Counting, generating and sampling tree alignments

Cedric Chauve, Julien Courtiel, Yann Ponty

Counting, generating and sampling tree alignments

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

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

Abstract. Pairwise ordered tree alignment are combinatorial objects that appear in important applications, such as RNA secondary structure comparison. However, the usual representation of tree alignments as supertrees is ambiguous, i.e. two distinct supertrees may induce identical sets of matches between identical pairs of trees. This ambiguity is uninformative, and detrimental to any probabilistic analysis.
In this work, we consider tree alignments up to equivalence. Our first result is a precise asymptotic enumeration of tree alignments, obtained from a context-free grammar by mean of basic analytic combinatorics. Our second result focuses on alignments between two given ordered trees S and T. By refining our grammar to align specific trees, we obtain a decomposition scheme for the space of alignments, and use it to design an efficient dynamic programming algorithm for sampling alignments under the Gibbs-Boltzmann probability distribution. This generalizes existing tree alignment algorithms, and opens the door for a probabilistic analysis of the space of suboptimal alignments.

1 Introduction

Tree alignments are the natural analog of sequence alignments, and have been introduced by Jiang, Wang and Zhang [9] to model and quantify the similarity between two (ordered[4]) trees. Initially proposed as an alternative to tree-edit distance, the tree alignment model has proven more robust, allowing for the inclusion of complex local operations [2], and for being generalized to multiple input trees [8]. Consequently, tree alignment has been used in a wide array of applicative contexts ranging from web crawling [17] to software engineering [16] through RNA Bioinformatics [7]. The minimal cost tree alignment between two trees of size n_1 and n_2, under classic insertion/deletion/(mis)-match operations, can be computed using dynamic programming (DP). The current best algorithms have a worst-case time and space complexity respectively in $\mathcal{O}(n_1 n_2 (n_1 + n_2)^2)$ and $\mathcal{O}(n_1 n_2 (n_1 + n_2))$ [9] algorithms, and an average-case time and space complexity (on uniformly drawn instances) in $\mathcal{O}(n_1 n_2)$ [6].

In the context of sequence alignments, the enumeration of alignments has been the object of much interest in Computational Biology [4,12,1]. Alignments

[4] In this work, unless explicitly specified, all trees will be rooted and ordered.

between two sequences over an alphabet Σ can be encoded as sequences over an extended alphabet Σ_a, representing insertions, deletions and (mis)matches (*e.g.* $\Sigma = \{a, b\}$, $\Sigma_a = \{(a,-),(-,b),(a,b),(a,a),(b,a),(b,b)\}$). Many sequences over Σ_a are equivalent if one considers only (mis)matches of the alignments, *i.e.* they align sequence of same lengths and induce the same sets of matched positions (*e.g.* $(a,-),(-,b)$ and $(-,b),(a,-)$). It is a natural problem to enumerate distinct sequence alignments for two sequences of cumulated length n [14, pp. 188]. Beyond purely theoretical considerations, the decompositions introduced for enumerating distinct sequence alignments were adapted into DP algorithms, *e.g.* for probabilistic alignment based on expectation maximization [3], or to compute Gibbs-Boltzmann measures of reliability [13].

In the present work, we consider similar questions on *tree alignments*. We are first interested in counting distinct tree alignments, *i.e.* enumerating, up to equivalence, ordered trees whose vertices are labeled in Σ_a (called *supertrees* from now). For trees, the notion of equivalence of alignments generalizes that of sequence alignments, *i.e.* two alignments are *equivalent* when they align the same pairs of trees, and induce the same sets of (mis)matched positions. Unfortunately, contrasting with the case of sequence alignments, existing DP algorithms for computing an optimal tree alignment [9,2,11] cannot be easily adapted into enumeration schemes for tree alignments up to equivalence. This additional difficulty is due to the existence of ambiguities of different nature.

Our main contribution is a grammar for (distinct) tree alignments, which provably generates a single representative for each equivalence class. We use the symbolic method [5] to obtain the generating function of tree alignments, and asymptotic equivalents for various statistics of interest can easily be derived, such as the average number of alignments over trees of total size n. Finally, and, perhaps more importantly from an applied point of view, the grammar can be transformed into an unambiguous and complete DP algorithm for aligning two input trees. The resulting algorithm has the same asymptotic worst-case and average-case complexities, up to reasonable constants, as the current best – ambiguous – algorithm [9,2]. The main interest of such an algorithm is that it opens immediately the way to new applications for the tree alignment model, including a critical assessment of the reliability of optimal alignments, either obtained by counting co-optimal alignments, or by sampling suboptimal alignments according to a Gibbs-Boltzmann distribution (see [10] for an example of this approach for the RNA folding problem).

In Section 2 we introduce the main definitions about trees, supertrees and tree alignments. In Section 3, we provide a grammar that generates all tree alignments. In Section 4.1 we analyze this grammar from an enumerative point of view and give precise results on the number of alignments of fixed size. Finally, in Section 4.2 we show how to transform the tree alignments grammar into a dynamic programming algorithm to sample tree alignments between two specified trees.

2 Definitions

Trees and supertrees. Let Σ be an *alphabet*. A tree T on Σ is a rooted plane tree whose vertices are labeled by elements of Σ. We denote by V_T the set of vertices of T. We *remove a non-root vertex* v from a tree T by contracting the edge between v and its parent u, that keeps its label. Removing the root r of a tree consists in creating a forest composed of the subtrees rooted at the children of r. We denote the operation of removing a vertex v from T by $T - v$.

We denote by Σ_a the alphabet defined by $\Sigma_a = (\Sigma \cup \{-\})^2 - \{(-,-)\}$. An element $(x, y) \in \Sigma_a$ is an *insertion* (resp. *deletion*, *match*) if $y = -$ (resp. $x = -$, $(x, y) \in \Sigma^2$). A *supertree* A is a tree on Σ_a; a vertex of A is an insertion (resp. deletion, match) if its label is an insertion (resp. deletion, match). The size of a supertree A is the number of its insertions and deletions, plus twice the number of its matches. A *superforest* is an ordered sequence of supertrees.

Given a supertree A on Σ, we define two forests $\pi_1(A)$ and $\pi_2(A)$ as follows: $\pi_1(A)$ (resp. $\pi_2(A)$) is obtained by (1) iteratively removing all insertion (resp. deletions) of A, in an arbitrary order, and (2) replacing the label (x, y) of each remaining vertex by x (resp. y). We refer to Fig. 1 for an illustration. We extend the notations π_1 and π_2 on vertices: for a non-insertion (resp. non-deletion) vertex v of A, we denote by $\pi_1(v)$ (resp. $\pi_2(v)$) the corresponding vertex in $\pi_1(A)$ (resp. $\pi_2(A)$). A vertex x of $\pi_1(A)$ such that $\pi_1^{-1}(x)$ is an insertion (resp. match) is said to be inserted (resp. matched) in A. Similarly, a vertex y of $\pi_2(A)$ such that $\pi_2^{-1}(y)$ is a deletion (resp. match) is said to be deleted (resp. matched) in A.

Tree alignments. As forests $\pi_1(A)$ and $\pi_2(A)$ are embedded into the supertree A, the latter implicitly defines an *alignment* between the forests $\pi_1(A)$ and $\pi_2(A)$, *i.e.* a set of correspondences between vertices of $\pi_1(A)$ and $\pi_2(A)$, that is consistent with the structure of both forests [9]. We refer to Fig. 1 for an illustration.

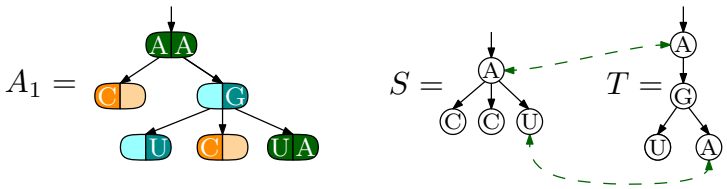

Fig. 1. A supertree A_1 with alphabet $\Sigma = \{A, C, G, U\}$, and the associated trees $S = \pi_1(A_1)$ and $T = \pi_2(A_1)$. The alignment of S and T defined by A is composed of two pairs of matched (A, A) and (U, A), indicated by dashed arrows.

We now turn to the central notion of *equivalent alignments*, *i.e.* alignments of identical pairs of trees, that contain exactly the same set of matched vertices. Given a supertree A, representing an alignment between two trees $S = \pi_1(A)$

and $T = \pi_2(A)$, the *set of matches of* A is formed by the elements (x, y) of $V_S \times V_T$ such that $\pi_1^{-1}(x) = \pi_2^{-1}(y)$ (*i.e.* there exists a vertex v of A such that $\pi_1(v) = x$ and $\pi_2(v) = y$). Two supertrees A_1 and A_2 are *equivalent* if $\pi_1(A_1) = \pi_1(A_2)$, $\pi_2(A_1) = \pi_2(A_2)$, and the sets of matches of A_1 and A_2 are identical (see Fig. 2 for an illustration).

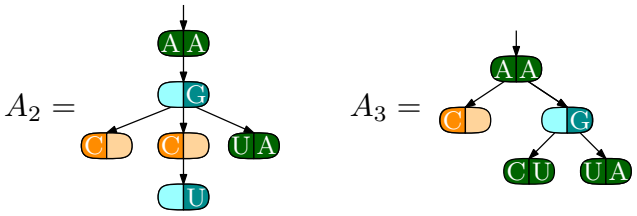

Fig. 2. Two non-equivalent supertrees, representing two different tree alignments. However, the supertree A_1 from Fig. 1 and the supertree A_2 are equivalent.

A *tree alignment* is then defined as an equivalence class over supertrees with respect to the above-defined equivalence relation, for which $\pi_1(A)$ and $\pi_2(A)$ are trees. The notion of *forest alignment* is similarly defined when $\pi_1(A)$ and $\pi_2(A)$ are not restricted to trees. Given a set $\mathcal{S}$ of tree (resp. forest) alignments, a set $\mathcal{T}$ of supertrees (resp. superforests) is said to be *representative of* $\mathcal{S}$ if it contains exactly one supertree (resp. superforest) for each alignment (*i.e.* equivalence classes of supertrees and forests) in $\mathcal{S}$. Tree alignments will now be the focus of our work.

3 A grammar for tree alignments

In this section, we describe a context-free grammar for a set $\mathcal{A}$ of supertrees that is representative of the set of all tree alignments.

We first define some basic operations on supertrees and superforests:

- The (ordered) concatenation of two (super)forests A and B is denoted by $A \circ B$. It creates a new superforest beginning by the supertrees of A, and ending by the supertrees of B.
- Given two disjoint sets $\mathcal{T}_1$ and $\mathcal{T}_2$ of supertrees or superforests, we denote by $\mathcal{T}_1 \oplus \mathcal{T}_2$ their (disjoint) union.
- For any superforest A and $a, b \in \Sigma$, $\mathrm{InsRoot}\,(A, a)$ (resp. $\mathrm{DelRoot}\,(A, b)$, $\mathrm{MatchRoot}\,(A, a, b)$) denotes the supertree whose root is the vertex $(a, -)$ (resp. $(-, b)$, (a, b)) and whose children are the supertrees in A, ordered with the same order that they have in A.

$$\mathcal{A} = \mathcal{V}^{\varnothing} \oplus \mathcal{T}_I \oplus \mathcal{T}_D \oplus \mathrm{InsRoot}\left(\mathcal{F}_I \circ \mathcal{T}_D\right) \tag{1}$$

$$\mathcal{T}_I = \mathrm{InsRoot}\left(\mathcal{F}_I\right), \quad \mathcal{F}_I = \{\text{empty superforest}\} \oplus \mathrm{InsRoot}\left(\mathcal{F}_I\right) \circ \mathcal{F}_I \tag{2}$$

$$\mathcal{T}_D = \mathrm{InsRoot}\left(\mathcal{F}_D\right), \quad \mathcal{F}_D = \{\text{empty superforest}\} \oplus \mathrm{InsRoot}\left(\mathcal{F}_D\right) \circ \mathcal{F}_D \tag{3}$$

$$\mathcal{V}^{\varnothing} = \mathcal{V}^{\uparrow} \oplus \mathrm{InsRoot}\left(\mathcal{V}\mathcal{H}\right) \tag{4}$$

$$\mathcal{V}^{\uparrow} = \mathrm{MatchRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}\right) \oplus \mathrm{DelRoot}\left(\mathcal{F}_D \circ \mathcal{V}^{\uparrow} \circ \mathcal{F}_D\right) \tag{5}$$

$$\mathcal{V}\mathcal{H} = \mathcal{F}_I \circ \mathcal{V}\mathcal{H} \oplus \mathcal{V}^{\varnothing} \circ \mathcal{F}_I \oplus \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{I|D},\leftrightarrow,\varnothing}\right) \circ \mathcal{F}_I \tag{6}$$

For every ν, M, M' with $\nu \in \{\mathsf{I|D}, \mathsf{D}\}$ and $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$:

$$\mathcal{H}_{\nu,M,M'} = \bigoplus \begin{cases} \{\text{empty superforest}\} & \text{if } (M,M') = (\varnothing,\varnothing) \\ \mathcal{T}_I \circ \mathcal{H}_{\nu,M,M'} & \text{if } \nu \neq \mathsf{D} \text{ and if } M \neq \leftrightarrow \\ \mathcal{T}_D \circ \mathcal{H}_{\mathsf{D},M,M'} & \text{if } M' \neq \leftrightarrow \\ \mathcal{V}^{\varnothing} \circ \overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{M,M'} & \\ \mathrm{InsRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\leftrightarrow}\right) \circ \overline{\mathcal{H}}^{\mathbf{1},+}_{M,M'} & \\ \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}\right) \circ \overline{\mathcal{H}}^{+,\mathbf{1}}_{M,M'} & \end{cases} \tag{7}$$

For every $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$ and $i, j \in \{\mathbf{1}, +\}$:

$$\overline{\mathcal{H}}^{i,j}_{M,M'} = \mathcal{H}_{\mathsf{I|D},\alpha(M),\alpha(M')} \oplus \begin{cases} \mathcal{F}_I & \text{if } M = \varnothing \text{ and } M' = \rightarrow \\ \mathcal{F}_I & \text{if } M = \varnothing, M' = \leftrightarrow \text{ and } j = + \\ \mathcal{F}_D & \text{if } M = \rightarrow \text{ and } M' = \varnothing \\ \mathcal{F}_D & \text{if } M = \leftrightarrow,\ M' = \varnothing \text{ and } i = + \\ \varnothing & \text{otherwise} \end{cases} \tag{8}$$

where $\alpha(\varnothing) = \varnothing$ and $\alpha(\leftrightarrow) = \alpha(\rightarrow) = \rightarrow$.

Fig. 3. A context-free grammar for $\mathcal{A}$, a representative set of all tree alignments.

– We naturally extend these operators to a set $\mathcal{T}$ of supertrees or superforests:
$\mathrm{InsRoot}\left(\mathcal{T}\right) = \bigoplus_{A \in \mathcal{T}, a \in \Sigma} \mathrm{InsRoot}\left(A, a\right)$, $\mathrm{DelRoot}\left(\mathcal{T}\right) = \bigoplus_{A \in \mathcal{T}, a \in \Sigma} \mathrm{DelRoot}\left(A, a\right)$,
$\mathrm{MatchRoot}\left(\mathcal{T}\right) = \bigoplus_{A \in \mathcal{T}, (a,b) \in \Sigma^2} \mathrm{MatchRoot}\left(A, a, b\right)$.

Our grammar is described in Fig. 3, and illustrated in Fig. 4.

Theorem 1. *The set of supertrees $\mathcal{A}$ generated by the grammar* (1)-(8) *is representative of the set of all tree alignments; i.e. $\mathcal{A}$ contains exactly one supertree for each equivalence class of supertrees.*

The key ingredient to prove Theorem 1 stems from the following (semantic) properties for the classes of supertrees and forests that appear in the grammar:

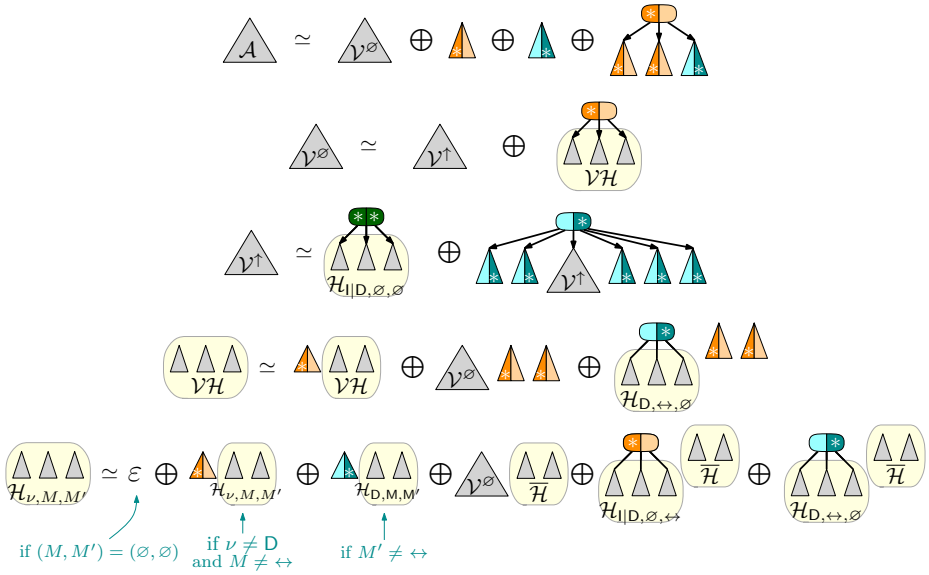

Fig. 4. A schematic illustration of the grammar for tree alignments.

1. Supertrees in $\mathcal{T}_I$ (resp. $\mathcal{T}_D$) contain only insertion (resp. deletion) vertices.
2. $\mathcal{F}_I$ (resp. $\mathcal{F}_D$) is the set of superforests formed by supertrees of $\mathcal{T}_I$ (resp. $\mathcal{T}_D$).
3. For $\mu \in \{\varnothing, \uparrow\}$, $\mathcal{V}^\mu$ is representative of the set of alignments A with at least one match, such that, if $\mu = \uparrow$, then the root of $\pi_1(A)$ is matched.
4. $\mathcal{VH}$ is representative of the set of forest alignments A with at least one match, such that $\pi_2(A)$ is a tree.
5. For $\nu \in \{\mathsf{I}|\mathsf{D}, \mathsf{D}\}$ and $(M, M') \in \{\varnothing, \leftrightarrow, \rightarrow\}^2$, $\mathcal{H}_{\nu,M,M'}$ is representative of the set of superforests A such that
   - if $\pi_1(A) \neq \varnothing$ and $\nu = \mathsf{D}$, then the first tree of $\pi_1(A)$ is matched in A;
   - if $M = \rightarrow$, then the last tree of $\pi_1(A)$ is matched in A (so $\pi_1(A) \neq \varnothing$);
   - if $M' = \rightarrow$, then the last tree of $\pi_2(A)$ is matched in A (so $\pi_2(A) \neq \varnothing$);
   - if $M = \leftrightarrow$, then the first and last trees in $\pi_1(A)$ are matched in A (so $\pi_1(A)$ has at least two trees);
   - if $M' = \leftrightarrow$, then the first and last trees in $\pi_2(A)$ are matched in A (so $\pi_2(A)$ has at least two trees).
6. For $i, j \in \{\mathbf{1}, +\}^2$, $\overline{\mathcal{H}}^{i,j}_{M,M'}$ is representative of superforests A' such that
   - there exists a superforest A such that $A \circ A' \in \mathcal{H}_{\mathsf{D},M,M'}$;
   - if $i = \mathbf{1}$ (resp. $+$), $\pi_1(A)$ is a tree (resp. a forest with at least two trees);
   - if $j = \mathbf{1}$ (resp. $+$), $\pi_2(A)$ is a tree (resp. a forest with at least two trees).

These properties can be verified recursively through a tedious analysis of the grammar, and imply quite straightforwardly that $\mathcal{A}$ contains one and exactly one supertree per equivalence class of supertrees.

Remark 1 *For sequences alignments, a grammar generating a representative set of sequence alignments can be easily adapted from the grammar generating all sequences over Σ_a, e.g. by preventing any occurrence to immediately precede an insertion. In the case of trees, the two-dimensional nature of the objects seems to forbid such a simple characterization, and seem to intrinsically mandate intricate combinatorial constructs/grammars. Note however, that our grammar, while complex, remains amenable to efficient computations (Section 4).*

4 Applications

4.1 Enumerating tree alignments

For the sake of simplicity, we will restrict our attention to $|\Sigma| = 1$, *i.e.* the alphabet is restricted to a single letter. The general case follows easily, and will be described in an extended version of the paper.

For a family $\mathcal{F}$ of superforests, we define a bivariate ordinary generating function

$$F(t,z) = \sum_{n\geq 0,\, k\geq 0} f_{n,k}\, t^n\, z^k$$

where $f_{n,k}$ is the number of superforests in $\mathcal{F}$ of size n with k matches.

Using the *symbolic method* [5], one classically translates the specification described by Eqs. (1)-(8) into a system of functional equations relating the generating functions of the sets of supertrees and forests. To that purpose, classes of objects are replaced by their generating function, disjoint unions (resp. concatenations) of two sets of supertrees are replaced by additions (resp. multiplications) of their generating functions, the addition of a root translates into a multiplication by a monomial tz (resp. t) if the root represents a match (resp. insertion/deletion), and empty superforests and sets translate into 1 and 0 respectively. The grammar is context-free, so the resulting system is algebraic and can be solved to yield the following characterization result.

Theorem 2. *The generating functions $T(t,z)$ and $F(t,z)$ of tree and forest alignments, whose size and number of matches are marked by t and z respectively, satisfy*

$$T(t,z) = \left(t^2 + t - t^2 z + \frac{t}{\sqrt{1-4t}}\right) F(t,z), \tag{9}$$

$$(tzC(t)^2 - t^2C(t)^2 + 2t)F(t,z)^2 + (t^2C(t)^4 - 2tC(t)^2 - 1)F(t,z) + C(t)^2 = 0, \tag{10}$$

where $C(t) = (1-\sqrt{1-4t})/2t$ is the generating function of Catalan numbers.

Solving the quadratic equation (10) leads to an explicit formula for FA (and hence TA), details of which are omitted due to space constraints. Nonetheless, these explicit expressions can be used to compute an asymptotic estimate using a *transfer theorem* [5, Cor. VI.1 p. 392].

Theorem 3. *The number of tree alignments of size n is asymptotically equivalent to $\kappa \times n^{-3/2} \times 6^n$, where $\kappa = \sqrt{2}(3-\sqrt{3})/(24\sqrt{\pi})$.*

Corollary 1 *The average number of tree alignments for a random pair of trees of cumulated size n is $\kappa' \times 1.5^n$, where $\kappa' = \sqrt{2}(3-\sqrt{3})/6$.*

Similar techniques can be used to characterize the distribution of the number of matches in a random tree alignment. A direct application of [5, Theorem IX.12 p. 676] indeed gives the following.

Proposition 2 *Let m_n be the random variable that counts the number of matches in a uniformly-drawn random tree alignment. The variable m_n follows a Normal law of mean $\mathbb{E}(m_n) \sim n/6$ and variance $\mathbb{V}(m_n) \sim n/6$.*

4.2 Sampling alignments between two given trees

We now consider two *fixed* trees S and T, and consider the task of sampling a tree alignment A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, with respect to the Gibbs-Boltzmann probability distribution. This can be used to assess the stability of a prediction. We refer the interested reader to our introduction for examples of further motivation and possible applications.

Preliminaries. Let $\mathcal{T}_{S,T}$ be the set of all supertrees A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, and $\mathcal{A}_{S,T}$ be a representative set of $\mathcal{T}_{S,T}$. In other words, $\mathcal{A}_{S,T}$ can be interpreted as the set of all alignments between S and T. For any supertree $A \in \mathcal{T}_{S,T}$, we define its *edit score* $s(A)$ as the sum of the number of insertions, deletions and matches (x, y) such that $x \neq y$.[5]

For a given positive constant $k\theta$, the *partition function* $Z_{S,T}$ of $\mathcal{A}_{S,T}$ and the *Gibbs-Boltzmann probability* $\Pr(A)$ of an alignment $A \in \mathcal{A}_{S,T}$ are defined as

$$Z_{S,T} = \sum_{A \in \mathcal{A}_{S,T}} e^{-s(A)/k\theta}, \quad \Pr(A) = \frac{e^{-s(A)/k\theta}}{Z_{S,T}}.$$

When $k\theta$ tends to 0, this distribution tends to the uniform distribution over supertrees of minimum edit score, while, when $k\theta$ tends to $+\infty$, it tends toward the uniform distribution over $\mathcal{A}_{S,T}$.

We consider the following problem: given two trees S and T, and a positive constant $k\theta$, design a sampling algorithm for alignments between S and T under the Gibbs-Boltzmann probability distribution. This problem generalizes the classic combinatorial optimization problem of computing a tree alignment between S and T having minimum edit score.

[5] The present results can be trivially extended to any edit scoring system that is a positive linear combination of the numbers of insertions, deletions and matches.

To address this problem, we rely on dynamic programming, by the approach described, among others, in [10] for RNA folding. We begin by adapting the grammar introduced in Section 3 into a grammar for $\mathcal{A}_{S,T}$, then detail how this grammar leads to an efficient sampling algorithm.

A grammar for $\mathcal{A}_{S,T}$. In order to guarantee that each supertree A indeed aligns two input trees S and T (namely $\pi_1(A) = S$ and $\pi_2(A) = T$), we need to restrict which rules in the grammar can be used, conditionally to which trees and forests are currently being generated. To that purpose, we introduce, for each set $\mathcal{S}$ in the previous grammar, an indexed version $\mathcal{S}_{[u,v]}$ which denotes the restriction of $\mathcal{S}$ to alignments between u and v two forests in S and T.

Slightly abusing previous notations, we denote by $a(u)$ the tree whose root is a vertex a and whose (forest of) children is u. Finally, for every tree/forest X, $\mathrm{Ins}(X)$ (resp. $\mathrm{Del}(X)$) represents the supertree/superforest obtained from X by inserting (resp. deleting) each of its elements. If X is empty, $\mathrm{Ins}(X)$ and $\mathrm{Del}(X)$ denote the empty superforest. The grammar for $\mathcal{A}_{S,T}$ is described in Fig. 5.

Theorem 4. *Let S and T be non-empty trees. The set of supertrees $\mathcal{A}_{S,T}$ generated by grammar* (11)-(18) *is representative of $\mathcal{T}_{S,T}$ the tree alignments between S and T.*

Applications to dynamic programming. The grammar defined by Equations (11)-(18) is a decomposition scheme for the alignments between S and T. It can easily be transformed into an algorithm for computing the partition function $Z_{S,T}$. Indeed, $Z_{S,T}$ is simply a weighted sum over all possible supertrees of $\mathcal{A}_{S,T}$, which is a set generated by the grammar. Now consider the image of the grammar as a set of numerical equations, obtained by syntactically replacing:

- The operators $(\oplus, \circ)$ with $(\sum, \times)$ respectively;
- The empty set $\varnothing$ with 0;
- Inserted/Deleted trees/forests $\mathrm{Ins}(X)$ and $\mathrm{Del}(X)$ with $e^{-|X|/k\theta}$,
- Match $\mathrm{MatchRoot}\,(V, a, a)$ events with V, $\forall a \in \Sigma$ and any expression V;
- Insertion $\mathrm{InsRoot}\,(V, a)$ events, deletion $\mathrm{DelRoot}\,(V, a)$ events, and mismatch $\mathrm{MatchRoot}\,(V, a, b)$ events with $e^{-1/k\theta} \times V$, $\forall a \neq b \in \Sigma$ and any V.

Theorem 4 immediately implies that the resulting set is a dynamic programming scheme that computes $Z_{S,T}$ instead of $\mathcal{A}_{S,T}$.

Moreover, each non-terminal term of the modified grammar now contains the partition function of the set of supertrees associated to this non-terminal term in the set-theoretic grammar, *e.g.* a term $\mathcal{VH}[a(u) \circ X, b(v)]$. This information can then be used to define an algorithm to sample supertrees from $\mathcal{A}_{S,T}$ under the Gibbs-Boltzmann distribution, following the *recursive* method for random generation [15].

To do so, it suffices to reinterpret the grammar defined by Equations (11)-(18) as a branching process: each $\oplus$ operator is replaced by a branching operator

$$\mathcal{A}_{\substack{S,T \\ S\equiv r_S(X_S)}} = \mathcal{V}^{\varnothing}[S,T] \oplus \mathrm{InsRoot}\left(\mathrm{Ins}(X_S)\circ\mathrm{Del}(T), r_S\right) \tag{11}$$

$$\mathcal{V}^{\varnothing}[a(u),b(v)] = \mathcal{V}^{\uparrow}[a(u),b(v)] \oplus \mathrm{InsRoot}\left(\mathcal{VH}[u,b(v)],a\right) \tag{12}$$

$$\mathcal{V}^{\uparrow}[a(u),b(v)] = \bigoplus \begin{cases} \mathrm{MatchRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}[u,v],a,b\right) \\ \bigoplus\limits_{Y\circ c(w)\circ Y'=v} \mathrm{DelRoot}\left(\mathrm{Del}(Y)\circ\mathcal{V}^{\uparrow}[a(u),c(w)]\circ\mathrm{Del}(Y'),b\right) \end{cases} \tag{13}$$

$$\mathcal{VH}[\varnothing,b(v)] = \varnothing \tag{14}$$

$$\mathcal{VH}[a(u)\circ X,b(v)] = \bigoplus \begin{cases} \mathrm{Ins}(a(u))\circ\mathcal{VH}[X,b(v)] \\ \bigoplus\limits_{\substack{X'\circ X''=a(u)\circ X \\ |X'|\geq 2}} \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{I|D},\leftrightarrow,\varnothing}[X',v],b\right)\circ\mathrm{Ins}(X'') \\ \mathcal{V}^{\varnothing}[a(u),b(v)]\circ\mathrm{Ins}(X) \end{cases} \tag{15}$$

For every ν, M, M' with $\nu \in \{\mathsf{I|D},\mathsf{D}\}$ and $M, M' \in \{\varnothing,\leftrightarrow,\rightarrow\}$:

$$\mathcal{H}_{\nu,M,M'}[X,\varnothing] = \begin{cases} \mathrm{Ins}(X) & \text{if } (M,M') = (\varnothing,\varnothing), \\ \varnothing & \text{otherwise,} \end{cases} \tag{16}$$

$$\mathcal{H}_{\nu,M,M'}[\varnothing,Y] = \begin{cases} \mathrm{Del}(Y) & \text{if } (M,M') = (\varnothing,\varnothing), \\ \varnothing & \text{otherwise,} \end{cases} \tag{17}$$

$$\mathcal{H}_{\nu,M,M'}[a(u)\circ X,b(v)\circ Y] =$$

$$\bigoplus \begin{cases} \mathrm{Ins}(a(u))\circ\mathcal{H}_{\nu,M,M'}[X,b(v)\circ Y] & \text{if } \nu \neq \mathsf{D} \text{ and if } M \neq\leftrightarrow, \\ \mathrm{Del}(b(v))\circ\mathcal{H}_{\mathsf{D},M,M'}[a(u)\circ X,Y] & \text{if } M' \neq \leftrightarrow, \\ \mathcal{V}^{\varnothing}[a(u),b(v)]\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X),\alpha(M',Y)}[X,Y] & \\ \bigoplus\limits_{\substack{Y'\circ Y''=b(v)\circ Y \\ |Y'|\geq 2}} \mathrm{InsRoot}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\leftrightarrow}[u,Y'],a\right)\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X),\alpha(M',Y'')}[X,Y''] & \\ \bigoplus\limits_{\substack{X'\circ X''=a(u)\circ X \\ |X'|\geq 2}} \mathrm{DelRoot}\left(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}[X',v],b\right)\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X''),\alpha(M',Y)}[X'',Y] & \end{cases} \tag{18}$$

where $\alpha(\varnothing,X) = \varnothing$ and $\alpha(\leftrightarrow,X) = \alpha(\rightarrow,X) = \begin{cases} \varnothing & \text{if } X = \varnothing, \\ \rightarrow & \text{otherwise.} \end{cases}$

Fig. 5. A grammar for $\mathcal{A}_{S,T}$, a representative set of all tree alignments between two fixed trees S and T.

that, instead of joining sets of supertrees into a larger set of supertrees, chooses one of the sets according to the weight of its partition function. For instance, assume we have a grammar rule $U = V \oplus W$: the sampling algorithm will select one of the sets V, W, with V being chosen with probability $Z_V/(Z_V + Z_W)$, and W with probability $Z_W/(Z_V + Z_W)$, provided that $Z_V,\ Z_W$ and Z_X have been previously computed. Recursive calls will then result into a supertree, which is provably randomly generated under the Gibbs-Boltzmann distribution.

Theorem 5. *Let S and T be two trees of respective sizes n_S and n_T. The above-defined branching process adapted from grammar* (11)-(18) *defines an algorithm that samples a supertree from $\mathcal{A}_{S,T}$ under the Gibbs-Boltzmann distribution. The worst-case time and space complexities of the algorithm are in $\mathcal{O}(n_S\, n_T\, (n_S + n_T)^2)$, while the average-case time and space complexities are in $\mathcal{O}(n_S\, n_T)$.*

The correctness of the algorithm immediately follows from Theorem 4. Its complexities are identical to [9,6] since the structure of the DP scheme essentially remains the same; only the number of DP tables is increased (by a constant factor). This implies that our algorithm, while solving a much more general problem, retains the same asymptotic complexity (up to constants) than the current tree alignment algorithms that are limited to computing a single optimal tree alignment.

5 Conclusion and discussion

Following a classical line of research in string algorithms, we introduced the notion of equivalence for tree alignments, and described a context-free grammar for a representative set of all possible alignments. We also showed how this grammar can be used to derive asymptotic properties of alignments, and design an efficient dynamic programming sampling algorithm for alignments between two given trees.

Our proposed grammar for tree alignments is much more complex than the grammars used to generate a representative set of sequence alignments, although dynamic programming for computing optimal sequences and trees alignments are very similar. This is due to the fact that it is particularly hard to characterize a representative set of tree alignments (see Remark 1). It thus remains an open problem to design a representative set of tree alignment that would be amenable to enumeration using a simpler grammar. However, it is important to remark that, despite its apparent complexity, our grammar leads to algorithms with an asymptotic complexity of the same order than existing optimization algorithms.

From a theoretical point of view, we believe that tree alignments as defined in this work form an interesting combinatorial family whose properties deserve to be explored in depth. In terms of future developments, our work was largely motivated by applications to RNA bioinformatics. Following the program outlined in [10], we are currently using our grammar and derived dynamic programming schemes to revisit the alignment of 3D models of RNA structures. More generally, it would be interesting to characterize the conditions under which an instance-agnostic grammar, enumerating a search space, could be adapted into a decomposition for a specific instance. Such a theory, at the confluence of enumerative combinatorics and algorithmic design, could provide another principled ways to design dynamic-programming algorithms.

References

1. Andrade, H., Area, I., Nieto, J.J., Torres, A.: The number of reduced alignments between two dna sequences. BMC Bioinformatics 15, 94 (2014), http://dx.doi.org/10.1186/1471-2105-15-94
2. Blin, G., Denise, A., Dulucq, S., Herrbach, C., Touzet, H.: Alignments of RNA structures. IEEE/ACM Trans. Comput. Biology Bioinform. 7(2), 309–322 (2010), http://doi.acm.org/10.1145/1791396.1791409
3. Do, C., Gross, S., Batzoglou, S.: Contralign: Discriminative training for protein sequence alignment. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P., Waterman, M. (eds.) Research in Computational Molecular Biology, Lecture Notes in Computer Science, vol. 3909, pp. 160–174. Springer Berlin Heidelberg (2006)
4. Dress, A., Morgenstern, B., Stoye, J.: The number of standard and of effective multiple alignments. Applied Mathematics Letters 11(4), 43 – 49 (1998), http://www.sciencedirect.com/science/article/pii/S0893965998000548
5. Flajolet, P., Sedgewick, R.: Analytic combinatorics. Cambridge University Press, Cambridge (2009)
6. 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(26-28), 2423–2432 (2010), http://dx.doi.org/10.1016/j.tcs.2010.01.014
7. Höchsmann, M., Töller, T., Giegerich, R., Kurtz, S.: Local similarity in rna secondary structures. Proc IEEE Comput Soc Bioinform Conf 2, 159–168 (2003)
8. Höchsmann, M., Voss, B., Giegerich, R.: Pure multiple rna secondary structure alignments: A progressive profile approach. IEEE/ACM Trans. Comput. Biol. Bioinformatics 1(1), 53–62 (Jan 2004), http://dx.doi.org/10.1109/TCBB.2004.11
9. Jiang, T., Wang, L., Zhang, K.: Alignment of trees - an alternative to tree edit. Theor. Comput. Sci. 143(1), 137–148 (1995), http://dx.doi.org/10.1016/0304-3975(95)80029-9
10. Ponty, Y., Saule, C.: A combinatorial framework for designing (pseudoknotted) RNA algorithms. In: Przytycka, T.M., Sagot, M. (eds.) Algorithms in Bioinformatics - 11th International Workshop, WABI 2011, Saarbrücken, Germany, September 5-7, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6833, pp. 250–269. Springer (2011), http://dx.doi.org/10.1007/978-3-642-23038-7_22
11. Schirmer, S., Giegerich, R.: Forest alignment with affine gaps and anchors, applied in RNA structure comparison. Theor. Comput. Sci. 483, 51–67 (2013), http://dx.doi.org/10.1016/j.tcs.2012.07.040
12. Torres, A., Cabada, A., Nieto, J.J.: An exact formula for the number of alignments between two dna sequences. DNA Seq 14(6), 427–430 (Dec 2003)
13. Vingron, M., Argos, P.: Determination of reliable regions in protein sequence alignments. Protein Engineering 3(7), 565–569 (1990), http://peds.oxfordjournals.org/content/3/7/565.abstract
14. Waterman, M.S.: Introduction to Computational Biology: Maps, Sequences, and Genomes. CRC Press (1995)
15. Wilf, H.S.: A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. Advances in Mathematics 24, 281–291 (1977)
16. Yang, W.: Identifying syntactic differences between two programs. Softw. Pract. Exper. 21(7), 739–755 (Jun 1991), http://dx.doi.org/10.1002/spe.4380210706
17. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. In: Proceedings of the 14th International Conference on World Wide Web. pp. 76–85. WWW '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1060745.1060761