally address the case of derivations from a DP cell from a VH table, $\mathsf{VH}\,[a(u) \circ X, b(v)]$ and of Specification 2; we want to prove that any derivation contains a match with a vertex of $b(v)$. This follows from induction for derivations obtained through equation (12a) (term $\mathsf{VH}\,[X, b(v)]$) and for derivations obtained through equation (12b) (term $\mathsf{H}\big[X', v\big]^{\mathsf{I}|\mathsf{D}}_{\leftrightarrow,\varnothing}$). For equation (12c), it follows from the previous claim that Specification 1 is satisfied.

Proof of Proposition 1 We remind that we want to prove the following statement: for any derivations $E, E' \in \mathsf{H}\big[S, T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$, $E \neq E' \Rightarrow \Phi(E) \neq \Phi(E')$ (Proposition 1, reformulated in an enumeration framework).

To do so, we will prove by induction a more general statement. We want to prove that for any indexing pair of forests (W, Z) for a DP table C and any $E, E' \in C[W, Z]$, $E \neq E' \Rightarrow \Phi(E) \neq \Phi(E')$. We assume that all derivations in $C[W, Z]$ indeed satisfy the constraints on the DP table C described in Specifications 1, 2 and 3, which will prove to be crucial in the proof. Note also that the constraints on derivations contained in $C[W, Z]$ are constraints on matches, insertions and deletions and are thus also applicable to alignments.

We denote by n_W the number of vertices of W, n_Z the number of vertices of Z and $n = n_W + n_Z$.

Base case. We consider the cases where either $n = 0$, or at least one of n_W and n_Z is equal to 0, or $n_W = n_Z = 1$. The statement holds obviously in such cases.

Induction hypothesis. For any indexing pair of forests (W, Z) for a DP table C, with size $n \geq 0$, and any $E, E' \in C[W, Z]$, $E \neq E' \Rightarrow \Phi(E) \neq \Phi(E')$.

Induction argument. Let (W, Z) be an indexing pair for a DP table C, of size $n \geq 1$.

(1). If $C[W,Z] = \mathsf{V}\big[a(u),b(v)\big]^{\uparrow}$, then, by induction, the only possibility for $E \neq E'$ such that $\Phi(E) = \Phi(E')$, would be that $E \in (-,b) \times \mathsf{H}\big[\varnothing, Y\big]^{\mathsf{I|D}}_{\varnothing,\varnothing} \times \mathsf{V}\big[a(u),c(w)\big]^{\uparrow} \times \mathsf{H}\big[\varnothing, Y'\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$ and $E' \in (a,b) \times \mathsf{H}\big[u,v\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$. But in the first case b does not belong to a matching while in the second it does, so $\Phi(E) \neq \Phi(E')$.

(2). If $C[W,Z] = \mathsf{V}\big[a(u),b(v)\big]^{\varnothing}$, then by induction applied to equation (10a) and by point (1) above applied to equation (10b), we need only to consider the case where $E \in (a,-) \times \mathsf{VH}\,[u,b(v)]$ and $E' \in \mathsf{V}\big[a(u),b(v)\big]^{\uparrow}$. By Spec. 1, $\Phi(E')$ contains a match involving a, while a is in an insertion in $\Phi(E)$, which again implies that $\Phi(E) \neq \Phi(E')$.

(3). We now consider the case where $C[W,Z] = \mathsf{VH}\,[a(u) \circ X, b(v)]$ and look if it is possible that for some derivations $E \neq E'$ we have $\Phi(E) = \Phi(E')$.
We first consider the case where E, E' are generated by the same DP equation. For equation 12a, as $\mathsf{H}\big[u,\varnothing\big]^{\varnothing}_{\varnothing,\mathsf{I|D}}$ contains a single derivation, this would imply that two different derivations of $\mathsf{VH}\,[X,b(v)]$ define the same alignment, which is impossible by induction. For equation 12b, the constraint $M = \leftrightarrow$ in the term $\mathsf{H}\big[X',v\big]^{\mathsf{I|D}}_{\leftrightarrow,\varnothing}$, together with the fact that any derivation in $\mathsf{H}\big[X'',\varnothing\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$ contains a match, implies that no tow derivations defined by different partitions $X' \circ X''$ can define the same alignment; then induction allows to conclude. Finally the case of 12c, has been addressed by point (1) above. So we can assume that E and E' belong to derivations generated from different DP equations.
The fact that any derivation in rhs(12c) contains a match involving a (Spec. 1) implies that $\Phi(E) = \Phi(E')$ cannot happen if $E \in rhs$(12a), $E' \in$ (12c).
Next, the fact that $|X'| \geq 2$ in (12b), together with Spec. 3 implies that the first tree of X contains a match in derivations generated by equation (12b), while the term $\mathsf{H}\big[X,\varnothing\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$ in (12c) prevents this. So $\Phi(E) = \Phi(E')$ cannot happen if $E \in rhs$(12b), $E' \in$ (12c).
The last case to consider is thus $E \in rhs$(12a), $E' \in$ (12b). Spec. 3 imply that $\Phi(E')$ contains a match in u while $a(u)$ is fully inserted in $\Phi(E)$.
This allows to conclude that there cannot be $E \neq E'$ in $\mathsf{VH}\,[a(u) \circ X, b(v)]$ such that $\Phi(E) = \Phi(E')$.

(4). We finally consider the case where $C[W,Z] = \mathsf{H}\big[a(u) \circ X, b(v) \circ Y\big]^{\alpha}_{M,M'}$.

Similarly to case (3) above, it follows from (i) Spec. 1, 2 and 3, (ii) the induction hypothesis, and (iii) cases (1), (2) and (3) already proved, that if E and E' are generated by the same DP equation, $E \neq E' \rightarrow \Phi(E) \neq \Phi(E')$. So we concentrate on the case where E and E' are generated by different DP equations, and consider

all possible cases and for each outline the properties of E and E' that imply that $\Phi(E) \neq \Phi(E')$. Our argument holds for any values of (α, M, M').

(a). $E \in rhs(8a),\ E' \in rhs(8b)$. $a(u)$ is fully inserted in $\Phi(E)$, while it is not in $\Phi(E')$ due to the constraint D.

(b). $E \in rhs(8a),\ E' \in rhs(8c)$. $a(u)$ is fully inserted in $\Phi(E)$, and so no vertex from $a(u)$ is in a match, while in $\Phi(E')$, the constraint $M' = \leftrightarrow$ in the term $\mathsf{H}\big[u, Y'\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\leftrightarrow}$ implies $\Phi(E')$ contains a match involving a vertex from u.

(c). $E \in rhs(8a),\ E' \in rhs(8d)$. $a(u)$ is fully inserted in $\Phi(E)$, and so no vertex from $a(u)$ is in a match, while in $\Phi(E')$, the constraint $M = \leftrightarrow$ in the term $\mathsf{H}\big[X', v\big]^{\mathsf{I}|\mathsf{D}}_{\leftrightarrow,\varnothing}$ implies $\Phi(E')$ contains a match involving a vertex from u.

(d). $E \in rhs(8a),\ E' \in rhs(8e)$. $a(u)$ is fully inserted in $\Phi(E)$, and so no vertex from $a(u)$ is in a match, while in $\Phi(E')$ there is a match composed of a vertex of $a(u)$ and of a vertex of $b(v)$.

(e,f,g). For the cases $E \in rhs(8b),\ E' \in rhs(8c)$, $E \in rhs(8b),\ E' \in rhs(8d)$ and $E \in rhs(8b),\ E' \in rhs(8e)$. We can apply similar arguments then above, involving $b(v)$ instead of $a(u)$ in a symmetric way.

(h). $E \in rhs(8c),\ E' \in rhs(8d)$. The alignment $\Phi(E)$ contains a match (x, y) with $x \in u$ and $y \in b(v)$ and a match (x', y') with $x' \in u$ and $y' \notin b(v)$ (because $|Y'| \geq 2$). However, in the alignment $\Phi(E')$ all matches (x, y) with $x \in a(u)$ satisfy that $y \in v$. So $\Phi(E) \neq \Phi(E')$.

(i). $E \in rhs(8c),\ E' \in rhs(8e)$. The same argument than case (h) applies: in $\Phi(E)$ matches involving $a(u)$ are spread over at least two trees of $b(v) \circ Y$, while they involve only vertices of $b(v)$ in $\Phi(E')$.

(j). $E \in rhs(8d),\ E' \in rhs(8e)$. The argument is symmetric to the one use in case (i).

Proof of Proposition 2 We want to prove that for any alignment A between S and T, there is at least one derivation $E \in \mathsf{H}\big[S, T\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ such that $\Phi(E) = A$ (Proposition 2, reformulated in an enumeration framework). In order to do so, we will proceed by induction on the size n of S and T (*i.e.* the total number of vertices in both S and T) and consider a more general statement that considers the constraints on alignments induced by Specifications 1, 2 and 3.

Let (W, Z) be an indexing pair for a DP table C. The constraints on derivations contained in $C[W, Z]$ are constraints on matches, insertions and deletions and are thus also applicable to alignments; so they define implicitly a set of alignments of the forests W and Z, that we denote by $C_A(W, Z)$, which is the subset of all alignments of W and Z that respect the constraints of the table C. The statement we want to prove by induction is then that for any alignment $A \in C_A(W, Z)$, there exists $E \in C[W, Z]$ such that $\Phi(E) = A$.

Through the proof, we assume again that all derivations in $C[W, Z]$ indeed satisfy the constraints on the DP table C described in Specifications 1, 2 and 3 (Prop. 8).

Base case. We consider the cases where either $n = 0$, or at least one of n_W and n_Z is equal to 0, or $n_W = n_Z = 1$. The statement holds obviously in such cases.

Induction hypothesis. For any indexing pair of forests (W, Z) for a DP table C, with size $n \geq 0$, and any alignment $A \in C_A(W, Z)$, there exists $E \in C[W, Z]$ such that $\Phi(E) = A$.

Induction argument. Let (W, Z) be an indexing pair for a DP table C, of size $n \geq 1$.

(1). We first consider the case where $C[W, Z] = \mathsf{V}\big[a(u), b(v)\big]^{\uparrow}$. From Spec. 1, we know that a must belong to a match in A. If this match is (a, b), the remaining forests u and v must then be aligned without additional constraint. This is achieved in (11b) by invoking H with $(\alpha, M, M') = (\mathsf{I}|\mathsf{D}, \varnothing, \varnothing)$, to which the induction hypothesis on the size of the indexing pair of forests applies to ensure that the alignment A without match (a, b), which is an unconstrained alignment of u and v, is the image by Φ of an derivation from $\mathsf{H}\big[u, v\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$.

If a is not matched with b, then it must be matched with a vertex in a subtree rooted at a child c of b. In that case, the ancestrality condition forces both b and all subtrees of $b(v)$ rooted at the siblings of c to be deleted in A. This is verified by induction on the three recursive calls of (11a): $\mathsf{H}\big[\varnothing, Y\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ (base case), $\mathsf{H}\big[\varnothing, Y'\big]^{\mathsf{I}|\mathsf{D}}_{\varnothing,\varnothing}$ (base case) and $\mathsf{V}\big[a(u), c(w)\big]^{\uparrow}$ (induction hypothesis).

This completes the proof of completeness for $C[W, Z] = \mathsf{V}a(u)b(v)\uparrow$.

(2). We now consider the case where $C[W, Z] = \mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$. We assume that $a(u)$ and $b(v)$ are matched. If a is matched, then (10b) calls $\mathsf{V}\big[a(u), b(v)\big]^{\uparrow}$ and point (1) above allows to conclude that A is indeed the image by Φ of some $E \in \mathsf{V}\big[a(u), b(v)\big]^{\uparrow}$ which implies also that $E \in \mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$.

Otherwise, a is inserted in A and a the remaining of the alignment A is an alignment A' between u and $b(v)$ with at least one match. By induction we can assume that there exists $E' \in \mathsf{VH}\,[u, b(v)]$ such that $A' = \Phi(E')$ and Equation (10a) ensures that there exists $E = (a, -).E'$ in $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$.

(3). We now consider the case where $C[W, Z] = \mathsf{VH}\,[a(u) \circ X, b(v)]$. The constraint on A here is that there is a vertex in $b(v)$ that belongs to a match.

Then we can consider three mutually exclusive cases: (i) all matches in A also involve vertices from $a(u)$, (ii) matches in A involve vertices in $a(u)$ and in X (so assuming that $|X > 0)$, or (iii) matches in A do not involve any vertex from $a(u)$ (here again $|X| > 0$).

Case (i), in which the vertices from X can only belong to insertions, is handled by equation (12c). Alignment A is composed of two sub-alignments: A' that contains all matches, insertions and deletions involving vertices of $a(u)$ and $b(v)$, and A'' composed of insertions for all vertices of X. Point (1) above ensures that A' is the image of an derivation of $\mathsf{V}\big[a(u), b(v)\big]^{\varnothing}$ while the base case applied to the recursive call $\mathsf{H}\big[X, \varnothing\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$ ensures that A'' is the image of an derivation of $\mathsf{H}\big[X, \varnothing\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$.

Case (ii) is handled by equation (12b). The ancestrality condition imposes that b is deleted in A, which is indeed ensured by this equation. The constraint $M = \leftrightarrow$ in the recursive call $\mathsf{H}\big[X', v\big]^{\mathsf{I|D}}_{\leftrightarrow,\varnothing}$ to which the induction hypothesis applies as the indexing pair is strictly smaller, ensures that indeed the set of matches of A are found in an derivation contained in this DP cell.

Last, case (iii) requires that all vertices of $a(u)$ are inserted. So A is composed of a set A' of insertions (vertices of A) and of an alignment A'' of X and $b(v)$ with at least one match. Equation 12a ensures that indeed there exists $E \in \mathsf{VH}\left[a(u) \circ X, b(v)\right]$ such that $\Phi(E) = A$. Indeed, $E = E' \times E''$, where $\Phi(E') = A'$ and $\Phi(E'') = A''$ and induction applied to $(a, -) \times \mathsf{H}\big[u, \varnothing\big]^{\mathsf{I|D}}_{\varnothing,\varnothing}$ and $\mathsf{VH}\left[X, b(v)\right]$ ensures that E' and E'' exist.

(4). We now consider the case where $C[W, Z] = \mathsf{H}\big[a(u) \circ X, b(v) \circ Y\big]^{\alpha}_{M,M'}$.

We can consider five mutually exclusive cases regarding the structure of A, that are handled respectively by the five equations (8a), (8e), (8d), (8c) and (8b): (i) the tree $a(u)$ is fully inserted, (ii) the tree $b(v)$ is fully deleted, (iii) $a(u)$ is matched and some matches involve vertices outside of $b(v)$ (the order condition condition implies that all matches involving vertices from $b(v)$ also involve vertices of $a(u)$), (iv) $b(v)$ is matched and some matches involve vertices outside of $a(u)$ (this implies that all matches involving vertices from $a(u)$ also involve vertices of $b(v)$), and finally (v) all matches of $a(u)$ involve $b(v)$ and conversely.

Cases (i) and (ii) are handled by equations (8a), (8e), and induction applies immediately, for all possible values of (α, M, M'), so we concentrate on the three other cases.

For case (iii), the matches involving $a(u)$ implicitly define a partition $Y' \circ Y'' = b(v) \circ Y$, where the last tree of Y' contains vertices matched with $a(u)$ while no tree of Y'' does, and $|Y'| \geq 2$. So A can be decomposed into an alignment A' of $a(u)$ with Y' with matches in $a(u)$ and the last tree of Y', and an alignment A'' of X with Y''. Now, the ancestrality condition, together with $|Y'| \geq 2$ implies that a is inserted in A', and induction applied to $\mathsf{H}\big[u, Y'\big]^{\mathsf{I|D}}_{\varnothing,\leftrightarrow}$ ensures there exists E' in this DP cell such that $A' = \{(a, -)\} \cup \Phi(E')$. So we consider the question of the existence of $E'' \in \mathsf{H}\big[X, Y''\big]^{\mathsf{I|D}}_{\delta_{M,X},\delta_{M',Y''}}$ such that $\Phi(E'') = A''$.

If $(M, M') = (\varnothing, \varnothing)$, A'' is unconstrained and induction applies immediately, for both possible values of α as there is an ensured match in u.
If $M = \rightarrow$, then A contains a match in the last tree of X if $|X| > 0$ (which implies that $|Y''| > 0$) and in $a(u)$ if $|X| = 0$ (which implies that A'' is empty). The latter case is already covered by the general argument. For the former case, $\delta_{\rightarrow,X} = \rightarrow$, so induction applied to $\mathsf{H}\big[X, Y''\big]^{\mathsf{I}\|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y''}}$ allows to conclude. The previous argument holds for all values of M'
Finally, if $M = \leftrightarrow$, the same argument applies as there is already en ensured match in u.

A symmetric argument applies to case (iv), and for case (v) the result follows again immediately by the previous analysis of the table VH (for the part of the alignment A that involves trees $a(u)$ and $b(v)$) and by induction on the term $\mathsf{H}\big[X, Y\big]^{\mathsf{I}\|\mathsf{D}}_{\delta_{M,X},\delta_{M',Y}}$.

Proof of Corollary 1 This Corollary states that our DP scheme indeed computes the cost of an optimal alignment between S and T. It follows immediately from Proposition 2 that states that all alignments between S and T are considered by our DP scheme: for any alignment A between S and T, there exists a derivation E of our DP scheme such that $\Phi(E) = A$. As the cost of an alignment is obtained trivially from E by summing the respective costs of matches, insertions and deletions, then our DP scheme , that takes the minimum among all derivations, indeed computes the cost of an optimal alignment.

Proof of Corollary 2 A suffix forest is fully characterized by a single node (the leftmost sibling rooting the first tree of the forest), while an infix forest requires two. So the number of pairs of forests satisfying condition (i) (resp. (ii)) of Proposition 3 is bounded by $n_1\, n_2^2$ (resp. $n_1^2\, n_2$). Moreover, the size of the subforests decomposed in the right-hand sides of the DP equations can not exceed n_1 and n_2 respectively. By Lemma 4, the complexity on every pair of trees of respective sizes n_1 and n_2 is in $\mathcal{O}\left(n_1\, n_2\, (n_1 + n_2)^2\right)$. This upper-bound is reached for pairs of trees that both have a node with a linear number of children.

Proof of Proposition 5 Let $I(z, u)$ and $S(z, u)$ be the generating functions

$$I(z,u) = \sum_{n,k\geq 1} i_{n,k}\, z^n\, u^k \quad \text{and} \quad S(z,u) = \sum_{n,k\geq 1} s_{n,k}\, z^n\, u^k,$$

where $i_{n,k}$ (resp. $s_{n,k}$) denotes the number of rooted ordered trees of size n with a marked infix forest (resp. suffix forest) composed of k trees.

By Lemma 4, the average-case time complexity is given, up to a constant factor, by

$$\frac{1}{\mathcal{C}_{n_1}\,\mathcal{C}_{n_2}} \sum_{\substack{S \text{ tree} \\ \text{of size } n_1}} \sum_{\substack{T \text{ tree} \\ \text{of size } n_2}} \sum_{\substack{(X,Y) \text{ satisfying} \\ \text{(i) or (ii)}}} |X| + |Y|.$$

This last expression can be rewritten in terms of the coefficients of the previous generating functions, as

$$\frac{i_{n_1} s'_{n_2} + i'_{n_1} s_{n_2} + s'_{n_1} i_{n_2} + s_{n_1} i'_{n_2} - \left(s_{n_1} s'_{n_2} + s'_{n_1} s_{n_2}\right)}{\mathcal{C}_{n_1}\,\mathcal{C}_{n_2}} \tag{19}$$

where i_n (resp. s_n, i'_n, s'_n) denotes the nth coefficient of $I(z,1)$ (resp. $S(z,1)$, $\frac{\partial I}{\partial u}(z,1)$, $\frac{\partial S}{\partial u}(z,1)$). Note that $s_{n_1}\, s'_{n_2} + s'_{n_1}\, s_{n_2}$ is subtracted to avoid counting each pair of suffix forests twice.

Using the symbolic method, we can characterize the generating functions $I(z,u)$ and $S(z,u)$. More precisely, we have

$$S(z,u) = \frac{u\,C(z)\,C'(z)}{1-u\,C(z)} \qquad \text{and} \qquad I(z,u) = \frac{S(z,u)}{1-C(z)},$$

where $C(z) = (1-\sqrt{1-4\,z})/2$ is the well-known generating function of rooted ordered trees.

Basic singularity analysis provides asymptotic estimates for all the terms appearing in Equation (19). Upon suitable simplifications, we find Eq. (19) to be asymptotically equivalent to $12\,n_1\,n_2$, i.e. the average-case time complexity of our algorithm is in $\Theta(n_1\,n_2)$.

The average space complexity is bounded from above by the time complexity, and from below by the number of pairs of nodes in $S \times T$, i.e. by $n_1\,n_2$, and is therefore also in $\Theta(n_1\,n_2)$.

Proof of proposition 6 The proof relies once again on generating functions, as recurrence relations can be naturally translated into functional equations on generating functions.

Let $H^{\alpha}_{M,M'}(z)$, $V^{\beta}(z)$ and $VH(z)$ be the generating functions of derivations that satisfy the restrictions of the DP tables $\mathsf{H}^{\alpha}_{M,M'}$, V^{β} and VH. For instance, $H^{\mathsf{D}}_{\varnothing,\leftrightarrow}(z)$ is the generating function of the number of derivations starting from forests X and Y such that the first tree of X is not inserted, and the first and last trees of Y are matched.

Equations (5)-(12c) can be translated into a system of 21 functional equations which, combined with the integral nature of their coefficients, uniquely characterize these generating functions. With the help of `maple`, this system can be explicitly solved, and the asymptotic estimate (13) can derived using singularity analysis tools [26]. The number of derivations is also the number of alignments, allowing us to conclude. □