RNA Tree Comparisons via Unrooted Unordered Alignments

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

Abstract. We generalize some current approaches for RNA tree alignment, which are traditionally confined to *ordered rooted* mappings, to also consider *unordered unrooted* mappings. We define the *Homeomorphic Subtree Alignment* problem, and present a new algorithm which applies to several modes, including global or local, ordered or unordered, and rooted or unrooted tree alignments. Our algorithm generalizes previous algorithms that either solved the problem in an asymmetric manner, or were restricted to the rooted and/or ordered cases. Focusing here on the most general unrooted unordered case, we show that our algorithm has an $O(n_T n_S \min(d_T, d_S))$ time complexity, where n_T and n_S are the number of nodes and d_T and d_S are the maximum node degrees in the input trees T and S, respectively. This maintains (and slightly improves) the time complexity of previous, less general algorithms for the problem. **Supplemental materials, source code, and web-interface for our tool are found in** `http://www.cs.bgu.ac.il/~negevcb/FRUUT`**.**

1 Introduction

Secondary structure of RNA molecules serves important functions in many non-coding RNAs [2]. Functional constraints lead to evolutionary structural conservation that in many cases exceeds the level of sequence conservation. Thus, detecting similarity between RNA secondary structures is of major importance in functional RNA research [3,4]. A mainstream approach for (pseudoknot free) RNA secondary structure comparison represents them as trees, and applies tree comparison algorithms [5–7]. Several variants of tree edit distance and alignment problems were previously studied. These variants differ in the type of trees they examine (ordered/unordered, rooted/unrooted), and in the type of edit operations or alignment restrictions they apply [8].

Currently available bioinformatic softwares for alignment of RNA trees usually assume *rooted ordered tree alignment* [3, 9–11]. However, there are known evolutionary phenomena, such as segment insertions, translocations and reversals,

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

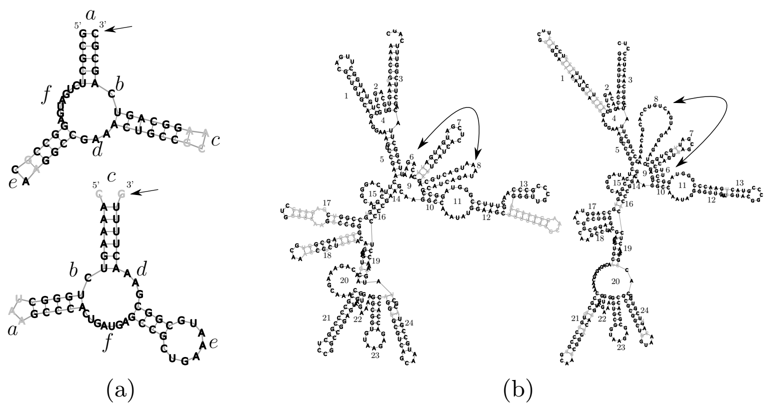

Fig. 1. Unrooted and unordered RNA similarities. Nodes of the RNA trees are clustered to motifs marked by letters or numbers (stems, loops, and unpaired nucleotide intervals), where aligned motifs share the same annotation, and unaligned nodes are in gray. Nomenclature is according to [1]. (a) An unrooted alignment between Hammerhead RNAs: *PDB_00693* (Type I, top) and *RFA_00388* (Type III, bottom). Arrows mark the external loops, which are usually chosen as roots by rooted RNA alignment tools. The unrooted mode of FRUUT identifies the high similarity between the molecules, not being restricted to align external loops to each other. (b) An unordered alignment between RNAse P RNAs: *ASE_00047* (left) and *ASE_00334* (right). In the unordered mode of FRUUT, the aligned motifs marked by 6 and 8 do not preserve order. In both molecules, pseudoknots occur between intervals annotated by 8 and 2 (see Figure S8), asserting the validity of the alignment.

which may result in a reordering or re-rooting of RNA structural elements [12]. These events can yield two similarly structured motifs, which are rooted differently (with respect to the standard "external loop" corresponding roots) [13], or permuted with respect to branching order. There are known examples of such unrooted/unordered RNA structural conservations [14,15] (Figure 1a), therefore, it is possible that searching for unordered and unrooted structural similarity may reveal new relations between RNA molecules that were previously undetected.

The general unordered tree edit distance problem is MAX-SNP hard [16], promoting the study of constrained variants. The *Subtree Isomorphism* problem [17] is, given a pattern tree S and a text tree T, to find if there is some subtree T' of T which is isomorphic to S. The *Subtree Homeomorphism* problem [18] is a variant of the former problem, where degree-2 nodes may be deleted from the selected subtree T' of the text. Pinter et al. [19] efficiently solved the Subtree Homeomorphism problem, under the unrooted unordered settings. In addition, their algorithm assigns costs for alignments and finds an alignment of minimum cost, thus solving a weighted variant of the problem. The running time of the algorithm of [19] is $O(n_S^2 n_T + n_S n_T \log n_T)$, where n_T and n_S are the number of nodes in T and S, respectively (improved time complexities under some scoring scheme restrictions were also shown in [19]). The *Constrained Edit Distance Between Unordered Labeled Trees* problem, presented by Zhang in [20], is a restricted version of rooted unordered tree edit distance, which allows the edit operations of node relabeling, subtree pruning, and deletions of degree-2 nodes (where in the general edit distance variant nodes of arbitrary degrees may be deleted). Zhang gave an $O(n_T n_S (d_T + d_S) log(d_T + d_S))$ time algorithm for this variant, where d_T and d_S are the maximum node degrees in T and S respectively. In this sense, the algorithm of [20] can be viewed as a symmetric (allowing deletions from both input trees), yet rooted variant of the algorithm of [19].

Our Contribution. We propose an efficient algorithm for comparing unordered unrooted trees. Specifically, we define the *Homeomorphic Subtree Alignment* problem, for which we give an $O(n_T n_S min(d_T, d_S))$ running time algorithm. Our approach can be viewed as a generalization of the two previous works of [19] and [20], which relaxes the asymmetric "text-pattern" restriction of [19], as well as the rooting restriction of [20]. Both algorithms in [19] and [20], as well as the algorithm presented here, make use of subroutines for solving the *Minimum Cost Bipartite Matching* problem which dictate their time complexities. Here, we define the *All Pairs Cavity Bipartite Matching* problem, a generalization of the *All Cavity Bipartite Matching* problem [21], design an efficient algorithm for it and show how to integrate it into our tree alignment algorithm. This modification allows our algorithm to match, and even improve, the running times of the previous (less general) algorithms of [19] and [20]. We apply our new publicly available tool to the comparison of RNA trees, and report on some results from a preliminary experiment, revealing similarities which may not be detected by the traditional rooted ordered alignment approaches.

Due to space constraints figures, pseudocode, and proofs for all theorems and lemmas are deferred to an online supplemental materials found in our website.

2 Preliminaries

2.1 Tree Notations

A *tree* $T = (V, E)$ is a connected, undirected and acyclic graph. For a node $v \in V$, denote by $N(v)$ the set of *neighbors* of v: $N(v) = \{u \in V : (v, u) \in E\}$. Denote by $d_v = |N(v)|$ the *degree* of v. A node v for which $d_v \leq 1$ is called a *leaf* in T. For simplicity, we henceforth use the notation $v \in T$ and $(v, u) \in T$ to imply that v is a node and (v, u) is an edge in a tree T. We use the notation $(v \to u)$ to indicate that the generally undirected edge (v, u) is being considered with respect to the specific direction from v to u. Denote by n_T and d_T the number of nodes and the maximum degree of a node in T, respectively.

A *rooted tree* is a tree in which one of the nodes is selected as its *root*. Denote by T^v the tree T when rooted upon the node $v \in T$. An *ordered* tree is a tree T in which for each node $v \in T$, the elements in $N(v)$ are ordered. In this work we consider *unrooted unordered* trees, *rooted unordered* trees, *unrooted ordered* trees, and *rooted ordered* trees. If no indication is given, we assume that the mentioned trees are unrooted and unordered.

Let $T = (V, E)$ be a tree. A *smoothing of a node* v of degree 2 in T is obtained by removing v from T and connecting its two neighbors by an edge. A *smoothing* of T is a tree obtained by smoothing zero or more nodes in T. A *subtree* of T is a connected subgraph of T. For an edge $(v \to u) \in T$, denote by T_u^v the *rooted subtree* of T induced by v as a root, and all nodes x in T such that the path between v and x in T starts with $(v \to u)$. Since a tree T with n nodes contains $n-1$ undirected edges, the total number of directed edges, and hence the number of rooted subtrees of the form T_u^v, is $2(n-1)$.

A *pruning* of a tree T with respect to an edge $(v \to u)$ is the removal from T of all nodes in T_u^v, except for v. Observe that every nonempty subtree of T is obtained by pruning T with respect to zero or more edges.

2.2 Homeomorphic Subtree Alignment

An *isomorphic alignment* between two trees $T = (V, E)$ and $S = (V', E')$ is a bijection $A : V \to V'$, such that for every pair of nodes $v, u \in V$ we have that $(v, u) \in E \Leftrightarrow (A(v), A(u)) \in E'$. A *homeomorphic alignment* between T and S is an isomorphic alignment between some smoothing T' of T and some smoothing S' of S, and a *homeomorphic subtree alignment* (HSA) between T and S is a homeomorphic alignment between some subtree T' of T and some subtree S' of S (Figure S1). For short, we write $(v, v') \in A$ to indicate that $A(v) = v'$.

Let T' and S' be the subtrees of T and S, and let T'' and S'' be the smoothings of T' and S', respectively, such that A is an isomorphic alignment between T'' and S''. Say that a node $v \in T$ is *aligned* by A if $v \in T''$, and that v is *smoothed* by A if $v \in T'$ and $v \notin T''$. Say that a subtree T_u^v is *pruned* by A if $v \in T'$ and $u \notin T'$. Let $prune(T_u^v)$ be a cost associated with pruning the subtree T_u^v from T, $smooth(v)$ be a cost associated with smoothing node v, and $align(v, v')$ be a cost associated with aligning node v against some node v'. Definitions for S are similar. Denote by $\pi(A)$ the set of pruned subtrees and by $\delta(A)$ the set of smoothed nodes of T and S, with respect to A. Define the *alignment cost*:

$$w(T, S, A) = \sum_{(v,v') \in A} align(v, v') + \sum_{T' \in \pi(A)} prune(T') + \sum_{v \in \delta(A)} smooth(v) \quad (1)$$

Denote by $HSA(T, S)$ the minimum alignment cost of an HSA between T and S, and call an HSA A *optimal* with respect to T and S if $w(T, S, A) = HSA(T, S)$. The *Min-Cost HSA* problem is, given a pair of trees T and S, to compute $HSA(T, S)$.

Rooted and Ordered Alignments. In addition to the general Min-Cost HSA problem, we also consider special cases of the problem in which the two input trees are rooted and/or ordered, and the alignment is required to satisfy certain restrictions with respect to these additional properties. For two rooted trees T^v and $S^{v'}$, say that A is a *rooted HSA* between T^v and $S^{v'}$ if A is an HSA between T^v and $S^{v'}$, and $(v, v') \in A$. The definition of *ordered HSA* requires some additional formalism, related to *bipartite matchings.*

Let X and Y be two sets. A *bipartite matching* M between X and Y is a set of pairs $M \subseteq X \times Y$, such that each element in $X \cup Y$ participates in at most one pair in M. If some element $z \in X \cup Y$ does not participate in any pair in M, we say that z is *unmatched* by M and denote $z \notin M$. When $X = \langle x_1, x_2, \ldots, x_n \rangle$ and $Y = \langle y_1, y_2, \ldots, y_m \rangle$ are ordered sets, say that M *preserves linear order* if for every $(x_i, y_j), (x_{i'}, y_{j'}) \in M$, $i \leq i' \Leftrightarrow j \leq j'$. Say that M *preserves cyclic order* if M preserves linear order with respect to X and some rotation $Y_k = \langle y_k, y_{k+1}, \ldots, y_m, y_1, \ldots, y_{k-1} \rangle$ of Y.

Let A be an HSA between the trees T and S. For an edge $(v \to u) \in T$, say that u is a *relevant neighbor* of v (with respect to A) if there is some $x \in T^v_u, x \neq v$, which is aligned by A. Define relevant neighbors in S similarly.

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 $(y, y') \in A$, $y \in T^v_u \Leftrightarrow y' \in S^{v'}_{u'}$.*

Lemma 1 implies that for $(v, v') \in A$, the alignment induces a bipartite matching $M^A_{v,v'}$ between $N(v)$ and $N(v')$ in which the matched elements are exactly those relevant neighbors of v and v' (Figure S2). Say that A is *ordered* if T and S are ordered trees, and for every $(v, v') \in A$, the corresponding bipartite matching $M^A_{v,v'}$ is cyclically ordered. Now, we can define three additional variants of the HSA problem. Let T^v and $S^{v'}$ be rooted and ordered trees. Denote by *Ordered-HSA*(T, S), *Rooted-HSA*$(T^v, S^{v'})$ and *Ordered-Rooted-HSA*$(T^v, S^{v'})$ the minimum costs of an ordered HSA, a rooted HSA, and an ordered and rooted HSA between T^v and $S^{v'}$, respectively. Define the corresponding variants of the Min-Cost HSA problem whose goals are to compute these values.

2.3 Min-cost Bipartite Matching

Similarly to previous tree alignment and edit distance algorithms [19–21], the algorithm presented here makes use of min-cost bipartite matching algorithms as subroutines. We next define an extended variant of the matching problem, in which the score incorporates penalties for unmatched elements, in addition to standard element matching scoring terms.

Let X and Y be two sets. A (generalized) *matching cost function* w for X and Y assigns costs $w(x, y)$ for every $(x, y) \in X \times Y$ and costs $w(z)$ for every $z \in X \cup Y$. The *cost* of a bipartite matching M between X and Y with respect to w is given by $w(M) = \sum\limits_{(x,y) \in M} w(x, y) + \sum\limits_{z \in X \cup Y,\, z \notin M} w(z)$.

A *matching input instance* is a triplet (X, Y, w), where X and Y are two sets, and w is a matching cost function for X and Y. The *Min-Cost Bipartite Matching* problem (*MCM*) is, given a matching instance (X, Y, w), to find the minimum cost of a matching between X and Y with respect to w. Denote by *MCM*(X, Y, w) the *solution* of the *MCM* problem for the instance (X, Y, w), and call a matching whose cost equals to the solution *optimal*.

Numerous works study and suggest algorithms for the *MCM* problem, usually when no unmatched element costs are taken into account [22–25]. A standard approach is to reduce *MCM* to the *Min-Cost Max-Flow* problem, which may be solved in a cubic time with respect to the size of the input sets. In [20], an adapted reduction was presented which also generalizes the problem definition to incorporate unmatched element costs and runs in $O(nm(n+m))$ time, where n and m are the sizes of the two sets in the matching instance. In the supporting materials we give a refined reduction for the generalized variant, which obtains the running time of $O(nm\min(n,m))$.

In addition to the definition above, the tree alignment definitions and algorithms we present in this work make use of the following derivatives of the *MCM* problem. The *Linear Ordered* MCM problem (*Linear-MCM*) and the *Cyclic Ordered* MCM problem (*Cyclic-MCM*) are defined similarly to *MCM*, with the restrictions that the considered matchings have to preserve linear or cyclic order, respectively.

3 A Basic Algorithm for Homeomorphic Subtree Alignment

In this section we describe a basic algorithm for HSA for its unordered unrooted variant (though it is adequate for the other variants as well).

Let A be an HSA between T and S. Let $(v,v') \in A$, and $M^A_{v,v'}$ the corresponding bipartite matching between $N(v)$ and $N(v')$, as defined in Section 2.2. Note that A can be viewed as a rooted alignment between T^v and $S^{v'}$, which is the union of a set of rooted sub-alignments A^v_u between rooted subtree pairs of the form T^v_u and $S^{v'}_{u'}$, where $(u,u') \in M^A_{v,v'}$ (Figure S2). The alignment cost can therefore be obtained by summing the costs of these sub-alignments, which cover all scoring terms implied by matching nodes, smoothing nodes, and pruning subtrees by the corresponding sub-alignments, and the additional pruning costs of pruned subtrees of the forms T^v_u and $S^{v'}_{u'}$ (where u,u' are unmatched by $M^A_{v,v'}$). Note that the pair (v,v') belongs by definition to each of the sub-alignments A^v_u. In order to avoid multiple additions of the term $align(v,v')$ when summing sub-alignment costs, define $w^{\text{-}r}(T^v,S^{v'},A) = w(T^v,S^{v'},A) - align(v,v')$. The cost $w(T^v,S^{v'},A)$ can then be written as follows:

$$w(T^v,S^{v'}\!,A) = w^{\text{-}r}(T^v,S^{v'}\!,A) + align(v,v') \tag{2}$$

$$w^{\text{-}r}(T^v,S^{v'}\!,A) = \sum_{\substack{u\in N(v),\\ u\notin M^A_{v,v'}}} prune(T^v_u) + \sum_{\substack{u'\in N(v'),\\ u'\notin M^A_{v,v'}}} prune(S^{v'}_{u'}) + \sum_{(u,u')\in M^A_{v,v'}} w^{\text{-}r}(T^v_u,S^{v'}_{u'},A^v_u) \tag{3}$$

Call a rooted alignment *non-trivial* if it aligns at least one additional pair of nodes besides the roots. Note that every rooted sub-alignment A^v_u is non-trivial (since u and u' are relevant neighbors of v and v'). Denote by *Rooted-HSA*$^{\text{-}r}\left(T^v_u,S^{v'}_{u'}\right)$ the minimum $w^{\text{-}r}$ cost of a non-trivial rooted alignment between T^v_u and $S^{v'}_{u'}$. Clearly, if A is an optimal rooted HSA between T^v and $S^{v'}$, then for each $(u,u') \in M^A_{v,v'}$ $w^{\text{-}r}\left(T^v_u,S^{v'}_{u'},A^v_u\right) =$ *Rooted-HSA*$^{\text{-}r}\left(T^v_u,S^{v'}_{u'}\right)$ (otherwise, it is possible to produce a rooted alignment with a better cost than A for

T^v and $S^{v'}$). Define the bipartite matching instance $(N(v), N(v'), w_{v,v'})$, where for $u \in N(v),\ u' \in N(v')$, set $w_{v,v'}(u,u') = \textit{Rooted-HSA}^{\text{-}r}(T^v_u, S^{v'}_{u'}),\ w_{v,v'}(u) = \textit{prune}(T^v_u)$, and $w_{v,v'}(u') = \textit{prune}(S^{v'}_{u'})$. Observe that the right-hand side of Equation 3 equals to the cost of the bipartite matching $M^A_{v,v'}$ for the matching instance $(N(v), N(v'), w_{v,v'})$ (see Fig. S2b). In addition, every bipartite matching between $N(v)$ and $N(v')$ corresponds to some valid rooted HSA between T^v and $S^{v'}$, so that the matching and alignment costs are equal. Therefore, a minimum cost bipartite matching induces a minimum cost alignment, and we get that

$$\textit{Rooted-HSA}(T^v, S^{v'}) = \textit{align}(v, v') + \textit{MCM}\ (N(v), N(v'), w_{v,v'}) \tag{4}$$

Assuming the non-degenerate case where an optimal HSA between T and S contains at least one pair (v, v'), we can compute the cost of an optimal HSA by considering all possible such pairs for all the sub-instances:

$$\textit{HSA}(T, S) = \min_{v \in T, v' \in S} \textit{Rooted-HSA}(T^v, S^{v'}) \tag{5}$$

In order to obtain cost functions of the form $w_{v,v'}$ for the computation of Equation 4, we need to compute solutions of the form $\textit{Rooted-HSA}^{\text{-}r}(T^v_u, S^{v'}_{u'})$ for sub-instances of the input. When u and u' are leaves, the only non-trivial rooted alignment between T^v_u and $S^{v'}_{u'}$ contains both pairs (v, v') and (u, u'), and therefore we get that $\textit{Rooted-HSA}^{\text{-}r}\left(T^v_u, S^{v'}_{u'}\right) = \textit{align}(u, u')$. Otherwise, Equation 6, whose correction is shown in the supporting materials, computes $\textit{Rooted-HSA}^{\text{-}r}\left(T^v_u, S^{v'}_{u'}\right)$ recursively (see Figure S3), where $w^{v,v'}_{u,u'}$ is defined similarly to $w_{u,u'}$ with respect to the sets $N(u) \setminus \{v\}$ and $N(u') \setminus \{v'\}$.

$$\textit{Rooted-HSA}^{\text{-}r}\left(T^v_u, S^{v'}_{u'}\right) =$$
$$\min \left\{ \begin{array}{l} \textit{I. smooth}(u) + \min\limits_{x \in N(u)\setminus\{v\}} \left(\textit{Rooted-HSA}^{\text{-}r}\left(T^u_x, S^{v'}_{u'}\right) + \sum\limits_{y \in N(u)\setminus\{v,x\}} \textit{prune}(T^u_y) \right) \\ \textit{II. smooth}(u') + \min\limits_{x' \in N(u')\setminus\{v'\}} \left(\textit{Rooted-HSA}^{\text{-}r}\left(T^v_u, S^{u'}_{x'}\right) + \sum\limits_{y' \in N(u')\setminus\{v',x'\}} \textit{prune}(S^{u'}_{y'}) \right) \\ \textit{III. align}(u, u') + \textit{MCM}\left(N(u) \setminus \{v\}, N(u') \setminus \{v'\}, w^{v,v'}_{u,u'}\right) \end{array} \right\} \tag{6}$$

The computation of $w^{v,v'}_{u,u'}$ requires the computation of scores of the form $\textit{Rooted-HSA}^{\text{-}r}\left(T^u_x, S^{u'}_{x'}\right)$ for all $x \in N(u) \setminus \{v\}$ and all $x' \in N(u') \setminus \{v'\}$. It can be shown that all $\textit{Rooted-HSA}^{\text{-}r}$ solutions required for the computation of the right-hand side of the equation are for strictly smaller sub-instances than the sub-instance appearing in the left-hand side, thus the termination of the recursive computation is guaranteed. Equation 6 can be efficiently computed using Dynamic Programming (DP), as summarized by Algorithm 1 below.

Algorithm 1. *HSA*(T, S)

```
1 Construct a DP matrix $H$ of size $2(n_T - 1) \times 2(n_S - 1)$, where $n_T$ and $n_S$ are
  the numbers of nodes in $T$ and $S$, respectively. The rows of $H$ correspond to
  subtrees $T_u^v$ and the columns correspond to subtrees $S_{u'}^{v'}$, ordered with
  nondecreasing sizes (see Figure S4c);
2 Fill the entries of $H$ row by row with increasing row indices, each row column
  by column, with increasing column indices. Each entry is filled according to
  Equation 6 with respect to the pair of subtrees corresponding to its row and
  column (all solutions for relevant instances in the right-hand side of the
  equation are already computed and stored in $H$, due to the order-by-size
  organization of the subtrees along the rows/columns of the matrix);
3 Compute Equation 4 for every $T^v$, $S^{v'}$ such that $v \in T, v' \in S$;
4 Compute Equation 5 to return $HSA(T, S)$;
```

In the supporting materials we show that a straightforward implementation of Algorithm 1 obtains the running time of $O(n_T n_S d_T d_S \min(d_T, d_S))$. For some trees with $O(n)$ nodes and a maximum node degree of $O(n)$ (e.g. "star" trees), this implies an $O(n^5)$ running time. In the following section, we show how to improve this time bound and obtain a cubic time algorithm for the problem.

We note that Algorithm 1 generalizes to also solve the ordered unrooted, unordered rooted, and ordered rooted variants of *Min-Cost HSA* in polynomial time. In case a rooted alignment is sought, the algorithm can compute Equation 4 in line 3 only with respect to the two roots, and avoid the computation of Equation 5. In case an ordered alignment is sought, the *MCM* application in Equations 4 can be replaced by *Cyclic-MCM*, and in Equation 6 *MCM* can be replaced by *Linear-MCM*, similarly to [20]. Traditionally, ordered matchings are implemented via reduction to sequence alignment [8,19,26]. Similarly to the algorithm given in the next section for unordered tree alignments, fast incremental/decremental versions of ordered matchings [27–29] can be integrated into the ordered variant of our algorithm to improve its time complexity. However, the detailed description of these techniques are beyond the scope of this paper.

3.1 Improving the Algorithm's Time Complexity

Next, we show how to improve the time complexity of Algorithm 1 by incorporating new *cavity matching* algorithms. The *All-Cavity-MCM* problem [21] is, given a matching instance (X, Y, w), to compute $MCM(X, Y \setminus \{y\}, w)$ for all $y \in Y$. The *All-Pairs-Cavity-MCM* problem is, given a matching instance (X, Y, w), to compute $MCM(X \setminus \{x\}, Y \setminus \{y\}, w)$ for all $x \in X$ and $y \in Y$. Clearly, algorithms for both *All-Cavity-MCM* and *All-Pairs-Cavity-MCM* problems can be implemented by repeatedly running an algorithm for *MCM* on all required inputs. In [21], an algorithm for *All-Cavity-MCM* was proposed which is more efficient than the naïve algorithm, and retains the same cubic running time as the standard algorithm for *MCM*. To the best of our knowledge, no algorithm for *All-Pairs-Cavity-MCM* which improves upon the naïve algorithm was previously described. Algorithm 2 in the supplemental materials efficiently solves the *All-Pairs-Cavity-MCM* problem in the same time complexity as the standard *MCM* algorithm. Intuitively, this algorithm obtains a cubic running time by exploiting an observation that a solution for the sub-instance $(X \setminus \{x\}, Y \setminus \{y\}, w)$ is obtained by summing the solutions for the complete instance (X, Y, w) with the minimum cost of a path from y to x in the corresponding residual flow network. The running times of all three unordered bipartite matching problems are summarized in the following theorem.

Theorem 1. *Let (X, Y, w) be a matching instance. Then, each one of the problems* MCM*,* All-Cavity-MCM*, and* All-Pairs-Cavity-MCM *over the instance (X, Y, w) can be solved in $O(|X||Y|\min(|X|, |Y|))$ running time.*

The following lemma identifies *MCM* computations as the time consuming bottleneck of Algorithm 1.

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.*

It thus remains to analyze the time required for executing *MCM* computations. Such computations are applied when computing term *III* of Equation 6 in line 2, and when computing Equation 4 in line 3. In what follows, we explain how cavity-matching algorithms can be used to speed-up these computations.

Let $index(T_u^v)$ and $index(S_{u'}^{v'})$ denote the row and column indices of T_u^v and $S_{u'}^{v'}$ in H, respectively. Let $u \in T$ and $u' \in S$ be a pair of nodes. The set of subtrees T_u^v for $v \in N(u)$ corresponds to a subset of rows in H. Similarly, the set of subtrees $S_{u'}^{v'}$ for $v' \in N(u')$ corresponds to a subset of columns in H, and thus all solutions of the form $Rooted\text{-}HSA^{\text{-}r}\left(T_u^v, S_{u'}^{v'}\right)$ are stored in a sub-matrix of H of size $d_u \times d_{u'}$ (Figure S5). Let $H_{u,u'}$ denote this sub-matrix, and let v_i and v'_j denote nodes in $N(u)$ and $N(u')$ such that $T_u^{v_i}$ and $S_{u'}^{v'_j}$ correspond to the i-th row and j-th column in $H_{u,u'}$, respectively (i.e. $index(T_u^{v_1})$ is the first row in $H_{u,u'}$, $index(T_u^{v_2})$ is the second row, etc.).

The following observation identifies special properties of the second column and second row in $H_{u,u'}$, which are exploited later on for the incorporation of cavity matching subroutines within the DP framework of Algorithm 1. Observe that for every $1 < i \leq d_u$, $T_{v_i}^u$ is a subtree of $T_u^{v_1}$, and therefore $index(T_{v_i}^u) < index(T_u^{v_1})$. Also, $T_{v_1}^u$ is a subtree of $T_u^{v_2}$, and therefore $index(T_{v_1}^u) < index(T_u^{v_2})$. Since $index(T_u^{v_1}) < index(T_u^{v_2})$, we get the following observation (Figure S4):

Observation 1. *For every $1 \leq i \leq d_u$, $index(T_{v_i}^u) < index(T_u^{v_2})$. Similarly, for every $1 \leq j \leq d_{u'}$, $index(S_{v'_j}^u) < index(S_{u'}^{v'_2})$.*

Consider the computation of term *III* of Equation 6 for instances in the first row in $H_{u,u'}$. Note that for the entries in this row, the first group in the bipartite matching instance is fixed and equals to $N(u) \setminus \{v_1\}$, whereas for each column j, the second group in the matching instance is $N(u') \setminus \{v'_j\}$. The first entry in this row is computed by solving the *MCM* problem directly for the matching instance $(N(u) \setminus \{v_1\}, N(u') \setminus \{v'_1\}, w^{v_1,v'_1}_{u,u'})$ (Figures S5a, S5b). Based on Observation 1, upon reaching the second entry in this row, all solutions *Rooted-HSA*$^{\text{-}r}\left(T^u_{v_i}, S^{u'}_{v'_j}\right)$ for $i > 1$ and $j \geq 1$ are computed and stored in H. Therefore, the *All-Cavity-MCM* problem can be solved for the matching instance $(N(u) \setminus \{v_1\}, N(u'), w^{v_1}_{u,u'})$, where $w^{v_1}_{u,u'}$ is defined similarly as $w_{u,u'}$ with respect to the sets $N(u) \setminus \{v_1\}$ and $N(u')$. This allows to compute term III for each one of the remaining entries in this row of $H_{u,u'}$ in $O(1)$ time (Figures S5c, S5d).

The first entry of the second row in $H_{u,u'}$ is again computed directly by solving the *MCM* problem for the matching instance $(N(u) \setminus \{v_2\}, N(u') \setminus \{v'_1\}, w^{v_2,v'_1}_{u,u'})$. Upon reaching the second entry of the second row of $H_{u,u'}$, Observation 1 implies that all solutions *Rooted-HSA*$^{\text{-}r}\left(T^u_{v_i}, S^{u'}_{v'_j}\right)$ for $i, j \geq 1$ are already computed and stored in H. Therefore, the *All-Pairs-Cavity-MCM* problem can be solved for the matching instance $(N(u), N(u'), w_{u,u'})$, allowing to compute term III for each one of the remaining entries in $H_{u,u'}$ in $O(1)$ time (Figures S5e, S5f). Thus, computing term III for all entries in $H_{u,u'}$ is done by solving the *MCM* problem twice, solving the *All-Cavity-MCM* problem once, and solving the *All-Pairs-Cavity-MCM* problem once, where the sizes of the two groups in the matching instances for these problems are at most d_u and $d_{u'}$. Based on Theorem 1, this whole computation takes $O(d_u d_{u'} \min(d_u, d_{u'}))$ time. As all computations of the *MCM* problem conducted by the algorithm are done either in the computation of term III for some entry in a submatrix $H_{u,u'}$, or for some pair $v \in T$, $v' \in S$ when computing Equation 4 in line 3, we can argue that in total, for each pair $u \in T$, $u' \in S$, the algorithm spends $O(d_u d_{u'} \min(d_u, d_{u'}))$ in computations of the *MCM* problem. As $\sum_{u \in T} \sum_{u' \in S} d_u d_{u'} \min(d_u, d_{u'}) \leq \min(d_T, d_S) \sum_{u \in T} d_u \sum_{u' \in S} d_{u'} = O\left(\min(d_T, d_S) n_T n_S\right)$, the total running time for solving the *MCM* problem along the entire run of the algorithm is $O(n_T n_S \min(d_T, d_S))$. Based on Lemma 2, all other operations require $O(n_T n_S)$ time, and we get the following theorem:

Theorem 2. *Algorithm 1 can be implemented with an $O(n_T n_S \min(d_T, d_S))$ time complexity.*

In the supplemental materials we give a refined time complexity analysis showing that the algorithm's time is in fact $O(n_T n_S + L_T L_S \min(d_T, d_S))$, where L_T and L_S are the number of leaves in T and S, respectively. In addition, we show that for RNA trees T in our specific application it is expected that $L_T \ll n_T$.

4 Results

Our algorithm is implemented (in Java) as a tool called FRUUT (Fast RNA Unordered Unrooted Tree mapper). The RNA tree representation and scoring scheme employed by FRUUT are described in detail in Section S5. FRUUT allows the user to select any alignment mode (rooted/unrooted, ordered/unordered, local/global) and to compute optimal pairwise alignments of RNA trees. We also provide an interactive PHP web-server for running FRUUT in our website (RNA plots are are generated by the Vienna Package [30]).

RNase P is the endoribonuclease responsible for the 5' maturation of tRNA precursors [15]. Secondary structures of bacterial RNase P RNAs have been studied in detail, primarily using comparative methods [31], and were shown to share a common core of primary and secondary structure. In bacteria, synthetic minimal RNase P RNAs consisting only of these core sequences and structures were shown to be catalytically proficient. Sequences encoding RNase P RNAs of various genomes have been determined and a huge database established [32], which consists of a compilation of ribonuclease P RNA sequences, sequence alignments, secondary structures, three-dimensional models and accessory information.

We conducted a preliminary experiment, intended to identify examples of pairs of RNA trees for which an RNA structural comparison approach supporting unrooting and branch shuffling may detect (otherwise hidden) structural similarity. To achieve this, we ran a benchmark of all-against-all pairwise alignments of bacterial RNAse P RNA secondary structure trees, using our tool's different tree-alignment modes and comparing the differences between the obtained alignment costs. The alignment cost functions and parameters used in our experiment are given in Section S5.2.

Our benchmark was based on 470 RNAse P structures, ranging across various organisms, taken from the RNAse P database [32] (molecule naming conventions are according to [33]). After filtering out partial and environmental sequences, 170 distinct structures remained, yielding 14,365 distinct pairs of trees. The sizes of the trees in this dataset ranged from 82 to 230 nodes, averaging at 141.99. The total running time of the benchmark was approximately 33 minutes on a single Xeon X5690 using around 300Mb of memory.

Each pair of trees T, S was compared in two modes to obtain the corresponding scores and alignments: rooted-ordered (RO) and rooted-unordered (RU). Within each mode, we used a relative score formula described by Höchsmann et al. [34] to assess the similarity of two trees, normalizing the alignment cost by the average of the self-alignment costs of the compared trees. Let $HSA_m(T, S)$ denote the optimal alignment cost of trees T and S in alignment mode m, where $m \in \{RO, RU\}$. Let $RelScore_m(T, S)$ denote the relative score of T and S in alignment mode m, given by $RelScore_m(T, S) = \frac{2HSA_m(T,S)}{HSA_m(T,T)+HSA_m(S,S)}$. The scoring scheme we have satisfies that for every tree T, $HSA_m(T, T) < 0$, and for every pair of trees T, S, $HSA_m(T, T), HSA_m(S, S) \leq HSA_m(T, S)$. Under these conditions, the relative score for any pair of trees is upper bounded by 1, and the similarity of the trees increases as the score approaches 1.

Our goal in this experiment was to identify evolutionary events that can be explained by unordered alignments. Thus, we sought pairs of RNAseP RNAs that are highly conserved, and yet their alignment can still benefit from unordered mappings. To achieve this, we removed from the set pairs of trees for which

$RelScore_{RU}(T,S) < 0.5$. We sorted the remaining pairs of trees according to $RelScore_{RU}(T,S) - RelScore_{RO}(T,S)$.

When examining the top 50 alignments carefully, two distinct types of mapping patterns were observed among them, where each of the top 50 pairs belongs (with slight variations) to one of these two types (33 to Type 1 and 17 to Type 2). In Section S6, we exemplify the highest ranking alignment of each of the two types (the first type is shown in Figure 1b). As mentioned before, the input for FRUUT alignments consisted only of sequence and secondary structure information. The tertiary structure (pseudoknot annotations) for the top-ranking alignments were only considered later, during the alignment interpretations.

Another type of homology detected by our tool is exemplified in the HammerHead 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. Additional details regarding the experiment for detecting such unroooted alignments are given in the supplemental materials.

Acknowledgments. The authors are grateful to the anonymous WABI referees for their helpful comments, and to Naama Amir for pointing out challenges in extending previous algorithms to allow deletions from both trees. This research was partially supported by ISF grant 478/10 and by the Frankel Center for Computer Science at Ben Gurion University of the Negev.

References

1. Andronescu, M., Bereg, V., Hoos, H., Condon, A.: Rna strand: the rna secondary structure and statistical analysis database. BMC Bioinformatics 9, 340 (2008)
2. Agmon, I., Auerbach, T., Baram, D., Bartels, H., Bashan, A., Berisio, R., Fucini, P., Hansen, H., Harms, J., Kessler, M., et al.: On peptide bond formation, translocation, nascent protein progression and the regulatory properties of ribosomes. European Journal of Biochemistry 270, 2543–2556 (2003)
3. Hofacker, I., Fontana, W., Stadler, P., Bonhoeffer, L., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. Monatshefte fur Chemie/Chemical Monthly 125, 167–188 (1994)
4. Steffen, P., Voss, B., Rehmsmeier, M., Reeder, J., Giegerich, R.: RNAshapes: an integrated RNA analysis package based on abstract shapes (2006)
5. Hochsmann, M., Toller, T., Giegerich, R., Kurtz, S.: Local similarity in RNA secondary structures. In: Proceedings of the 2003 IEEE Bioinformatics Conference, CSB 2003, pp. 159–168. IEEE (2003)
6. Jiang, T., Lin, G., Ma, B., Zhang, K.: A general edit distance between RNA structures. Journal of Computational Biology 9, 371–388 (2002)
7. Zhang, K., Wang, L., Ma, B.: Computing Similarity between RNA Structures. In: Crochemore, M., Paterson, M. (eds.) CPM 1999. LNCS, vol. 1645, pp. 281–293. Springer, Heidelberg (1999)
8. Bille, P.: A survey on tree edit distance and related problems. Theoretical Computer Science 337, 217–239 (2005)

9. Schirmer, S., Giegerich, R.: Forest Alignment with Affine Gaps and Anchors. In: Giancarlo, R., Manzini, G. (eds.) CPM 2011. LNCS, vol. 6661, pp. 104–117. Springer, Heidelberg (2011)
10. Allali, J., Sagot, M.-F.: A Multiple Graph Layers Model with Application to RNA Secondary Structures Comparison. In: Consens, M.P., Navarro, G. (eds.) SPIRE 2005. LNCS, vol. 3772, pp. 348–359. Springer, Heidelberg (2005)
11. Blin, G., Denise, A., Dulucq, S., Herrbach, C., Touzet, H.: Alignments of RNA structures. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7, 309–322 (2010)
12. Jan, E.: Divergent ires elements in invertebrates. Virus Research 119, 16–28 (2006)
13. Perreault, J., Weinberg, Z., Roth, A., Popescu, O., Chartrand, P., Ferbeyre, G., Breaker, R.: Identification of hammerhead ribozymes in all domains of life reveals novel structural variations. PLoS Computational Biology 7, e1002031 (2011)
14. Birikh, K., Heaton, P., Eckstein, F.: The structure, function and application of the hammerhead ribozyme. European Journal of Biochemistry 245, 1–16 (1997)
15. Haas, E., Brown, J.: Evolutionary variation in bacterial RNase P RNAs. Nucleic Acids Research 26, 4093–4099 (1998)
16. Zhang, K., Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees. Information Processing Letters 49, 249–254 (1994)
17. Shamir, R., Tsur, D.: Faster subtree isomorphism. J. of Algorithms 33, 267–280 (1999)
18. Chung, M.: O (n2. 5) time algorithms for the subgraph homeomorphism problem on trees. Journal of Algorithms 8, 106–112 (1987)
19. Pinter, R.Y., Rokhlenko, O., Tsur, D., Ziv-Ukelson, M.: Approximate labelled subtree homeomorphism. Journal of Discrete Algorithms 6, 480–496 (2008)
20. Zhang, K.: A constrained edit distance between unordered labeled trees. Algorithmica 15, 205–222 (1996)
21. Kao, M., Lam, T., Sung, W., Ting, H.: Cavity matchings, label compressions, and unrooted evolutionary trees. Arxiv preprint cs/0101031 (2001)
22. Edmonds, J., Karp, R.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) 19, 248–264 (1972)
23. Fredman, M., Tarjan, R.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) 34, 596–615 (1987)
24. Gabow, H., Tarjan, R.: Faster scaling algorithms for network problems. SIAM Journal on Computing 18, 1013 (1989)
25. Orlin, J., Ahuja, R.: New scaling algorithms for the assignment and minimum mean cycle problems. Mathematical Programming 54, 41–56 (1992)
26. Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recognition 28, 463–474 (1995)
27. Maes, M.: On a cyclic string-to-string correction problem. Information Processing Letters 35, 73–78 (1990)
28. Schmidt, J.P.: All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. SIAM J. of Computing 27, 972–992 (1998)
29. Tiskin, A.: Semi-local string comparison: Algorithmic techniques and applications. Mathematics in Computer Science 1, 571–603 (2008)
30. Hofacker, I.: Vienna RNA secondary structure server. Nucleic Acids Research 31, 3429 (2003)

31. Pace, N.R., Brown, J.W.: Evolutionary perspective on the structure and function of ribonuclease P, a ribozyme. J. Bacteriol. 177, 1919–1928 (1995)
32. Brown, J.: The ribonuclease p database. Nucleic acids research 27, 314 (1999)
33. Andronescu, M., Bereg, V., Hoos, H.H., Condon, A.: RNA STRAND: the RNA secondary structure and statistical analysis database. BMC Bioinformatics 9, 340 (2008)
34. Höchsmann, M.: The tree alignment model: algorithms, implementations and applications for the analysis of RNA secondary structures. PhD thesis, Universitätsbibliothek Bielefeld (2005)