RNA Tree Comparisons Via Unrooted Unordered Alignments
(Supplemental Materials)

Nimrod Milo[1⋆], Shay Zakov[2⋆], Erez Katzenelson[1], Eitan Bachmat[1], Yefim Dinitz[1], and Michal Ziv-Ukelson[1⋆⋆]

[1] Dept. of Computer Science, Ben-Gurion University of the Negev, Israel
{milon, erezkatz, ebachmat, dinitz, michaluz}@cs.bgu.ac.il
[2] Dept. of Computer Science and Engineering, UC San Diego, La Jolla, CA, USA
szakov@eng.ucsd.edu

S1 Supplementary Figures

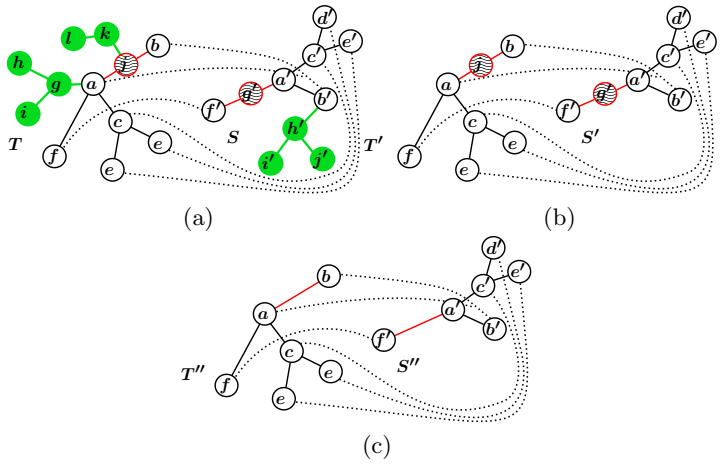


Fig. S1: Homeomorphic Subtree Alignment. Dashed lines represent the alignment (a) An HSA $A = \{(a, a'), (b, b'), \ldots, (g, g')\}$ between two trees T and S. The set of pruned subtrees (in green fillings) with respect to A is $\pi(A) = \left\{T_h^a, T_l^k, S_{i'}^{c'}\right\}$. The set of smoothed nodes (in red and waved) with respect to A is $\delta(A) = \{k, h'\}$. (b) The subtrees T' and S' of T and S, respectively, obtained after pruning the subtrees in $\pi(A)$. (c) The smoothings T'' and S'' of T' and S', respectively, obtained after smoothing the nodes in $\delta(A)$. It may be asserted that A is an isomorphic alignment between T'' and S''.

⋆ These authors contributed equally to the paper.
⋆⋆ Corresponding author.

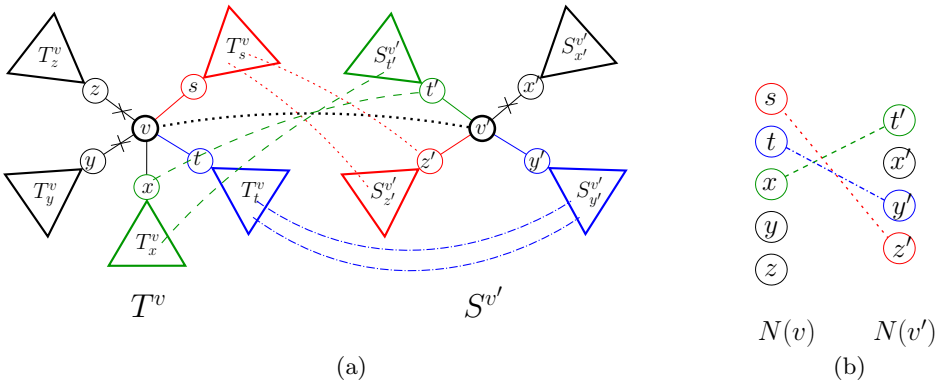


Fig. S2: An illustration of a rooted alignment A between T^v and $S^{v'}$. (a) Each subtree of the form T_u^v ($S_{u'}^{v'}$) is either pruned by the alignment, or matched to exactly one subtree of the form $S_{u'}^{v'}$ (T_u^v). In this example, T_s^v is matched to $S_{z'}^{v'}$ (in red), T_t^v is matched to $S_{y'}^{v'}$ (in blue), and T_x^v is matched to $S_{t'}^{v'}$ (in green). These three subtree-matchings induce three sub-alignments of A: A_s^v, A_t^v, and A_x^v, respectively, where A is the union of these three sub-alignments and the pair (v, v'). The pruned subtrees in this example are T_y^v, T_z^v, and $S_{x'}^{v'}$. (b) The corresponding bipartite matching $M_{v,v'} = \{(s, z'), (t, y'), (x, t')\}$ between $N(v)$ and $N(v')$.

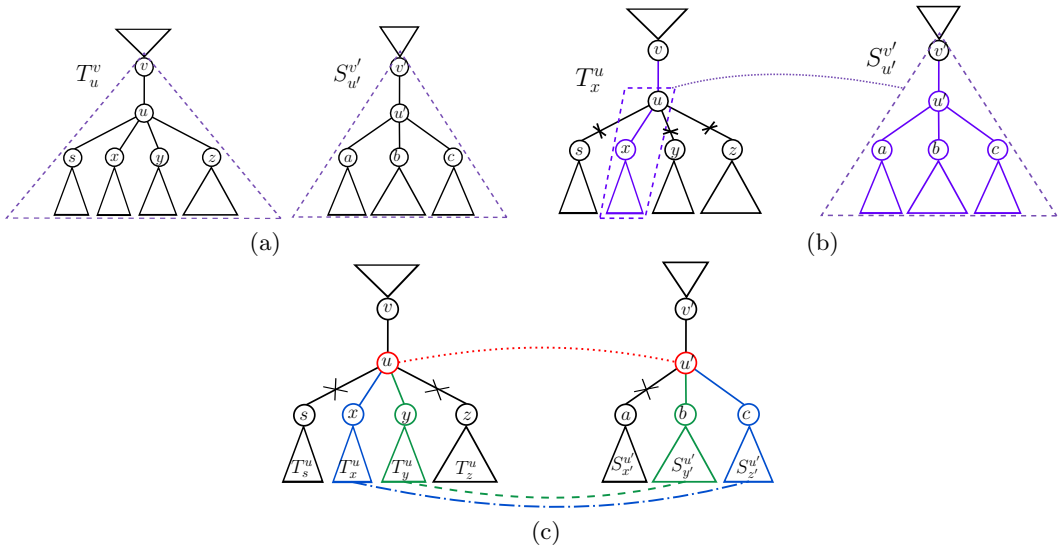


Fig. S3: An illustration of the computation of Equation 6. (a) The instance $\left(T_u^v, S_{u'}^{v'}\right)$ in the left-hand side of the equation. (b) The computation of term I in the right-hand side of the equation. The term considers the best alignment score under the assumption that u is smoothed. In this case, the score is obtained by taking the smoothing cost of u, and summing it for some $x \in N(u) \setminus \{v\}$ with the alignment score between T_x^u and $S_{u'}^{v'}$ and the pruning cost of all subtrees T_y^u for $y \in N(u) \setminus \{v, x\}$. x is chosen to be the neighbor of u that induces a minimum cost with respect to this computation. The computation of term II is symmetric. (c) The computation of term III in the right-hand side of the equation. This term considers the case were neither u nor u' are smoothed, and therefore these two nodes are aligned to each other. In this case, the score is computed similarly to the computation in Equation 4, with respect to the sets $N(u) \setminus \{v\}$ and $N(u') \setminus \{v'\}$.

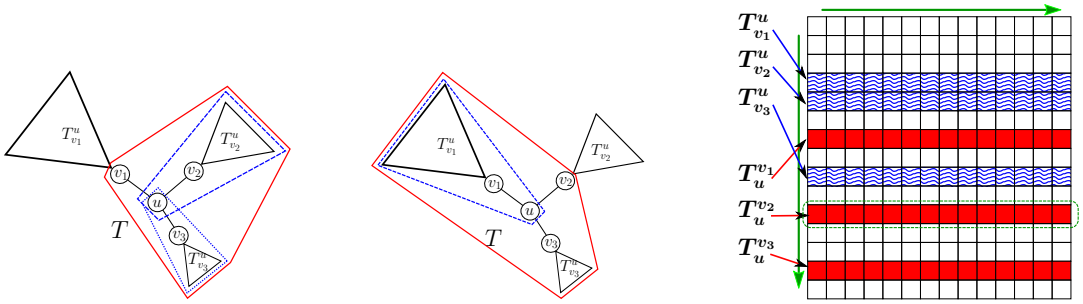


Fig. S4: An illustration of Observation 1. (a) A node u in a tree T, such that $N(u) = \{v_1, v_2, v_3\}$, and $|T_u^{v_1}| \leq |T_u^{v_2}| \leq |T_u^{v_3}|$. The subtree $T_u^{v_1}$ (bounded by a solid red line) contains $T_{v_2}^u$ and $T_{v_3}^u$ as subtrees (bounded by dashed and dotted blue lines, respectively), and therefore $|T_{v_2}^u|, |T_{v_3}^u| \leq |T_u^{v_1}| \leq |T_u^{v_2}|$. (b) The subtree $T_u^{v_2}$ (bounded by a solid red line) contains $T_{v_1}^u$ as a subtree (bounded by a dashed blue line), and therefore $|T_{v_1}^u|, |T_{v_2}^u|, |T_{v_3}^u| \leq |T_u^{v_2}|$. (c) The DP matrix H. The rows of the matrix correspond to subtrees of T, sorted from top to bottom with increasing tree size. Rows corresponding to subtrees of the form $T_u^{v_i}$ are in solid red, and rows corresponding to subtrees of the form $T_{v_i}^u$ are in waved-blue. All waved-blue rows have smaller indices than the row corresponding to $T_u^{v_2}$ (circled with a dashed green line).

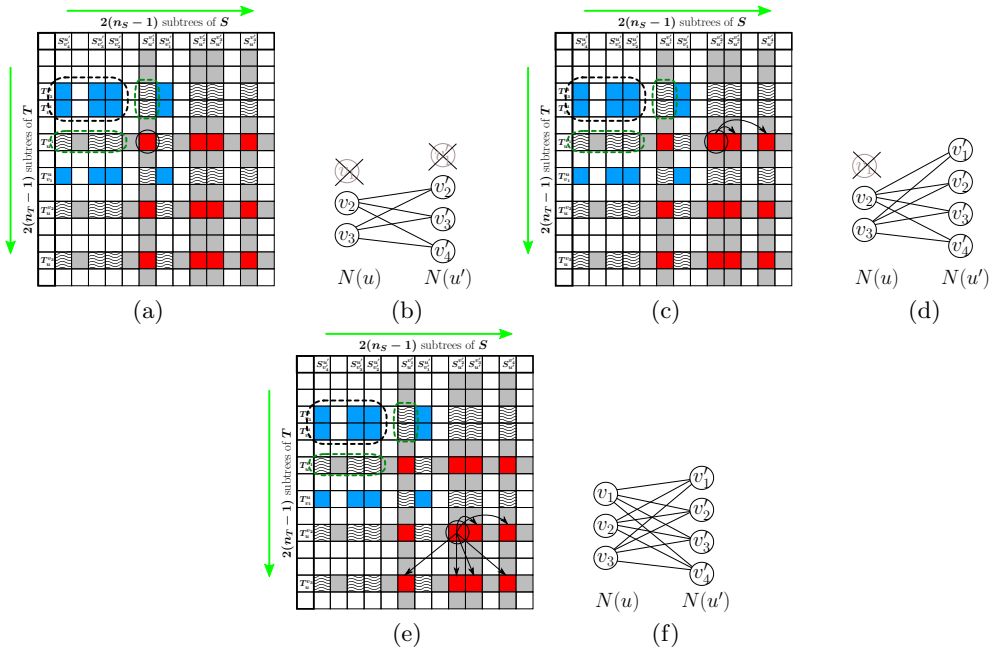


Fig. S5: The incorporation of cavity matching subroutines in the DP algorithm. In (a), (c), and (e), the DP matrix H is illustrated, where the solid red entries correspond to entries in a sub-matrix $H_{u,u'}$ for some $u \in T$ and $u' \in S$. In this example, $N(u) = v_1, v_2, v_3$, and $N(u') = v'_1, v'_2, v'_3, v'_4$, where $|T_u^{v_1}| \leq |T_u^{v_2}| \leq |T_u^{v_3}|$, and $|S_{u'}^{v'_1}| \leq |S_{u'}^{v'_2}| \leq |S_{u'}^{v'_3}| \leq |S_{u'}^{v'_4}|$. The solid blue entries correspond to computed values of the form *Rooted-HSA*$^{-r}(T_{v_i}^u, S_{v'_j}^{u'})$, which are required for the computation of term III in Equation 6 for entries in $H_{u,u'}$. The waived entries correspond to computed values of the form *Rooted-HSA*$^{-r}(T_{v_i}^u, S_{u'}^{v'_j})$ and *Rooted-HSA*$^{-r}(T_u^{v_i}, S_{v'_j}^{u'})$, which are required for the computation of terms I and II in Equation 6 for entries in $H_{u,u'}$. (a) The computation of *Rooted-HSA*$^{-r}(T_u^{v_1}, S_{u'}^{v'_1})$ according to Equation 6. The entry corresponding to this sub-instance is marked with a solid black circle. In order to compute term I of the equation, there is a need to examine values of the form *Rooted-HSA*$^{-r}(T_{v_j}^u, S_{u'}^{v'_1})$ for $v_j \in N(u) \setminus \{v_1\}$. As each $T_{v_j}^u$ is a subtree of $T_u^{v_1}$, the rows corresponding to these subtrees have smaller indices than the row of $T_u^{v_1}$, and so the required solutions are already computed and stored in H (waved entries marked with a dashed green circle, in the same column above the computed entry). Similarly, for computing term II, there is a need to examine solutions *Rooted-HSA*$^{-r}(T_u^{v_1}, S_{v'_j}^{u'})$ for $v'_j \in N(u') \setminus \{v'_1\}$, appearing at the same row and to the left of the computed entry. For computing term III, there is a need to construct the matching cost function $w_{u,u'}^{v_1,v'_1}$, which assigns for each $v_j \in N(u) \setminus \{v_1\}$ and $v'_{j'} \in N(u') \setminus \{v'_1\}$ the matching cost $w_{u,u'}^{v_1,v'_1}(v_j, v'_{j'}) =$ *Rooted-HSA*$^{-r}(T_{v_j}^u, S_{v'_{j'}}^{u'})$ (the corresponding matching instance is shown in (b)). These required values appear in blue entries marked by a dashed black circle. Due to the order in which entries are being traversed, all required values were previously computed by the algorithm and stored in H, thus it is possible to compute *Rooted-HSA*$^{-r}(T_u^{v_1}, S_{u'}^{v'_1})$ at this stage. (c) Upon reaching the entry corresponding to $T_u^{v_1}$ and $S_{u'}^{v'_2}$, all values of the form *Rooted-HSA*$^{-r}(T_{v_j}^u, S_{v'_{j'}}^{u'})$ for $v_j \in N(u) \setminus \{v_1\}$ and $v'_{j'} \in N(u')$ are computed (see (d)). This allows to compute all values $MCM(N(u) \setminus \{v_1\}, N(u') \setminus \{v'_j\}, w_{u,u'}^{v_1,v'_j})$, by solving the *All-Cavity-MCM* problem over the instance $(N(u) \setminus \{v_1\}, N(u'), w_{u,u'}^{v_1})$, and thus computing term III with respect to all remaining entries in the first row of $H_{u,u'}$. (e) Upon reaching the entry corresponding to $T_u^{v_2}$ and $S_{u'}^{v'_2}$, all values of the form *Rooted-HSA*$^{-r}(T_{v_j}^u, S_{v'_{j'}}^{u'})$ for $v_j \in N(u)$ and $v'_{j'} \in N(u')$ are computed (see (f)). This allows to compute all values $MCM(N(u)\setminus\{v_j\}, N(u')\setminus\{v'_{j'}\}, w_{u,u'}^{v_j,v'_{j'}})$, by solving the *All-Pairs-Cavity-MCM* problem over the instance $(N(u), N(u'), w_{u,u'})$, and thus computing term III with respect to all remaining entries in $H_{u,u'}$.

S2 Omitted proofs

Before proving Lemma 1, we formulate an auxiliary observation.

Observation 1 *Let A be an HSA between trees T and S, and $(v,v'),(x,x'),(y,y') \in A$. The path between x and y in T goes through v if and only if the path between x' and y' in S goes through v'.*

The correction of the observation can be asserted from the fact that A is an isomorphic alignment between a smoothed subtree of T that contains v, x, and y, and a smoothed subtree of S that contains v', x', and y'. The full details of the proof are excluded.

Lemma 1. *Let $(v,v') \in A$, and let u be a relevant neighbor of v. Then, there is a unique relevant neighbor u' of v' such that for every $(y,y') \in A$, $y \in T_u^v \Leftrightarrow y' \in S_{u'}^{v'}$.*

Proof. Since u is a relevant neighbor of v, there is a node $x \in T_u^v$, such that $x \neq v$ and x is aligned by A. Let $x' = A(x)$, and let u' be the relevant neighbor of v' such that $x' \in S_{u'}^{v'}$. For $(y,y') \in A$, observe that $y \notin T_u^v$ if and only if the path between x and y in T passes through v. Similarly, $y' \notin S_{u'}^{v'}$ if and only if the path between x' and y' in S passes through v'. Applying Observation 1, this implies that $y \in T_u^v \Leftrightarrow y' \in S_{u'}^{v'}$. □

Proof of Equation 6. By definition of the *Rooted-HSA*$^{\text{-}r}\left(T_u^v, S_{u'}^{v'}\right)$ score, it corresponds to the score of the best non-trivial alignment between T_u^v and $S_{u'}^{v'}$, minus the root alignment cost term $align(v,v')$. We may divide the set of all possible non-trivial rooted alignments between T_u^v and $S_{u'}^{v'}$ into three sets: (a) alignments in which u is unmatched, (b) alignments in which u' is unmatched, and (c) alignments in which both u and u' are matched (note that there might be an intersection between groups (a) and (b)). We show that each one of the terms I, II, and III in the right-hand side of Equation 6 computes the minimum cost of an alignment in each one of the groups (a), (b), and (c), respectively, and therefore the minimum among all these terms gives the correct value *Rooted-HSA*$^{\text{-}r}\left(T_u^v, S_{u'}^{v'}\right)$.

We start by showing term I computes the minimum cost of an alignment in group (a). Let A be an alignment in group (a) of minimum cost. Since A is non-trivial, and u is smoothed in A, u has exactly one additional relevant neighbor x^* besides v. It therefore follows that all nodes in T_u^v except for v which are matched by A belong to the subtree $T_{x^*}^u$. Defining $A_{x^*}^u = (A \setminus \{(v,v')\}) \cup \{(u,v')\}$, we have that $A_{x^*}^u$ is a non-trivial rooted HSA between $T_{x^*}^u$ and $S_{u'}^{v'}$. Therefore, the cost of A is obtained by the summation of pruning costs of all subtrees T_y^u for $y \in N(u) \setminus \{v, x^*\}$, the cost of smoothing u, and the cost $w^{\text{-}r}\left(T_{x^*}^u, S_{u'}^{v'}, A_{x^*}^u\right)$ (which counts for all cost terms corresponding to node matchings, node smoothings, and subtree prunings, implied by the sub-alignment $A_{x^*}^u$). Since A is optimal, it is clear that $w^{\text{-}r}\left(T_{x^*}^u, S_{u'}^{v'}, A_{x^*}^u\right) = \textit{Rooted-HSA}^{\text{-}r}\left(T_{x^*}^u, S_{u'}^{v'}\right)$ (otherwise, it is possible to improve the cost of A by replacing the sub-alignment $A_{x^*}^u$ with an optimal alignment for the corresponding sub-instance). Thus, $w^{\text{-}r}\left(T_u^v, S_{u'}^{v'}, A\right) = smooth(u) + \textit{Rooted-HSA}^{\text{-}r}\left(T_{x^*}^u, S_{u'}^{v'}\right) + \sum_{y \in N(u)\setminus\{v,x^*\}} prune(T_y^u)$. Since for every $x \in N(u)\setminus\{v\}$ the term $smooth(u) + \textit{Rooted-HSA}^{\text{-}r}\left(T_x^u, S_{u'}^{v'}\right) + \sum_{y \in N(u)\setminus\{v,x\}} prune(T_y^u)$ is the $w^{\text{-}r}$ cost of a possible non-trivial alignment between T_u^v and $S_{u'}^{v'}$ in which u is smoothed (where the subtree T_x^u is optimally aligned to $S_{u'}^{v'}$), we get that x^* satisfies that

$$smooth(u) + \sum_{y \in N(u) \setminus \{v, x^*\}} prune(T^u_y) + \textit{Rooted-HSA}^{\text{-}r}\left(T^u_{x^*}, S^{v'}_{u'}\right) =$$
$$smooth(u) + \min_{x \in N(u) \setminus \{v\}} \left(\sum_{y \in N(u) \setminus \{v, x\}} prune(T^u_y) + \textit{Rooted-HSA}^{\text{-}r}\left(T^u_x, S^{v'}_{u'}\right) \right),$$

hence the correctness of term I.

The proof that term II of the equation computes the minimum cost of an alignment in group (b) is symmetric to the proof of term I. As for term III, note that the case where u and u' are matched, but not to each other, implies a contradiction to Observation 1. Therefore, for any alignment in group (c), and in particular for such an alignment A of minimum cost, we have that $(u, u') \in A$. In this case, the optimal alignment cost is obtained immediately from applying Equation 4 for this specific sub-instance, as formulated by term III in the equation. □

Lemma 2. *It is possible to implement Algorithm 1 so that all operations, besides computation of solutions to the* MCM *problem, require $O(n_T n_S)$ running time.*

Proof. It is simple to observe that the computations conducted in lines 1 and 4 of the algorithm consume $O(n_T n_S)$ time (the computation of all subtree sizes and their sorting can be implemented in a linear time in a straightforward manner, where the details are omitted from this text). As computations of Equation 4 in line 3 and of term III of Equation 6 in line 2 are dominated by *MCM* computations, it is left to show that it is possible to compute terms I and II of Equation 6 in line 2 of the algorithm in $O(n_T n_S)$ along the entire run of the algorithm.

Consider a pair of nodes $u \in T$ and $u' \in S$, and the corresponding sub-matrix $H_{u,u'}$ (see Section 3.1 for definition of $H_{u,u'}$). It is simple to observe that an explicit computation of term I of the equation with respect to some subtrees T^v_u and $S^{v'}_{u'}$ can be conducted in $O(d_u)$ time. Nevertheless, we next show how to conduct this computation in $O(1)$ amortized time. Fix an index $1 \leq j \leq d_{u'}$. Let $x^* = \text{argmin}_{x \in N(u)}\left(\textit{Rooted-HSA}^{\text{-}r}\left(T^u_x, S^{v'_j}_{u'}\right) - prune(T^u_x)\right)$, and denote $\alpha = \textit{Rooted-HSA}^{\text{-}r}\left(T^u_{x^*}, S^{v'_j}_{u'}\right) - prune(T^u_{x^*})$. As $N(u) \setminus \{v\} \subset N(u)$ for every $v \in N(u)$, it is clear that for $v \neq x^*$, $\min_{x \in N(u) \setminus \{v\}} \left(\textit{Rooted-HSA}^{\text{-}r}\left(T^u_x, S^{v'_j}_{u'}\right) - prune(T^u_x)\right) = \alpha$. Similarly, denote $\beta = \sum_{x \in N(u)} prune(T^u_x)$, and observe that $\sum_{x \in N(u) \setminus \{v\}} prune(T^u_x) = \beta - prune(T^u_v)$ for every $v \in N(u)$. Thus, for $v \neq x^*$, term I of Equation 6 can be written as $smooth(u) + \beta - prune(T^u_v) + \alpha$, and given the values α and β, be computed in $O(1)$ time. Due to Observation 1, when the algorithm is about to compute the entry in the second row and j-th column of $H_{u,u'}$ (i.e. the entry corresponding to $T^{v_2}_u$ and $S^{v'_j}_{u'}$), all required values for computing x^*, α, and β, are already stored in H, and therefore these values may be computed in $O(d_u)$ time. Once computing these values, it is possible to compute term I for each of the reminding rows $i > 1$ in column j of $H_{u,u'}$, except for row i such that $v_i = x^*$, in $O(1)$ time each. Additional $O(d_u)$ operations are required for computing the term for row 1 and the row i such that $v_i = x^*$, and therefore the total number of operations for computing the term for all d_u entries in column j of $H_{u,u'}$ is $O(d_u)$ time. This implies that the amortized time for computing term I for each entry in $H_{u,u'}$ is $O(1)$, and therefore the amortized time for computing term I for each entry in H is $O(1)$. The proof for term II is symmetric. All

in all, we get that the running time for all operations conducted by the algorithm, besides *MCM* computations, is $O(n_T n_S)$. □

S3 Running time analysis of Algorithm 1

From Lemma 2, all operations in the algorithm besides *MCM* computations require $O(n_T n_S)$ time. It thus remains to count the number of operations applied in *MCM* computations. Such computations are applied when computing term III of Equation 6, or when computing Equation 4. Using the algorithm described in Section S4.1, each *MCM* instance can be solved in $O(nm \min(n,m))$, where n and m are the sizes of the two groups in the instance. Term III of Equation 6 is computed once for every entry in H in line 2 of the algorithm, where the group sizes of the corresponding matching instances are at most $d_T - 1$ and $d_S - 1$. Thus, the total time for computing term III for all $O(n_T n_S)$ entries in H is $O(n_T n_S d_T d_S \min(d_T, d_S))$. A tighter analysis, similar to that preformed in Section 3.1 (i.e. counting differently the cases in which $\min(d_u, d_{u'}) \leq 2$ and the cases in which $d_u > 2$ and $d_{u'} > 2$), can show that this computation is actually conducted in $O(L_T L_S d_T d_S \min(d_T, d_S))$ time. Equation 4 is computed once for each $v \in T$ and $v' \in S$ in line 3, where the corresponding matching instance's group sizes are d_v and $d_{v'}$. Summing the total number of operations in the implied *MCM* computations, we get $\sum_{v \in T} \sum_{v' \in S} d_v d_{v'} \min(d_v, d_{v'}) \leq \min(d_T, d_S) \sum_{v \in T} d_v \sum_{v' \in S} d_{v'} = \min(d_T, d_S) n_T n_S$. Therefore, the overall running time of the algorithm is dictated by the the bottleneck expression $O(n_T n_S + L_T L_S d_T d_S \min(d_T, d_S))$.

S4 Algorithms for bipartite matching problems

In this section we show efficient algorithms for the *MCM*, *All-Cavity-MCM*, and *All-Pairs-Cavity-MCM* problems defined in Section 2.3. The well known approach for solving *MCM* is to reduce the problem to the Min-Cost Max-Flow problem. The first efficient implementation of this reduction was described by Edmonds and Karp in [1], where in [2] Zhang showed a modified reduction, which takes into account also unmatched element penalties. Here, we further refine the reduction in order to improve the running time of the algorithm. For two groups of sizes n and m, where $n \leq m$, our modification reduces the $O(nm^2 + m^2 \log m)$ running time of the algorithm of [2] to $O(n^2 m)$. Similarly to Kao et al. [3], we show that solutions for instances of the form $(X, Y \setminus \{y\}, w)$ correspond to certain shortest paths in the residual flow network obtained when solving the instance (X, Y, w). In addition, we show such correlations also with respect to solutions for instances of the form $(X \setminus \{x\}, Y \setminus \{y\}, w)$.

Since the problems and techniques mentioned above are well known to computer scientists, we confine ourselves to discuss the main modifications and to elaborate on certain essential points, while only briefly explaining the rest. For a definition of the *Min-Cost Max-Flow* problem, related theorems, and a thorough discussion of its properties, please refer to other works, e.g. [1, 4, 5].

S4.1 Reducing *MCM* to *Min-Cost Max-Flow*

Let (X, Y, w) be a matching instance, and denote $n = \min(|X|, |Y|), m = \max(|X|, |Y|)$. We describe the reduction for the case where $|X| = n$, where the construction in the case where $|Y| = n$ is symmetric.

The reduction builds the flow network $N = (G, s, t, c, w')$, where G is the network's graph, s and t are the source and sink nodes respectively, and c and w' are the edge capacity and cost functions respectively. The graph $G = (V, E)$ is defined as follows (Figure S6a):

- $V = X \cup Y \cup \{s, t, \phi\}$, where s, t, and ϕ are unique nodes different from all nodes in X and Y. Note that we use the same notations for elements in X and Y and their corresponding nodes in V, where ambiguity can be resolved by the context.
- $E = E_1 \cup E_2$, where $E_1 = \{(x, y) : x \in X, y \in Y\}$, and $E_2 = \{(s, x) : x \in X\} \cup \{(y, t) : y \in Y\} \cup \{(x, \phi) : x \in X\} \cup \{(\phi, t)\}$.

The capacity function c assigns unit capacities to all edges in E, with the exception that $c(\phi, t) = n$. The cost function w' assigns to edges $(x, y) \in E_1$ the costs $w(x, y) - w(x) - w(y)$, and zero costs to all edges in E_2.

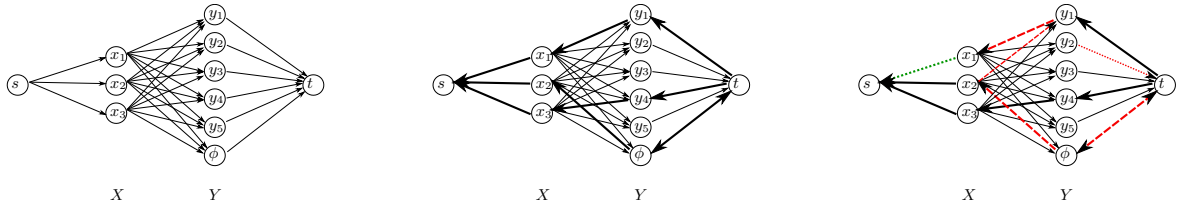


Fig. S6: The reduction from *MCM* to *Min-Cost Max-Flow*. (a) The graph G constructed in the case where $|X| = 3$ and $|Y| = 5$. All edge capacities are 1, except for the edge (ϕ, t) whose capacity is $c(\phi, t) = |X| = 3$. The cost function w' assigns to edges of the form (x_i, y_j) the costs $w'(x_i, y_j) = w(x_i, y_j) - w(x_i) - w(y_j)$, and zero costs to all other edges. (b) The residual network after finding a minimum cost maximum flow f^* in the network. Edges over which there is a flow (depicted as thickened edges) reverse their direction (the edge (ϕ, t) is not saturated, and therefore both (ϕ, t) and (t, ϕ) belong to the graph). In this example, the flow implies the minimum cost matching $M_{f^*} = \{(x_1, y_1), (x_3, y_4)\}$, where the elements x_2, y_2, y_3, and y_5 are unmatched. (c) A minimum cost path from y_2 to x_1 in the residual network. The path, depicted in dashed and dotted red, is $P = y_2 \to t \to \phi \to x_2 \to y_1 \to x_1$. The corresponding return path P' from t to s is obtained by removing from p the first edge $y_2 \to t$ (in dotted red), and adding at the end the edge $x_1 \to s$ (in dotted green). The flow f' obtained by returning one flow unit from t to s along P' is an optimal flow in the network N' obtained by removing y_2 and x_1 from the graph (see proof of Lemma 2). The corresponding matching for this flow is $M_{f'} = \{(x_2, y_1), (x_3, y_4)\}$.

Call a minimum-cost maximum-flow in N an *optimal* flow with respect to N. Proposition 1 shows the relation between optimal flows in f and optimal matchings of (X, Y, w).

Proposition 1. *Let (X, Y, w) be a matching instance, N the corresponding network, and f^* an optimal flow in N. Then,* $\mathrm{MCM}(X, Y, w) = w'(f^*) + \sum_{a \in X \cup Y} w(a)$, *and f^* defines an optimal matching as containing all pairs (x, y) such that $f^*(x, y) = 1$.*

Proof. Let M^* be an optimal matching of (X, Y, w). We show that $w(M^*) = w'(f^*) + \sum_{a \in X \cup Y} w(a)$.

Define the flow f_{M^*} in N as the flow which is obtained by transmitting one flow unit on every path of the form $s \to x \to y \to t$ for all $(x,y) \in M$, and one flow unit on every path of the form $s \to x \to \phi \to t$ for all $x \in X$ such that $x \notin M$. It is straightforward to observe that f_{M^*} is a valid flow in N (satisfying the capacity and flow conservation constraints) of maximum size (saturating the cut $(\{s\}, V \setminus \{s\})$), and that the cost of f_{M^*} is given by

$$\begin{aligned} w'(f_{M^*}) &= \sum_{(x,y)\in M^*} (w(x,y) - w(x) - w(y)) \\ &= \sum_{(x,y)\in M^*} (w(x,y) - w(x) - w(y)) + \sum_{\substack{a\in X\cup Y \\ a\notin M^*}} w(a) - \sum_{\substack{a\in X\cup Y \\ a\notin M^*}} w(a) \\ &= w(M^*) - \sum_{a\in X\cup Y} w(a) = MCM(X,Y,w) - \sum_{a\in X\cup Y} w(a). \end{aligned}$$

Thus, $w'(f^*) + \sum_{a\in X\cup Y} w(a) \leq w'(f_{M^*}) + \sum_{a\in X\cup Y} w(a) = MCM(X,Y,w)$.

On the other hand, define the matching M_{f^*} of (X,Y,w) which contains all pairs $(x,y) \in X \times Y$ such that $f^*(x,y) = 1$ (the validity of M_{f^*} can be easily asserted, see Figure S6b), and similarly observe that $w'(f^*) + \sum_{a\in X\cup Y} w(a) = w(M_{f^*}) \geq MCM(X,Y,w)$. Thus, $MCM(X,Y,w) = w'(f^*) + \sum_{a\in X\cup Y} w(a)$, while M_{f^*} is an optimal matching for (X,Y,w). □

Time complexity. Given a matching instance (X,Y,w), the flow network N can be constructed in $O(nm)$ running time, by the reduction. An optimal flow in N can be found with the algorithm of Edmonds and Karp [1,6]. Essentially, this is an iterative algorithm that maintains a valid flow f in N. The algorithm computes in each iteration a minimum cost path from s to t in the current residual network N^f in $O(|E| + |V|\log|V|) = O(nm + m\log m)$ time, using the algorithm of [6]. Negative edge costs, which are not allowed in the minimum cost path algorithm of [6], are handled by using a *node labeling* function as explained in [1], where initial labels for each $v \in V$ can be computed as the minimum cost of a path from s to v in G. Since G is acyclic, these costs can be computed in time $O(|V|+|E|) = O(nm)$. The labeling function can be maintained without increasing the time bound of iterations [1]. Once a minimum cost path from s to t in N^f is computed, the algorithm updates the flow by transmitting one flow unit along this path. Since each iteration increases the flow size by one and the maximum flow size is n, the time of the complete algorithm is $O(n^2m + nm\log m)$.

A more involved implementation eliminates the $nm\log m$ term from the running time bound. This is obtained by first selecting for each $x \in X$ a subset $Y_x \subseteq Y$ of n smallest cost matches for x in Y (with respect to the w' matching costs), setting $Y' = \bigcup_{x\in X} Y_x$, and solving the MCM problem with respect to the sets X and Y'. It can be shown that there exists an optimal matching between X and Y such that all matched elements in Y are from the subset Y' (since if in some optimal matching there is an element $x \in X$ which is matched to an element $y \in Y \setminus Y'$, then there is an unmatched element $y' \in Y_x$, and replacing the matching of x from y to y' does not increase the matching cost). Therefore, an optimal matching between X and Y' is an optimal matching between X and Y. The computation of each subset Y_x can be done naively in $O(nm)$ time (or $O(m)$ time using the algorithm of [7]), and thus Y' can be computed in $O(n^2m)$ time (or $O(nm)$

time, using [7]). For $m' = |Y'| \leq n^2$, solving the *MCM* with respect to the sets X and Y' takes $O(n^2m' + nm' \log m') = O(n^2m')$ time (since $\log m' \leq \log n^2 = o(n)$), and thus the total running time of the algorithm is $O(n^2m + n^2m') = O(n^2m)$.

S4.2 Efficient algorithms for *All-Cavity-MCM* and *All-Pairs-Cavity-MCM*

We now present Algorithms 1 and 2, which solve *All-Cavity-MCM* and *All-Pairs-Cavity-MCM*, respectively. In what follows, given a network N and a flow f in it, we denote by $d_{r,q}$ the minimum cost of a path from r to q in the residual network N^f.

Algorithm 1: *All-Cavity-MCM*(X, Y, w)

1 Construct the network N that corresponds to (X, Y, w), and compute an optimal flow f in N;
2 Set $MCM(X, Y, w) = w(f) + \sum_{a \in X \cup Y} w(a)$;
3 Compute for every $y \in Y$ the values $d_{y,t}$;
4 For every $y \in Y$, report $MCM(X, Y \setminus \{y\}, w) = MCM(X, Y, w) - w(y) + d_{y,t}$;

Algorithm 2: *All-Pairs-Cavity-MCM*(X, Y, w)

1 Construct the network N that corresponds to (X, Y, w), and compute an optimal flow f in N;
2 Set $MCM(X, Y, w) = w(f) + \sum_{a \in X \cup Y} w(a)$;
3 Compute for every $x \in X$ and every $y \in Y$ the values $d_{y,x}$;
4 For every $x \in X$ and every $y \in Y$, report
$MCM(X \setminus \{x\}, Y \setminus \{y\}, w) = MCM(X, Y, w) - w(x) - w(y) + d_{y,x}$;

Correctness of these algorithms is implied by the following analysis. We will use the following auxiliary lemma:

Lemma 1. *Let $G = (V, E)$ be a directed graph with a cost function w over its edges and no negative cost cycles. Let P be a minimum cost path from a node r to a node q in G. Let $G' = (V, E')$ be the graph obtained from G by replacing every edge $(u, v) \in P$ with the reversed edge (v, u) (if not already in G), whose cost is defined to be $w(v, u) = -w(u, v)$. Then, G' contains no negative cost cycles.*

Proof. Following [1], we call a flow *extreme* if it is of minimum cost among all flows of the same size. By [1], a flow is extreme if and only if the corresponding residual network contains no negative cost cycle. In addition, a flow obtained by increasing an extreme flow along a minimum cost augmentation path from the source to the sink of the network, is also extreme.

Let N be the flow network defined by the graph G, the source node r, the sink node q, the cost function w, and the capacity function c which assigns unit capacities to all edges in G. Since there is no negative cost cycle in N, the zero flow is extreme with respect to zero-size flows. Let f be

the unit flow along P in N, and note that the residual graph of N^f is identical to G' by definition. Since P is a minimum cost path from r to q in N, f is extreme with respect to all flows of size 1. Thus, N^f contains no negative cost cycle.

Now, we show the relation between *MCM* solutions for instances (X, Y, w) and solutions for sub-instances of the forms $(X \setminus \{x\}, Y, w)$, $(X, Y \setminus \{y\}, w)$, and $(X \setminus \{x\}, Y \setminus \{y\}, w)$. Let N be the network corresponding to a matching instance (X, Y, w). Let f be an optimal flow in N, and N^f be the residual network of N with respect to f (Figure S6b). Since f is a maximum flow in N it saturates the minimum cut $(\{s\}, V \setminus \{s\})$, that is for every $x \in X,\ f(s, x) = 1$. Therefore, from flow conservation considerations, for every $x \in X$ there exists a single $z \in Y \cup \{\phi\}$ such that $f(x, z) = 1$, and $f(z, t) \geq 1$. This implies that in the residual network N^f the only edge into x is the residual edge (z, x). For elements $y \in Y$, either there exist some $x \in X$ such that $f(x, y) = 1$ (and then $f(y, t) = 1$), or the flow through y is 0. In the former case, the only edge out from y in N^f is (y, x), while in the latter case the only edge out from y in is (y, t).

Lemma 2. *1. For every* $x \in X,\ \mathrm{MCM}(X \setminus \{x\}, Y, w) = \mathrm{MCM}(X, Y, w) - w(x) + d_{t,x}$.
2. For every $y \in Y,\ \mathrm{MCM}(X, Y \setminus \{y\}, w) = \mathrm{MCM}(X, Y, w) - w(y) + d_{y,t}$.
3. For every $x \in X, y \in Y,\ \mathrm{MCM}(X \setminus \{x\}, Y \setminus \{y\}, w) = \mathrm{MCM}(X, Y, w) - w(x) - w(y) + d_{y,x}$.

Proof. We start by showing item 3 of the lemma. Let N' be the network that corresponds to the instance $(X \setminus \{x\}, Y \setminus \{y\}, w)$. In order to prove item 3, we construct a flow f' in N' such that $w(f') = w(f) + d_{y,x}$, and prove that f' is optimal with respect to N'.

First, we show that there is a path from y to x in N^f. Let $z \in Y \cup \{\phi\}$ be the node such that $f(x, z) = 1$. If $z = y$ then the residual edge (y, x) exists in N^f, composing the path $y \to x$. Else, if there is some $x' \in X$ such that $f(x', y) = 1$, then the path $y \to x' \to z \to x$ belongs to N^f. Otherwise, N^f contains the path $y \to t \to z \to x$.

Let P be a minimum cost path from y to x in N^f. Define the graph $\hat{N^f}$ which is obtained from N^f by reversing all edges along P. Since N^f contains no negative cost cycle (due to the optimality of f), Lemma 1 indicates that $\hat{N^f}$ contains no negative cost cycle. In addition, there is no edge into x and no edge out of y in $\hat{N^f}$ (since the only edges of these forms were reversed).

Define the flow f' which is obtained from f by returning one flow unit from t to s along the path P' as follows: if P starts with (y, t), then P' is obtained by removing (y, t) from P and concatenating (x, s) at its end (see Figure S6c). Else, P' is obtained by concatenating (t, y), P, and (x, s). Observe that f' is a valid flow in N' (since f' passes no flow units through x and y), the cost of f' is $w(f') = w(f) + d_{y,x}$, and its size is $|f'| = |f| - 1 = n - 1$ (which is the maximum flow size in N'). Also, observe that $N'^{f'}$ is a sub-graph of $\hat{N^f}$, and therefore it contains no negative cycles. Thus, f' is an optimal flow in N' of cost $w(f) + d_{y,x}$.

Hence, we get that

$$\begin{aligned} MCM(X \setminus \{x\}, Y \setminus \{y\}, w) &= w'(f') + \sum_{a \in (X \setminus \{x\}) \cup (Y \setminus \{y\})} w(a) \\ &= w'(f) + d_{y,x} - w(x) - w(y) + \sum_{a \in X \cup Y} w(a) \\ &= MCM(X, Y, w) - w(x) - w(y) + d_{y,x}. \end{aligned}$$

The proof for the other two items is similar, with the following minor differences. For item 1, P is defined to be the minimum cost path from t to x in N^f, and P' is the concatenation of P with

(x, s). For item 2, P is defined to be the minimum cost path from y to t in N^f. If this path is (y, t), then P' is empty, and otherwise P' is the cycle obtained by concatenating (t, y) to P (in this case the size of f' remains the same as that of f).

Based on Lemma 2, Algorithms 1 and 2 solve *All-Cavity-MCM* and *All-Pairs-Cavity-MCM*, respectively.

Efficient implementation and time complexity analysis. Recall that $n = \min(|X|, |Y|)$ and $m = \max(|X|, |Y|)$. Lines 1 and 2 of both Algorithms 1 and 2 can be implemented in $O(n^2 m)$ running time, as explained in Section S4.1. Line 4 takes in Algorithm 1 $O(m)$ time, and in Algorithm 2 $O(nm)$. Single source shortest paths in the residual graph can be computed in $O(|E| + |V| \log |V|) = O(nm + m \log m)$ using Dijkstra's algorithm [6]. This algorithm can be applied in line 3 of Algorithm 1 to compute minimum cost paths from all nodes into t, and in Algorithm 2 it can be applied n times either to compute minimum cost paths from all nodes into each $x \in X$ (in the case where $n = |X|$), or to compute minimum cost paths into all nodes from each $y \in Y$ (in the case where $n = |Y|$). In order to eliminate the $m \log m$ term from the running time bound of the shortest paths algorithm, we next consider two cases: the case where $|Y| = m$, and the case where $|X| = m$.

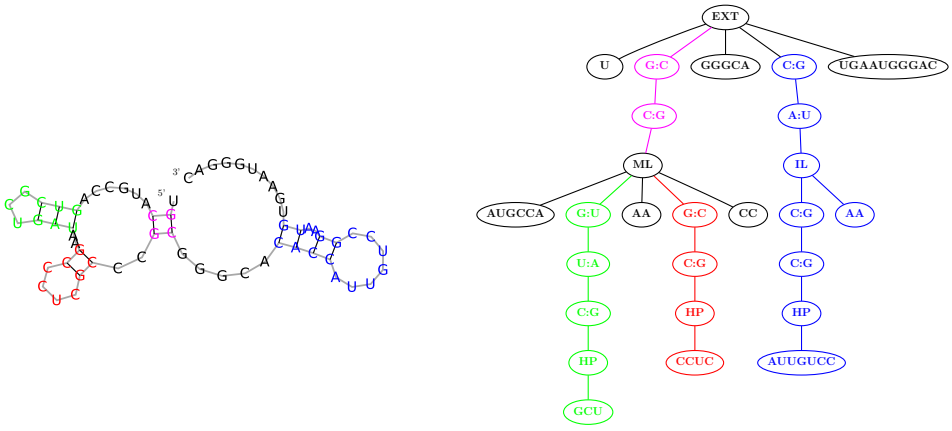
In the case where $|Y| = m$, the graph constructed by the reduction corresponds to the sets X and $Y' \subseteq Y$, where $|Y'| = m' \leq n^2$, as explained in the time complexity analysis in Section S4.1. We call this graph the *reduced graph*, as opposed to the *full graph* which contains all nodes in Y. Note that the terms $d_{y,t}$ or $d_{y,x}$ in line 3 of both algorithms should be computed with respect to the full graph. Observe that in the full graph, each node $y \in Y \setminus Y'$ has exactly one outgoing edge - the edge (y, t), and ingoing edges of the form (x, y) for all $x \in X$. Therefore, each sub-path of the form $x \to y \to t$ in the full graph can be implemented in the reduced graph by adding an edge $x \to t$ whose cost is the cost of the path $x \to y \to t$, so that a minimum cost path between any two nodes in the reduced graph has the same cost as a minimum cost path between the corresponding nodes in the full graph. As this process induces for a single $x \in X$ multiple edges into t, it is enough to add a single edge (x, t) for each $x \in X$, and set its cost to the minimum cost of a path $x \to y \to t$ for all $y \in Y \setminus Y'$, in $O(nm)$ time. Solving the single source shortest paths problem over the reduced graph can be used for computing all values $d_{y,t}$ and $d_{y,x}$ for $y \in Y'$. For nodes $y \in Y \setminus Y'$, the minimum cost path into some $x \in X$ in the full graph is given by $d_{t,x}$ in the reduced graph (since such path in the full graph starts with the edge (y, t), whose cost is 0), and the minimum cost of a path into t is simply 0 (since the path starts with the edge (y, t), and contains no negative cost cycles). Thus, single source shortest paths computations can be implemented in $O(nm+nm'+m' \log m') = O(nm)$ (as $\log m' \leq \log n^2 = o(n)$), and therefore line 3 requires $O(nm)$ time in Algorithm 1, and $O(n^2 m)$ time in Algorithm 2. The case where $|X| = m$ is solved similarly, replacing X by a subset $X' \subseteq X$ of size $|X'| = m' \leq n^2$ in a reduced graph, and using the observation that each $x \in X \setminus X'$ has only one ingoing edge in the full graph - the edge (ϕ, x) (since in the corresponding flow in the full network there is a flow unit transmitted along the path $s \to x \to \phi \to t$). The rest of the details for solving this case are omitted. In all, the running times of both algorithms 1 and 2 is $O(n^2 m)$.

S5 Implementation details

S5.1 RNA tree representation

There are several previous models for representing a pseudoknot-free RNA secondary structure (example in Fig. S7a) as an ordered, rooted tree [8–14]. Our tree representation uses a similar modeling as in Höchsman et al. [15], with some variations we describe next (see Fig. S7b). Given an RNA secondary structure, each loop, base-pair, and interval of unpaired bases generates a node in the tree representation of the structure. Labels were assigned to tree nodes in order to indicate their types and content. Node types that were used are: `BP` (base-pair), `UPI` (unpaired-base interval), `HP` (hairpin), `IL` (internal loop or bulge), `ML` (multi-loop), and `EXT` (external loop). For a `UPI` node, the label also includes the 5' to 3' base-sequence of the corresponding interval, and for a `BP` node the label includes the corresponding bases as well as the indices of the two endpoints of the base-pair (see Fig. S7).

Each loop node (`HP`, `IL`, `ML`, and `EXT`) is connected, in 5' to 3' sequence order, to all `UPI` nodes which correspond to intervals of unpaired bases associated to the loop, and to all `BP` nodes which correspond to stem-terminating base-pairs adjacent to the loop. `BP` nodes are nodes of degree 2 (except for the special case of a base pair which is a leaf), where the two neighbors of such nodes are the `BP` nodes that correspond to adjacent stacked base-pairs, or loop nodes in the case where the `BP` node terminates a stem. The set of leaves in the tree correspond to the set of `UPI` nodes or `BP` nodes terminating loop-free stems.



(a) (b)

Fig. S7: (a). RNA secondary structure as presented by [16]. (b). The tree representation of (a) in our model. Each base-pairs stacking is matched by color to it's representing branch in the tree

S5.2 Alignment cost functions

Node smoothing costs were set to 3 for `BP`, 11 for `ML`/`EXT` and 5 for `IL`. For subtree pruning costs, we have designed a cost function that counts the occurrences of different types of elements appearing

in the subtree and deduces a corresponding penalty. The function is of the form: $prune(T) = C_{up} \cdot N_{up} + C_{bp} \cdot N_{bp} + C_{hp} \cdot N_{hp} + \ldots$, where the values of C_x are constant penalty factors, N_{up} is the number of unpaired bases in T, N_{bp} is the number of base-pairs, N_{hp} is the number of hairpins, and so on (The specific C_x values are given in Table S1a). Table S1b summarizes the costs of matching node pairs by the alignment.

Table S1: (a). Element pruning penalty. (b). Node matching costs. Sequence alignment costs and base-pair alignment costs were set using the RIBOSUM85-60 scoring matrix [17].

(a)

Factor	C_{up}	C_{bp}	C_{hp}	C_{ml}	C_{il}	C_{int}	C_{ext}
Value	1	2.5	5	3	2	2	0

(b)

Type	UPI	BP	HP	IL	ML	EXT
UPI	Sequence alignment	∞	∞	∞	∞	∞
BP	∞	Base-pair alignment	∞	∞	∞	∞
HP	∞	∞	-10	0	0	0
IL	∞	∞	0	-5	∞	∞
ML	∞	∞	0	∞	-7	-7
EXT	∞	∞	0	∞	-7	-10

S6 Results analysis

Type 1: loop swapping in main multiloop accompanied by hairpin deletion in P17.1 The first type of alignment pattern is characterized by comparisons between a green sulfur bacteria *Chlorobium* (e.g. *ASE_00047*) and gamma purple bacterial RNAse P RNAs. This alignment pattern is exemplified in Figure 1b. When examining the corresponding tertiary structure information (Figure S8), the transformations predicted by FRUUT seem to make sense: we observe that *ASE_00047* has an additional duplex (named P2) connecting loop l_1 to l_2 while in *ASE_00334* the corresponding duplex in 3D connects l_1 to l_3. Notice that the alignment between the two trees maps l_2 to l_3 while preserving all the other information surrounding the loops - thus providing a unordered alignment between the trees.

Type 2: hairpin swapping between P17 and P17.1 The second type of alignment pattern is characterized by comparisons between *Agrobacterium tumefaciens* (*ASE_00018*) and several *Chlamydia trachomatis* members. This alignment pattern is exemplified in Figure S9 . An interesting element-twist transformation is observed here in hairpins $P17$ and $P17.1$ of *ASE_00018*, which are mapped onto their corresponding hairpins $P17.1$ and $P17$ in *ASE_00070*, respectively, via a subtree reordering mapping operation. When examining the corresponding tertiary structure information (Figure S9b), we observe that the loops of the hairpins $P17$ and $P17.1$ are engaged in pseudoknots with loops L and l (named P6), respectively.

S6.1 Ordered Unrooted Hammerhead Ribosyme Tree Similarities

Another type of homology detected by our tool is exemplified in the Hammerhead Ribosyme family, which is characterized by two distinct transcript types yielding the same functional RNA (Figure 1a). Our tool can detect the similarity between these two Hammerhead types by applying the unrooted tree alignment mode.

It seems reasonable that some RNAs in nature share a similar secondary structure, which may or may not consider base sequence, and are transcribed starting from different sequential locations with respect to the resulting structure. The rationale for this assumption is that catalytic sites and binding sites do not necessarily need conserved 5' or 3' ends to operate, but rather depend on the structural context in which they reside. In what follows, the hammerhead is given to exemplify structural similarities that cannot be detected or are poorly detected using the conventional method of rooted tree alignment, and to demonstrate that such similarities can be detected using unrooted tree alignment (see Figure 1a).

The Hammerhead Ribozyme, a derivative of several self-cleaving satellite virus RNAs [18] is a single strand RNA with autocatalytic capabilities. Naturally, it has a highly specific self-cleavage site at C17, operating via isomeration and rearrangement of the linkage phosphodiester bond [19]. Furthermore, Birikh et al. [20] suggested that the Hammerhead Ribozyme may undergo synthetic modification by removing the loop of one of its helical arms thus making it catalytically active and able to cleave other RNAs. Hammerheads are therefore widely used in the biotechnological industry as biosensors, enzymes for specific RNAs and gene discovery agents.

The tertiary structure of the minimal version of the Hammerhead Ribozyme has been thoroughly studied by [21, 22]. It is composed of three base paired helices, entitled I, II, III, according to their position in the sequence. There are highly conserved sequences within the multi-loop between stem I and II (containing the sequence box $CUGA$), between stem II and III (containing the box $GAAA$) and between stem III and I (containing the cleavage reaction site, C). Evidently, there are two types of Hammerheads: type I, where stem I starts at the 5' and 3' ends of the strand, and type III, where stem III starts at the 5' and 3' ends of the strand. As of today, no natural type II Hammerhead have been found.

Our benchmark was based on 146 Hammerhead structures, ranging across various organisms, taken from the RNA Strand [23]. This yields total of 10,585 pairs of trees, with tree sizes ranging from 17 to 48 nodes, averaging at 25.5.

Each pair of trees T, S was compared in two modes to obtain the corresponding scores and alignments: rooted-ordered (RO) and unrooted-ordered (UO). Within each mode, we used a relative score formula described by Höchsmann et al. [24] to assess the similarity of two trees, normalizing the alignment cost by the average of the self-alignment costs of the compared trees. We divided all the pairs into two groups, the first containing all pairs where both members are from the same type and the second group containing mixed pairs where the two members in each pair belong to different types. We calculated the average normalized similarity score for each group in both alignment modes. The results, Summarized in Table S2, show that the UO tree alignment mode describes the similarity between the two different types (type I and type III) better than the RO mode.

Table S2: The UO alignment mode helps to emphasize the similarity between the two types of Hammerhead molecules. Each cell represent the average over all pairs in each group (same / different types)

Mode\Type	Same Type	Different Type
RO	0.55	0.19
UO	0.56	0.35

We further analyzed the results by applying a ranking method similar to the one given in Section 4, however in this analysis the aligned tree-pairs were ranked by the relative improvement of the unrooted ordered alignment in comparison to rooted ordered alignment. When examining the top-ranking pairs in this ranking, we came across many pairs consisting of Hammerhead structures spanning across the two different types. A more careful observation of the results shows that none of the pairs in the top 2000 results consists of two trees of the same Hammerhead type. This suggests that when comparing two Hammerhead trees of different types using rooted alignment, one would observe a mild similarity score. However, when comparing the same pair using unrooted alignment, the score improves greatly.

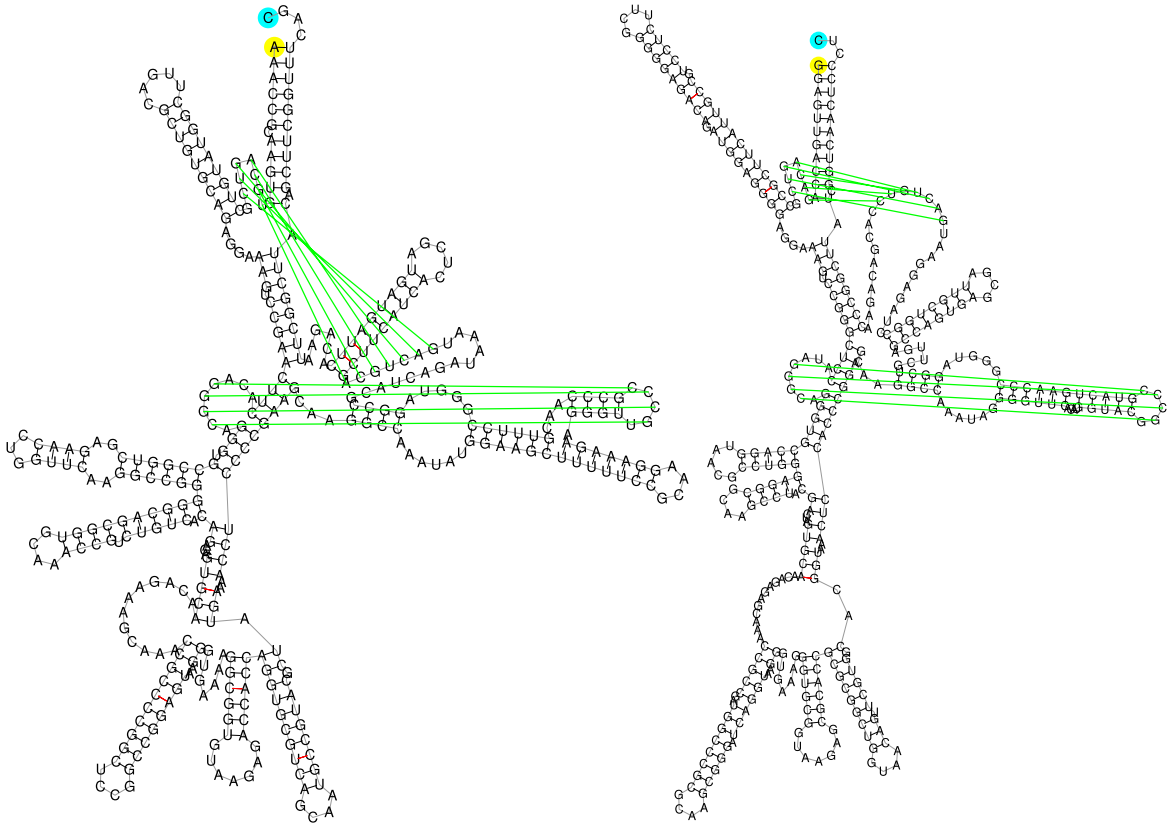
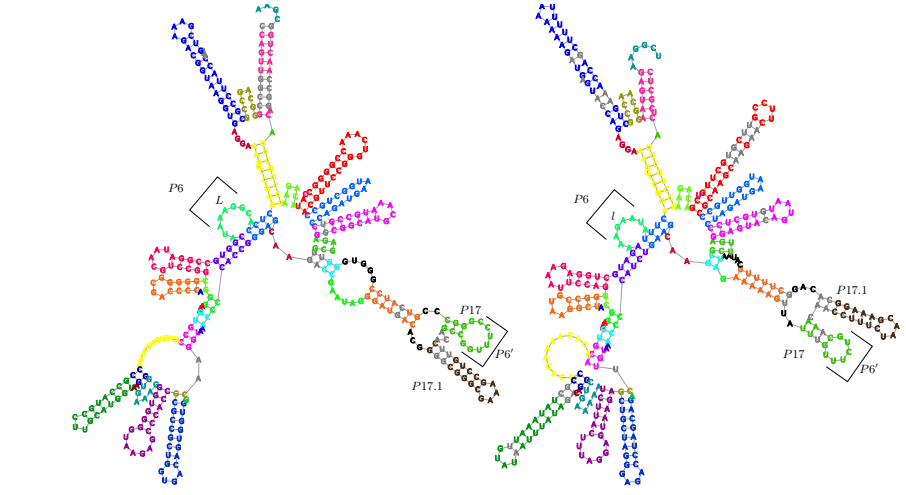
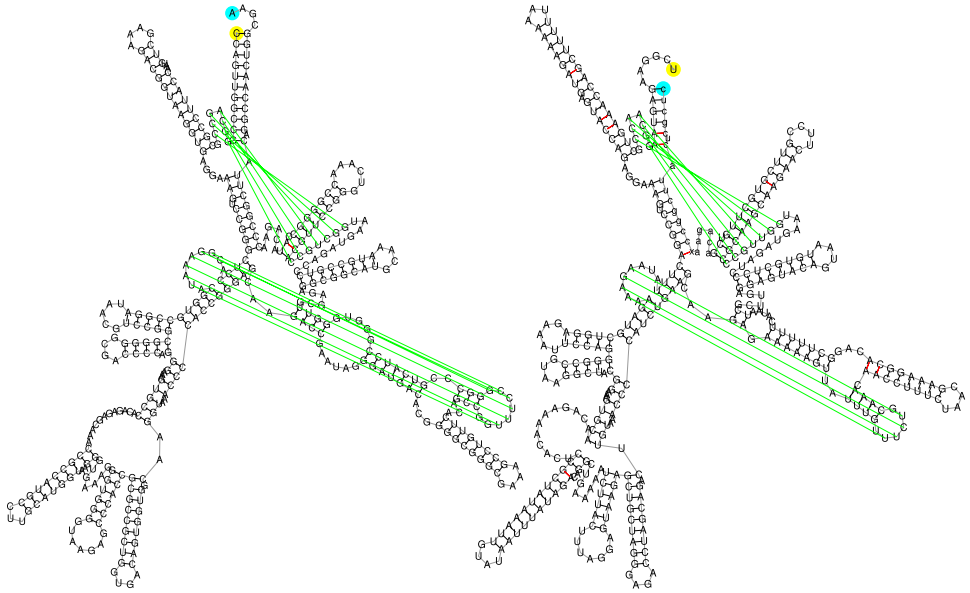


Fig. S8: Tertiary structural information for *ASE_00147* and *ASE_00334*, taken from the RNAse P Database [25].



(a)



(b)

Fig. S9: (a) An example of unordered alignment between two RNAse P RNAs: *ASE_00018* and *ASE_00070*. Grey colored bases in the FRUUT alignment graphics represent deletions. *P*6 is a pseudoknot marking of the tertiary structure information. (b) Tertiary structural information for *ASE*_00018 and *ASE*_00070, taken from the RNAse P Database [25].

References

1. Edmonds, J., Karp, R.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) **19** (1972) 248–264

2. Zhang, K.: A constrained edit distance between unordered labeled trees. Algorithmica **15** (1996) 205–222
3. Kao, M., Lam, T., Sung, W., Ting, H.: Cavity matchings, label compressions, and unrooted evolutionary trees. Arxiv preprint cs/0101031 (2001)
4. Tarjan, R.: Data structures and network algorithms. Volume 44. Society for Industrial Mathematics (1983)
5. Ahuja, R., Magnanti, T., Orlin, J., Weihe, K.: Network flows: theory, algorithms and applications. ZOR-Methods and Models of Operations Research **41** (1995) 252–254
6. Fredman, M., Tarjan, R.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) **34** (1987) 596–615
7. Blum, M., Floyd, R., Pratt, V., Rivest, R., Tarjan, R.: Time bounds for selection. Journal of Computer and System Sciences **7** (1973) 448–461
8. Shapiro, B.: An algorithm for comparing multiple RNA secondary structures. Computer Applied Bioscience **4** (1986) 387–393
9. Waterman, M.: Secondary structure of single-stranded nucleic acids. Adv. math. suppl. studies **1** (1978) 167–212
10. Zhang, K.: Computing similarity between RNA secondary structures. In: INTSYS '98: Proceedings of the IEEE International Joint Symposia on Intelligence and Systems, Washington, DC, USA, IEEE Computer Society (1998) 126
11. Le, S., Nussinov, R., Maizel, J.: Tree graphs of RNA secondary structures and their comparisons. Computers and biomedical research **22** (1989) 461–473
12. Fontana, W., Konings, D., Stadler, P., Schuster, P.: Statistics of RNA secondary structures. Biopolymers **33** (1993) 1389–1404
13. Steffen, P., Voss, B., Rehmsmeier, M., Reeder, J., Giegerich, R.: RNAshapes: an integrated RNA analysis package based on abstract shapes (2006)
14. Liu, J., Wang, J., Hu, J., Tian, B.: A method for aligning RNA secondary structures and its application to RNA motif detection. BMC bioinformatics **6** (2005) 89
15. Höchsmann, M., Voss, B., Giegerich, R.: Pure multiple RNA secondary structure alignments: a progressive profile approach. IEEE Transactions on Computational Biology and Bioinformatics (2004) 53–62
16. Hofacker, I.: Vienna RNA secondary structure server. Nucleic acids research **31** (2003) 3429
17. Klein, R., Eddy, S.: RSEARCH: finding homologs of single structured RNA sequences. BMC bioinformatics **4** (2003) 44
18. Murray, J., Terwey, D., Maloney, L., Karpeisky, A., Usman, N., Beigelman, L., Scott, W.: The structural basis of hammerhead ribozyme self-cleavage. Cell **92** (1998) 665–673
19. Hean, J., Weinberg, M.: The hammerhead ribozyme revisited: New biological insights. RNA and the regulation of gene expression: a hidden layer of complexity (2008) 1
20. Birikh, K., Heaton, P., Eckstein, F.: The structure, function and application of the hammerhead ribozyme. European Journal of Biochemistry **245** (1997) 1–16
21. Pley, H., Lindes, D., DeLuca-Flaherty, C., McKay, D.: Crystals of a hammerhead ribozyme. Journal of Biological Chemistry **268** (1993) 19656
22. Scott, W., Finch, J., Klug, A.: The crystal structure of an aii-rnahammerhead ribozyme: A proposed mechanism for rna catalytic cleavage. Cell **81** (1995) 991–1002
23. Andronescu, M., Bereg, V., Hoos, H.H., Condon, A.: RNA STRAND: the RNA secondary structure and statistical analysis database. BMC bioinformatics **9** (2008) 340
24. Höchsmann, M.: The tree alignment model: algorithms, implementations and applications for the analysis of RNA secondary structures. PhD thesis, Universitätsbibliothek Bielefeld (2005)
25. Brown, J.: The ribonuclease p database. Nucleic acids research **27** (1999) 314