CoSTAR: Coarse Stem-Topology Alignment of Pseudoknotted RNA Structures by Relation-Constrained Search

Finn Archinuk
Department of Biomedical Engineering, University of Alberta, Edmonton, Canada

Hosna Jabbari[1]
Department of Biomedical Engineering, University of Alberta, Edmonton, Canada

Abstract

RNA structural alignment is a central task in comparative RNA analysis, but many efficient methods achieve tractability by restricting the class of admissible structures, often excluding pseudoknots. This exclusion is limiting for viral and regulatory RNAs, where conserved structure can remain informative even when sequence conservation is weak. We introduce a coarse RNA structural alignment algorithm that aligns secondary structures by searching over partial maps between stems rather than nucleotides. Each input structure is decomposed into stems, annotated with nucleotide-level features, and encoded by pairwise topological relations among stems. Alignment is formulated as a cost-minimizing partial stem map with skip operations, and the search tree is pruned by RNA-specific directionality and topological constraints derived from already aligned stems. For the stated cost function and over the class of injective, direction-preserving, topologically consistent stem maps, the search is exact. This shifts the dominant computational dependence from sequence length to the number and arrangement of stems. We evaluated the method on 2100 pairwise alignments sampled from seven Rfam families spanning 40–224 nucleotides and 2–15 stems. Across these benchmarks, the algorithm returned terminal coarse alignments in which every stem was either matched or skipped. We measured running time and search-tree width to characterize performance on diverse family-to-family comparisons. The experiments also show that ordering the input structures affects efficiency: using the structure with more stems as the search-driving structure reduces tree width. The resulting partial stem map is directly interpretable for RNA annotation and can be projected to nucleotide resolution for downstream sequence–structure analysis. The source code for CoSTAR is available at: `https://github.com/TheCOBRALab/CoSTAR`

2012 ACM Subject Classification Applied computing → Bioinformatics; Theory of computation → Design and analysis of algorithms

Keywords and phrases RNA structural alignment, pseudoknots, RNA secondary structure, branch-and-bound search, stem topology

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.24

Related Version *Preliminary Version*: `https://doi.org/10.64898/2026.05.22.727263`

Supplementary Material *Software (Source Code)*: `https://github.com/TheCOBRALab/CoSTAR` archived at `swh:1:dir:d00d8facb68d10226195689f8d8ff14666d61172`

1 Introduction

RNA molecules perform diverse informational, catalytic, and regulatory functions in the cell. Protein-coding mRNAs serve as templates for translation. Non-coding RNAs contribute to translation as ribosomal and transfer RNAs, to RNA processing as spliceosomal and small nucleolar RNAs, to gene regulation as small RNAs and long non-coding RNAs, to

[1] corresponding author

genome organization and defense, and to catalysis as ribozymes [19, 20, 4, 7]. Across these roles, RNA activity often depends not only on primary sequence, but also on structures formed within the molecule and through interactions with other RNAs, DNA, proteins, or small molecules [26, 5, 12]. Thus, the analysis of RNA function often requires methods that compare molecules beyond their primary sequences.

One important level of RNA structural organization is secondary structure, defined by the set of intramolecular base pairs in the molecule [26]. For many structural non-coding RNAs, functional constraints act on the base-pairing pattern at least as strongly as on the primary sequence. Consequently, homologous RNAs may show weak sequence identity while retaining corresponding base pairs. At a conserved base pair, a substitution at one paired position can be accompanied by a substitution at the other position so that base-pair complementarity is maintained. These coordinated changes are compensatory substitutions, and their statistical association across an alignment is referred to as covariation [10]. Comparative RNA methods use such covariation as evidence for conserved secondary structure [8]. Structural alignment is therefore a central computational primitive for comparing ncRNAs, annotating conserved structural elements, and transferring functional or mechanistic hypotheses between related RNAs [3].

The algorithmic difficulty of structural alignment depends strongly on the class of secondary structures handled. If all base pairs are non-crossing, the structure is pseudoknot-free and has a nested topology. This nesting property supports efficient dynamic programming decompositions, tree representations, and grammar-based models [24, 25, 9]. RNAforester, for example, uses a tree representation to compute local similarity between RNA secondary structures [15], whereas BEAR enriches dot-bracket strings with contextual structural symbols and learned substitution scores [18]. However, these efficient approaches are designed for a restricted structural class and do not preserve crossing base-pair dependencies. Although pseudoknots constitute a minority of annotated structures in current datasets (bpRNA estimates that approximately 12% of annotated structures contain pseudoknots [6]), they occur in important viral regulatory mechanisms, including gene expression and replication [2]. Thus, the existing efficient structural comparison methods may miss the biologically important structural motifs that lie outside of their pseudoknot-free class of structures.

This limitation is especially relevant for viral non-coding and regulatory RNA elements. Viral genomes can accumulate substitutions rapidly; as sequence divergence increases, nucleotide-level alignments may fail to identify corresponding stems or structural motifs [21]. At the same time, particular stems and pseudoknotted motifs may be constrained by replication, translation, or regulatory function, and may therefore remain comparable even when the underlying nucleotides differ [2]. Some sequence–structure alignment methods can handle pseudoknotted inputs. LaRA 2, for example, aligns RNA sequences with pseudoknotted secondary structures [28]. Such methods return nucleotide-to-nucleotide alignments. In applications where the primary question is whether two RNAs contain corresponding stems, this resolution is not always necessary. We therefore consider a coarser alignment problem: compute a partial stem map between the stems of two RNA secondary structures, while preserving their relative topological relationships. This stem-level map can be used directly for structural annotation or as a set of anchors for subsequent nucleotide-level sequence–structure alignment.

Motivated by this setting, we formulate RNA structural alignment at the level of stems rather than nucleotides. For each input structure, short bulges and internal loops are first compressed according to a user-defined threshold, and the resulting structure is decomposed into stems. Each stem is represented by a feature vector that includes the number of base

pairs, the number of nucleotides in its two stem arms, and its stacking-energy magnitude. We also precompute a relation array for each structure. For an ordered pair of stems (A, B), this array records the topological relation of B with respect to A: 5′-disjoint, 3′-disjoint, inside A, containing A, crossing A from the 5′ side, or crossing A from the 3′ side. Given two input structures R^F and R^G, with ordered stem lists $\mathcal{F}$ and $\mathcal{G}$, a coarse alignment is a partial injective map from stems of $\mathcal{F}$ to stems of $\mathcal{G}$. Stems that are not included in the map are assigned skip costs, and matched stems are assigned costs derived from their feature vectors.

The computational bottleneck is the number of possible partial stem maps. If $\mathcal{F}$ has f stems and $\mathcal{G}$ has g stems, then, without any order constraint, the number of partial injective maps is $\sum_{k=0}^{\min(f,g)} \binom{f}{k}\binom{g}{k}k!$. Indeed, for each k, one chooses k stems from $\mathcal{F}$, k stems from $\mathcal{G}$, and an arbitrary bijection between the two chosen sets. CoSTAR restricts the search to direction-preserving maps: if stem F_i is matched before $F_{i'}$, then its image in $\mathcal{G}$ must also occur before the image of $F_{i'}$ in $\mathcal{G}$. Under this restriction, the two chosen k-element subsets determine a unique order-preserving bijection. Hence the number of terminal direction-preserving partial maps, before applying topological constraints, is $\sum_{k=0}^{\min(f,g)} \binom{f}{k}\binom{g}{k} = \binom{f+g}{f}$, where the equality follows from Vandermonde's identity. This space remains exponential in the number of stems, and the generated search tree also contains prefixes of such maps. CoSTAR therefore expands partial alignments in the 5′-to-3′ order of $\mathcal{F}$. When assigning the next stem F_i, a candidate G_j is considered only if j is larger than every previously matched index in $\mathcal{G}$. The candidate is admissible only if, for every previously matched pair (F_p, G_q), the relations of the next stem $\rho_F(p, i)$ are identical to those of the candidate stems $\rho_G(q, j)$, where $\rho_F(p, i)$ and $\rho_G(q, j)$ denote the corresponding ordered stem relations in the two input structures. These precomputed local consistency checks reduce the branching factor without requiring either structure to be pseudoknot-free.

Contributions. We introduce CoSTAR, a relation-constrained search algorithm for coarse stem-topology alignment of RNA secondary structures that may contain pseudoknots. CoSTAR represents each input as an ordered list of stems with nucleotide-derived features and precomputed pairwise topological relations. It then solves a minimum-cost alignment problem over partial injective stem maps with explicit skip operations, using 5′-to-3′ order and relation-array consistency to prune infeasible extensions. We evaluate CoSTAR on 2100 pairwise alignments from seven Rfam families with 2–15 stems, characterizing running time and search-tree width and showing that placing the structure with more stems in the search-driving role empirically reduces search-tree width. The output is an interpretable partial stem map for annotation or downstream nucleotide-level sequence–structure alignment.

2 Related Work

RNA comparison methods differ in the information they use, the structural classes they support, and the resolution of the output. Classical sequence–structure alignment methods extend sequence alignment by incorporating base-pairing information into the objective. The Sankoff formulation gives a general framework for simultaneous alignment and folding, but its full dynamic program is computationally expensive [24]. LocARNA implements an efficient Sankoff-style approach for local sequence–structure alignment of pseudoknot-free RNAs and has been used for structure-based clustering of non-coding RNAs [27]. LaRA and LaRA 2 address fixed sequence–structure alignment using combinatorial optimization. In particular, LaRA models sequence–structure alignment as an integer linear program, and LaRA 2

accelerates pairwise alignment by using parallel execution and vectorized alignment kernels [1, 28]. These methods are appropriate when the desired output is a nucleotide-to-nucleotide alignment. Our objective is different: given two fixed structures, we compute a partial stem map between stems. This coarser output records which stems correspond and which are skipped, and can later be projected to nucleotide resolution if needed.

Several methods compare RNA secondary structures primarily by transforming or enriching structural representations. RNAforester represents pseudoknot-free secondary structures as ordered forests and computes local or global structural alignments [15]. BEAR replaces standard dot-bracket notation with a context-aware alphabet that distinguishes paired and unpaired positions according to their structural context, and then compares the resulting strings using learned substitution scores [18]. More recently, bpRNA-align uses feature-specific structural arrays, context-dependent affine gap penalties, and a structural substitution matrix to compute global similarity scores between RNA secondary structures [16]. These approaches provide useful structural similarity measures, but their alignments are defined over nucleotide positions or position-level structural symbols. In contrast, our method searches directly over partial maps between stems, which are the objects returned by the output alignment.

Pseudoknotted structures require representations that do not rely on purely nested topology. ASPRAlign compares secondary structures with arbitrary pseudoknots by transforming them into algebraic or structural RNA trees and computing the ASPRA distance [22]. This provides an efficient structure-only distance for pseudoknotted RNAs. However, the ASPRA distance is a global dissimilarity measure rather than an explicit feature-weighted correspondence between stems. Our method instead keeps pseudoknotted relations as pairwise topological constraints during search, while retaining stem-level quantitative features such as stem length and stacking energy.

Descriptor-based RNA motif search provides another relevant abstraction. In RNArobo, a user specifies a motif descriptor containing sequence and structural constraints, and the algorithm searches genomic sequences for matching regions [23]. Although descriptor search is not a pairwise alignment problem, it illustrates the utility of representing RNA structure in terms of ordered helices and other structural descriptors rather than individual nucleotides. Our reduced topology graph follows this element-level view, but uses it to align two complete structures rather than to search a sequence database for occurrences of a query descriptor.

Topology-based abstractions are also closely related. RNA abstract shapes collapse secondary structures into branching forms, enabling clustering of structures that differ in local base-pair details but share a global organization [13]. RNA-as-Graphs represents RNA secondary structures as graph topologies, including pseudoknotted structures through dual graphs [11]. Such abstractions are intentionally non-injective: distinct structures can map to the same abstract object when they differ in stem length, nucleotide composition, or energetic stability but not in topology. Super N-motifs takes a different alignment-free approach by decomposing structures into simple motifs and constructing vector representations from informative motif combinations [14]. These representations are useful for clustering or large-scale comparison, but they do not directly return a one-to-one correspondence between stems in two RNAs.

Our method occupies an intermediate point between nucleotide-level alignment and alignment-free topology comparison. It supports pseudoknotted inputs, uses topology to constrain the search, and uses nucleotide-derived stem features to avoid purely topological degeneracy. The resulting output is an explicit partial map between stems, rather than only a nucleotide alignment, a string alignment, or a scalar distance. This partial map can be projected to a nucleotide-level alignment which we will use to demonstrate alignment quality.

3 Data Representation

In this section we define the representation used by our CoSTAR algorithm. The input is an RNA sequence together with a fixed secondary structure. The output of the preprocessing step is an ordered stem list, a reduced topology graph, a relation array over ordered pairs of stems, and a feature vector for each stem.

3.1 RNA sequence and secondary structure

An RNA sequence $S = S_1, \ldots, S_n$ is a string over the alphabet $\{A, C, G, U\}$. The indices $1, \ldots, n$ are ordered from the 5′ end to the 3′ end. For $1 \leq i \leq j \leq n$, let $S_{i,j} = S_i, \ldots, S_j$ denote the subsequence of S, and let $[i, j]$ denote the corresponding interval of nucleotide positions.

A base pair of S is an ordered pair (i, j), where $1 \leq i < j \leq n$. In a canonical sequence-compatible structure, S_i and S_j are complementary, i.e., $\{S_i, S_j\}$ is one of $\{A, U\}$, $\{C, G\}$, or $\{G, U\}$. The alignment algorithm below is specified for the base-pair set itself, so annotated non-canonical pairs can also be admitted provided that the pairing relation is well defined. A secondary structure R for S is a set of base pairs such that each nucleotide participates in at most one pair:

$$\{i, j\} \cap \{i', j'\} = \emptyset \quad \text{for all distinct } (i, j), (i', j') \in R.$$

A secondary structure is pseudoknot-free if it does not contain two base pairs (i, j) and (i', j') such that $i < i' < j < j'$ or $i' < i < j' < j$. Otherwise, the structure is pseudoknotted. Pseudoknotted structures are admissible inputs to our method.

3.2 Stems and bulge compression

A stem is a maximal chain of base pairs that are consecutive, or nearly consecutive after ignoring a bounded number of unpaired nucleotides. Let $\beta \geq 0$ be a user-defined bulge compression parameter. Consider two base pairs (i, j) and (k, ℓ) in R, with $i < k < \ell < j$. They are stacked if $k = i + 1$ and $\ell = j - 1$. They are β-adjacent if all nucleotides in the intervals $[i + 1, k - 1]$ and $[\ell + 1, j - 1]$ are unpaired in R, and $(k - i - 1) + (j - \ell - 1) \leq \beta$. Thus, $\beta = 0$ recovers the ordinary definition of consecutive stacked base pairs. Larger values of β allow short bulges or internal loops to be compressed when defining coarse stems. The original sequence and base-pair set are not changed; the parameter β only determines which base pairs are grouped into the same stem. As default we set $\beta = 2$.

A stem h is a maximal ordered sequence of base pairs $h = \big((i_1, j_1), (i_2, j_2), \ldots, (i_r, j_r)\big)$ such that $i_1 < i_2 < \cdots < i_r < j_r < \cdots < j_2 < j_1$, and (i_{t-1}, j_{t-1}) and (i_t, j_t) are β-adjacent for every $1 \leq t \leq r$. The integer r is the number of base pairs in the stem.

Each stem has two arms. The 5′ arm of h is the interval $h^- = [i_1, i_r]$, and the 3′ arm is the interval $h^+ = [j_r, j_1]$. We refer to the ordered pair (h^-, h^+) as the arm representation of h. The 5′ arm always precedes the 3′ arm in the sequence order.

Let $\mathcal{H}(R, \beta) = (h_1, \ldots, h_M)$ be the ordered stem list obtained from R using compression parameter β, where stems are ordered by the left endpoint of their 5′ arms, and M is the total number of stems. When β is fixed by context, we write simply $\mathcal{H}(R)$.

3.3 Reduced topology graph

The reduced topology graph records the order of stem arms along the RNA backbone and the pairing relationship between the two arms of each stem. Let $V = \{h_i^-, h_i^+ : 1 \leq i \leq M\}$ be the set of arm intervals. Sort these $2M$ intervals by their left endpoint in the 5′-to-3′

order, and denote the resulting sequence by $v_1, v_2, \ldots, v_{2M}$. The reduced topology graph is $T_R = (V, E_{\text{bb}} \cup E_{\text{stem}})$, where $E_{\text{bb}} = \{(v_t, v_{t+1}) : 1 \le t < 2M\}$ is the set of directed backbone edges, and $E_{\text{stem}} = \{(h_i^-, h_i^+) : 1 \le i \le M\}$ is the set of stem edges. Unpaired regions that do not belong to compressed stems are not represented as vertices. Hence T_R has $2M$ vertices, $2M-1$ backbone edges, and M stem edges. This representation is related to descriptor-based RNA motif representations, where structures are described through ordered helices and intervening regions [23]; here we retain only the stem arms and their order because the alignment algorithm operates on stems rather than on unstructured intervals. The graph T_R is used to define and visualize the reduced topology; the search algorithm uses the ordered stem list, the feature vectors, and the relation array induced by T_R.

3.4 Pairwise topological relations

The alignment algorithm uses pairwise topological relations between stems as constraints. We first define an ordering relation on arm intervals. For two arm intervals x and y, write $x \prec y$ if interval x ends before interval y begins.

Let $A = (a, a')$ and $B = (b, b')$ be two distinct stems in arm representation, where a and b are the $5'$ arms and a' and b' are the $3'$ arms. We use the ordered relation-label tuple $\Lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6)$ defined by the following mutually exclusive cases:

ℓ	condition	relation of B with respect to A	code alias
1	$b' \prec a$	$5'$-disjoint	$5'$SL (stemloop)
2	$b \prec a \prec b' \prec a'$	$5'$-crossing	$5'$PK (pseudoknot)
3	$b \prec a \prec a' \prec b'$	contains	outer
4	$a \prec b \prec a' \prec b'$	$3'$-crossing	$3'$PK
5	$a' \prec b$	$3'$-disjoint	$3'$SL
6	$a \prec b \prec b' \prec a'$	inside	inner.

The semantic labels are used in the text. The code aliases are retained only for consistency with the implementation and figures.

For a structure R with ordered stem list $\mathcal{H}(R) = (h_1, \ldots, h_M)$, define $\rho_R(i, j)$ as the unique label in Λ giving the relation of stem h_j with respect to stem h_i, for $i \neq j$. For any ordered pair of distinct stems, exactly one label in Λ applies. The reciprocal relation is obtained by reversing the ordered pair. For example, if $\rho_R(i, j) = \lambda_1$, then $\rho_R(j, i) = \lambda_5$; if $\rho_R(i, j) = \lambda_3$, then $\rho_R(j, i) = \lambda_6$; and if $\rho_R(i, j) = \lambda_2$, then $\rho_R(j, i) = \lambda_4$.

We precompute ρ_R for all ordered pairs of stems. The relation array for structure R is the Boolean tensor $\Gamma_R \in \{0, 1\}^{M \times M \times 6}$, where for $i, j \in \{1, \ldots, M\}$ and $\ell \in \{1, \ldots, 6\}$,

$$\Gamma_R[i, j, \ell] = 1 \quad \Longleftrightarrow \quad i \neq j \text{ and } \rho_R(i, j) = \lambda_\ell.$$

Diagonal entries $\Gamma_R[i, i, \ell]$ are set to 0 and are not used. Since there are $M(M-1)$ ordered pairs of distinct stems, the relation array is computed in $O(M^2)$ time and stored in $O(M^2)$ space. This array is independent of the second structure being aligned, so it can be cached and reused across multiple alignments involving the same RNA structure.

3.5 Stem features

The reduced topology graph and relation array encode the relative arrangement of stems, but they do not distinguish stems with the same topological relation profile and different physical properties. Therefore each stem is also assigned a feature vector derived from the original nucleotide-level structure.

For a stem $h = \big((i_1, j_1), \ldots, (i_r, j_r)\big)$, we define three features. The first is the number of base pairs, $\phi_1(h) = r$. The second is the number of nucleotide positions between the two stem arms, $\phi_2(h) = j_r - i_r - 1$. This value provides geometric spacing considerations. The third is the magnitude of the stacking free energy contributed by adjacent stacked base-pair steps in the stem. Let $e_s(i, j)$ denote the nearest-neighbor stacking free energy for the adjacent stacked base-pair step $(i, j), (i + 1, j - 1)$, using standard thermodynamic parameters [17]. Then,

$$\phi_3(h) = - \sum_{\substack{1 \le t < r \\ i_{t+1} = i_t + 1,\ j_{t+1} = j_t - 1}} e_s(i_t, j_t).$$

If no adjacent stacked base-pair step occurs in h, then $\phi_3(h) = 0$. The negative value is used so that larger feature values correspond to larger stabilizing contributions.

The feature vector of h is $\phi(h) = \big(\phi_1(h), \phi_2(h), \phi_3(h)\big)$. All feature vectors are computed once during preprocessing and are used later to define match and skip costs in the alignment algorithm.

3.6 Preprocessing output

For an input sequence–structure pair (S, R) and bulge compression parameter β, the preprocessing step returns

$$\mathcal{D}(S, R, \beta) = \big(\mathcal{H}(R, \beta), T_R, \Gamma_R, \Phi_R\big),$$

where $\mathcal{H}(R, \beta)$ is the ordered stem list, T_R is the reduced topology graph, Γ_R is the relation array, and $\Phi_R = (\phi(h_1), \ldots, \phi(h_M))$ is the list of stem feature vectors. The alignment algorithm in the next section takes two such preprocessed representations as input.

4 CoSTAR Algorithm

Here we define the cost function and the algorithm that finds its minimum. Over the class of injective, direction-preserving, topologically consistent stem maps, the CoSTAR search is exact. Let (S^F, R^F) and (S^G, R^G) be two input sequence–structure pairs, and let

$$\mathcal{D}_F = \mathcal{D}(S^F, R^F, \beta) = (\mathcal{F}, T_F, \Gamma_F, \Phi_F) \quad \text{and} \quad \mathcal{D}_G = \mathcal{D}(S^G, R^G, \beta) = (\mathcal{G}, T_G, \Gamma_G, \Phi_G)$$

be their preprocessed representations. Here $\mathcal{F} = (F_1, \ldots, F_f)$ and $\mathcal{G} = (G_1, \ldots, G_g)$ are the ordered stem lists of the two structures.

4.1 Coarse alignments

A coarse alignment is a partial one-to-one correspondence between stems of $\mathcal{F}$ and stems of $\mathcal{G}$. Formally, let $I_F = \{1, \ldots, f\}$ and $I_G = \{1, \ldots, g\}$ be the stem-index sets of the two structures. A partial map

$$\mu : D \to I_G, \qquad D \subseteq I_F,$$

is injective if $\mu(i) \neq \mu(i')$ for all distinct $i, i' \in D$. We interpret $\mu(i) = j$ as matching stem F_i to stem G_j. Stems in $I_F \setminus D$ and $I_G \setminus \mu(D)$ are skipped.

The search is restricted to direction-preserving maps. A partial map μ is direction-preserving if, for all $i, i' \in D$, $i < i' \implies \mu(i) < \mu(i')$. This constraint preserves the $5'$-to-$3'$ order of matched stems. It does not require the structures to be pseudoknot-free; pseudoknotted relationships are represented separately by the pairwise relation arrays.

A direction-preserving map is topologically consistent if every pair of matched stems has the same ordered relation in both structures. A partial map μ is topologically consistent if $\rho_F(p,i) = \rho_G(\mu(p),\mu(i))$ for every distinct $p, i \in D$. Equivalently, $\Gamma_F[p,i,\cdot] = \Gamma_G[\mu(p),\mu(i),\cdot]$ for every ordered pair of matched stems. The alignment problem is to find a minimum-cost partial map that is injective, direction-preserving, and topologically consistent.

4.2 Match and skip costs

The cost of matching F_i to G_j is the weighted ℓ_1 distance between their feature vectors:

$$\delta_{ij} = \sum_{\ell=1}^{3} \omega_\ell \, |\phi_\ell(F_i) - \phi_\ell(G_j)|$$

where $\omega_1, \ldots, \omega_d \geq 0$ are feature weights (with default values of 1). All pairwise match costs are precomputed in an $f \times g$ matrix $\Delta = (\delta_{ij})$.

Skipping a stem is modeled as matching it to the empty stem. The skip costs are

$$\sigma_i^F = \sum_{\ell=1}^{3} \omega_\ell \phi_\ell(F_i), \qquad \sigma_j^G = \sum_{\ell=1}^{3} \omega_\ell \phi_\ell(G_j).$$

For a terminal coarse alignment represented by $\mu : D \to I_G$, the alignment cost is

$$C(\mu) = \sum_{i \in D} \delta_{i,\mu(i)} + \sum_{i \in I_F \setminus D} \sigma_i^F + \sum_{j \in I_G \setminus \mu(D)} \sigma_j^G.$$

Thus the objective is $\min_\mu C(\mu)$, where the minimum is taken over all injective, direction-preserving, and topologically consistent partial maps.

4.3 Search states

The algorithm searches over partial alignments in the $5'$-to-$3'$ order of $\mathcal{F}$. A search node is a triple $u = (t, \mu, m)$, where $t \in \{1, \ldots, f\}$ is the number of stems of $\mathcal{F}$ whose status has already been decided, μ is a topologically consistent partial map with domain $D \subseteq \{1, \ldots, t\}$, and

$$m = \begin{cases} \max \mu(D), & D \neq \emptyset, \\ 0, & D = \emptyset. \end{cases}$$

The value m is the largest index of a stem in $\mathcal{G}$ that has been matched so far. By directionality, any unmatched stem G_j with $j < m$ is already skipped and cannot be used later.

Equivalently, a node can be represented by two alignment arrays. The $\mathcal{F}$-array has length f and stores, for each i, either the matched index $\mu(i)$, a skip symbol $\perp$, or an undecided symbol $*$. The $\mathcal{G}$-array is defined analogously. In the implementation these two symbols are encoded by sentinel values; the mathematical description uses $\perp$ and $*$.

For a node $u = (t, \mu, m)$, define the fixed cost

$$C_{\text{fix}}(u) = \sum_{i \in D} \delta_{i,\mu(i)} + \sum_{i \in \{1,\ldots,t\} \setminus D} \sigma_i^F + \sum_{j \in \{1,\ldots,m\} \setminus \mu(D)} \sigma_j^G.$$

This is the cost of all decisions that can no longer change.

The remaining undecided stems are

$$U_F(u) = \{t+1, \ldots, f\}, \qquad U_G(u) = \{m+1, \ldots, g\}.$$

To prioritize the search, we use a lower bound on the cost of completing a partial alignment. For $i \in U_F(u)$, define

$$\eta_F(i;u) = \min\left\{\sigma_i^F, \min_{j\in U_G(u)} \delta_{ij}\right\},$$

and for $j \in U_G(u)$, define

$$\eta_G(j;u) = \min\left\{\sigma_j^G, \min_{i\in U_F(u)} \delta_{ij}\right\}.$$

The node key is

$$L(u) = C_{\text{fix}}(u) + \max\left\{\sum_{i\in U_F(u)} \eta_F(i;u), \sum_{j\in U_G(u)} \eta_G(j;u)\right\}.$$

The two sums are not added, since doing so would double-count potential future matches. Their maximum is still a valid lower bound and is sufficient for branch-and-bound pruning.

▶ **Lemma 1.** *For every search node u, $L(u)$ is a lower bound on the cost of every terminal coarse alignment extending u.*

Proof. The term $C_{\text{fix}}(u)$ is the exact cost of all matched and skipped stems whose status has already been decided. Consider any terminal extension of u. Each remaining stem $F_i \in U_F(u)$ is either skipped, contributing σ_i^F, or matched to some remaining stem G_j, contributing δ_{ij}. Hence its contribution is at least $\eta_F(i;u)$, and the remaining cost is at least $\sum_{i\in U_F(u)} \eta_F(i;u)$. The same argument applied to the remaining stems of $\mathcal{G}$ shows that the remaining cost is also at least $\sum_{j\in U_G(u)} \eta_G(j;u)$. Therefore it is at least the maximum of these two lower bounds. Adding $C_{\text{fix}}(u)$ gives $L(u)$. ◀

4.4 Node expansion

Let $u = (t, \mu, m)$ be a non-terminal node, so $t < f$. The next stem to be decided is F_i with $i = t + 1$. There is always a skip child $(t + 1, \mu, m)$, whose fixed cost increases by σ_i^F.

A match child is generated for a candidate G_j, with $j > m$, only if the candidate satisfies all topological constraints induced by the stems already matched in μ. Specifically, $F_i \mapsto G_j$ is admissible if

$$\Gamma_F[p, i, \cdot] = \Gamma_G[\mu(p), j, \cdot] \qquad \forall p \in D.$$

If this condition holds, the child node is

$$(t + 1, \mu \cup \{i \mapsto j\}, j).$$

The fixed cost of this child increases by the match cost δ_{ij} and by the skip costs of any stems of $\mathcal{G}$ that are passed over:

$$\delta_{ij} + \sum_{q=m+1}^{j-1} \sigma_q^G.$$

The skipped stems $G_{m+1}, \ldots, G_{j-1}$ cannot be used later because of the direction-preserving constraint.

Since every pair of matched stems is checked when the later stem in $\mathcal{F}$ is added, every terminal node produced by this expansion rule is topologically consistent.

4.5 Alignment tree and initialization

The search space is a rooted tree. The root is $u_0 = (0, \emptyset, 0)$. A root-to-leaf path decides the status of each stem $F_1, \ldots, F_f$ exactly once. When a leaf has $t = f$, all remaining unmatched stems $G_{m+1}, \ldots, G_g$ are skipped. Therefore the terminal cost of a leaf $u = (f, \mu, m)$ is

$$C_{\text{term}}(u) = C_{\text{fix}}(u) + \sum_{j=m+1}^{g} \sigma_j^G.$$

This is equal to the terminal alignment cost $C(\mu)$.

For implementation, we use a shortcut initialization rather than generating leading skip decisions one at a time. The root children are all possible first matched pairs $F_i \mapsto G_j$, with $1 \le i \le f$ and $1 \le j \le g$. In the child corresponding to $F_i \mapsto G_j$, stems $F_1, \ldots, F_{i-1}$ and $G_1, \ldots, G_{j-1}$ are immediately fixed as skipped. Subsequent expansion then proceeds from F_{i+1} and considers only stems of $\mathcal{G}$ with index larger than j. Thus the shortcut suppresses paths whose only purpose is to choose the first matched pair after a prefix of skipped stems.

This initialization restricts the implemented search to nonempty partial stem maps: every generated terminal alignment contains at least one matched pair. The all-skip map $\mu = \emptyset$, with cost $C(\emptyset) = \sum_{i=1}^{f} \sigma_i^F + \sum_{j=1}^{g} \sigma_j^G$, is therefore not generated by the shortcut initialization.

4.6 Branch-and-bound traversal

The traversal procedure is given in Algorithm 1. The priority queue stores active search nodes ordered by their lower-bound key $L(u)$. The variable C^* stores the best terminal alignment cost found so far, and $\mathcal{S}^*$ stores all terminal alignments attaining this cost. A node is discarded whenever its lower bound exceeds C^*, since no completion of that node can improve the current best solution. To enumerate all optimal alignments, nodes with lower bound equal to C^* are retained until they are either expanded or shown not to lead to another terminal alignment of cost C^*.

If only one optimal alignment is required, the traversal may stop once a terminal coarse alignment of cost C^* has been found and every active node has lower bound at least C^*. To enumerate all optimal alignments, the search continues to expand nodes with lower bound equal to C^*, and records every terminal alignment with terminal cost C^*.

▶ **Theorem 2.** *The branch-and-bound traversal returns a minimum-cost alignment over the set of injective, direction-preserving, and topologically consistent stem maps.*

Proof. The expansion rule enumerates exactly the direction-preserving choices for each stem F_i: either F_i is skipped, or it is matched to a later unused stem of $\mathcal{G}$. A match child is generated only when its relation-array entries agree with all previously matched stems, so every terminal node is topologically consistent. Conversely, any direction-preserving and topologically consistent complete map determines a unique root-to-leaf path in the tree.

By Lemma 1, $L(u)$ is a lower bound on the cost of every terminal coarse alignment extending u. Therefore, when a node has $L(u) > C^*$, no descendant of that node can improve the best terminal coarse alignment already found. The traversal only prunes such nodes. Hence no alignment with cost smaller than C^* is discarded. When the queue is empty, or when all remaining nodes have lower bound greater than C^*, no unexplored completion can have cost below C^*. Thus C^* is the optimal alignment cost. ◀

Algorithm 1 Best-first branch-and-bound traversal of the alignment tree.

Require: Root node u_0, lower-bound function $L(\cdot)$, terminal-cost function $C_{\text{term}}(\cdot)$
Ensure: Set $\mathcal{S}^*$ of minimum-cost coarse alignments and optimal cost C^*

$Q \leftarrow$ empty priority queue ordered by increasing $L(\cdot)$
$C^* \leftarrow +\infty$
$\mathcal{S}^* \leftarrow \emptyset$
insert u_0 into Q with key $L(u_0)$
while $Q \neq \emptyset$ and $\min_{u \in Q} L(u) \leq C^*$ **do**
    $u \leftarrow \textsc{ExtractMin}(Q)$
    if $L(u) > C^*$ **then**
        continue
    end if
    if u is terminal **then**
        $c \leftarrow C_{\text{term}}(u)$
        if $c < C^*$ **then**
            $C^* \leftarrow c$
            $\mathcal{S}^* \leftarrow \{\mu_u\}$
        else if $c = C^*$ **then**
            $\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \{\mu_u\}$
        end if
    else
        for all $v \in \textsc{Expand}(u)$ **do**
            if $L(v) \leq C^*$ **then**
                insert v into Q with key $L(v)$
            end if
        end for
    end if
end while
return $(C^*, \mathcal{S}^*)$

4.7 Example of topologically constrained expansion

Figure 1 illustrates one expansion step. For a partial alignment with matched-domain D, the next stem F_i defines a constraint vector $\big(\Gamma_F[p, i, \cdot]\big)_{p \in D}$. For each candidate G_j, the algorithm constructs the corresponding vector $\big(\Gamma_G[\mu(p), j, \cdot]\big)_{p \in D}$. The candidate G_j is accepted exactly when these two vectors are identical. Otherwise, matching F_i to G_j would change at least one previously established topological relation, and the child is not generated. The skip child for F_i is always generated.

5 Experimental Setup

Table 1 summarizes the Rfam families used in the benchmark and the range of sequence lengths and stem counts after preprocessing.

5.1 Benchmark instances

For each Rfam family, we selected the first ten sequences and extracted the consensus structure, converting it from WUSS notation into dotbracket and applied the preprocessing procedure described in Section 3. The resulting structures from the same family were not

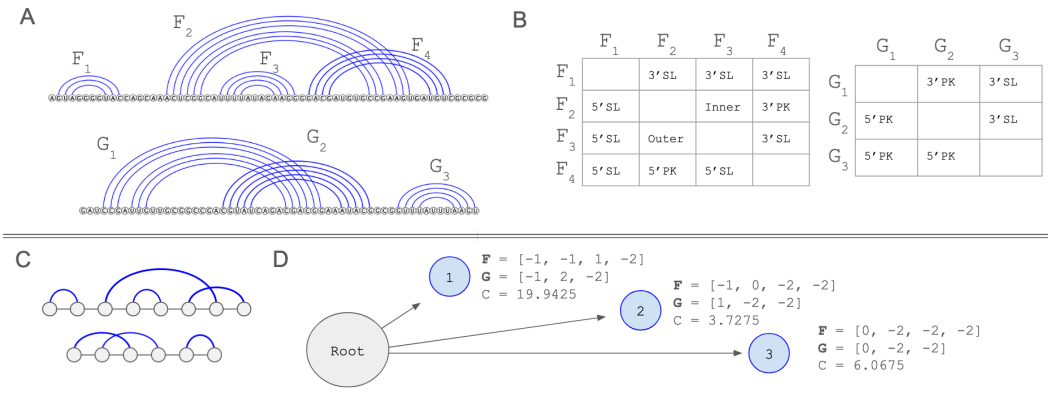

Figure 1 Example of topologically constrained expansion. **A**) Two RNA structures F and G as arc diagrams (each arc connects two ends of a base pair) with stems labelled. **B**) Relation-array summaries. Rows give the reference stem, and columns give the relation of the column stem with respect to the row stem. Reciprocal entries contain reciprocal relation labels. **C**) Reduced topology representations of F and G used by the alignment algorithm. **D**) Example search-tree nodes. Each node stores the current partial stem map, skipped stems, undecided stems, and its lower-bound key. A candidate child is generated only if its relation-array constraints match those induced by the previously matched stems.

necessarily identical at the stem level: insertions, deletions, and small bulges can change how base pairs are grouped into stems under the chosen bulge compression parameter. We then constructed cross-family alignment instances. For each unordered pair of distinct Rfam families, we formed all 10×10 cross-family comparisons by aligning each selected structure from one family with each selected structure from the other. This produced $\binom{7}{2} \times 10 \times 10 = 2100$ cross-family pairwise alignment instances. We used cross-family rather than within-family pairs because they give a more stringent test of the search procedure: stem features, stem counts, and pairwise topological relations are expected to be less conserved across families.

5.2 Measured quantities

For a pair of preprocessed inputs $\mathcal{D}_F = (\mathcal{F}, T_F, \Gamma_F, \Phi_F)$, $\mathcal{D}_G = (\mathcal{G}, T_G, \Gamma_G, \Phi_G)$, let $\mathcal{F} = (F_1, \ldots, F_f)$ and $\mathcal{G} = (G_1, \ldots, G_g)$. We use $I(\mathcal{F}, \mathcal{G}) = fg$ as a simple proxy for the number of initial match choices in the shortcut initialization of the alignment tree. This quantity does not determine the full search-tree size, but it captures the first-order dependence on the number of stems in the two inputs.

For each ordered input pair, let $\mathcal{T}_{\mathcal{F},\mathcal{G}}$ be the alignment tree generated by CoSTAR. We recorded the number of generated search nodes and the maximum tree width

$$W(\mathcal{F}, \mathcal{G}) = \max_d \left| \{ u \in \mathcal{T}_{\mathcal{F},\mathcal{G}} : \mathrm{depth}(u) = d \} \right| .$$

Thus $W(\mathcal{F}, \mathcal{G})$ is the largest number of partial alignments present at any search depth. To measure the size of the relation-consistent search space rather than the behavior of a particular branch-and-bound run, we disabled only the cost-based pruning rule $L(u) > C^*$ for these measurements. The directionality constraint and the relation-array constraints remained active. Consequently, the reported node counts and tree widths measure the search tree induced by valid topological extensions.

Table 1 Rfam families used in the benchmark and their preprocessing statistics. The first column represent the Rfam family ID; the second column (N. Seq) presents the number of sequences in that family; the third column (PK) identifies whether the family contains a pseudoknot. We use 10 samples from each family and the following columns characterize those selected samples: the fourth column (Seq. Id.) identifies the sequence similarity in the family subset; the fifth and the sixth columns present the minimum and maximum lengths of sequences in the family subset; and the last two columns present the minimum and maximum number of stems in the family subset as identified in the Rfam structure.

Rfam family	N. Seq.	PK	Seq. Id.	Min. NT	Max. NT	Min. stems	Max. stems
RF00001	712	No	0.92	116	118	5	6
RF00003	100	No	0.87	162	164	5	5
RF00008	85	No	0.57	40	82	3	4
RF00165	52	Yes	0.71	61	63	2	2
RF00174	434	Yes	0.72	180	224	12	15
RF00381	13	Yes	0.85	56	59	2	2
RF00507	51	Yes	0.69	78	81	3	4

6 Results

We evaluated the coarse alignment algorithm on the Rfam benchmark described in Section 5.

Search-tree growth. Figure 2 summarizes the dependence of running time and tree width on stem-level input size. The left panel plots running time as a function of $I(\mathcal{F}, \mathcal{G}) = fg$. Instances with larger $I(\mathcal{F}, \mathcal{G})$ tend to induce wider search trees and longer running times. This is consistent with the structure of the algorithm: after preprocessing, the dominant cost is not sequence length, but the number of partial stem maps that survive the directionality and topological consistency constraints. The dependence is not solely a function of f and g, since two pairs with the same stem counts can have different relation arrays and therefore different numbers of admissible extensions. For this panel, the search-driving structure $\mathcal{F}$ is set as the structure with more stems.

Effect of input ordering. The alignment problem is symmetric in its final objective, but the search tree is not symmetric in the ordered inputs. The algorithm decides stems of $\mathcal{F}$ sequentially and branches over admissible unused stems of $\mathcal{G}$. Thus, placing the structure with more stems in the search-driving role $\mathcal{F}$ increases the depth of the tree but decreases the number of candidate targets considered at each expansion. In the benchmark instances, this ordering reduced maximum tree width. The right panel of Figure 2 compares the alignment tree width and alignment time to calculate $\mathcal{F} \to \mathcal{G}$ with the width and time for $\mathcal{G} \to \mathcal{F}$, where $\mathcal{F}$ is the structure with more stems. Color indicates the number of stems in $\mathcal{F}$ minus the number in $\mathcal{G}$. When the search-driving structure has more stems, the tree width tends to be lower.

Preprocessing cost. The non-search preprocessing steps are comparatively inexpensive. For a structure with n nucleotides, p base pairs, and M stems, parsing the structure and computing stem features are linear in the nucleotide-level representation after the base pairs have been ordered. The relation array is computed by comparing all ordered pairs of stems, requiring $O(M^2)$ time and $O(M^2)$ space. Since the relation array Γ_R and feature list Φ_R depend only on a single input structure, they can be cached and reused when the same RNA participates in multiple pairwise alignments.

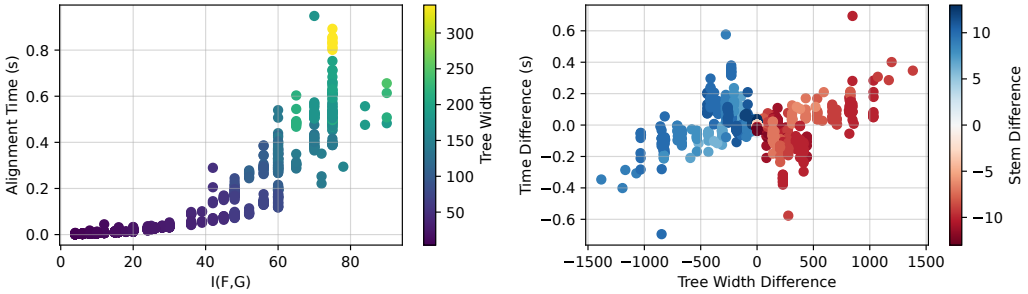

Figure 2 Empirical search behavior on 2100 cross-family Rfam alignment instances. **Left:** Alignment time as a function of the initial stem-pair count $I(\mathcal{F}, \mathcal{G}) = fg$. Points are colored by maximum tree width $W(\mathcal{F}, \mathcal{G})$, computed with cost-based pruning disabled but with directionality and topological constraints enforced. **Right:** Effect of input structure ordering on alignment tree. One structure is set as the search-driving structure ($\mathcal{F} \to \mathcal{G}$) and tree width and alignment time are calculated. The reciprocal ($\mathcal{G} \to \mathcal{F}$) is also evaluated. For a pair of structures we plot two points showing the change in tree width and alignment time relative to the alternative structure ordering. Color shows which structure has more stems where positive values (blue) indicates the search-driving structure has more stems.

The alignment tree is the main source of computational cost. If relation-array constraints are ignored but directionality is retained, the number of complete direction-preserving partial maps is $\sum_{k=0}^{\min(f,g)} \binom{f}{k}\binom{g}{k}$, because choosing k stems from $\mathcal{F}$ and k stems from $\mathcal{G}$ determines a unique order-preserving matching. The generated search tree also contains prefixes of such maps. Relation-array constraints reduce this space in an input-dependent way by rejecting candidate matches that would alter the pairwise topology of already matched stems.

Adversarial structures. The topological constraints are least effective when many stems have indistinguishable relation profiles. For example, if both inputs contain many repeated stems that are mutually disjoint along the backbone, then most pairwise relations are only $5'$-disjoint or $3'$-disjoint. In such cases, relation-array constraints reject few candidate extensions, and the search approaches the directionality-only regime.

As a stress test, we aligned two structures with 11 stems each whose relation patterns were largely repetitive and disjoint. With cost-based pruning disabled, the alignment required approximately six minutes in our implementation, generated about 10^6 search nodes, and reached a maximum tree width of approximately 2.8×10^5. This instance illustrates the data-dependent nature of the search: the number of stems determines the broad scale of the search tree, but the pairwise topology of those stems determines how much pruning is obtained from the relation-array constraints.

6.1 Qualitative comparison with LaRA

We first compare CoSTAR with LaRA, the closest available method for aligning RNA sequences with pseudoknotted secondary structures. LaRA optimizes a nucleotide-level sequence–structure alignment, whereas CoSTAR optimizes a partial map between stems. Thus, the two methods do not return the same type of object. We outline a simple projection method in appendix A that converts our mapping into a nucleotide-level alignment. The examples below show that, even when LaRA is run in a structure-prioritized setting, its nucleotide-level alignment can fail to recover the stem correspondences required for coarse structural comparison.

Synthetic pseudoknotted pair. We constructed a synthetic pair in which the two RNAs share a pseudoknotted structural core but differ in flanking and additional stems. This design tests whether an aligner can ignore non-corresponding structural elements and align the shared pseudoknotted block. Listing 1 shows the LaRA alignment. The shared paired regions are not aligned as coherent stem blocks: gaps are inserted within paired regions, and the pseudoknotted arms are fragmented across the alignment. As a result, the output does not provide a usable stem-to-stem correspondence for this example.

Listing 1 LaRA nucleotide-level alignment of the synthetic pseudoknotted pair.

```
F LaRA
--AGUAGCUCGGCCAUUUUAGAGUCGGACGAUUUGUGGCGAAGUGUGUCGGCGCGAUGCUUGACAGUUCGG----AUGAUCCGAUUG
__........(((.((((.....[[[.[[..........)))..))))..............]].]]]____...........
G LaRA
CAGACAGGUUUAACGA-------GCGGUU--UUUGAGUCAUCACAUGCCUAAACGAUGCUGGGCUCCGCGGCGACUCGCUCUACCGA
.(((....))).....(_______((((((__.....[[[.[[....[[[[.)))..))))]]]]...([]].]]]..)).......
```

Listing 2 shows the nucleotide-level projection obtained from the CoSTAR stem map on the same input pair. Vertical bars mark the boundaries induced by the coarse alignment. The shared pseudoknotted arms are aligned within corresponding blocks, while additional stems and unmatched regions are separated by gaps or distinct blocks. This illustrates the intended use of CoSTAR: the structural correspondence is determined at the stem level before nucleotide columns are introduced.

Listing 2 CoSTAR nucleotide-level projection of the synthetic pseudoknotted pair. Vertical bars mark blocks induced by the coarse stem map.

```
F CoSTAR
|AGUAGCUC_______|GGCCAUUU|UAGAG|UCGGAC|GAUUUGUGGCG|AAGUGUGUC|GGCGCGAUGCUUGACA|GUUCGG|AUGAUCCGAUUG|
|........_______|((((.((((|.....|[[[.[[|...........|)))..))))|.................|]].]]]|...........|
G CoSTAR
|CAGACAGGUUUAACG|AGC_GGUU|UUUGA|GUCAUC|ACAUGCCUA__|AACGAUGCU|GGGCUCCGC_______|GGCGAC|UCGCUCUACCGA|
|.(((....))).....|(((_((((|.....|[[[.[[|....[[[[.__|)))..))))|]]]].((_______|]].]]]|..)).........|
```

Stem-deletion perturbation. We next tested whether the methods can align a structure to a perturbed copy with a deleted stem. We selected a structure from Rfam family RF00507 and created a knockout by removing the first stem together with external unstructured nucleotides. The desired coarse alignment should skip the deleted 5′ stem and align the remaining stems after a leading gap.

Listing 3 shows the alignment produced by LaRA. LaRA introduces a large 5′ gap, but the gap is placed within the remaining paired region rather than before the corresponding structural block. Consequently, the deleted stem is not represented as a clean skipped structural element, and the alignment does not recover the intended stem-level offset.

Listing 3 LaRA nucleotide-level alignment of the RF00507 stem-deletion perturbation.

```
Reference LaRA
GGGUUCGGGGUACUAGUGUGAAUGCCCGGCUAGUACCCUGUGCUAGUGGUUUAUCUACUGAUGUUCAAUUAAGGGCAUUUG
.......((((((((((......[[[[[..)))))))))...(.((...((......))...)).).......]]]]]....
Knockout LaRA
UG--------------------C--CCGGCUAGUACCCUGUGCUAGUGGUUUAUCUACUGAUGUUCAAUUAAGGGCA----
((____________________(__((..............(.((...((......))...)).).......)))))____
```

CoSTAR instead represents the deletion as a skipped leading block and aligns the remaining stems directly, as shown in Listing 4. This is the behavior required for the coarse alignment task: the perturbation is localized to the deleted stem, while the rest of the stem map remains interpretable.

Listing 4 CoSTAR nucleotide-level projection of the RF00507 stem-deletion perturbation. The deleted leading stem is represented as a skipped block.

```
Reference CoSTAR
|GGGUUCGGGGUACUAGUGUGAA|UGCCC|GGCUAGUACCCUGU|GCUA|GUG|GU|UUAUCU|AC|UGA|UGUU|CAAUUAA|GGGCA|UUUG|
|.......((((((((((......|[[[[[|..)))))))))...|(.((|...|((|......|))|...|)).)|.......|]]]]]|....|
Knockout CoSTAR
|_____________________|UGCCC|GGCUAGUACCCUGU|GCUA|GUG|GU|UUAUCU|AC|UGA|UGUU|CAAUUAA|GGGCA|____|
|_____________________|(((((|.............|(.((|...|((|......|))|...|)).)|.......|)))))|____|
```

Together, these examples demonstrate the distinction between the two objectives. LaRA can align pseudoknotted RNA at nucleotide resolution, but the resulting alignment need not preserve the stem-level correspondences needed for coarse structural comparison. CoSTAR directly optimizes that stem-level correspondence and can then project the resulting map to nucleotide resolution.

7 Discussion and Future Work

We presented CoSTAR, a coarse alignment algorithm for RNA secondary structures that may contain pseudoknots. CoSTAR represents each input by its ordered stems, pairwise stem relations, and nucleotide-derived stem features, and returns a partial injective map between stems. The method therefore lies between topology-only representations, such as RNA abstract shapes and RNA-as-Graph [13, 11], and nucleotide-level sequence–structure alignment methods such as LaRA 2 [28]. Its main computational advantage is that the dominant input size is the number of stems, not the number of nucleotides: stem features are computed once, relation arrays require $O(M^2)$ time and space for M stems, and these precomputed objects can be cached across alignments.

The search over partial stem maps remains the computational bottleneck and can grow exponentially in the number of stems in the worst case. The empirical results show, however, that directionality and relation-array constraints reduce the explored search space on the benchmark instances. Because the search is asymmetric in its ordered inputs, using the structure with more stems as the search-driving input $\mathcal{F}$ typically reduces tree width, as observed in the reciprocal-orientation experiments. The least favorable cases are structures with many repeated, mutually disjoint stems, where most relations are $5'$-disjoint or $3'$-disjoint and topological constraints reject few candidates.

Several modeling choices can be refined. The current lower bound is admissible but relaxed: it allows each remaining stem to choose its cheapest match or skip cost without enforcing future injectivity or topological consistency. Tighter admissible bounds, such as a minimum-cost bipartite matching over the remaining stems with skip vertices, could reduce the number of expanded nodes. The cost function is also intentionally simple, using weighted distances between stem features and corresponding skip costs. Since RNA similarity is task-dependent [3], future scoring models could incorporate sequence composition, covariation, learned feature weights, or stronger penalties for disrupting pseudoknotted topology.

CoSTAR can also be used as an anchoring step for nucleotide-level alignment. If the coarse map matches F_i to G_j, then the nucleotide positions in the arms of F_i can be constrained or encouraged to align to the corresponding arms of G_j, and intervals between matched stems can be solved as smaller sequence–structure alignment subproblems. We provide a simple nucleotide-projection method, though future work should evaluate this projection on curated homologous RNAs, viral pseudoknot benchmarks, and comparisons against nucleotide-level aligners and topology-based distance methods. Additional extensions include studying the effect of the bulge-compression parameter β, which trades off fewer stems and faster search against the risk of merging stems that should remain distinct.

References

1 Markus J. Bauer, Gunnar W. Klau, and Knut Reinert. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics*, 8:271, 2007. doi:10.1186/1471-2105-8-271.

2 Ian Brierley, Simon Pennell, and Robert J. C. Gilbert. Viral RNA pseudoknots: versatile motifs in gene expression and replication. *Nature Reviews. Microbiology*, 5:598–610, 2007.

3 James W. Brown, Amanda Birmingham, Paul E. Griffiths, Fabrice Jossinet, Rym Kachouri-Lafond, Rob Knight, B. Franz Lang, Neocles Leontis, Gerhard Steger, Jesse Stombaugh, and Eric Westhof. The RNA structure alignment ontology. *RNA*, 15(9):1623–1631, 2009. doi:10.1261/rna.1601409.

4 Thomas R. Cech and Joan A. Steitz. The noncoding RNA revolution—trashing old rules to forge new ones. *Cell*, 157(1):77–94, 2014. doi:10.1016/j.cell.2014.03.008.

5 José Almeida Cruz and Eric Westhof. The dynamic landscapes of RNA architecture. *Cell*, 136(4):604–609, 2009. doi:10.1016/j.cell.2009.02.003.

6 Padideh Danaee, Mason N. Rouches, Michelle Wiley, Dezhong Deng, Liang Huang, and David A. Hendrix. bpRNA: large-scale automated annotation and analysis of RNA secondary structure. *Nucleic Acids Research*, 46:5381–5394, 2018.

7 Jennifer A. Doudna and Thomas R. Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002. doi:10.1038/418222a.

8 Sean R. Eddy. Computational analysis of conserved RNA secondary structure in transcriptomes and genomes. *Annual Review of Biophysics*, 43:433–456, 2014. doi:10.1146/annurev-biophys-051013-022950.

9 Sean R. Eddy and Richard Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994. doi:10.1093/nar/22.11.2079.

10 Stéfan Engelen and Fariza Tahi. Predicting RNA secondary structure by the comparative approach: how to select the homologous sequences. *BMC Bioinformatics*, 8:464, 2007. doi:10.1186/1471-2105-8-464.

11 Daniela Fera, Namhee Kim, Nir Shiffeldrim, Joshua Zorn, Uri Laserson, Hin Hark Gan, and Tamar Schlick. RAG: RNA-As-Graphs web resource. *BMC Bioinformatics*, 5:88, 2004. doi:10.1186/1471-2105-5-88.

12 Laura R. Ganser, Megan L. Kelly, Daniel Herschlag, and Hashim M. Al-Hashimi. The roles of structural dynamics in the cellular functions of RNAs. *Nature Reviews Molecular Cell Biology*, 20(8):474–489, 2019. doi:10.1038/s41580-019-0136-0.

13 Robert Giegerich, Björn Voß, and Marc Rehmsmeier. Abstract shapes of RNA. *Nucleic acids research*, 32 16:4843–51, 2004.

14 Jean-Pierre Séhi Glouzon, Jean-Pierre Perreault, and Shengrui Wang. The super-n-motifs model: a novel alignment-free approach for representing and comparing RNA secondary structures. *Bioinformatics*, 33:1169–1178, 2017.

15 Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNAsecondary structures. *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, pages 159–168, 2003. doi:10.1109/CSB.2003.1227315.

16 Brittany Lasher and David A. Hendrix. bpRNA-align: improved RNA secondary structure global alignment for comparing and clustering RNA structures. *RNA*, 29(5):584–595, 2023. doi:10.1261/rna.079211.122.

17 David H. Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of molecular biology*, 288 5:911–40, 1999.

18 Eugenio Mattei, Gabriele Ausiello, Fabrizio Ferrè, and Manuela Helmer-Citterich. A novel approach to represent and compare RNA secondary structures. *Nucleic Acids Research*, 42:6146–6157, 2014.

19 John S. Mattick and Igor V. Makunin. Non-coding RNA. *Human Molecular Genetics*, 15(suppl_1):R17–R29, 2006. doi:10.1093/hmg/ddl046.

20 Kevin V. Morris and John S. Mattick. The rise of regulatory RNA. *Nature Reviews Genetics*, 15(6):423–437, 2014. doi:10.1038/nrg3722.

21 Kayla M. Peck and Adam S. Lauring. Complexities of viral mutation rates. *Journal of Virology*, 92, 2018.

22 Michela Quadrini, Luca Tesei, and Emanuela Merelli. ASPRAlign: a tool for the alignment of RNA secondary structures with arbitrary pseudoknots. *Bioinformatics*, 36(11):3578–3579, 2020. doi:10.1093/bioinformatics/btaa147.

23 Ladislav Rampásek, Randi M. Jimenez, Andrej Lupták, Tomás Vinař, and Broňa Brejová. RNA motif search with data-driven element ordering. *BMC Bioinformatics*, 17, 2016.

24 David Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985. doi:10.1137/0145048.

25 Bruce A. Shapiro and Kaizhong Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Bioinformatics*, 6(4):309–318, 1990. doi:10.1093/bioinformatics/6.4.309.

26 Ignacio Jr. Tinoco and Carlos Bustamante. How RNA folds. *Journal of Molecular Biology*, 293(2):271–281, 1999. doi:10.1006/jmbi.1999.3001.

27 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. doi:10.1371/journal.pcbi.0030065.

28 Jörg Winkler, Gianvito Urgese, Elisa Ficarra, and Knut Reinert. LaRA 2: parallel and vectorized program for sequence–structure alignment of RNA sequences. *BMC Bioinformatics*, 23, 2022. doi:10.1186/S12859-021-04532-7.

A Postprocessing Projection

An intuitive way to represent the output of our stem mapping is by projecting the mapping back to a nucleotide-level alignment. The alignment is broken up into segments of mapped and unmapped regions. Segments are separated by pipes for visual convenience. Unmapped regions are those that contain either no structure, or contain structure that was not included in the mapping. An example of structure not part of the mapping can be observed in Listing 2 where an unmapped loop in the G structure is found in the most 5′ section. By construction there are an odd number of segments in a projection, alternating between unmapped and mapped, with both 5′ and 3′ extremes being unmapped.

When introducing spacer symbols to the two inputs, we follow two principles: 1) in an unmapped section, spacers are added to the shorter of the two inputs on the 3′ side equal to the difference; 2) in mapped sections, spacers fill internal loops and bulges first, then additional spacers are added to the 3′ side. Since our focus is on structures, we do not further refine the nucleotide sequences of the unmapped segments, which could be performed by merging gaps with adjacent segments or applying existing pairwise sequence alignment tools.

B Examples of Nucleotide-Level Alignment Behavior

LaRA 2 was used as a point of comparison because it supports alignment of RNA sequences with pseudoknotted secondary structures [28]. However, LaRA solves a nucleotide-level sequence–structure alignment problem, whereas our method computes a stem-level structural map. The two outputs are therefore not identical objects. The examples below illustrate cases in which a nucleotide-level alignment can be difficult to interpret when the intended comparison is a coarse stem correspondence.

Let S^F, R^F and S^G, R^G denote the two input sequence–structure pairs. A nucleotide-level aligner returns aligned strings $\widetilde{S}^F, \widetilde{R}^F, \widetilde{S}^G, \widetilde{R}^G$, where gap symbols are inserted into the sequence and structure strings. A base pair $(i, j) \in R^F$ is preserved by the alignment if the columns containing S^F_i and S^F_j are aligned to two non-gap nucleotides S^G_p and S^G_q such that $(p, q) \in R^G$. In the examples below, we use this criterion only as a qualitative diagnostic: the purpose is not to benchmark LaRA, but to show why a stem-level output can be more appropriate for some structural comparison tasks.

LaRA has three relevant parameters in these examples. The parameter `balance` controls the relative contribution of sequence and structure. We set `balance=INF` to prioritize structural agreement. The parameters `gapOpen` and `gapExtend` define the affine cost for opening and extending gaps, respectively.

Default gap parameters. The first run used the two structures shown in Figure 1, with `balance=INF` and the default LaRA gap parameters. Lines labelled F and G give the aligned sequence and structure strings for the two inputs.

Listing 5 LaRA output with `balance=INF` and default gap parameters.

```
F sequence:  AGUAGGGGUACCAGCAAACUCGGCAUUUUAUAGAAGGGGACGAUGUGCCGAAGUGAUGUCGCGCG
F structure: .(((....))).....((((((..((((....)))).[[[[.....))).))))..]]]].....
G sequence:  GAUCCGAUUGUU_GCGGCCGACGUAUCAGAC_GACGGAAAUACGGCGGUUUAUUUA______AGU
G structure: ...(((.(((..._........[[[[[....)_)))))..]]]]]...((((....)______)))
```

In this output, gaps are inserted inside structural regions of G. For example, an insertion occurs within a bracketed segment of the G structure, splitting a stem arm in the aligned representation. A second long insertion appears near the end of the structure string. As a result, the displayed alignment does not preserve the intended correspondence between the stem arms and pseudoknotted arms visible in Figure 1. This is a nucleotide-level behavior relative to the stem-level objective considered in this paper: the alignment is syntactically valid, but it does not provide a useful mapping between stems.

More permissive gap parameters. We repeated the same comparison with lower gap costs, using `gapOpen=3` and `gapExtend=1`. This makes long gaps less costly and should, in principle, allow the nucleotide-level alignment to place larger insertions outside conserved structural blocks.

Listing 6 LaRA output with `balance=INF`, `gapOpen=3`, and `gapExtend=1`.

```
F sequence:  AGU__AGGGGUACCAGCAAACUCGGCAUUUUAUAGAAGGGGACGAUGUGCCGAAGUGAUGUCGCGCG
F structure: .((__(....))).....((((((..((((....)))).[[[[.....))).))))..]]]].....
G sequence:  GAUCCGAUUGUUGCGGCCGAC___GUAUCAGAC_GA___CGGAAAUACGGCGGU_UUAU_UUAAGU_
G structure: ...(((.(((.........[___[[[[....)_))___)))..]]]]]...((_((..._..))))_
```

Lowering the gap penalties does not remove this behavior in the example. Instead, gaps are still placed inside bracket runs, and the resulting alignment continues to split regions that should be treated as coherent stems. This behavior reflects the difference between optimizing a nucleotide-level alignment objective and directly optimizing a map between stems.

Projection of the coarse alignment. For comparison, Listing 7 shows a nucleotide-level projection of the coarse alignment produced by our method. The method first computes a partial map between stems. Matched stems are then projected to nucleotide resolution by aligning corresponding stem arms as blocks. The vertical bars mark the boundaries of coarse stem blocks.

Listing 7 Nucleotide-level projection of the stem-level alignment. Vertical bars mark coarse stem boundaries.

```
F sequence:  |AGUAGGGGUACCAGCAA|ACU_CGG|CAUUUUAUAGAAGGG|GACG_|AUGUG|CCGAAGU|GA|UGUC_|GCGCG__________|
F structure: |.(((....))).....|(((_(((|..((((....)))).|[[[[_|.....|))).)))|..|]]]]_|.....__________|
G sequence:  |GAU_____________|CCGAUUG|UUGCGGCCGA_____|CGUAU|CAGA_|CGA_CGG|AA|AUACG|GCGGUUUAUUUAAGU|
G structure: |..._____________|(((.(((|.........._____|[[[[[|...._|)))_)))|..|]]]]]|...((((....))))|
```

The projection is not intended to replace a full nucleotide-level sequence–structure aligner. Rather, it shows the type of information produced by the coarse algorithm: corresponding stems are identified first, and nucleotide-level columns are introduced only after the stem map has been fixed. This avoids placing gaps inside a stem arm when the downstream task is to compare stems as coherent units.

In other LaRA runs, we also observed a 5′-anchoring behavior, where an early nucleotide was aligned before a leading gap even though introducing the gap would better align the downstream stems. Such cases further motivate the use of stem-level alignment as a preprocessing or anchoring step when the primary biological question concerns structural correspondence rather than nucleotide identity.

C Rfam Samples

In our results we evaluated our method on Rfam RNA structures, specifically the first 10 of each family. Table 2 contains the accession numbers for the used samples.

Table 2 Rfam family sample accession numbers.

RF00001	RF00003	RF00008	RF00165	RF00174	RF00381	RF00507
X01556.1	X06810.1	D00685.1	NC_022103.1	AF010496.1	BC056833.1	NC_006577.2
3-118	261-421	1-46	27934-27996	39869-39653	241-299	13601-13681
X55260.1	X14417.1	AJ247116.1	NC_009657.1	AF010496.1	AF242520.1	NC_003045.1
3-119	177-340	133-53	27966-28028	105318-105540	225-280	13342-13422
M16174.1	Z11883.1	Y14700.1	NC_028814.1	AF010496.1	AF258463.1	NC_006213.1
3-119	3441-3603	133-53	27384-27446	116971-117193	230-288	13341-13421
X55267.1	X14419.1	AJ247123.1	NC_038294.1	AF193754.1	D10706.1	NC_026011.1
3-119	178-338	132-52	29868-29928	4343-4143	256-314	13485-13565
M16172.1	X14416.1	AJ247121.1	NC_019843.3	AF193754.1	AC004152.1	NC_004718.3
3-119	171-331	133-53	29875-29935	24966-24789	16641-16699	13399-13476
AF001265.1	X14415.1	AJ247122.1	NC_003436.1	AF263012.1	AF291576.1	NC_022103.1
6033-6149	175-335	132-52	27820-27882	9229-9016	215-270	12533-12607
X05057.1	X14414.1	AJ247113.1	NC_038861.1	AF306632.1	D11373.1	NC_002306.3
3-119	152-312	134-53	28363-28425	382-182	473-530	12410-12484
X15126.1	Z11883.1	AJ536620.1	NC_028806.1	AJ000758.1	BC054690.1	NC_038861.1
3-120	2839-2998	1-40	27893-27955	1191-1369	204-259	12339-12413
Z50737.1	Z11883.1	AJ536617.1	NC_002306.3	AJ000758.1	AB017117.1	NC_028806.1
3-119	1496-1656	1-40	29138-29200	1489-1667	228-286	12325-12399
X55261.1	Z11883.1	Y12833.1	NC_032107.1	AJ278144.1	AF196332.1	NC_017083.1
3-119	361-521	339-285	28249-28311	22351-22164	69-127	13520-13600