ISMB, 2022, pp. 1–9

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

LinearSankoff: Linear-time Simultaneous Folding and Alignment of RNA Homologs

Sizhen Li,[1] Ning Dai,[1] He Zhang,[1,2] Apoorv Malik,[1] David H. Mathews[3,4,5] and Liang Huang[1,*]

[1]School of Electrical Engineering & Computer Science, Oregon State University, Corvallis, 97330, OR, USA, [2]Baidu Research USA, Sunnyvale, 94089, CA, USA, [3]Dept. of Biochemistry & Biophysics, [4]Center for RNA Biology and [5]Dept. of Biostatistics & Computational Biology, University of Rochester Medical Center, Rochester, 14642, NY, 14642
[*]Corresponding author. liang.huang.sh@gmail.com

Abstract

The classical Sankoff algorithm for the simultaneous folding and alignment of homologous RNA sequences is highly influential, but it suffers from two major limitations in efficiency and modeling power. First, it takes $O(n^6)$ for two sequences where n is the average sequence length. Most implementations and variations reduce the runtime to $O(n^3)$ by restricting the alignment search space, but this is still too slow for long sequences such as full-length viral genomes. On the other hand, the Sankoff algorithm and all its existing implementations use a rather simplistic alignment model, which can result in poor alignment accuracy. To address these problems, we propose LinearSankoff, which seamlessly integrates the original Sankoff algorithm with a powerful Hidden Markov Model-based alignment module. This extension substantially improves alignment quality, which in turn benefits secondary structure prediction quality, confirmed over a diverse set of RNA families. LinearSankoff also applies beam search heuristics and the A* algorithm to achieve that runtime scales linearly with sequence length. LinearSankoff is the first linear-time algorithm for simultaneous folding and alignment, and the first such algorithm to scale to coronavirus genomes ($n \simeq 30,000nt$). It only takes 10 minutes for a pair of SARS-CoV-2 and SARS-related genomes, and outperforms previous work at identifying crucial conserved structures between the two genomes.

Key words: simultaneously folding and alignment, conserved secondary structure, structural alignment
Availability: Code: `https://github.com/LinearFold/LinearSankoff`; Web Server: `http://www.linearfold.org/linearsankoff`

Introduction

Many RNAs are involved in multiple cellular processes [1, 2], whose functions highly rely on their conserved structures. Therefore, there is a need to develop fast and accurate algorithms for conserved structure prediction over RNA homologs.

To automate comparative analysis, Sankoff [3] pioneered an algorithm to simultaneously fold and align homologous sequences. However, it takes $O(n^6)$ time for just two sequences with the average sequence length n, and $O(n^{3k})$ time for k sequences in general. To make it feasible, several implementations of the Sankoff algorithm [4, 5, 6, 7, 8, 9, 10, 11]. reduce the runtime to $O(n^3m^3)$ via banding the alignment region with a fixed width (m), which shrinks the alignment search space from $O(n^2)$ to $O(nm)$. Some of these tools, including PARTS [11], LocARNA [9] and SCARNA [12], also simplified the energy model using base-pairing probabilities. However, this cubic runtime is still intractable for long sequences such as full-length viral genomes.

Besides the intractability, there is yet another important limitation in the Sankoff framework and its implementations, where the alignment module is overly simplistic which scores matches, mismatches, and gaps independently of each other (e.g., using the classical Needleman–Wunsch [13] alignment). For example, the original Sankoff algorithm and Dynalign [5] only include gap penalty, while LocARNA and FoldAlign also include mismatch matrices. By contrast, the Hidden Markov Model (HMM) has been well studied and applied to align RNA sequences [14, 15] which score each alignment step (match, mismatch, or gap) depending on the previous step, therefore extending a gap is treated differently from starting a gap.

To address these problems, we propose LinearSankoff, which extends the original Sankoff algorithm (with full energy model) by incorporating a more powerful, HMM-based alignment model. This integration between Sankoff and HMM requires non-trivial generalizations to the original Sankoff-style dynamic programming algorithm. As a result, all existing variants of Sankoff are simplified versions of LinearSankoff in terms of either or both folding and alignment models. To make

language	RNA
single-sentence parsing	single-sequence folding
CKY $O(n^3)$	Nussinov/Zuker $O(n^3)$
context-free grammar (CFG)	CFG
synchronous parsing	homologous folding
(joint parsing & alignment)	(joint folding & alignment)
Wu [18] $O(n^6)$	Sankoff [3] $O(n^6)$
synchronous CFG	synchronous CFG

Table 1. Correspondence between natural language parsing and RNA folding. While the correspondence between single-sentence parsing and singe-sequence folding is well-known, our work is the first to establish the connection between synchronous parsing and homologous folding. This leads to our borrowing of synchronous context-free grammar from the former to the latter.

it efficient, we generalize the beam pruning technique of LinearFold [16] from single-sequence folding to homologous folding to make the LinearSankoff runtime scale linearly with the sum of sequence lengths. More interestingly, LinearSankoff also applies the A* algorithm with admissible heuristics [17] together with beam pruning to further speed up the search.

We make the following contributions:

- We provide the first rigorous formulation of simultaneous folding and alignment using synchronous context free grammars borrowed from computational linguistics.
- We integrate Sankoff with an HMM-based alignment model, which not only improves alignment quality but also in turn benefits folding quality, and generalize the Sankoff-style dynamic programming to keep track of HMM states.
- We extend the beam search heuristic from single-sequence folding to joint folding to achieve linear runtime, and further apply the A* algorithm to speed up the search.
- Overall, LinearSankoff achieves higher secondary structure prediction and alignment accuracies than three baseline models (LinearFold, Dynalign and LinearTurboFold).

Formulation and Modeling

Synchronous Context Free Grammar Formulation

While there have been many variants of the Sankoff algorithm in the literature [3, 4, 6, 7, 8, 9], there has not been a formal definition of joint folding and alignment. Therefore there is a need to develop such a formulation to provide mathematical rigor to this important area. Luckily, in the sister field of computational linguistics, there is a very similar problem "synchronous parsing", which jointly parses and aligns a sentence pair from two languages such as English and Chinese [18]; it basically extends single-sentence parsing to two sentences, just like homologous folding extends single-sequence folding to two sequences. Synchronous parsing is rigorously formulated by synchronous context-free grammars [19, 20, 21], which extend the well-known context-free grammars from one language to two languages. So we naturally borrow this concept to formulate homologous folding. See Table 1 for the correspondence between language parsing and RNA folding. Below we start with a quick review of context-free grammars for RNA folding.

For one RNA sequence $\mathbf{x} = x_1 x_2 \ldots x_n$ with each $x_i \in \{\text{A}, \text{U}, \text{C}, \text{G}\}$, the minimum free energy change (MFE) [22] structure $\hat{\mathbf{s}}$ is the best-scoring structure among all possible structures $\mathcal{S}(\mathbf{x})$:

$$\hat{\mathbf{s}} = \operatorname*{\mathbf{argmin}}_{\mathbf{s} \in \mathcal{S}(\mathbf{x})} \Delta G^\circ(\mathbf{x}, \mathbf{s}) \tag{1}$$

where $\Delta G^\circ(\mathbf{x}, \mathbf{s})$ is the free energy of the structure $\mathbf{s}$ for the sequence $\mathbf{x}$. The classical solution for finding the pseudoknot-free MFE structure is the $O(n^3)$-time dynamic programming algorithm [23, 24], whose search space is usually formulated by a context free grammar (CFG). Formally, a CFG is 4-tuple $G = \langle V, \Sigma, R, S \rangle$, where V is the set of nonterminals, Σ is the set of terminals ($\Sigma = \{\text{A}, \text{U}, \text{C}, \text{G}\}$), R is the set of production rules, and $S \in V$ is the start symbol. Each rule $r \in R$ has the form $A \to \alpha$, where $A \in V$ is rewritten into $\alpha \in (V \cup \Sigma)^*$ where * denotes zero or more repetitions.

As an example, Fig. 1A shows a CFG corresponding to the Nussinov algorithm [23]. These nonterminals represent structural components: S for an arbitrary span, P for a span with two ends paired, and N for an unpaired nucleotide. As a shorthand notation, we use a to represent a nucleotide, and aa' to represent a base pair. An RNA sequence $\mathbf{x}$ can be derived from the grammar G by applying a series of production rules ($S \overset{*}{\Rightarrow}_G \mathbf{x}$). Each derivation also implies a RNA secondary structure. Fig. 1B shows a derivation of G for sequence AACAAG along with the secondary structure "..(..)" in the dot-bracket format (in gray shades, where "." represents an unpaired position, and "(" and ")" indicate paired positions).

Now we extend this framework to handle two sequence folding, by extending CFG to synchronous CFG (SCFG) [21]. An SCFG G' is still a 4-tuple $\langle V, \Sigma', R', S \rangle$, where V and S remain unchanged, and the new terminal set $\Sigma' = \{\text{A}, \text{U}, \text{G}, \text{C}, -\}$ includes a gap symbol ($-$) for alignment. Each *synchronous production rule* in R' (see Fig. 1C) now has two parts on the right hand side to capture two sequences:

$$A \to \alpha,\ \beta$$

where $A \in V$ and $\alpha, \beta \in (V \cup \Sigma)^*$. For example, $S \to S\ P$ is extended to $S \to S\ P,\ S\ P$. Note that there is a one-to-one correspondence between the nonterminals in α and the nonterminals in β.

Although the grammar G' requires each nonterminal (structural component) in one structure to correspond to another nonterminal in the other structure, it allows some variation on the structures of two sequences to some extent by inserting base pairs and unpaired nucleotides. Specifically, the rule

$$P \to a\ S\ a',\ b\ S\ b'$$

indicates that the based pairs (a, a') and (b, b') are aligned (a with b and a' with b'). But the rule

$$P \to a\ P\ a',\ -\ P\ -$$

represents that one base pair (a, a') is inserted in the first sequence and gaps ($-$) are added to the second sequence for alignment. Similarly,

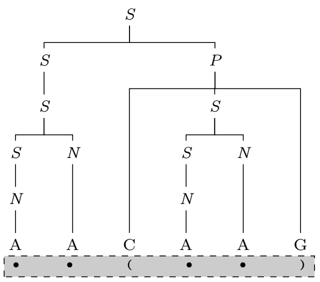
$$P \to -\ P\ -,\ b\ P\ b'$$

indicates that one base pair (b, b') is inserted in the second sequence. In addition, the productions derived from N provide flexibility on the length of the corresponding unpaired regions

A

$$\begin{aligned} S &\rightarrow S\ P \mid P \mid S\ N \mid N \\ P &\rightarrow a\ S\ a' \\ N &\rightarrow a \\ a &\in \{\text{A}, \text{U}, \text{G}, \text{C}\} \\ aa' &\in \{\text{AU}, \text{UA}, \text{GC}, \text{CG}, \text{GU}, \text{UG}\} \end{aligned}$$

B



C

$$\begin{aligned} S &\rightarrow S\ P, S\ P \\ &\rightarrow P, P \\ &\rightarrow S\ N, S\ N \\ &\rightarrow N, N \\ P &\rightarrow a\ S\ a',\ b\ S\ b' \\ &\rightarrow a\ P\ a', -\ P\ - \\ &\rightarrow -\ P\ -, b\ P\ b' \\ N &\rightarrow a,\ b \\ &\rightarrow a,\ - \\ &\rightarrow -,\ b \\ a, b &\in \{\text{A}, \text{U}, \text{G}, \text{C}\} \\ aa', bb' &\in \{\text{AU}, \text{UA}, \text{GC}, \text{CG}, \text{GU}, \text{UG}\} \end{aligned}$$

D

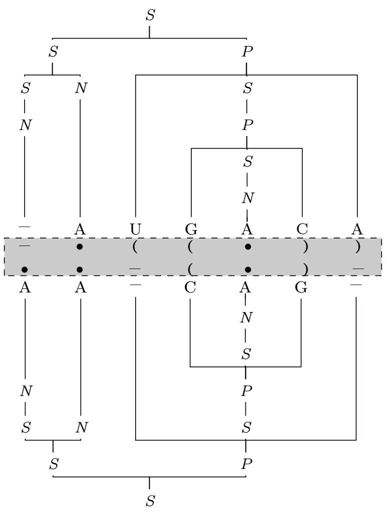


Fig. 1. (**A**) Context Free Grammar (CFG) formulation for parsing one sequence. (**B**) An example illustrates that CFG generates RNA sequence AACAAG via one possible derivation. The corresponding structure imposed by the derivation is annotated below the tree with gray background. (**C**) Synchronous Context Free Grammar (SCFG) formulation for simultaneously folding and aligning two sequences. (**D**) And example represents that SCFG yields an aligned sequence pair (–AUGACA, AA–CAG–) via one possible derivation. The original sequence pair (AUGACA, AACAG) can be obtained by removing gaps directly. The secondary structures are shown with gray background. Not that we do not consider sharpturn constraint on hairpins to simplify examples (**B** and **D**).

(N) by inserting/deleting a nucleotide in one sequence. For example,

$$N \rightarrow a, b$$

aligns two unpaired (".") nucleotides a and b from two sequences, while

$$N \rightarrow a, -$$

inserts one unpaired nucleotide a in the first sequence. Therefore, the SCFG G' folds two sequences with generally similar structures, but does not require them to be exactly the same. It allows freedom in the number of base pairs in corresponding helices, as well as the length of corresponding unpaired regions.

More formally, a derivation of SCFG, notated $S \overset{*}{\Rightarrow}_{G'} \langle \bar{\mathbf{x}}, \bar{\mathbf{y}} \rangle$, generates a pair of *aligned sequences*, along with one secondary structure for each sequence. Fig. 1D demonstrates one such derivation that generates the aligned sequence pair:

```
-.((.))
-AUGACA
AA-CAG-
..-(.)-
```

where both the sequences and structures are aligned by inserting gaps (–). The original sequences and structures can be obtained by removing gaps.

Integrating HMM-based Alignment Model

For a sequence pair $\langle \mathbf{x}, \mathbf{y} \rangle = \langle x_1 x_2 \dots x_{n_1},\ y_1 y_2 \dots y_{n_2} \rangle$ with sequence length n_1 and n_2, respectively, we denote a possible alignment $\mathbf{a}$ of two equal-length sequences with gaps, $\langle x'_1 x'_2 \dots x'_m,\ y'_1 y'_2 \dots y'_m \rangle$ with the same sequence length m ($m \geq \max(n_1, n_2)$) by inserting gaps in two sequences, thus $x'_i, y'_j \in \{\text{A}, \text{U}, \text{C}, \text{G}, -\}$. Naturally, with the same sequence length, the alignment can be treated as a sequence of pairs $\langle x'_1,\ y'_1 \rangle, \langle x'_2,\ y'_2 \rangle, \dots, \langle x'_m,\ y'_m \rangle$. We use a Hidden Markov Model to model the pairwise alignment, which consists of three hidden states: $\nearrow$, $\rightarrow$ and $\uparrow$ representing alignment of two nucleotides, inserting one nucleotide in the first sequence, and inserting one nucleotide in the second sequence, respectively. Correspondingly, the emission/observation is $\langle x'_i,\ y'_j \rangle$, $\langle x'_i, - \rangle$ and $\langle -, y'_j \rangle$, respectively, where x'_i and y'_j are nucleotides rather than gaps. The Viterbi alignment path is the most likely sequence of hidden states to generate two sequences (ignoring gaps) among all possible alignment paths $\mathcal{A}(\mathbf{x}, \mathbf{y})$:

$$\begin{aligned} \hat{\mathbf{a}} &= \underset{\mathbf{a} \in \mathcal{A}(\mathbf{x},\mathbf{y})}{\mathbf{argmin}}\ p(\mathbf{a}, \mathbf{x}, \mathbf{y}) \\ &= \underset{\mathbf{a} \in \mathcal{A}(\mathbf{x},\mathbf{y})}{\mathbf{argmin}}\ \Pi_{i=1}^{m} p_{\mathrm{t}}(h_i \mid h_{i-1}) p_{\mathrm{t}}(\langle x'_i, y'_i \rangle \mid h_i) \end{aligned} \qquad (2)$$

where $h_i \in \{\nearrow, \rightarrow, \uparrow\}$ is the hidden state, starting from $h_0 = \nearrow$, and $p_{\mathrm{t}}(h_i \mid h_{i-1})$ and $p_{\mathrm{e}}(\langle x'_i, y'_i \rangle \mid h_i)$ are the transition and the emission probabilities, respectively.

To formalize the integrated Sankoff+HMM framework, we need to explicitly generate structures and the alignment state sequence, so we further extend the 2-component SCFG G' to a 5-component SCFG $G'' = \langle V, \Sigma'', R'', S \rangle$.[1] The terminal set is extended to $\{\text{A}, \text{U}, \text{G}, \text{C}, ., (,), \nearrow, \rightarrow, \uparrow\}$, where ".", "(" and ")" represent structures, and $\nearrow$, $\rightarrow$ and $\uparrow$ represent alignment states (note that gap – is no longer needed). The production rules are further extended to have five parts on the right side:

$$A \rightarrow \alpha, \beta, \alpha', \beta', \theta$$

where $A \in V$, $\alpha, \beta \in (V \cup \{\text{A}, \text{U}, \text{G}, \text{C}\})^*$, $\alpha', \beta' \in (V \cup \{., (,)\})^*$, and $\theta \in (V \cup \{\nearrow, \rightarrow, \uparrow\})^*$. For example, we extend the "aligned pair" rule $P \rightarrow a\ P\ a',\ b\ P\ b'$ to

$$P \rightarrow a\ P\ a',\ b\ P\ b',\ (\ P\ ),\ (\ P\ ),\ \nearrow P \nearrow$$

[1] Such use of SCFG to explicitly model structures is also found in natural language, e.g., between syntax and semantics [25].

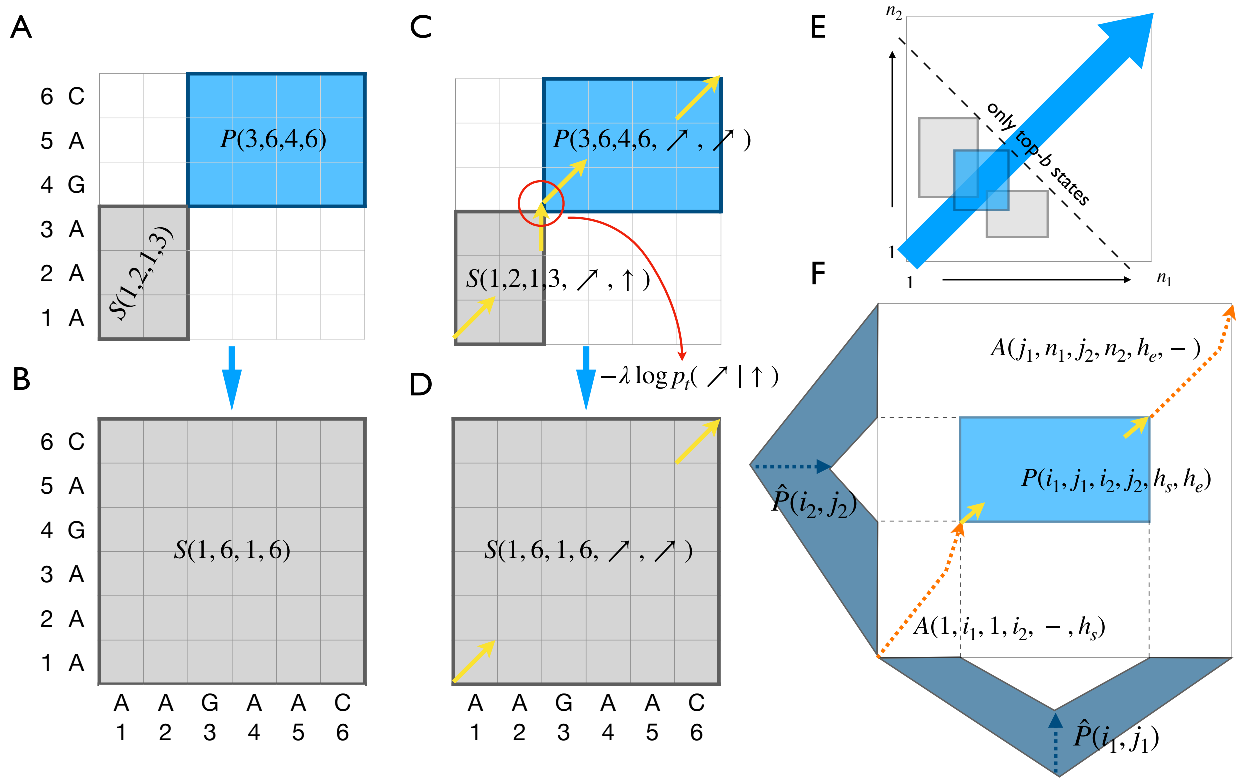


Fig. 2. (**A–B**) Deductive system of Nussinov model with a simple alignment model for simultaneously folding and alignment of two sequences. Concatenate two adjacent states $S(1,2,1,3)$ and $P(3,6,4,6)$ (**A**) to $S(1,6,1,6)$ (**B**). (**C–D**) Deductive system of Nussinov model with a HMM-based alignment model for simultaneously folding and alignment of two sequences. Two more dimensions are added in the states to indicates alignment hidden state of start and end positions. Concatenate two adjacent state $S(1,2,1,3,\nearrow,\uparrow)$ and $P(3,6,4,6,\nearrow,\nearrow)$ (**C**) to $S(1,6,1,6,\nearrow,\nearrow)$ (**D**). The solid yellow arrows show alignment hidden states of start and end positions. (**E**) LinearSankoff's computation along the diagonal (from bottom left to top right) makes it possible to further apply the beam pruning heuristic [26]. (**F**) LinearSankoff applies the A* algorithm during beam pruning to speed up searching. The admissible heuristic values includes $\hat{P}(i_1,j_1)$, $\hat{P}(i_2,j_2)$, $A(1,i_1,1,i_2,-,h_s)$ and $A(j_1,n_1,j_2,n_2,h_e,-)$. $\hat{P}(i_1,j_1)$ is the minimum free energy change of folding regions $x_1\ldots x_{i_1-1}$ and $x_{j_1+1}\ldots x_{n_1}$ of sequence $\mathbf{x}$ conditioned on (x_{i_1},x_{j_1}) forming a base pair. This score is obtained by folding $\mathbf{x}$ separately from pre-processing. A similar definition applies to $\hat{P}(i_2,j_2)$. $A(1,i_1,1,i_2,-,h_s)$ is the Viterbi alignment path of $x_1\ldots x_{i_1}$ and $y_1\ldots y_{i_2}$ with constrained alignment state h_s imposed on (x_{i_1},y_{i_2}). This probability can be computed in pre-processing. A similar idea applies to $A(j_1,n_1,j_2,n_2,h_e,-)$.

and the "pair-gap" rule $P \to a\ P\ a',\ -\ P\ -$ to

$$P \to a\ P\ a',\ P,\ (\ P\ ),\ P,\ \to P \to$$

where we remove the gaps in the sequences and structures. The "aligned unpaired" rule $N \to a, b$ becomes

$$N \to a,\ b,\ .,\ .,\ \nearrow$$

and the "unpaired-gap" rule $N \to -, b$ becomes

$$N \to \epsilon,\ b,\ \epsilon,\ .,\uparrow$$

where ϵ denotes empty string.

Now a derivation in G'' generates a 5-tuple:

$$S \overset{*}{\Rightarrow}_{G''} \langle \mathbf{x}, \mathbf{y}, \mathbf{s_x}, \mathbf{s_y}, \mathbf{a}\rangle$$

where $\mathbf{x}$ and $\mathbf{y}$ are the two input sequences (without gaps), $\mathbf{s_x}$ and $\mathbf{s_y}$ are their corresponding secondary structures, and $\mathbf{a}$ is the sequence of alignment hidden states.

Given two RNA homologous sequences $\mathbf{x}$ and $\mathbf{y}$, and a synchronous context free grammar G'', the goal of simultaneously folding and alignment of RNA sequences is to find the most likely derivation tree, i.e., secondary structures $\mathbf{s}_x$ and $\mathbf{s}_y$, to generate an alignment $\mathbf{a}$ of $\mathbf{x}$ and $\mathbf{y}$ with the minimum weighted sum of folding and alignment cost:

$$\min_{S \overset{*}{\Rightarrow}_{G''} \langle \mathbf{x},\mathbf{y},\mathbf{s_x},\mathbf{s_y},\mathbf{a}\rangle} \left[\Delta G^\circ(\mathbf{x},\mathbf{s}_x) + \Delta G^\circ(\mathbf{y},\mathbf{s}_y) - \lambda \log p(\mathbf{a},\mathbf{x},\mathbf{y})\right] \tag{3}$$

There is a trade-off between free energy changes $(\Delta G^\circ(\mathbf{x},\mathbf{s}_1)+\Delta G^\circ(\mathbf{y},\mathbf{s}_2))$ and the alignment cost $(\log p(\mathbf{a},\mathbf{x}_1,\mathbf{x}_2))$, which is balanced by the hyperparameter λ. In the complete model, $\Delta G^\circ(\mathbf{x},\mathbf{s}_1)$ and $\Delta G^\circ(\mathbf{y},\mathbf{s}_2)$ are calculated using loop-based Turner free-energy model [27, 28], and $p(\mathbf{a},\mathbf{x}_1,\mathbf{x}_2)$ is estimated based on the trained HMM parameters [5].

Efficient Algorithms and Implementation

Dynamic Programming

Using Nussinov algorithm as an example, we illustrate the deductive system of LinearSankoff in Fig. 2A–D.

With a simple alignment model, e.g., Needleman–Wunsch [13], whose alignment states are independent with neighbors, states $S(i_1, j_1, i_2, j_2)$ and $P(i_1, j_1, i_2, j_2)$ is the minimum cost of simultaneously folding and alignment of two spans $x_{i_1}x_{i_1+1}\ldots x_{j_1}$ and $y_{i_2}y_{i_2+1}\ldots y_{j_2}$ from two sequences, respectively. $P(i_1, j_1, i_2, j_2)$ requires at least one sequence forms a base pair at the two ends of span, either (x_{i_1}, x_{j_1}) or (y_{i_2}, y_{j_2}) forms a base pair, or both. As shown in Fig. 2A–B, concatenating two adjacent states just sums the cost of two states directly.

The HMM-based alignment model is more complicated due to the alignment state is the current alignment state is dependent on the previous state. Therefore, the states $S(i_1, j_1, i_2, j_2, h_s, h_e)$ and $P(i_1, j_1, i_2, j_2, h_s, h_e)$ are extended with two more dimensions h_s and h_e to indicate the alignment state of start position (x_{i_1}, y_{i_2}) and end position (x_{j_1}, y_{j_2}). Fig. 2C-D use solid yellow arrows to represent alignment states h_s and h_e of each state. The dotted yellow arrows are possible alignments insides each state. When two states are concatenated, e.g., $S(1, 2, 1, 3, \nearrow, \uparrow)$ and $P(3, 6, 4, 6, \nearrow, \nearrow)$, the free energy change of secondary structure can be added. However, the final alignment cost is the product of two probabilities of alignment path and a transition probability from $\uparrow$ to $\nearrow$. To get a larger S by concatenating two small states S and P, the state $S(1, 6, 1, 6, \nearrow, \nearrow)$ only keeps h_s ($\nearrow$) from $S(1, 2, 1, 3, \nearrow, \rightarrow)$ and h_e ($\nearrow$) from $P(3, 6, 4, 6, \nearrow, \uparrow)$ and ignores the intermediate alignment states (Fig. 2D).

Linearization

Inspired by LinearFold [16], the linear-time algorithm for single RNA sequence folding, we generalize the beam search heuristic from single-sequence folding to simultaneously folding two sequences to achieve linear runtime against the sum of sequence lengths. LinearSankoff parses two RNA sequences along diagonal (from bottom left to top right) (see Fig. 2E). Although, the current version of algorithm still runs in $O(n^6)$ time for two sequences, the diagonal direction allows us to further employ beam pruning heuristic [26, 16, 29], which reduces to linear runtime. More sepcifically, at each step s ($s = 1 \ldots n_1 + n_2$), for all candidates $P(i_1, j_1, i_2, j_2, h_s, h_e)$ ($j_1 + j_2 = s$), we only keep the b top-scoring states and prune less promising ones because they are less likely to be part of the optimal final results. This results in an approximate search algorithm in $O(nb^2)$ time.

A* Algorithm

LinearSankoff applies the A* algorithm to further accelerate searching during beam search. The heuristic values are from single sequence folding and sequence alignment. Formally, during beam search, for each step s (from 1 to $n_1 + n_2$), and each state candidate $P(i_1, j_1, i_2, j_2, h_s, h_e)$ with $j_1 + j_2 = s$, LinearSankoff builds a "global" cost by adding an approximately estimated distance to the destination. For folding, LinearSankoff gets $\hat{P}(i_1, j_1)$, which represents the minimum free energy change of folding regions $x_1 \ldots x_{i_1-1}$ and $x_{j_1+1} \ldots x_{n_1}$ for sequence $\mathbf{x}$ from single sequence folding in pre-processing. Similarly, LinearSankoff obtains $\hat{P}(i_2, j_2)$ as the minimum free energy change of folding regions $y_1 \ldots y_{i_2-1}$ and $y_{j_2+1} \ldots y_{n_2}$ for sequence $\mathbf{y}$. For alignment, LinearSankoff looks up the probability of the Viterbi alignment of two prefix sequences $x_1 \ldots x_{i_1}$ and $y_1 \ldots y_{i_2}$ as $A(1, i_1, 1, i_2, -, h_s)$, which constrains the alignment state of (x_{i_1}, y_{i_2}) to be h_s. LinearSankoff also pre-computes the probability of the Viterbi alignment of any two postfix sequences $x_{j_1} \ldots x_{n_1}$ and $y_{j_2} \ldots y_{n_2}$ as $A(j_1, n_1, j_2, n_2, h_e, -)$ from pre-processing, which limits the alignment state of (x_{j_1}, y_{j_2}) to be h_e.

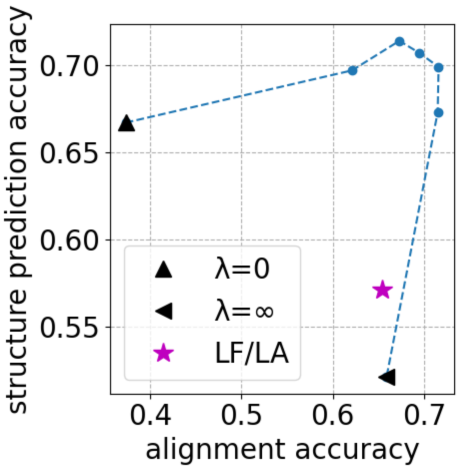


Fig. 3. Grid search for the hyperparameter λ. Structure prediction accuracy (F1 score) against alignment accuracy (F1 score) with λ values from 0 to ∞. LF/LA is the point that shows the structure prediction accuracy for single-sequence LinearFold calculations and alignment accuracy for LinearAlignment alignments that are not structurally informed.

LinearSankoff sums up the free energy change of three segments ($P(i_1, j_1, i_2, j_2, h_s, h_e)$, $\hat{P}(i_1, j_1)$ and $\hat{P}(i_2, j_2)$) as a "global" folding score of two whole sequences, and assembles probabilities of three alignment segments ($P(i_1, j_1, i_2, j_2, h_s, h_e)$, $A(1, i_1, 1, i_2, -, h_s)$ and $A(j_1, n_1, j_2, n_2, h_e, -)$) as a "global" alignment score between two whole sequences, then computes a "global" cost based on Equation 3. Note that, only the segment $P(i_1, j_1, i_2, j_2, h_s, h_e)$ is from simultaneous folding and alignment, the folding costs ($\hat{P}(i_1, j_1)$ and $\hat{P}(i_2, j_2)$) and alignment costs ($A(1, i_1, 1, i_2, -, h_s)$ and $A(j_1, n_1, j_2, n_2, h_e, -)$) are independent of each other. LinearSankoff further applies beam search heuristic regarding "global" costs.

Results

Hyperparameter Selection

The weight on alignment cost (λ) is selected empirically based on performance on the training dataset. We benchmarked LinearSankoff with different values of λ from 0 to infinity over four training families from RNAStralign following TurboFold II [30]: tRNA, 5S ribosomal RNA, tmRNA and Group I Intron RNA, and 20 sequence pairs were sampled randomly for each family.

Fig. 3 shows the secondary structure prediction accuracy (y axis) against alignment accuracy (x axis) with different values of λ. When λ is 0, i.e., LinearSankoff barely takes advantage of alignment information (close to Dynalign), structure prediction accuracy of LinearSankoff is still higher than single sequence folding (LinearFold) because LinearSankoff folds two sequences to generally similar structures even though the alignment is poor ($\lambda = 0$ in Fig. 3). Additionally, when λ is infinite, LinearSankoff only optimizes alignment and the alignment accuracy is close to the accuracy of sequence alignment (LinearAlignment, see $\lambda = \infty$ in Fig. 3). In between these

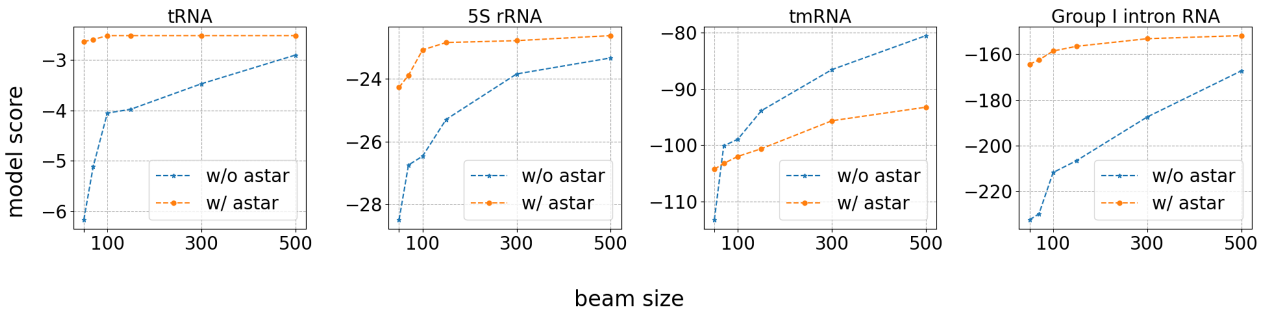


Fig. 4. Model score against beam size. The orange and blue curves represent LinearSankoff with and without applying A* algorithm, respectively.

extreme λ values, as the λ value increases, Fig. 3 illustrates a trend that both the structure prediction and alignment accuracies first increase then decrease. We choose $\lambda = 0.3$ as the default value which is the most closest to the top right corner.

Fig. 4 compares the model scores of LinearSankoff with and without the A* algorithm against the beam size over four training families. Both methods get higher model scores with a larger beam size. While LinearSankoff with A* algorithm leads to have a higher model score than the plain LinearSankoff with a small beam size, e.g., 50. In addition to the tmRNA family, as the beam size increase to 500, LinearSankoff with the A* algorithm still achieves higher model scores than the plain LinearSankoff, but the difference of model scores gets smaller. For all the results presented and discussed in the following parts are from LinearSankoff with the A* algorithm.

Efficiency and Scalability

To compare the runtime usage of LinearSankoff ($\lambda = 0.3$ and $b = 100$) and Dynalign, one practical implementation of the Sankoff algorithm, we collected a dataset that consists of sequence pairs from RNAStralign with the average sequence length ranging from 70 to 3000 *nt*. We used a Linux machine (CentOS 7.7.1908) with a 2.30 GHz Intel Xeon E5-2695 v3 CPU and 755 GB memory, and gcc 4.8.5 for benchmarks.

As we discussed above, Dynalign takes $O(n^3m^3)$ time, where m is the average width of alignment searching space, which correlates with the sequence identity. Dynalign has two modes to decide the value of m. One is to require users to specify the value of m, which is fixed along the sequence. Another mode is to generate a valid alignment searching space adaptively based on sequence identity. Dynalign first computes posterior alignment probabilities using the forward-backward algorithm, then prunes unlikely positions by a threshold, which is determined by sequence identity. LinearSankoff also has these two modes.

Fig. 5A uses the second mode, which restricts the alignment searching space adaptively based on sequence identity, to show runtime comparison between Dynalign and LinearSankoff. Dynalign took more than 800 minutes for two pairs of Group I Intron sequences with sequence length ~500 *nt*. The sequence identity of these two pairs is around 0.35 with m 45 *nt*. Even though 16S rRNA sequences (~1500 *nt*) are three times longer than Group I Intron sequences, Dynalign only took one third of the runtime spent on Group I Intron sequence pairs due to the high sequence identity (0.85) and a narrow searching space ($m = 4$ *nt*) of the 16S rRNA sequence pairs.

	SRP RNA	RNaseP RNA	telomerase RNA	16S rRNA	overall
sequence length	286	370	455	1140	
sequence identity	0.29	0.48	0.83	0.85	
m	25.4	18.8	3.4	3.7	
Structure Prediction Accuracy (F1 score)					
LinearFold [16]	72.1	59.0	54.3	46.1	58.0
Dynalign [4]	72.5	69.2	66.4	56.3	66.2
FoldAlign [7]	61.4	56.8	40.6	53.1	53.2
LocARNA [9]	70.9	60.0	61.7	58.6	63.0
SCARNA [10]	72.7	55.6	44.1	62.0	58.7
LinearTurboFold [31]	69.5	70.4	58.3	54.2	63.3
MAFFT+RNAalifold [32]	31.5	42.0	50.2	57.0	45.3
LinearSankoff (b=100)	75.4	70.4	**68.6**	**60.1**	68.6
LinearSankoff (b=∞)	**76.3**	**73.3**	67.5	58.8	**69.0**
Alignment Accuracy (F1 score)					
MAFFT [33]	44.4	70.1	93.1	97.3	76.2
Dynalign	43.2	56.6	70.5	90.1	65.1
FoldAlign	51.2	71.0	92.7	97.2	78.0
LocARNA	**54.8**	70.7	92.3	97.3	**78.8**
SCARNA	50.2	70.5	93.0	**97.4**	77.8
LinearTurboFold	50.8	69.0	**93.4**	97.3	77.6
LinearSankoff (b=100)	50.8	**73.0**	91.2	96.8	77.9
LinearSankoff (b=∞)	51.2	**73.0**	91.3	96.7	78.1

Table 2. Structure prediction and alignment accuracies on test set.

Thanks to the beam pruning, although LinearSankoff performs a more complicated alignment model than Dynalign, LinearSankoff is significantly faster than Dynalign, especially for diverse or long sequences. In Fig. 5A, for instance, Dynalign took ~13 hours for Group I Intron sequence pairs, which are the most diverse sequences among sampled data, and ~3 hours for 16S rRNA sequence pairs, which are the longest sequences. While, LinearSankoff only needs 5 minutes for Group I Intron sequence pairs and 30 seconds for 16S rRNA sequence pairs, respectively.

Both Dynalign and LinearSankoff's runtime correlate with both alignment searching space (m) and sequence length (n). Therefore, we fixed alignment searching space (Fig. 5B) and sequence length (Fig. 5C) to show correlation with two variables. With a fixed m, Dynalign scales cubically with sequence length, while LinearSankoff takes linear runtime with sequence length. With fixed sequence length and m varying from 1 to 20 *nt*, both Dynalign and LinearSankoff scales almost quadratically with m.

Folding and Alignment Accuracies

To evaluate LinearSankoff and several benchmarks, we first randomly sampled 80 sequence pairs from the other four families of the RNAStralign dataset: SRP RNA, telomerase

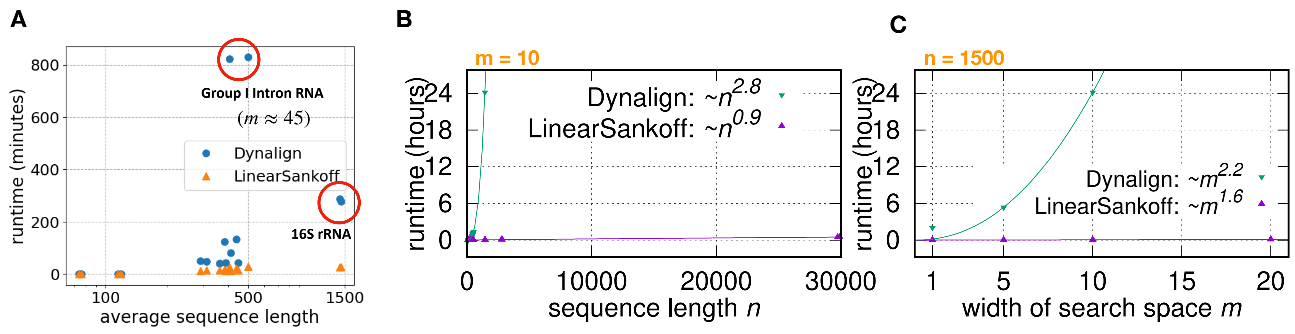


Fig. 5. Runtime comparisons between Dynalign and LinearSankoff (b=100). (**A**) The runtime of sampled pairs of sequences. Both Dynalign and LinearSankoff construct the alignment searching space adaptively based on sequence identity. (**B**) The runtime against sequence length (n) with the fixed width of alignment searching space (m=10 nt). (**C**) The runtime against the width of alignment search space (m) with fixed sequence length (n=1500 nt).

RNA, RNase P RNA and 16S rRNA. The first three rows in Tab. 2 summarize the basic information of these four families including the average sequence length, sequence identity, and the average alignment searching space (m). The benchmarks consist of LinearFold, MAFFT, Dynalign, FoldAlign, LocARNA, SCARNA, LinearTurboFold and RNAalifold, which are selected from several perspectives. LinearFold predicts structures for a single sequence, and MAFFT performs alignment only based on nucleotides. Dynalign, FoldAlign, LocARNA and SCARNA are representative implementations of the Sankoff algorithm. As a workaround of the Sankoff algorithm, LinearTurboFold iteratively performs folding and alignment modules to avoid strictly simultaneous computation. MAFFT + RNAalifold divides the task of simultaneous folding and alignment into two consecutive independent tasks: first aligning sequences then folding the alignment. We ran all the tools with default settings, only ran SCARNA with "-rfold" mode.

For secondary structure prediction, LinearSankoff (both b=100 and ∞) first perform better than single-sequence folding (LinearFold). LinearSankoff (both b=100 and ∞) achieves higher accuracy than Sankoff-style methods including Dynalign, FoldAlign, LocARNA and SCARNA on every test family. With infinite beam size, LinearSankoff leads to better performance on the SRP RNA and RNase P RNA families than beam size 100. These two families have relatively low sequence identity with large alignment searching space (m) among four test families, thus need a large beam size.

Regarding alignment accuracy, Dynalign obtains the lowest accuracy among all benchmarks on all four test families, which does not include terms for sequence identity. FoldAlign, SCARNA, LinearTurboFold achieve comparable alignment accuracy to LinearSankoff (both b=100 and ∞). LinearSankoff with infinite beam size is the second-best tool in terms of alignment quality, and its accuracy is only lower than LocARNA. While, LocARNA performs worse on secondary structure prediction than Dynalign, LinearTurboFold and LinearSankoff (both b=100 and ∞).

Domain Insertion

Most of implementations and variations of the Sankoff algorithm (Dynalign, FoldAlign, LocARNA, SCARNA and LinearSankoff) fold RNA homologous sequences to generally similar structures, i.e., a branch in one structure must have a corresponding branch in the other structure. However, it is not guaranteed that the one-to-one correspondence always exists. For instance, some structures contains insertion or deletion of a whole branch. As the ground truth shown in Fig. 6B, one structure (on the top of the matrix) contains one more branch (covered by a red box) than the other structure on the right side. The yellow curve in the black matrix represents the alignment between two sequences with a continuous long insertion corresponds to insertion of a whole branch into the top structure. Dynalign II [6] extends the Dynalign to model domain insertion. Following Dynalign II, LinearSankoff † is able to model inserted branches as well.

To evaluate the modeling ability of LinearSankoff †, we collected a specific dataset by sampling sequence pairs from tRNA and SRP RNA families. From the tRNA family, we sampled sequence pairs, whose structures consist of four

A

	tRNA (4- vs. 5-branches)			SRP RNA (2- vs. 3-branches)		
	PPV	sensitivity	F1	PPV	sensitivity	F1
Dynalign	71.4	70.8	71.1	59.3	59.2	59.2
LinearSankoff	87.6	82.6	85.0	43.4	37.1	40.0
Dynalign II	82.7	84.3	83.5	70.4	72.9	71.6
LinearSankoff †	91.6	91.6	91.6	73.7	71.3	72.5

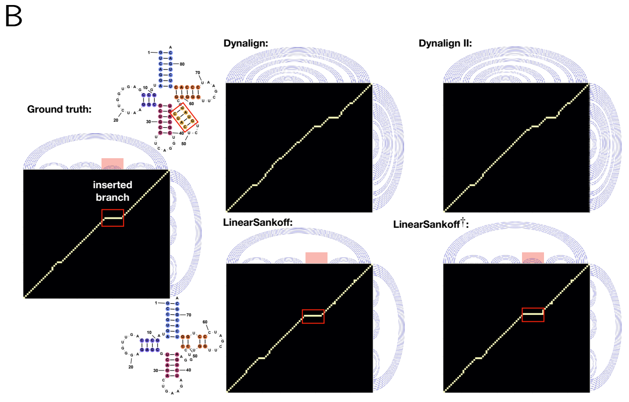
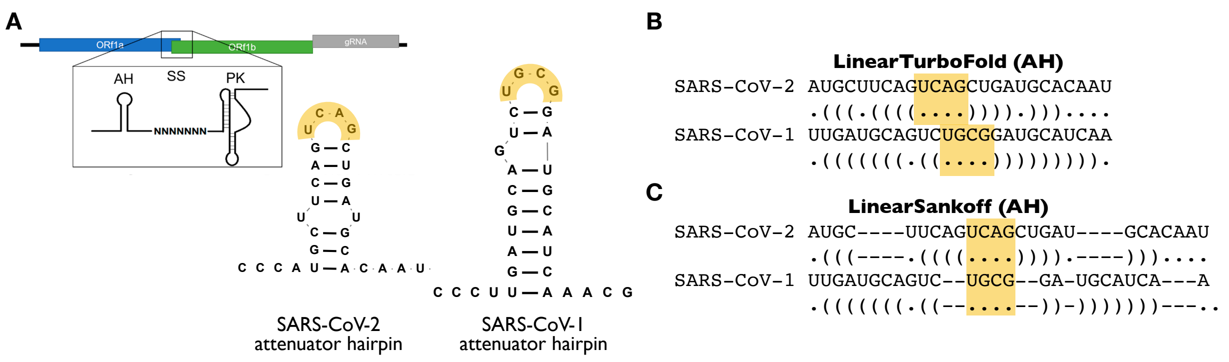


Fig. 6. (**A**) Structure prediction accuracies on families with domain insertion. Both Dynalign and LinearSankoff are unable to capture the inserted branch, while Dynalign II and LinearSankoff † have the advanced domain insertion feature. (**B**) The ground truth and the predictions of Dynalign, Dynalign II, LinearSankoff and LinearSankoff † for a sequence pair (tdbD00005111 and tdbD00001770) from the tRNA family.



```
SARS-CoV-2  AUGCUUCAGUCAGCUGAUGCACAAU
            .(((.((((....)))).)))....
SARS-CoV-1  UUGAUGCAGUCUGCGGAUGCAUCAA
            .(((((((.((....))))))))).
```

```
SARS-CoV-2  AUGC----UUCAGUCAGCUGAU----GCACAAU
            .(((----.((((....)))).----)))....
SARS-CoV-1  UUGAUGCAGUC--UGCG--GA-UGCAUCA---A
            .(((((((.((--....--))-)))))))---..
```

Fig. 7. The alignment of attenuator hairpins in the frameshifting element (FSE). (**A**) Canonical structures of attenuator hairpins in SARS-CoV-2 and SARS-CoV-1 frameshifting element. (**B**) Alignment of attenuator hairpins generated by LinearTurboFold over 25 SARS-CoV-2 and SARS-related genomes. (**C**) Alignment of attenuator hairpins generated by LinearSankoff. Clearly, the alignment from LinearSankoff is more accurate than the output of LinearTurboFold for the attenuator hairpins.

branches and five branches, respectively. For the SRP RNA family, we sampled sequence pairs from two subfamilies (archael and long bacterial). Compared to the structure of the archael, one branch is deleted from a three-branch multiloop in the structure of the long bacterial. Fig. 6A shows performance of Dynalign, Dynalign II, LinearSankoff and LinearSankoff † on two families. Overall, with the help of HMM-based alignment model, LinearSankoff † achieves higher structure prediction accuracy than Dynalign II after adding the feature of domain insertion.

Without the feature to model domain insertion, LinearSankoff achieves higher accuracy than Dynalign on the tRNA family due to the powerful HMM-based alignment model. Although it is out of the scope of LinearSankoff to predict insertion of a whole branch, the HMM-based alignment model captures the signal from sequences and LinearSankoff just leaves the corresponding region unpaired (see red boxes in LinearSankoff prediction in Fig. 6B). LinearSankoff even obtains higher accuracy than Dynalign II. We observed that Dynalign II predicts same structures as Dynalign for some tRNA sequence pairs (see Dynalign and Dynalign II predictions in Fig. 6B), which is highly because the default penalty for domain insertion is relatively large for tRNA sequences (only $\sim$70 *nt*). In other words, the free energy change of adding a new branch can not make up for the penalty of domain insertion. While LinearSankoff does not reply on any penalty for domain insertion, only the probability of the alignment path.

Application to SARS-CoV-2 genomes

We further applied LinearSankoff to the SARS-CoV-2 reference genome (NC_045512.2) with the SARS-CoV-1 reference genome (NC_004718). The attenuator hairpin (AH) in the frameshifting element (FSE) are conserved among SARS-CoV-2 and SARS-related genomes and its structures are well established as shown in Fig 7A. However, LinearTurboFold cannot align these two attenuator hairpins from SARS-CoV-2 and SARS-CoV-1 correctly due to some extent of disagreement between folding and alignment (Fig 7B). Thanks to the strong coupling between folding and alignment in LinearSankoff, Fig 7C shows LinearSankoff aligns two structures properly, and we can further extract conserved structures directly based on LinearSankoff's prediction without any extra manual work.

Conclusion

We focus on simultaneous folding and alignment of RNA homologous sequences. Formally, we borrowed synchronous context free grammars from computational linguistics to formulate homologous folding of two RNA sequences. We proposed LinearSankoff, which enhances the modeling capacity of the Sankoff algorithm by integrating it with an HMM-based alignment model. We devised a dynamic programming algorithm tailored to this combined Sankoff+HMM model. In addition, LinearSankoff generalizes beam search heuristic from single-sequence folding to parsing two sequences simultaneously, which make its runtime scale linearly with sequence length. LinearSankoff further applies A* algorithm to conduct more efficient searching together with beam pruning. Based on evaluation on four test families and comparison with a variety of benchmarks, LinearSankoff achieves significantly better secondary structure accuracy than other benchmarks, and comparable alignment accuracy to most of the Sankoff-style tools. LinearSankoff is also the first joint folding and alignment algorithm to scale to full-length SARS-CoV-2 genomes, and outperforms other tools in identifying crucial conserved structures between SARS-CoV-2 and SARS-CoV-1.

LinearSankoff is in principle extendable to multiple sequences. with several solutions. One option is to replace Dynalign with LinearSankoff in Multilign [34], which progressively constructs a conserved structure to multiple sequences by conducting pairwise alignment using Dynalign. Another option is to generalize LinearSankoff to take not only single sequences but also multiple sequence alignments (MSA) as input, i.e., simultaneously folding and alignment of MSAs. With such generalizations, LinearSankoff can progressively build the MSA along a phylogenetic tree. We leave these endeavors to future work.

References

1. S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, 2(12):919–929, 2001.
2. Jennifer A. Doudna and Thomas R. Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002.

3. David Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810—-825, 1985.
4. David H Mathews and Douglas H Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of molecular biology*, 317(2):191–203, 2002.
5. Arif Ozgun Harmanci, Gaurav Sharma, and David H Mathews. Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics*, 8(1):130, 2007.
6. Yinghan Fu, Gaurav Sharma, and David H Mathews. Dynalign ii: common secondary structure prediction for rna homologs with domain insertions. *Nucleic acids research*, 42(22):13939–13948, 2014.
7. Jakob H Havgaard, Elfar Torarinsson, and Jan Gorodkin. Fast pairwise structural rna alignments by pruning of the dynamical programming matrix. *PLOS computational biology*, 3(10):e193, 2007.
8. Chuong B Do, Mahathi SP Mahabhashyam, Michael Brudno, and Serafim Batzoglou. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome research*, 15(2):330–340, 2005.
9. Sebastian Will, Kristin Reiche, Ivo L Hofacker, Peter F Stadler, and Rolf Backofen. Inferring noncoding rna families and classes by means of genome-scale structure-based clustering. *PLoS computational biology*, 3(4):e65, 2007.
10. Yasuo Tabei, Koji Tsuda, Taishin Kin, and Kiyoshi Asai. SCARNA: fast and accurate structural alignment of RNA sequences by matching fixed-length stem fragments. *Bioinformatics*, 22(14):1723–1729, 2006.
11. Arif Ozgun Harmanci, Gaurav Sharma, and David H Mathews. Parts: probabilistic alignment for rna joint secondary structure prediction. *Nucleic acids research*, 36(7):2406–2417, 2008.
12. Yasuo Tabei, Hisanori Kiryu, Taishin Kin, and Kiyoshi Asai. A fast structural multiple alignment method for long RNA sequences. *BMC Bioinformatics*, 9(1):33, 2008.
13. Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.
14. Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.
15. Arif O Harmanci, Gaurav Sharma, and David H Mathews. Turbofold: iterative probabilistic estimation of secondary structures for multiple RNA sequences. *BMC bioinformatics*, 12(1):108, 2011.
16. Liang Huang, He Zhang, Dezhong Deng, Kai Zhao, Kaibo Liu, David Hendrix, and David Mathews. LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics*, 35(14):i295–i304, 07 2019.
17. Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
18. Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403, 1997.
19. Philip M Lewis and Richard Edwin Stearns. Syntax-directed transduction. *Journal of the ACM (JACM)*, 15(3):465–488, 1968.
20. Alfred V. Aho and Jeffrey D. Ullman. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56, 1969.
21. David Chiang. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228, 2007.
22. David H Mathews and Douglas H Turner. Prediction of RNA secondary structure by free energy minimization. *Curr. Opin. Struct. Biol.*, 16(3):270–278, 2006.
23. Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *PNAS*, 77(11):6309–6313, 1980.
24. Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *NAR*, 9(1):133–148, 1981.
25. Stuart M. Shieber and Yves Schabes. Synchronous Tree-Adjoining Grammars. In *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*, 1990.
26. Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*, page 1077–1086, Uppsala, Sweden, 2010. ACL.
27. David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288(5):911–940, 1999.
28. David Mathews et al. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *PNAS*, 101(19):7287–7292, 2004.
29. He Zhang, Liang Zhang, David H Mathews, and Liang Huang. LinearPartition: linear-time approximation of RNA folding partition function and base-pairing probabilities. *Bioinformatics*, 36(Supplement_1):i258–i267, 2020.
30. Zhen Tan, Yinghan Fu, Gaurav Sharma, and David H. Mathews. TurboFold II: RNA structural alignment and secondary structure prediction informed by multiple homologs. *Nucleic Acids Research*, 45(20):11570–11581, 09 2017.
31. Sizhen Li, He Zhang, Liang Zhang, Kaibo Liu, Boxiang Liu, David H Mathews, and Liang Huang. Linearturbofold: Linear-time global prediction of conserved structures for rna homologs with applications to sars-cov-2. *Proceedings of the National Academy of Sciences*, 118(52):e2116269118, 2021.
32. Stephan H Bernhart, Ivo L Hofacker, Sebastian Will, Andreas R Gruber, and Peter F Stadler. RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics*, 9(1):1–13, 2008.
33. Kazutaka Katoh and Daron M Standley. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular Biology and Evolution*, 30(4):772–780, 2013.
34. Zhenjiang Xu and David H Mathews. Multilign: an algorithm to predict secondary structures conserved in multiple RNA sequences. *Bioinformatics*, 27(5):626–632, 2011.