Normalized Similarity of RNA Sequences

Rolf Backofen*, Danny Hermelin**,
Gad M. Landau***, and Oren Weimann**

Abstract. We introduce a normalized version of the LCS metric as a new local similarity measure for comparing two RNAs. An $\mathcal{O}(n^2 m \lg m)$ time algorithm is presented for computing the maximum normalized score of two RNA sequences, where n and m are the lengths of the sequences and $n \leq m$. This algorithm has the same time complexity as the currently best known global LCS algorithm.

1 Introduction

Sequence comparison is an extensively studied topic with many applications, especially in biology. One commonly used metric is *longest common subsequence* (LCS) [2, 10, 11] which measures the longest subsequence of symbols that appears in both input sequences. While the LCS metric is a suitable metric for global comparison, in many real-life applications one is often interested in finding local regions of high similarity [16]. One approach for transforming the global LCS metric into a local version, is to calculate the *normalized longest common subsequence* [3, 7]. Here, one divides the LCS score of two substrings by the sum of their lengths. This approach overcomes various weaknesses that are inherent in the standard local similarity algorithm [16].

In RNA sequences, as in other biological applications, it is not sufficient to perform pure sequence-based comparisons without respecting the underlying semantics of the sequences. RNAs are polymers consisting of four nucleotides A,C,G and U which are connected linearly via a backbone. In addition, the complementary nucleotides A—U, G—C and G—U can form bonds, which define the secondary structure of the RNA. In recent years, RNA sequences gained increasing interest due to numerous discoveries of biological functions which are associated with them. Consequently, research on small RNAs has been elected as the scientific breakthrough of the year 2002 by the readers of Science [6].

One major challenge of this research is to find common patterns in RNAs, since they suggest functional similarities. For this purpose, one has to investigate

* Institute of Computer Science, Friedrich-Schiller Universität Jena, Jena Center for Bioinformatics, Germany. **backofen@inf.uni-jena.de**.

** Department of Computer Science, University of Haifa, Israel. Partially supported by the Israel Science Foundation grant 282/01. **danny@cri.haifa.ac.il**, **oweimann@cs.haifa.ac.il**.

*** Department of Computer Science, University of Haifa, Haifa - Israel, and Department of Computer and Information Science, Polytechnic University, New York - USA. Partially supported by the Israel Science Foundation grant 282/01. **landau@cs.haifa.ac.il**.

not only sequential features, but also structural features for the following reasons. First, a major fraction of the function of an RNA is determined by its secondary structure [14]. Second, it is known that RNA structure is often more conserved than the sequence during evolution [5].

One promising approach for comparing RNAs while considering their sequence and secondary structure is to use appropriate variants of LCS-like metrics. This has been widely studied in the literature. One variant which is known under the term "*longest common subsequence for arc-annotated sequences*" (LAPCS), was first introduced by Evans [8], and then later extensively studied in [1, 9, 12]. However, the major downfall of this variant is that a base-pair, *i.e.* hydrogen bond, is not regarded as a whole entity. For example, in comparison of two RNAs, one nucleotide in a base-pair can be matched in the LCS, while the other nucleotide unmatched. In this paper we adopt a different variant introduced by Zhang [17] which treats base-pairs as a whole. This method is closer to the spirit of the comparative analysis method currently being used in the analysis of RNA secondary structures, either manually or automatically.

All known approaches for transferring the LCS metric to a metric of comparing RNA sequences have been restricted to global comparisons so far. However, as in almost all biological applications, global similarity is inferior to local similarity when comparing RNA sequences. In this paper, we consider a local variant of the LCS metric. Specifically, we consider the local normalized LCS metric for RNA sequences which measures the highest LCS scoring consecutive subsequences divided by their length. The advantages of the normalized approach in the context of strings [3, 7] also apply for RNAs. We present an $\mathcal{O}(n^2 m \lg m)$ time algorithm for this problem which is conceptually inspired by the algorithm given in [7]. Its time complexity origins in the global LCS algorithm presented in [13], which is used as a preprocessing procedure in the algorithm. Therefore, our algorithm can compute the local normalized LCS score at the cost of computing the global LCS score.

This paper is organized as follows. In the following section we introduce definitions and terminology which will be used throughout the paper. In Section 3, we describe methods which provide the basis of our algorithm. The algorithm itself is later presented in Section 4.

2 Preliminaries

2.1 RNA sequences

An RNA sequence $\mathcal{R}$ is an ordered pair (S, P), where $S = s_1 \cdots s_{|S|}$ is a string over the alphabet $\Sigma = \{A, C, G, U\}$, and $P \subseteq \{1, \ldots, |S|\} \times \{1, \ldots, |S|\}$ is the set of hydrogen bonds between bases of $\mathcal{R}$ (*i.e.* the secondary structure), such that $\forall (i, i') \in P : i < i'$. Any base in $\mathcal{R}$ can bond with at most one other base, therefore we have $\forall\ (i_1, i'_1), (i_2, i'_2) \in P$, $i_1 = i_2 \Leftrightarrow i'_1 = i'_2$. Furthermore, following Zuker [18, 19], we assume a model where the bonds in P are *non crossing*, *i.e.* for any $(i_1, i'_1), (i_2, i'_2) \in P$, we cannot have $i_1 < i_2 < i'_1 < i'_2$ nor $i_2 < i_1 < i'_2 < i'_1$.

We refer to a bond $(i, i') \in P$ as an *arc*, where i (i') is referred to as the *left* (*right*) *endpoint* of the arc. Finally, we let $|\mathcal{R}|$ denote the number of nucleotides in $\mathcal{R}$, *i.e.* $|\mathcal{R}| = |S|$.

We will require a notion similar to that of a substring for RNA sequences. Therefore, for any $1 \leq i < i' \leq |S|$, we let $\mathcal{R}[i, i']$, the *consecutive subsequence* of $\mathcal{R}$, be the RNA (S', P') with $S' = S_i \cdots S_{i'}$ and $P' = P \cap \{i, \ldots, i'\} \times \{i, \ldots, i'\}$. Note that arcs of $\mathcal{R}$ with one endpoint in $\mathcal{R}[i, i']$ are absent in $\mathcal{R}[i, i']$. If there are no such arcs, then $\mathcal{R}[i, i']$ is said to be *arc complete*.

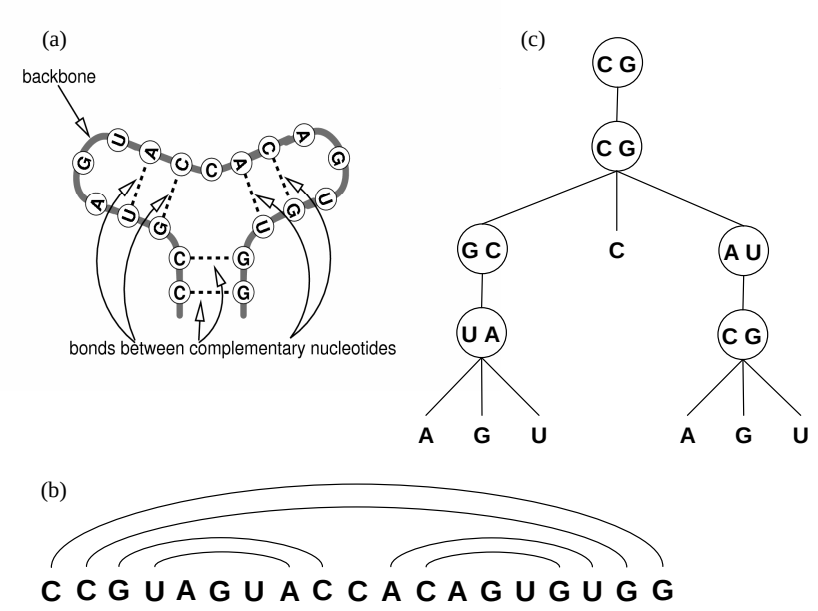

Fig. 1. Three different ways of viewing an RNA sequence. In (a), a schematic 2-dimensional description of an RNA folding. In (b), a linear representation of the RNA. In (c), the RNA as a rooted ordered tree.

2.2 LCS similarity

This paper deals with comparing two RNA sequences. We denote these two RNAs by $\mathcal{R}_1 = (S_1, P_1)$ and $\mathcal{R}_2 = (S_2, P_2)$, and we set $|\mathcal{R}_1| = |S_1| = n$ and $|\mathcal{R}_2| = |S_2| = m$. Furthermore, we assume $n \leq m$.

A base in one sequence can be matched to an identical base in the other sequence if they are both non arc endpoints. In the case of arcs, given $(i, i') \in P_1$ and $(j, j') \in P_2$, we require that (i, j) be matched if and only if (i', j') is matched. This captures the notion of arcs as single entities. A match between two left

(right) endpoints of arcs is called a *left arc match* (respectfully *right arc match*). Note that for every left arc match (i, j) there exists exactly one corresponding right arc match (i', j'), such that $(i, i') \in P_1$ and $(j, j') \in P_2$. In this case we let $right(i, j) = (i', j')$ and $left(i', j') = (i, j)$.

An RNA sequence $\mathcal{R}'$ is a *common subsequence* of $\mathcal{R}_1$ and $\mathcal{R}_2$ if it can be obtained by omitting unpaired bases and arcs (along with their endpoints) in both $\mathcal{R}_1$ and $\mathcal{R}_2$. Alternatively, $\mathcal{R}'$ is a set of matches $\mathcal{M} = (i_1, j_1), \ldots, (i_p, j_p)$, such that $i_k < i_{k+1}$, $j_k < j_{k+1}$ for all $1 \leq k < p$, and $(i, j) \in \mathcal{M} \Leftrightarrow right(i, j) \in \mathcal{M}$ for any left arc match (i, j). The longest common subsequence (LCS) of $\mathcal{R}_1$ and $\mathcal{R}_2$, denoted $LCS(\mathcal{R}_1, \mathcal{R}_2)$, is a common subsequence of $\mathcal{R}_1$ and $\mathcal{R}_2$ of maximum cardinality.

The non crossing formation formed by the arcs in both $\mathcal{R}_1$ and $\mathcal{R}_2$ conveniently allows representing these RNAs as trees [17]. Each arc is represented by an internal node in the tree, and each unpaired base by a leaf. The set of ordered children of an internal node which is associated with the arc (i, i') is all unpaired bases i'' such that $i < i'' < i'$, and all arcs (l, l') such that $i < l < l' < i'$ (see Fig. 1). In [15], an algorithm for tree editing was presented which was later improved in [13]. This algorithm can be used to determine the minimum number of unpaired base and arc deletions needed in order to obtain a common subsequence of $\mathcal{R}_1$ and $\mathcal{R}_2$ in $\mathcal{O}(n^2 m \lg m)$ time, and is currently the fastest (worst-case) algorithm for computing the global LCS score of two RNA sequences [4]. Furthermore, an important property of this algorithm is that it computes the score between every pair of subtrees of the two given trees. In our setting this means that $LCS(\mathcal{R}[i, i'], \mathcal{R}[j, j'])$ is computed between all pairs of arcs $(i, i') \in P_1$ and $(j, j') \in P_2$ in a single execution of this algorithm. The importance of this property will become apparent later on.

2.3 Normalized similarity

We next present an extension of a local similarity metric for strings that uses normalization [7], to a metric for RNA sequences. Following this, we define the computational problem considered in this paper.

Definition 1 (Normalized LCS score). *The normalized LCS score of two RNA sequences $\mathcal{R}'_1$ and $\mathcal{R}'_2$ is given by*

$$\frac{|LCS(\mathcal{R}'_1, \mathcal{R}'_2)|}{|\mathcal{R}'_1| + |\mathcal{R}'_2|}.$$

The above definition is of a global nature. We therefore define the *local* normalized LCS score of two RNA sequences, $\mathcal{R}_1$ and $\mathcal{R}_2$, as the normalized LCS score of the two highest scoring consecutive subsequences of $\mathcal{R}_1$ and $\mathcal{R}_2$. More formally:

Definition 2 (Local normalized LCS score). *The local normalized LCS score of two RNA sequences, $\mathcal{R}_1$ and $\mathcal{R}_2$, is the maximal value of*

$$\frac{|LCS(\mathcal{R}_1[i, j], \mathcal{R}_2[i', j'])|}{|\mathcal{R}_1[i, j]| + |\mathcal{R}_2[i', j']|}$$

where $LCS(\mathcal{R}_1[i,j],\mathcal{R}_2[i',j'])$ *is also a common subsequence of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$*, and* $1 \le i \le j \le n$*,* $1 \le i' \le j' \le m$*.*

Note the requirement of $LCS(\mathcal{R}_1[i,j],\mathcal{R}_2[i',j'])$ being a common subsequence of $\mathcal{R}_1$ and $\mathcal{R}_2$ is crucial in case either $\mathcal{R}_1[i,i']$ or $\mathcal{R}_2[j,j']$ are not arc complete. In this case, the above requirement prevents matching an endpoint of an arc whose other endpoint is absent in the consecutive subsequence, thereby ensuring that local solutions are valid also as global solutions.

Furthermore, notice that by the above definition, one single match gets the optimal normalized LCS score (1/2). To solve this problem, we require that $|LCS(\mathcal{R}_1[i,j],\mathcal{R}_2[i',j'])| \ge I$, where I is some integer (perhaps dependent of n) predefined according to the application at hand.

Definition 3 (The local normalized LCS problem). *Given two RNA sequences* $\mathcal{R}_1 = (S_1, P_1)$ *and* $\mathcal{R}_2 = (S_2, P_2)$*, and an integer* I*, the local normalized LCS problem asks to compute the local normalized LCS score of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$*.*

3 Decomposing common subsequences

In the following section we present techniques for decomposing common subsequences of our two RNA sequences $\mathcal{R}_1 = (S_1, P_1)$ and $\mathcal{R}_2 = (S_2, P_2)$. We will be interested in a small fraction of such subsequences. These can intuitively be thought of as locally optimal common subsequences which contain exactly k matches, for all $1 \le k \le n$. In [7], a *k-Chain* is defined as a common subsequence of two given strings which consists of k matches. We adopt this terminology for our case as follows.

Definition 4. $k\text{-}Chain^{(i,j)}_{(i',j')}$ *is a common subsequence of* $\mathcal{R}_1[i,i']$ *and* $\mathcal{R}_2[j,j']$ *which consists of* k *matches.*

- $k\text{-}Chain^{(i,j)}_{(i',j')}$ starts *at* (i,j) *and* ends *at* (i',j')*.*
- *The* head *of* $k\text{-}Chain^{(i,j)}_{(i',j')}$ *is the first match in* $k\text{-}Chain^{(i,j)}_{(i',j')}$*.*
- *The* tail *of* $k\text{-}Chain^{(i,j)}_{(i',j')}$ *is the last match in* $k\text{-}Chain^{(i,j)}_{(i',j')}$*.*
- *The* length *of* $k\text{-}Chain^{(i,j)}_{(i',j')}$ *is the sum of* $|\mathcal{R}_1[i,i']|$ *and* $|\mathcal{R}_2[j,j']|$, i.e. $j'-j+i'-i$*.*
- *The* normalized score *of* $k\text{-}Chain^{(i,j)}_{(i',j')}$ *is given by* $\frac{k}{j'-j+i'-i}$*.*

This definition differs from the definition in [7], since $k\text{-Chain}^{(i,j)}_{(i',j')}$ is defined there only when (i,j) and (i',j') are matches. Next we define the best scoring k-Chain that starts at (i,j).

Definition 5. $k\text{-}Chain^{(i,j)}$ *is* $k\text{-}Chain^{(i,j)}_{(i',j')}$ *with the highest normalized score (shortest length) over all* $i' \le n$ *and* $j' \le m$*.*

A major obstacle in constructing k-Chains, is that any attempt to construct k-Chain$^{(i,j)}$ simply by tying another match to the tail of $(k-1)$-Chain$^{(i,j)}$ will not necessarily result in the optimal k-Chain. We therefore take the opposite approach. From among all chains which start at (i', j'), $i \leq i', j \leq j$ and $(i, j) \neq (i', j')$, we choose the one that when concatenated to (i, j), creates k-Chain$^{(i,j)}$.

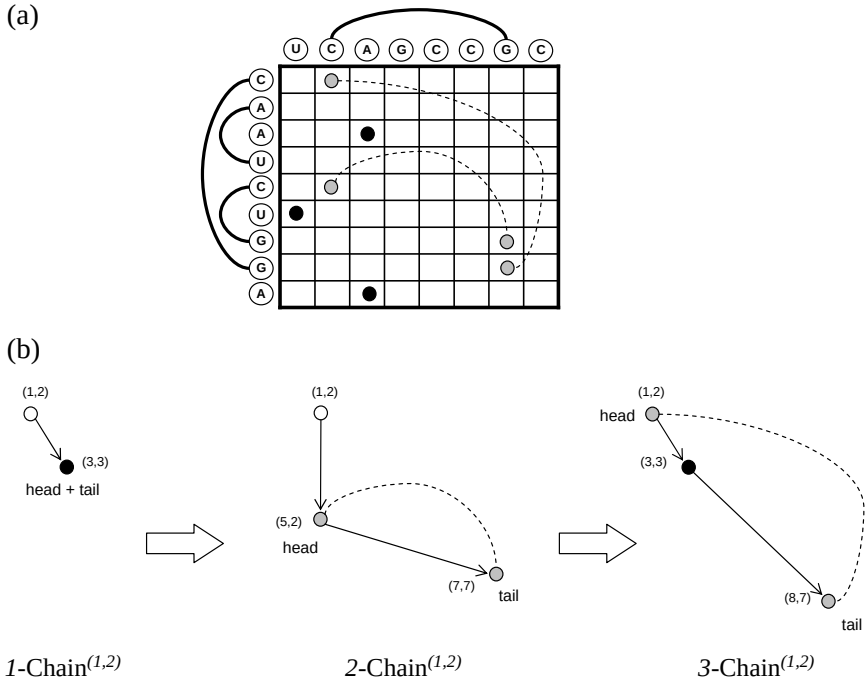

Fig. 2. Constructing k-Chains. (a) The matches of two RNA sequences. (b) The construction of 3-Chain$^{(1,2)}$.

In order to construct k-Chain$^{(i,j)}$, we distinguish between four different cases depending on (i, j). Indeed, (i, j) can either be a mismatch, a non arc match, a right arc match, or a left arc match. Note that if (i, j) is a non arc match then we can assume that it is the head of k-Chain$^{(i,j)}$, since otherwise by replacing the head with (i, j), we obtain a k-Chain with the same score. Furthermore, if (i, j) is a right arc match, then (i, j) cannot be the head of k-Chain$^{(i,j)}$, since the left arc match corresponding to (i, j) is not in k-Chain$^{(i,j)}$, and by definition, (i, j) cannot appear alone in any common subsequence.

In the following we give further details concerning each one of these four cases. Later these will provide the basis for a dynamic programming procedure which we design for solving the local normalized LCS problem.

For a pair of matches $(i,j),(i',j') \in \{1,\ldots,n\} \times \{1,\ldots,m\}$, we refer to the value $|i'-i|+|j'-j|$ as the *distance* between (i,j) and (i',j').

Definition 6 (k-closest). *The* k-closest *chain to* (i,j) *is the* k*-Chain with minimum distance between its tail and* (i,j) *among all* k*-Chains starting at* (i',j') *with* $i \leq i', j \leq j'$*, and* $(i',j') \neq (i,j)$*.*

Case 1: (i,j) *is a mismatch.* In this case, by definition, k-Chain$^{(i,j)}$ consists of the k matches of the k-closest chain to (i,j).

Case 2: (i,j) *is a non arc match.* In this case, (i,j) is the head of k-Chain$^{(i,j)}$. Therefore, k-Chain$^{(i,j)}$ consists of (i,j) and the k–1 matches of the $(k$–1$)$-closest chain to (i,j) starting at (i',j') such that $i < i'$ and $j < j'$.

Case 3: (i,j) *is a right arc match.* In this case, (i,j) cannot be the head of k-Chain$^{(i,j)}$. Furthermore, any match (i,j') or (i',j) is also a right arc match. Therefore, k-Chain$^{(i,j)}$ consists of the k matches of the k-closest chain to (i,j) starting at (i',j') such that $i < i'$ and $j < j'$.

Case 4: (i,j) *is a left arc match.* This is the most delicate case. Indeed, k-Chain$^{(i,j)}$ may or may not include (i,j). If $(i,j) \notin k$-Chain$^{(i,j)}$, then the head of k-Chain$^{(i,j)}$ can still be (i,j') or (i',j) for some $i' > i$ and $j' > j$. Therefore, in this case k-Chain$^{(i,j)}$ consists of the k matches of the k-closest chain to (i,j).

In case $(i,j) \in k$-Chain$^{(i,j)}$, then $(i',j') = right(i,j)$ is also in k-Chain$^{(i,j)}$ by definition. Therefore, k-Chain$^{(i,j)}$ is of length at least $i'-i+j'-j$. Let $\mathcal{M} = LCS(\mathcal{R}_1[i,i'],\mathcal{R}_2[j,j'])$. From the optimality of k-Chain$^{(i,j)}$, it follows that in this case $k > |\mathcal{M}|-2$, since otherwise there exists a shorter k-Chain which doesn't include (i,j) nor (i',j'). If $k = |\mathcal{M}|$, then k-Chain$^{(i,j)}$ consists exactly of the matches in $\mathcal{M}$. If $k \geq |\mathcal{M}|$, then k-Chain$^{(i,j)}$ consists of all matches of $\mathcal{M}$ and all matches of $(k-|\mathcal{M}|)$-Chain$^{(i',j')}$. The case $k = |\mathcal{M}|-1$ is eccentric. Here, k-Chain$^{(i,j)}$ consists of all but one match of $\mathcal{M}$ (which is neither (i,j) or $right(i,j)$). In this case, $(k+1)$-Chain$^{(i,j)}$ has a higher normalized score than k-Chain$^{(i,j)}$, and so this case is mentioned only for completeness.

4 The algorithm

We are now in position to describe our algorithm for computing the local normalized similarity score of our two given RNA sequences $\mathcal{R}_1 = (S_1,P_1)$ and $\mathcal{R}_2 = (S_2,P_2)$. Recall that we are looking for the highest normalized scoring k-Chain such that $k \geq I$.

Our algorithm begins in a preprocessing stage which consists of two phases. In the first phase, for each $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,m\}$, the algorithm classifies (i,j) as one of the following four types: mismatch, non arc match, right arc match, or left arc match. In the second phase, the algorithm computes and stores the longest common subsequence of $\mathcal{R}_1[i,i']$ and $\mathcal{R}_2[j,j']$ for every $(i,i') \in P_1$

and $(j, j') \in P_2$. As mentioned in Section 2.2, this can be done by a single execution of the algorithm given in [13].

The second phase is the bottleneck of our algorithm. Nevertheless, this computation is necessary for efficiently constructing k-Chains which start at left arc matches. According to the fourth case above, if (i, j) is a left arc match and $right(i, j) = (i', j')$, then the computation of k-Chain$^{(i,j)}$ is based on $\mathcal{M} = LCS(\mathcal{R}_1[i, i'], \mathcal{R}_2[j, j'])$. In the dynamic programming computation below, $\mathcal{M}$ is computed for any pair of arcs $(i, i') \in P_1$ and $(j, j') \in P_2$. Using [13] allows us to compute this efficiently, in some cases at the cost of computing $\mathcal{M}$ for a single pair. On the other hand, in cases where the number of nesting edges is small, one can use standard LCS algorithms.

After the preprocessing stage is complete, the algorithm computes and stores the score of k-Chain$^{(i,j)}$, for all $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$ and for all $1 \leq k \leq n$. More precisely, the algorithm computes and stores the lengths of all these k-Chains, since the score of a k-Chain can be derived from its length and vice versa. Let $DP^k[i, j]$ denote the length of k-Chain$^{(i,j)}$. For $k = 1$, the recursion of $DP^1[i, j]$ is given by:

$$DP^1[i,j] = \begin{cases} 2 & (i,j) \text{ is a non arc match,} \\ \min \begin{cases} DP^1[i{+}1, j{+}1] + 2 \\ DP^1[i, j{+}1] + 1 \\ DP^1[i{+}1, j] + 1 \end{cases} & \text{Otherwise.} \end{cases}$$

For $k > 1$, we let $(i', j') = right(i, j)$ and $\mathcal{M} = LCS(\mathcal{R}_1[i, i'], \mathcal{R}_2[j, j'])$. The recursion of $DP^k[i, j]$ is then given by:

$$DP^k[i,j] = \begin{cases} \min \begin{cases} DP^k[i{+}1, j{+}1] + 2 \\ DP^k[i, j{+}1] + 1 \\ DP^k[i{+}1, j] + 1 \end{cases} & (i,j) \text{ is a mismatch.} \\ DP^{k-1}[i{+}1, j{+}1] + 2 & (i,j) \text{ is a non arc match.} \\ DP^k[i{+}1, j{+}1] + 2 & (i,j) \text{ is a right arc match.} \\ \min \begin{cases} DP^k[i{+}1, j{+}1] + 2 \\ DP^k[i, j{+}1] + 1 \\ DP^k[i{+}1, j] + 1 \\ j' - j + i' - i + DP^{k-|\mathcal{M}|}[i', j'] \end{cases} & (i,j) \text{ is a left arc match and } k > |\mathcal{M}|. \\ \min \begin{cases} DP^k[i{+}1, j{+}1] + 2 \\ DP^k[i, j{+}1] + 1 \\ DP^k[i{+}1, j] + 1 \\ j' - j + i' - i \end{cases} & (i,j) \text{ is a left arc match and } k \leq |\mathcal{M}|. \end{cases}$$

The final stage of the algorithm consists of analyzing all DP tables and reporting a solution. Here, the algorithm can either report the normalized score

of the highest scoring k-Chain such that $k \geq I$, or the consecutive subsequences $\mathcal{R}_1[i, i']$ and $\mathcal{R}_2[j, j']$ that correspond to this chain. Furthermore, if required, the algorithm can be modified to report all chains with a score higher than some given threshold, *e.g.* 80%.

Correctness of our algorithm follows from the discussion in Section 3.

Time complexity. The preprocessing stage can be done in $\mathcal{O}(n^2 m \lg m)$ time using the algorithm in [13]. Furthermore, computing all DP requires $\mathcal{O}(n^2 m)$ time. Hence, the total time complexity of our algorithm is $\mathcal{O}(n^2 m \lg m)$.

Acknowledgments

We would like to thank Dennis Shasha and Kaizhong Zhang for fruitful discussions. Furthermore, we owe our gratitude to an anonymous referee for pointing out an error in the preliminary version of this paper.

References

1. Alber J., J. Gramm, J. Guo and R. Niedermeier. Towards optimally solving the longest common subsequence problem for sequences with nested arc annotations in linear time. *Proc. of the 13th Combinatorial Pattern Matching conference (CPM 2002)*, LNCS vol. 2373, 99-114, 2002.
2. Apostolico A. and C. Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2:315-336, 1987.
3. Arslan A.N., Ö. Eğecioğlu and P.A. Pevzner. A new approach to sequence alignment: normalized sequence alignment. *Bioinformatics*, 17(4):327-337, 2001.
4. Bille P. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217-239, 2005.
5. Chartrand P., X-H. Meng, R.H. Singer and R.M. Long. Structural elements required for the localization of ASH1 mRNA and of a green fluorescent protein reporter particle *in vivo*. *Current Biology*, 9:333-336, 1999.
6. Couzin J. Breakthrough of the year. Small RNAs make big splash. *Science*, 298(5602):2296-2297, 2002.
7. Efraty N. and G.M. Landau. Sparse normalized local alignment. *Proc. of the 15th Combinatorial Pattern Matching conference (CPM 2004)*, LNCS vol. 3109, 333-346, 2004.
8. Evans P.A. Algorithms and complexity for annotated sequence analysis. *PhD thesis, University of Alberta*, 1999.
9. Gramm J., J. Guo and R. Niedermeier. Pattern matching for arc annotated sequences. *Proc. of the 22nd Foundations of Software Technologies and Theoretical Computer Science conference (FSTTSC 2002)*, LNCS vol. 2556, 182-193, 2002.
10. Hirschberg D.S. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664-675, 1977.
11. Hunt J.W. and T.G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350-353, 1977.
12. Jiang T., G-H. Lin, B. Ma and K. Zhang. The longest common subsequence problem for arc-annotated sequences. *Proc. of the 11th Combinatorial Pattern Matching conference (CPM 2000)*, LNCS vol. 1848, 154-165, 2000.

13. Klein P.N. Computing the Edit-Distance between Unrooted Ordered Trees. *Proc. of the 6th European Symposium on Algorithms conference (ESA 1998)*, LNCS vol. 1461, 91-102, 1998.
14. Moore P.B. Structural motifs in RNA. *Annual review of biochemistry*, 68:287-300, 1999.
15. Shasha D. and K. Zhang. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245-1262, 1989.
16. Smith T.F. and M.S. Waterman. The identification of common molecular subsequences. *Journal of Molecular Biology* , 147:195-197, 1981.
17. Zhang K. Computing similarity between RNA secondary structures. *Proc. of the IEEE joint symposium on Intelligence and Systems conference*, 126-132, 1998.
18. Zuker M. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48-52, 1989.
19. Zuker M. and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133-148, 1981.