Journal of Discrete Algorithms 9 (2011) 2–11

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Fast RNA structure alignment for crossing input structures

Rolf Backofen [a], Gad M. Landau [b,c,1], Mathias Möhl [d], Dekel Tsur [e,*], Oren Weimann [f]

[a] Bioinformatics, Institute of Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany
[b] Department of Computer Science, University of Haifa, Haifa 31905, Israel
[c] Department of Computer Science and Engineering, NYU-Poly, Six MetroTech Center, Brooklyn, NY 11201-3840, USA
[d] Programming Systems Lab, Saarland University, Saarbrücken, Germany
[e] Ben-Gurion University, Beer-Sheva, Israel
[f] Massachusetts Institute of Technology, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Available online 19 August 2010

Keywords:
RNA
Sequence structure alignment
Simultaneous alignment and folding

ABSTRACT

The complexity of pairwise RNA structure alignment depends on the structural restrictions assumed for both the input structures and the computed consensus structure. For arbitrarily crossing input and consensus structures, the problem is NP-hard. For non-crossing consensus structures, Jiang et al.'s (2002) [9] algorithm computes the alignment in $O(n^2m^2)$ time where n and m denote the lengths of the two input sequences. If the input structures are also non-crossing, the problem corresponds to tree editing which can be solved in $O(m^2n(1+\log\frac{n}{m}))$ time (Demaine et al., 2007) [3]. We present a new algorithm that solves the problem for d-crossing structures in $O(dm^2n\log n)$ time, where d is a parameter that is one for non-crossing structures, bounded by n for crossing structures, and much smaller than n on many practical examples. Crossing input structures allow for applications where the input is not a fixed structure but is given as base-pair probability matrices.

1. Introduction

With the recent focus on non-protein-coding RNA (ncRNA) genes, interest in detecting novel ncRNAs has rapidly emerged. A recent screen on ncRNAs has detected more than 30 000 putative ncRNAs in human genome [14], most of them with unknown function. Since the structure of RNA is evolutionarily more conserved than its sequence, predicting the RNA's secondary structure is the most important step towards its functional analysis [2].

The secondary structure of an RNA molecule can be calculated from its nucleotide sequence by determining a folding with minimal free energy [15,12,19,1,16]. Albeit this so-named thermodynamic approach is a success story in the analysis of RNA, it is known that predicting the secondary structure from a single sequence is error-prone, where the best available approaches can correctly predict only up to 73% of the base-pairs [4]. This situation can be improved by taking phylogenetic information into account, i.e., by predicting a common consensus structure from a whole set of evolutionary related RNA sequences.

There are several approaches for the comparative RNA structure prediction (see [7] for an overview). One approach is to predict for every input sequence the minimum free-energy non-crossing structure (in $O(n^3)$ time), and then perform

* Corresponding author.
E-mail addresses: backofen@informatik.uni-freiburg.de (R. Backofen), landau@cs.haifa.ac.il (G.M. Landau), mmohl@ps.uni-sb.de (M. Möhl), dekelts@cs.bgu.ac.il (D. Tsur), oweimann@mit.edu (O. Weimann).

1 Partially supported by the National Science Foundation Award 0904246, Israel Science Foundation grants 35/05 and 347/09, the Israel–Korea Scientific Research Cooperation, Yahoo, Grant No. 2008217 from the United States–Israel Binational Science Foundation (BSF) and DFG.

1570-8667/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2010.07.004

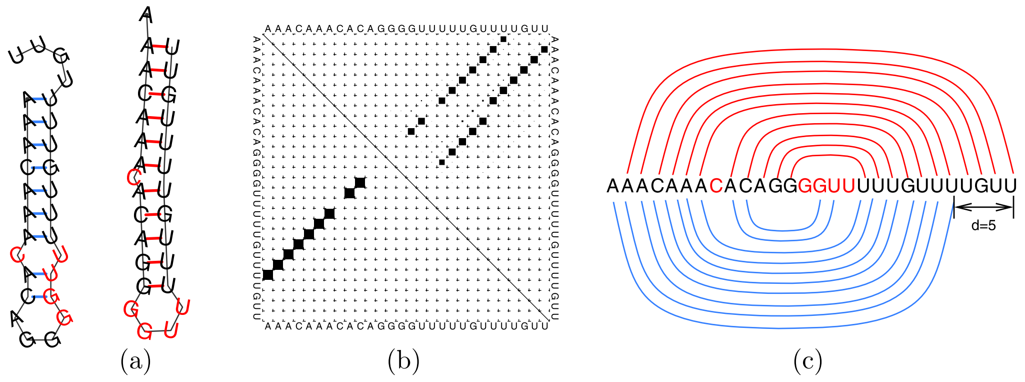

Fig. 1. (a) Two structures for the sequence AAACAAACACAGGGGUUUUUGUUUUGUU with similar free energy. The stem in the second sequence is shifted by 5 nucleotides. (b) Associated base-pair probability matrix (upper triangle) and minimum free energy structure (lower triangle). The shifted stem is indicated by two parallel diagonals, a pattern often seen in RNA-structures. (c) Both nested structures together form a 5-crossing structure. Note that this structure forms a two-page embedding (or is 2-colorable, as it is called in [5]), but our approach is not restricted to this class of structures.

pairwise sequence–structure alignments. The problem of aligning two non-crossing structures corresponds to tree editing and can be solved in $O(n^3)$ time [3]. However, this approach crucially depends on the quality of the initial structure prediction, which is error-prone.

Hence, the gold standard are Sankoff-like approaches [13,11,8,18,17] which *simultaneously* align and fold the sequences. This approach can be viewed as performing sequence–structure alignments, where the structure of each input sequence consists of all possible base pairs. The complexity of the alignment in this approach is $O(n^6)$. Will et al. [17] reduced this complexity to $O(n^4)$ using the following approach: For each input sequence, compute a base-pair probability matrix. Then, build a crossing structure for each sequence by taking base pairs whose probabilities are above some threshold. Each structure contains $O(n)$ arcs, and therefore aligning the structures takes $O(n^4)$ time.

In this work, we shorten the gap between structure alignment of non-crossing structures (with a complexity of $O(n^3)$), and the Sankoff-like approaches (with a complexity of $O(n^4)$ for alignment of sparse crossing structures) for a practical application scenario. In many practical cases, the base-pair probability matrix gives a main structure that allows for a small deviation. As shown in the example in Fig. 1, the alternative structures together form a crossing input structure, where the offset between crossing arcs is small. In this paper, we introduce a measurement for this deviation (d-crossing), and introduce an efficient alignment algorithm with complexity $O(n^3 \log n)$ given that the deviation is small. The fast available structure alignment methods for non-crossing input structures [10,3] rely on a heavy path decomposition which was so far only available for tree-like structures. Our approach generalizes this to d-crossing structures.

2. Preliminaries

An *arc-annotated sequence* is a pair (S, P), where S is a string over the set of bases $\{A, U, C, G\}$ and P is a set of arcs (l, r) with $1 \leqslant l < r \leqslant |S|$ representing bonds between bases. We allow more than one arc to be adjacent to one base, but require that $|P| \in O(|S|)$, that is, on average each base is adjacent to only a constant number of arcs. We denote the i-th symbol of S by $S[i]$ and the substring from symbol i to symbol j with $S[i \ldots j]$. For an arc $p = (l, r)$, we denote its left end l and right end r by p^{L} and p^{R}, respectively. The *span* of p is defined as $\mathrm{span}(p) = p^{\mathrm{R}} - p^{\mathrm{L}} + 1$.

Two arcs p_1 and p_2 in an arc-annotated sequence (S, P) are *crossing* if $p_1^{\mathrm{L}} \leqslant p_2^{\mathrm{L}} \leqslant p_1^{\mathrm{R}} \leqslant p_2^{\mathrm{R}}$ or $p_2^{\mathrm{L}} \leqslant p_1^{\mathrm{L}} \leqslant p_2^{\mathrm{R}} \leqslant p_1^{\mathrm{R}}$. Two crossing arcs p_1 and p_2 are *d-crossing* if $|p_1^{\mathrm{L}} - p_2^{\mathrm{L}}| < d$ and $|p_1^{\mathrm{R}} - p_2^{\mathrm{R}}| < d$. An arc p_1 is *nested* in an arc p_2 if $p_2^{\mathrm{L}} < p_1^{\mathrm{L}} < p_1^{\mathrm{R}} < p_2^{\mathrm{R}}$. An arc p_1 *precedes* an arc p_2 if $p_1^{\mathrm{R}} < p_2^{\mathrm{L}}$. For every two arcs, either the two arcs are crossing, one of the arc is nested in the other, or one of the arc precedes the other. An arc-annotated sequence (S, P) containing crossing arcs is called *crossing*, otherwise *non-crossing* or *nested*. A *d-crossing sequence* is a crossing sequence in which every two crossing arcs are d-crossing.

3. Problem definition

An *alignment* A of two arc-annotated sequences (S_1, P_1) and (S_2, P_2) is a set $A = A_{\text{match}} \uplus A_{\text{gap}}$. The set $A_{\text{match}} \subseteq [1, n] \times [1, m]$ of *match pairs* satisfies that for all $(i, j), (i', j') \in A$, (1) $i > i'$ implies $j > j'$, and (2) $i = i'$ if and only if $j = j'$. Given A_{match}, the set of *gap pairs* is implied as $A_{\text{gap}} := \{(i, -) \mid i \in [1, n] \wedge \nexists j.(i, j) \in A_{\text{match}}\} \cup \{(-, j) \mid j \in [1, m] \wedge \nexists i.(i, j) \in A_{\text{match}}\}$. A *consensus structure* for an alignment A is a matching $P \subseteq P_1 \times P_2$ that satisfies $(p_1, p_2) \in P \Rightarrow (p_1^{\mathrm{L}}, p_2^{\mathrm{L}}) \in A \wedge (p_1^{\mathrm{R}}, p_2^{\mathrm{R}}) \in A$. We require a consensus structure to be non-crossing, namely $\{(p_1, p_2), (p_1', p_2')\} \subseteq P \Rightarrow p_1$ and p_1' do not cross.

Each alignment together with some consensus structure has an associated cost based on functions $\gamma_1 \in [1, n] \to \mathbb{N}$, $\gamma_2 \in [1, m] \to \mathbb{N}$, $\beta \in [1, n] \times [1, m] \to \mathbb{N}$, and $\alpha \in ([1, n])^2 \times ([1, m])^2 \to \mathbb{N}$. $\gamma_k(i)$ denotes the cost to align position i of sequence k to a gap, $\beta(i, j)$ the cost for a base match, i.e. cost to align position i of the first sequence to position j of the

$$M[i,i',j,j'] = \min \begin{cases} M[i,i'-1,j,j'] + \gamma_1(i') & \text{I} \\ M[i,i',j,j'-1] + \gamma_2(j') & \text{II} \\ M[i,i'-1,j,j'-1] + \beta(i',j') & \text{III} \\ \text{for all } p_1 = (i_0,i') \in P_1,\ p_2 = (j_0,j') \in P_2 \text{ with } i \leqslant i_0,\ j \leqslant j_0 & \text{IV} \\ \quad M[i,i_0-1,j,j_0-1] + M[i_0+1,i'-1,j_0+1,j'-1] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 2. Recurrence for the table M.

second sequence, provided arcs adjacent to i and j are not contained in the consensus structure, and $\alpha(p_1,p_2)$ denotes the cost to match arcs p_1 and p_2 in the consensus structure. The cost of an alignment A with consensus structure P, denoted $C_P(A)$, is

$$\sum_{(i,-)\in A} \gamma_1(i) + \sum_{(-,j)\in A} \gamma_2(j) + \sum_{(p_1,p_2)\in P} \alpha(p_1,p_2) + \sum_{(i,j)\in A'} \beta(i,j),$$

where A' is the set of all pairs $(i,j) \in A_{\text{match}}$ such that no arc in P is adjacent to i or to j. Note that this scoring scheme can easily be instantiated with the edit distance scoring scheme of Jiang et al. [9] if each base is adjacent to at most one arc. For this case we set $\gamma_1(i) = w_d + \psi_1(i)(\frac{w_r}{2} - w_d)$, $\gamma_2(j) = w_d + \psi_2(j)(\frac{w_r}{2} - w_d)$, $\beta(i,j) = \chi(i,j)w_m + (\psi_1(i) + \psi_2(j))\frac{w_b}{2}$, and $\alpha((i,j),(i',j')) = (\chi(i,j) + \chi(i',j'))\frac{w_{am}}{2}$ where ψ_1, ψ_2, χ, w_d, w_r, w_m, w_b, and w_{am} are defined as in [9]. However, we formulate the algorithm with the more general scoring scheme, since $\alpha((i,j),(i',j'))$ can be used to encode base pair weights which is more suitable in the presence of several adjacent arcs per base that represent alternative structures.

The *RNA structure alignment problem* is given two arc-annotated sequences (S_1,P_1) and (S_2,P_2), to find an alignment A and a consensus structure P such that $C_P(A)$ is minimal. For the remainder of this paper we fix two arc-annotated sequences (S_1,P_1) and (S_2,P_2) with $|S_1| = n$, $|S_2| = m$, $|P_1| \in O(n)$ and $|P_2| \in O(m)$ and assume that (S_1,P_1) is d-crossing. We assume w.l.o.g. that P_1 contains an arc $(1,n)$.

Arc annotated sequences are often classified as PLAIN, NEST, CROSS or UNLIM, as originally proposed in [6]. We solve for our scoring scheme the edit problem for a class that fully contains EDIT(NEST, NEST) and partially contains EDIT(UNLIM, UNLIM) (namely those instances where one structure is d-crossing and where on average each base is adjacent to only a constant number of arcs).

4. The algorithm

The algorithm consists of two stages. The first stage computes the optimal costs to align certain fragments that are required for the second stage.

4.1. Stage 1

In the first stage, the algorithm computes a table M analogously to the recurrence of Jiang et al. [9]. Let $\text{OPT}(i,i',j,j')$ denote the minimal cost of an alignment between $(S_1[i\ldots i'], P_1 \cap [i,i']^2)$ and $(S_2[j\ldots j'], P_2 \cap [j,j']^2)$. The entry $M[i,i',j,j']$ stores the value $\text{OPT}(i,i',j,j')$.

The base cases where $i' = i-1$ and $j' = j-1$ are initialized with $M[i,i-1,j,j-1] = 0$, the other entries are computed recursively as defined in Fig. 2. In the recursive computation, cases that rely on invalid items (i.e. where any of i, i', j, j' are not within their allowed range) are implicitly skipped. While Jiang et al.'s algorithm computes the entire alignment based on this recurrence, we only compute entries of M for short fragments of the first sequence that have a length of at most $2d+2$, i.e. for $1 \leqslant i \leqslant n$, $i-1 \leqslant i' \leqslant \min(i+2d+1,n)$, $1 \leqslant j \leqslant m$, and $j-1 \leqslant j' \leqslant m$.

4.2. Stage 2

For non-crossing input structures, the correspondence of these structures to trees allows for alignment methods that are asymptotically faster than the recurrence used in the first stage [10,3]. In our approach we apply a similar technique, but since our input structures do not correspond to trees, we select a subset $P_T \subseteq P_1$ of the arcs. The arcs in P_T do not cross and at most one of them is adjacent to each base. Hence, the arcs in P_T form a tree structure that guides the recursive decomposition during the computation of the alignment.

4.2.1. Construction of P_T

Define the *inner d-range* of an arc p (with span at least $2d+1$) as $I_d(p) = [p^{\text{L}}+1, p^{\text{L}}+d-1] \times [p^{\text{R}}-d+1, p^{\text{R}}-1]$. For a set of arcs $P \subseteq P_1$, the set $\text{tree}(P)$ is defined recursively as follows. If $P = \emptyset$ or all arcs in P have span at most $2d$ then $\text{tree}(P) = \emptyset$. Otherwise, let p be some arc in P with maximum span (ties are broken arbitrarily), and

$$\text{tree}(P) = \{p\} \cup \text{tree}\big(P \cap \big[1, p^{\text{L}}-1\big]^2\big) \cup \text{tree}\big(P \cap \big[p^{\text{R}}+1, n\big]^2\big) \cup \text{tree}\big(\big(P \cap \big[p^{\text{L}}+1, p^{\text{R}}-1\big]^2\big) \setminus I_d(p)\big).$$

Lemma 1. *Every arc in P crosses at most one arc in* $\operatorname{tree}(P)$.

Proof. Let p_1 and p_2 be two arcs in $\operatorname{tree}(P)$, and assume w.l.o.g. that $p_1^{\mathrm{L}} < p_2^{\mathrm{L}}$. We have that either p_2 is nested in p_1 or p_1 precedes p_2.

If p_2 is nested in p_1 then by the definition of $\operatorname{tree}(P)$, either $p_2^{\mathrm{L}} - p_1^{\mathrm{L}} \geqslant d$ or $p_1^{\mathrm{R}} - p_2^{\mathrm{R}} \geqslant d$. Suppose w.l.o.g. that $p_2^{\mathrm{L}} - p_1^{\mathrm{L}} \geqslant d$. Let p be an arc that crosses p_1. If $p^{\mathrm{L}} \leqslant p_1^{\mathrm{L}}$ then $|p^{\mathrm{L}} - p_2^{\mathrm{L}}| \geqslant p_2^{\mathrm{L}} - p_1^{\mathrm{L}} \geqslant d$, so p does not cross p_2. If $p^{\mathrm{L}} > p_1^{\mathrm{L}}$ then $p^{\mathrm{L}} \leqslant p_1^{\mathrm{L}} + d - 1 < p_2^{\mathrm{L}}$ and $p^{\mathrm{R}} \geqslant p_1^{\mathrm{R}} > p_2^{\mathrm{R}}$. Therefore, p_2 is nested in p, and in particular, p does not cross p_2.

If p_1 precedes p_2 then $p_2^{\mathrm{L}} > p_1^{\mathrm{R}} = p_1^{\mathrm{L}} + \operatorname{span}(p_1) - 1 \geqslant p_1^{\mathrm{L}} + 2d$. Therefore, for every arc p, either $|p^{\mathrm{L}} - p_1^{\mathrm{L}}| \geqslant d$, or $|p^{\mathrm{L}} - p_2^{\mathrm{L}}| \geqslant d$. We conclude that p cannot cross both p_1 and p_2. $\square$

Lemma 2. *An arc $p \in P$ satisfies $p \in I_d(p')$ for at most one arc $p' \in$* $\operatorname{tree}(P)$. *If p does not cross an arc in* $\operatorname{tree}(P)$ *then $p \in I_d(p')$ for a unique arc $p' \in$* $\operatorname{tree}(P)$.

Proof. To prove the first part of the lemma, let p_1 and p_2 be two arcs in $\operatorname{tree}(P)$ with $p_1^{\mathrm{L}} < p_2^{\mathrm{L}}$. Either p_2 is nested in p_1 or p_1 precedes p_2. If p_2 is nested in p_1 then either $p_2^{\mathrm{L}} - p_1^{\mathrm{L}} \geqslant d$ or $p_1^{\mathrm{R}} - p_2^{\mathrm{R}} \geqslant d$. In the former case, the intervals $[p_1^{\mathrm{L}} + 1, p_1^{\mathrm{L}} + d - 1]$ and $[p_2^{\mathrm{L}} + 1, p_2^{\mathrm{L}} + d - 1]$ are disjoints, and therefore $I_d(p_1) \cap I_d(p_2) = \phi$. Similarly, $I_d(p_1) \cap I_d(p_2) = \phi$ when $p_1^{\mathrm{R}} - p_2^{\mathrm{R}} \geqslant d$ or when p_1 precedes p_2. Thus, p cannot be both in $I_d(p_1)$ and $I_d(p_2)$.

We prove the second part of the lemma using induction on $|P|$. Let $P \subseteq P_1$ be a nonempty set of arcs, and let p be some arc in P that does not cross an arc in $\operatorname{tree}(P)$. Let p' be the maximum span arc in P that is chosen when computing $\operatorname{tree}(P)$. Recall that $\operatorname{tree}(P) = \{p'\} \cup \operatorname{tree}(P^1) \cup \operatorname{tree}(P^2) \cup \operatorname{tree}(P^3)$ where $P^1 = P \cap [1, p'^{\mathrm{L}} - 1]^2$, $P^2 = P \cap [p'^{\mathrm{R}} + 1, n]^2$, and $P^3 = (P \cap [p'^{\mathrm{L}} + 1, p'^{\mathrm{R}} - 1]^2) \setminus I_d(p')$. If $p \in I_d(p')$ we are done. Otherwise, since p does not cross p' and $p \notin I_d(p')$, we have that p is in some set P^i. Since $|P^i| < |P|$, by the induction hypothesis there is an arc $p'' \in \operatorname{tree}(P^i)$ such that $p \in I_d(p'')$. $\square$

We define $P_T = \operatorname{tree}(P_1)$, and we call the arcs in P_T *tree arcs*. For every $p \in P_1$ we define $T(p)$ to be the unique tree arc p' such that p crosses p', if such arc exists. Otherwise, $T(p)$ is the unique tree arc p' such that $p \in I_d(p')$. The definition of $T(\cdot)$ is valid due to Lemma 2.

Lemma 3. *For every $p \in P_1$, $|p^{\mathrm{L}} - T(p)^{\mathrm{L}}| < d$ and $|p^{\mathrm{R}} - T(p)^{\mathrm{R}}| < d$.*

Proof. If p crosses $T(p)$ then the inequalities of the lemma are satisfied since (S_1, P_1) is d-crossing. Otherwise, $p \in I_d(T(p))$, and the inequalities of the lemma are satisfied by the definition of $I_d(\cdot)$. $\square$

Lemma 4. *Let $p \in P_1$ and let $p' \in P_T$ such that $p' \neq p$ and p' is nested in $T(p)$. Then, p' is nested in p.*

Proof. Let p and p' be two arcs satisfying the conditions of the lemma. From the definition of $T(\cdot)$, p cannot cross p'. Moreover, from Lemma 3 and the fact that $\operatorname{span}(p) > 2d$, p' cannot precede p, or vice versa. $\square$

For every tree arc $p \in P_T$ we select a tree arc denoted $\operatorname{hchild}(p)$ such that $\operatorname{hchild}(p)$ is nested in p and $\operatorname{span}(\operatorname{hchild}(p))$ is maximum (if there is such an arc). For $p \in P_T$ and $p \neq (1, n)$, define $\operatorname{parent}(p)$ to be the minimum span tree arc that p is nested in. We define $\operatorname{parent}((1, n)) = (1, n)$.

4.2.2. Recurrence

For each $p \in P_T$ we build two tables L^p and R^p. Intuitively, one obtains the optimal alignments of the area below p or any arc crossing p by first extending the optimal alignments of $\operatorname{hchild}(p)$ or any arc crossing $\operatorname{hchild}(p)$ to the left (with L^p) and then to the right (with R^p). The algorithm computes the tables in an order such that for each p, L^p is computed before R^p, and the tables of all $p' \in P_T$ that are nested in p are computed before the tables of p.

The table entries $L^p[i, i', j, j']$ and $R^p[i, i', j, j']$ have the same semantics as $M[i, i', j, j']$ and only differ in the domains of the indices i, i', j, j' and the recurrences according to which they are computed. Let us first assume that $\operatorname{hchild}(p)$ is defined for p. Then, $L^p[i, i', j, j']$ is defined for

$$\begin{aligned}
&\max\bigl(p^{\mathrm{L}} - d, \operatorname{parent}(p)^{\mathrm{L}}\bigr) \leqslant i \leqslant \operatorname{hchild}(p)^{\mathrm{L}} - 1,\\
&\operatorname{hchild}(p)^{\mathrm{R}} + 1 \leqslant i' \leqslant \min\bigl(\operatorname{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}}\bigr),\\
&1 \leqslant j \leqslant m,\\
&j - 1 \leqslant j' \leqslant m
\end{aligned}$$

and for $R^p[i, i', j, j']$ the domains of j and j' are the same, but i and i' must satisfy

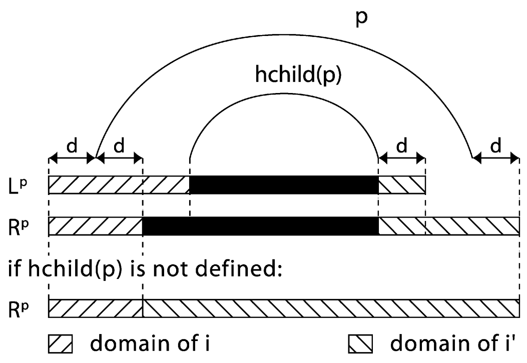

Fig. 3. Visualization of the domains for the different tables.

$$L^p[i,i',j,j'] = \min \begin{cases} L^p[i+1,i',j,j'] + \gamma_1(i) & \text{I} \\ L^p[i,i',j+1,j'] + \gamma_2(j) & \text{II} \\ L^p[i+1,i',j+1,j'] + \beta(i,j) & \text{III} \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j', & \\ \text{and hchild}(p) \text{ is nested in } p_1 & \text{IV} \\ \quad L^p[i+1,i_0-1,j+1,j_0-1] + M[i_0+1,i',j_0+1,j'] + \alpha(p_1,p_2) & \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j', & \\ \text{hchild}(p) \text{ is not nested in } p_1, \text{ and span}(p_1) \leqslant 2d & \text{V} \\ \quad M[i+1,i_0-1,j+1,j_0-1] + L^p[i_0+1,i',j_0+1,j'] + \alpha(p_1,p_2) & \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j', & \\ \text{hchild}(p) \text{ is not nested in } p_1, \text{ and span}(p_1) > 2d & \text{VI} \\ \quad R^{T(p_1)}[i+1,i_0-1,j+1,j_0-1] + L^p[i_0+1,i',j_0+1,j'] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 4. Recurrence for the table L^p.

$$\max\bigl(p^{\mathrm{L}} - d, \text{parent}(p)^{\mathrm{L}}\bigr) \leqslant i \leqslant \min\bigl(p^{\mathrm{L}} + d, \text{hchild}(p)^{\mathrm{L}} - 1\bigr),$$
$$\text{hchild}(p)^{\mathrm{R}} + 1 \leqslant i' \leqslant \min\bigl(p^{\mathrm{R}} + d, \text{parent}(p)^{\mathrm{R}}\bigr).$$

If hchild(p) is not defined for p, no L^p table is computed and the R^p tables contain entries for

$$\max\bigl(p^{\mathrm{L}} - d, \text{parent}(p)^{\mathrm{L}}\bigr) \leqslant i \leqslant p^{\mathrm{L}} + d,$$
$$p^{\mathrm{L}} + d \leqslant i' \leqslant \min\bigl(p^{\mathrm{R}} + d, \text{parent}(p)^{\mathrm{R}}\bigr)$$

and j, j' restricted as in the table R^p in the case where hchild(p) is defined. The domains of i and i' for the different cases are visualized in Fig. 3.

Computation of L^p. All entries $L^p[i,i',j,j]$ with $i \geqslant \max(\text{hchild}(p)^{\mathrm{L}} - d, p^{\mathrm{L}})$ are initialized as $L^p[i,i',j,j'] = R^{\text{hchild}(p)}[i,i',j,j']$. All other entries are computed according to the recurrence shown in Fig. 4. Again cases relying on invalid items are implicitly skipped. The last three cases of the recurrence are visualized in Fig. 5.

Computation of R^p. The computation of the R^p tables is similar to the computation of the L^p tables, only that the fragments are extended to the right instead of to the left. If hchild(p) is defined, we initialize all entries with $i' \leqslant \min(\text{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}})$ as $R^p[i,i',j,j'] = L^p[i,i',j,j']$. All other items are computed according to the recurrence shown in Fig. 6. If hchild(p) is not defined, we initialize all items with $i' = p^{\mathrm{L}} + d$ as $R^p[i,i',j,j'] = M[i,i',j,j']$. The recurrence for R^p in this case includes lines I, II, III, and V from Fig. 6.

Once the tables are computed, the actual alignment can be constructed using the usual backtrace technique.

4.3. Correctness

Let (A,P) be an optimal alignment and consensus structure for the fragments corresponding to some table entry $M[i,i',j,j']$, $L^p[i',i,j',j]$, or $R^p[i,i',j,j']$ (note the swapped indices in the entry of L^p). In all recurrences, lines I and II cover the cases where A aligns i' or j' to a gap. Line III covers the cases where $(i',j') \in A$ and no arcs of P are adjacent to i' or j'. Furthermore i' and j' can never be adjacent to arcs of the consensus structure whose other end is outside of the current fragment (due to the semantics of the table entries). Hence, the case that remains is where i' and j' are one

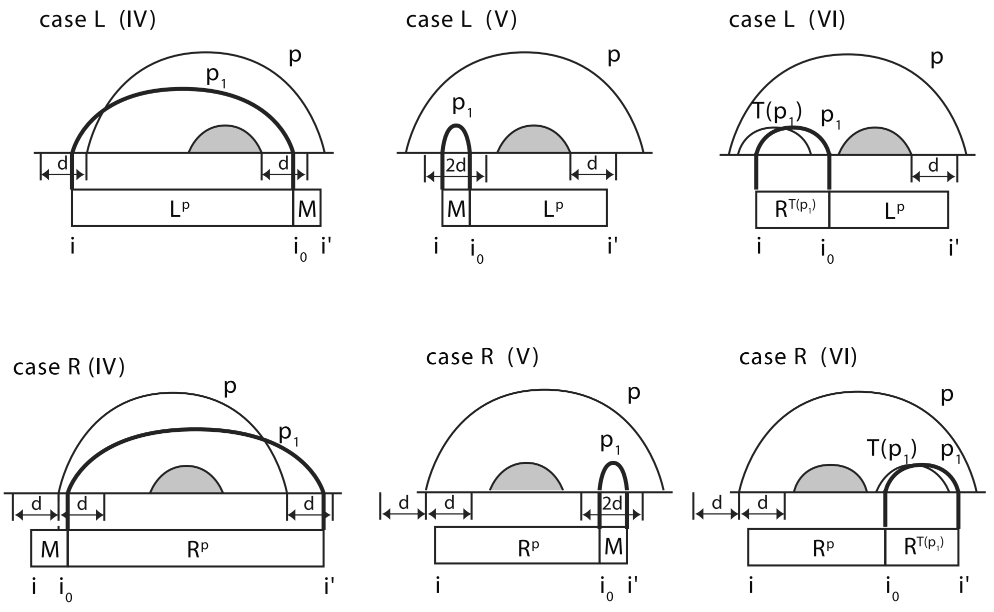

Fig. 5. Visualization of the recurrence cases. The arc bounding the gray area denotes hchild(p).

$$R^p[i,i',j,j'] = \min \begin{cases} R^p[i,i'-1,j,j'] + \gamma_1(i') & \text{I} \\ R^p[i,i',j,j'-1] + \gamma_2(j') & \text{II} \\ R^p[i,i'-1,j,j'-1] + \beta(i',j') & \text{III} \\ \text{for all } p_1=(i_0,i')\in P_1,\ p_2=(j_0,j')\in P_2 \text{ with } i\leqslant i_0,\ j\leqslant j_0, & \\ \text{and hchild}(p) \text{ is nested in } p_1 & \text{IV} \\ \quad M[i,i_0-1,j,j_0-1] + R^p[i_0+1,i'-1,j_0+1,j'-1] + \alpha(p_1,p_2) & \\ \text{for all } p_1=(i_0,i')\in P_1,\ p_2=(j_0,j')\in P_2 \text{ with } i\leqslant i_0,\ j\leqslant j_0, & \\ \text{hchild}(p) \text{ is not nested in } p_1, \text{ and span}(p_1)\leqslant 2d & \text{V} \\ \quad R^p[i,i_0-1,j,j_0-1] + M[i_0+1,i'-1,j_0+1,j'-1] + \alpha(p_1,p_2) & \\ \text{for all } p_1=(i_0,i')\in P_1,\ p_2=(j_0,j')\in P_2 \text{ with } i\leqslant i_0,\ j\leqslant j_0, & \\ \text{hchild}(p) \text{ is not nested in } p_1, \text{ and span}(p_1)> 2d & \text{VI} \\ \quad R^p[i,i_0-1,j,j_0-1] + R^{T(p_1)}[i_0+1,i'-1,j_0+1,j'-1] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 6. Recurrence for the table R^p.

end of some arc of the consensus structure whose other end is also contained in the current fragment. In the recurrence for M, this case is covered in line IV, and in the recurrences for L and R this case is further decomposed into subcases corresponding to lines IV to VI. In all those cases, the fragment is decomposed in the arc match (p_1, p_2), the fragment below the arc match and the fragment before it (or behind it, in the case of the table L). This decomposition is correct since the consensus structure is nested and hence cannot contain other arc pairs whose arcs cross p_1 and p_2 to connect the fragments before and below (p_1, p_2). It remains to show that in each case the table entries we recursively descend to exist.

Fix an arc $p \in P_T$ for which hchild(p) is defined (the case where hchild(p) is not defined is similar). Let $p_1 = (i_0, i')$ be an arc considered in lines IV to VI of the recurrence for R^p.

Lemma 5. *p_1 does not cross* hchild(p).

Proof. Since the case $i' \leqslant \min(\text{hchild}(p)^{\mathrm{R}}+d, p^{\mathrm{R}})$ is handled by the initialization of R^p, we have $i' > \min(\text{hchild}(p)^{\mathrm{R}}+d, p^{\mathrm{R}})$. Therefore, either $i' > \text{hchild}(p)^{\mathrm{R}} + d$ or $i' > p^{\mathrm{R}}$. In the former case we have from the assumption that (S_1, P_1) is d-crossing that p_1 does not cross hchild(p). In the latter case we also have that p_1 does not cross hchild(p) since otherwise, p_1 would also cross p, contradicting Lemma 1. □

By Lemma 5, either hchild(p) is nested in p_1 or hchild(p) precedes p_1. The case where hchild(p) is nested in p_1 is handled in line IV of the recurrence. In this case we have that either $T(p_1) = p$ or p is nested in p_1. In both cases we have that $i_0 \leqslant p^{\mathrm{L}} + d - 1$ (due to Lemma 3). From this inequality we obtain that $(i_0 - 1) - i = (i_0 - p^{\mathrm{L}}) + (p^{\mathrm{L}} - i) - 1 \leqslant 2d - 2$, so the entry $M[i, i_0 - 1, j, j_0 - 1]$ exists. Moreover, from the inequality $i_0 \leqslant p^{\mathrm{L}} + d - 1$ and the assumption that hchild(p) is nested in p_1 we obtain that the entry $R^p[i_0 + 1, i' - 1, j_0 + 1, j' - 1]$ exists.

Now consider the case where hchild(p) precedes p_1 which is handled in lines V and VI of the recurrence. In both lines, the common entry $R^p[i, i_0 - 1, j, j_0 - 1]$ exists.

If span$(p_1) \leqslant 2d$ then the entry $M[i_0 + 1, i' - 1, j_0 + 1, j' - 1]$ exists since $(i' - 1) - (i_0 + 1) = \mathrm{span}(p_1) - 3 \leqslant 2d - 3$. If span$(p_1) > 2d$ then we need to show that the entry $R^{T(p_1)}[i_0, i', j_0, j']$ exists. We have that $p_1^{\mathrm{L}} - p^{\mathrm{L}} > \mathrm{span}(\mathrm{hchild}(p)) > 2d$, and therefore p_1 does not cross p and $p_1 \notin I_d(p)$. It follows that $T(p_1) \neq p$. Therefore, $T(p_1)$ is nested in p, so the table $R^{T(p_1)}$ was already filled by the algorithm when the table R^p is filled. From Lemma 3 and Lemma 4 we conclude that the entry $R^{T(p_1)}[i_0, i', j_0, j']$ exists. The correctness arguments for the recurrence for L^p are analogous.

4.4. Time complexity

Let $d_k^R(i)$ (resp., $d_k^L(i)$) denote the number of arcs p in P_k with $p^{\mathrm{R}} = i$ (resp., $p^{\mathrm{L}} = i$) plus one. In stage 1, the time complexity for computing an entry $M[i, i', j, j']$ is $O(d_1^R(i')d_2^R(j'))$. For fixed i' and j', the number of entries of the form $M[i, i', j, j']$ that are computed by the algorithm is $O(dm)$. Therefore, the time complexity of stage 1 is

$$O\left(\sum_{i'=1}^{n}\sum_{j'=1}^{m} dm \cdot d_1^R(i')d_2^R(j')\right) = O\left(dm \sum_{i'=1}^{n} d_1^R(i') \sum_{j'=1}^{m} d_2^R(j')\right) = O(dnm^2).$$

For $p \in P_T$, the time complexity of computing an entry $L^p[i, i', j, j']$ is $O(d_1^L(i)d_2^L(j))$, and the time complexity of computing an entry $R^p[i, i', j, j']$ is $O(d_1^R(i')d_2^R(j'))$. Let $c_{i,j}^p$ denote the number of computed entries of the form $L^p[i, i', j, j']$ or $R^p[i', i, j', j]$. Then stage 2 requires $O(\sum_{p \in P_T} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j}^p \cdot (d_1^L(i)d_2^L(j) + d_1^R(i)d_2^R(j)))$ time.

For every $p \in P_T$ and every i and j, $c_{i,j}^p \in O(dm)$. Assuming i and j are fixed, we now count the number of arcs $p \in P_T$ for which $c_{i,j}^p > 0$. Let p_0 be the minimum span tree arc such that $i \in [p_0^{\mathrm{L}}, p_0^{\mathrm{R}}]$ (p_0 exists as $(1, n)$ is a tree arc).

Lemma 6. *Every tree arc p with $c_{i,j}^p > 0$ satisfies one of the following*:

1. *$p^{\mathrm{R}} < i$ and p^{R} is maximal among all tree arcs whose right ends are smaller than i.*
2. *$p^{\mathrm{L}} > i$ and p^{L} is minimal among all tree arcs whose left ends are bigger than i.*
3. *p_0 is nested in p and $i \notin [hchild(p)^{\mathrm{L}}, hchild(p)^{\mathrm{R}}]$.*

Proof. Suppose conversely that there is a tree arc p_1 with $c_{i,j}^{p_1} > 0$ that does not satisfy properties 1–3 above. If hchild(p_1) is defined, then by definition, $c_{i',j}^{p_1} = 0$ for every $i' \in [\mathrm{hchild}(p_1)^{\mathrm{L}}, \mathrm{hchild}(p_1)^{\mathrm{R}}]$, and therefore $i \notin [\mathrm{hchild}(p_1)^{\mathrm{L}}, \mathrm{hchild}(p_1)^{\mathrm{R}}]$. From the assumption that p_1 does not satisfy property 3, it follows that p_0 is not nested in p. Therefore, by the definition of p_0, $i \notin [p_1^{\mathrm{L}}, p_1^{\mathrm{R}}]$. Thus, either $p_1^{\mathrm{R}} < i$ or $p_1^{\mathrm{L}} > i$.

Without loss of generality, assume that $p_1^{\mathrm{R}} < i$. Let p be the arc such that $p^{\mathrm{R}} < i$ and p^{R} is maximal among all tree arcs whose right ends are smaller than i. By definition, $c_{i',j}^{p_1} = 0$ for every $i' > \min(p_1^{\mathrm{R}} + d, \mathrm{parent}(p_1)^{\mathrm{R}})$, so $i \leqslant \min(p_1^{\mathrm{R}} + d, \mathrm{parent}(p_1)^{\mathrm{R}})$. From the maximality of p we have that either p_1 is nested in p, or p_1 precedes p. In the former case we obtain a contradiction as $i > p^{\mathrm{R}} \geqslant \mathrm{parent}(p_1)^{\mathrm{R}}$. In the latter case, we obtain a contradiction as $i > p^{\mathrm{R}} \geqslant p_1^{\mathrm{R}} + \mathrm{span}(p) \geqslant p_1^{\mathrm{R}} + 2d + 1$. □

There are at most two arcs of types 1 and 2 above. Let $p_0, p_1, \ldots, p_k$ be all the tree arcs of the third type, such that p_i is nested in p_{i+1} for all i. Since $\mathrm{span}(p_i) \leqslant \mathrm{span}(\mathrm{hchild}(p_{i+1}))$, we have that $\mathrm{span}(p_{i+1}) > 2 \cdot \mathrm{span}(p_i)$ for all i and therefore $k < \log_2 n$. Thus, the time complexity of stage 2 is $O(\sum_{i=1}^{n} \sum_{j=1}^{m} dm \log n \cdot (d_1^L(i)d_2^L(j) + d_1^R(i)d_2^R(j))) = O(dm^2 n \log n)$.

4.5. Space complexity

We now show how to implement the algorithm in $O(nm + dm^2)$ space. The main idea is to split the tables M, $\{L^p\}_{p \in P_T}$ and $\{R^p\}_{p \in P_T}$ into smaller tables, and keep only one of these tables in memory at each point during the run of the algorithm. More precisely, we define tables $M_{i,j}$, where $M_{i,j}[i', j']$ stores the value of OPT(i, i', j, j'). The range of the indices i, i', j, j' is the same as in the definition of M. For every tree arc p for which hchild(p) is defined:

- We define tables $L_{i',j'}^p$, where $L_{i',j'}^p[i, j]$ stores the value of OPT(i, i', j, j'). The range of the indices i', j, j' is the same as in the definition of L^p. The range of i is

$$\max(p^{\mathrm{L}} - d, \mathrm{parent}(p)^{\mathrm{L}}) \leqslant i \leqslant \max(\mathrm{hchild}(p)^{\mathrm{L}} - d, p^{\mathrm{L}}).$$

$$M_{i,j}[i',j'] = \min \begin{cases} M_{i,j}[i'-1,j'] + \gamma_1(i') & \text{I} \\ M_{i,j}[i',j'-1] + \gamma_2(j') & \text{II} \\ M_{i,j}[i'-1,j'-1] + \beta(i',j') & \text{III} \\ \text{for all } p_1 = (i_0,i') \in P_1,\ p_2 = (j_0,j') \in P_2 \text{ with } i \leqslant i_0,\ j \leqslant j_0 & \text{IV} \\ \quad M_{i,j}[i_0-1,j_0-1] + A[p_1,p_2] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 7. Recurrence for the table $M_{i,j}$.

- We define tables $R^p_{i,j}$, where $R^p_{i,j}[i',j']$ stores the value of $\mathrm{OPT}(i,i',j,j')$. The range of the indices i, j, j' is the same as in the definition of R^p. The range of i' is

$$\min\bigl(\mathrm{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}}\bigr) \leqslant i' \leqslant \min\bigl(p^{\mathrm{R}} + d, \mathrm{parent}(p)^{\mathrm{R}}\bigr).$$

For a tree arc p for which $\mathrm{hchild}(p)$ is not defined, we define tables $R^p_{i,j}$, where $R^p_{i,j}[i',j']$ stores the value of $\mathrm{OPT}(i,i',j,j')$. The range of i,i',j,j' is the same as in the definition of R^p.

Since the algorithm discards the tables $M_{i,j}$, $L^p_{i',j'}$, and $R^p_{i,j}$ after it computes these tables, it needs to store some values from these table for later use. For this purpose, the algorithm keeps tables A, B^p_L and B^p_R that are defined below. The table B^p_L (resp., B^p_R) is used to initialize the $L^p_{i',j'}$ (resp., $R^p_{i,j}$) tables, and the A table is used in the recurrences of the $L^p_{i',j'}$ and $R^p_{i,j}$ tables. Moreover, some values from the $M_{i,j}$ tables will be recomputed when needed. This is done using the $M^p_{L,i',j'}$ and $M^p_{R,i,j}$ tables defined below. The additional tables are defined as follows. $A[p_1,p_2]$ stores the value of $\mathrm{OPT}(p_1^{\mathrm{L}}+1, p_1^{\mathrm{R}}-1, p_2^{\mathrm{L}}+1, p_2^{\mathrm{R}}-1)$ for every $p_1 \in P_1$ and $p_2 \in P_2$. For every tree arc p for which $\mathrm{hchild}(p)$ is defined, the following tables are used by the algorithm:

- $B^p_L[i',j,j']$ stores the value of $\mathrm{OPT}(\max(\mathrm{hchild}(p)^{\mathrm{L}} - d, p^{\mathrm{L}}), i', j, j')$. The range of the indices i', j, j' is the same as in the definition of L^p.
- $B^p_R[i,j,j']$ stores the value of $\mathrm{OPT}(i, \min(\mathrm{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}}), j, j')$. The range of the indices i, j, j' is the same as in the definition of R^p.
- $M^p_{L,i',j'}[i,j]$ stores the value of $\mathrm{OPT}(i,i',j,j')$. The range of the indices i', j, j' is the same as in the definition of L^p. The range of the index i is $\mathrm{hchild}(p)^{\mathrm{R}} + 1 \leqslant i \leqslant i' + 1$.
- $M^p_{R,i,j}[i',j']$ stores the value of $\mathrm{OPT}(i,i',j,j')$. The range of the indices i, j, j' is the same as in the definition of R^p. The range of the index i' is $i - 1 \leqslant i' \leqslant \min(p^{\mathrm{L}} + d, \mathrm{hchild}(p)^{\mathrm{L}} - 1)$.

At any step of the algorithm, the algorithm keeps only a constant number of the tables defined above. Since the size of every table is either $O(nm)$ or $O(dm^2)$, it follows that the space complexity is $O(nm + dm^2)$.

4.5.1. Stage 1

The algorithm first computes the $M_{i,j}$ tables. The order of computing these tables is arbitrary. Fix some i and j. The table $M_{i,j}$ is initialized with $M_{i,j}[i-1,j-1] = 0$, and the other entries are computed using the recurrence of Fig. 7 (which is straightforward adaptation of the recurrence of Fig. 2). During the computation of $M_{i,j}$, the algorithm copies values corresponding to the same subproblems from $M_{i,j}$ to A, namely, after computing an entry $M_{i,j}[i',j']$, if $(i-1,i'+1) \in P_1$ and $(j-1,j'+1) \in P_2$, the entry $M_{i,j}[i',j']$ is copied into $A[(i-1,i'+1),(j-1,j'+1)]$. After all the values of $M_{i,j}$ are computed, the table $M_{i,j}$ is discarded from memory. The table A is the only table kept in memory when stage 1 finishes.

4.5.2. Stage 2

In the second stage the algorithm computes the $L^p_{i',j'}$ and $R^p_{i,j}$ tables. Define the *heavy path* of a set of arcs $P \subseteq P_T$ as the sequence of arcs $p_1, p_2, \ldots, p_k$ that satisfies (1) p_1 be the maximum span arc in P, (2) $p_{i+1} = \mathrm{hchild}(p_i)$ for all i, and (3) $\mathrm{hchild}(p_k)$ is not defined. The computation order of the $L^p_{i',j'}$ and $R^p_{i,j}$ tables is defined recursively in Fig. 8.

Fix some tree arc p and suppose that $\mathrm{hchild}(p)$ is defined. Fix i' and j'. Before the algorithm computes the table $L^p_{i',j'}$, it computes the table $M^p_{L,i',j'}$ using the recurrence of Fig. 9. The table $L^p_{i',j'}$ is initialized with $L^p_{i',j'}[\max(\mathrm{hchild}(p)^{\mathrm{L}} - d, p^{\mathrm{L}}), j] = B^p_L[i',j,j']$ for all j. The other entries are computed using the recurrence of Fig. 10. The algorithm copies values from $L^p_{i',j'}$ into the tables A and B^p_R as follows: If $(i-1,i'+1) \in P_1$ and $(j-1,j'+1) \in P_2$, the entry $L^p_{i',j'}[i,j]$ is copied into $A[(i-1,i'+1),(j-1,j'+1)]$. Moreover, if $i' = \min(\mathrm{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}})$, $L^p_{i',j'}[i,j]$ is copied into $B^p_R[i,j,j']$. After all the values of $L^p_{i',j'}$ are computed, the tables $L^p_{i',j'}$ and $M^p_{L,i',j'}$ are discarded from memory. After the tables $L^p_{i',j'}$ are computed for all i' and j', the table B^p_L is discarded from memory.

The computation of an $R^p_{i,j}$ is done as follows. First, the table $M^p_{R,i,j}$ is computed using the recurrence of Fig. 11. Then, the table $R^p_{i,j}$ is initialized with $R^p_{i,j}[\min(\mathrm{hchild}(p)^{\mathrm{R}} + d, p^{\mathrm{R}}), j'] = B^p_R[i,j,j']$ for all j'. The other entries are computed using

```
1: Let $p_1, \dots, p_k$ be the heavy path of $P$.
2: for $p = p_1, p_2, \dots, p_{k-1}$ do
3:     Compute the tables for the arcs in $P \cap [p^{\mathrm{L}}+1, \mathrm{hchild}(p)^{\mathrm{L}}-1]^2$.
4:     Compute the tables for the arcs in $P \cap [\mathrm{hchild}(p)^{\mathrm{R}}+1, p^{\mathrm{R}}-1]^2$.
5: for $i = \max(p_k^{\mathrm{L}}+d, \mathrm{parent}(p_k)^{\mathrm{L}}), \dots, p_k^{\mathrm{L}}-d$ do
6:     for $j = m, \dots, 0$ do
7:         Compute $R_{i,j}^{p_k}$
8: for $p = p_{k-1}, p_{k-2}, \dots, p_1$ do
9:     for $i' = \mathrm{hchild}(p)^{\mathrm{R}}, \dots, \min(\mathrm{hchild}(p)^{\mathrm{R}}+d, p^{\mathrm{R}})$ do
10:        for $j' = 0, \dots, m$ do
11:            Compute $L_{i',j'}^{p}$
12:    for $i = \max(p^{\mathrm{L}}+d, \mathrm{parent}(p)^{\mathrm{L}}), \dots, \min(p^{\mathrm{L}}-d, \mathrm{hchild}(p)^{\mathrm{L}})$ do
13:        for $j = m, \dots, 0$ do
14:            Compute $R_{i,j}^{p}$
```

Fig. 8. Computation order of the tables $L_{i',j'}^p$ and $R_{i,j}^p$ for all $p \in P$.

$$M_{L,i',j'}^p[i,j] = \min \begin{cases} M_{L,i',j'}^p[i+1,j] + \gamma_1(i) & \text{I} \\ M_{L,i',j'}^p[i,j+1] + \gamma_2(j) & \text{II} \\ M_{L,i',j'}^p[i+1,j+1] + \beta(i,j) & \text{III} \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j' & \text{IV} \\ \quad M_{L,i',j'}^p[i_0+1, j_0+1] + A[p_1,p_2] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 9. Recurrence for the table $M_{L,i',j'}^p$.

$$L_{i',j'}^p[i,j] = \min \begin{cases} L_{i',j'}^p[i+1,j] + \gamma_1(i) & \text{I} \\ L_{i',j'}^p[i,j+1] + \gamma_2(j) & \text{II} \\ L_{i',j'}^p[i+1,j+1] + \beta(i,j) & \text{III} \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j', & \\ \text{and hchild}(p) \text{ is nested in } p_1 & \text{IV} \\ \quad A[p_1,p_2] + M_{L,i',j'}^p[i_0+1, j_0+1] + \alpha(p_1,p_2) & \\ \text{for all } p_1 = (i,i_0) \in P_1,\ p_2 = (j,j_0) \in P_2 \text{ with } i_0 \leqslant i',\ j_0 \leqslant j', & \\ \text{and hchild}(p) \text{ is not nested in } p_1 & \text{V} \\ \quad A[p_1,p_2] + L_{i',j'}^p[i_0+1, j_0+1] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 10. Recurrence for the table $L_{i',j'}^p$.

$$M_{R,i,j}^p[i',j'] = \min \begin{cases} M_{R,i,j}^p[i'-1,j'] + \gamma_1(i') & \text{I} \\ M_{R,i,j}^p[i',j'-1] + \gamma_2(j') & \text{II} \\ M_{R,i,j}^p[i'-1,j'-1] + \beta(i',j') & \text{III} \\ \text{for all } p_1 = (i_0,i') \in P_1,\ p_2 = (j_0,j') \in P_2 \text{ with } i \leqslant i_0,\ j \leqslant j_0 & \text{IV} \\ \quad M_{R,i,j}^p[i_0-1, j_0-1] + A[p_1,p_2] + \alpha(p_1,p_2) & \end{cases}$$

Fig. 11. Recurrence for the table $M_{R,i,j}^p$.

the recurrence of Fig. 12. If $(i-1, i'+1) \in P_1$ and $(j-1, j'+1) \in P_2$, the entry $R_{i,j}^p[i',j']$ is copied into $A[(i-1,i'+1),(j-1,j'+1)]$. Moreover, if $i = \max(p^{\mathrm{L}}-d, \mathrm{parent}(p)^{\mathrm{L}})$, $R_{i,j}^p[i',j']$ is copied into $B_L^{\mathrm{parent}(p)}[i',j,j']$. Finally, after all the values of $R_{i,j}^p$ are computed, the tables $R_{i,j}^p$ and $M_{R,i,j}^p$ are discarded from memory. Moreover, after the tables $R_{i,j}^p$ are computed for all i and j, the table B_R^p is discarded from memory.

We now analyze the time complexity of the new algorithm. Stage 1 of the algorithm is equivalent to stage 1 of the previous algorithm. That is, the total number of cells in all $M_{i,j}$ tables is the same as the number of cells in the M table, and the computation of a cell $M_{i,j}[i',j']$ has the same time complexity as the computation of $M[i,i',j,j']$. Therefore, the time complexity of stage 1 is $O(dnm^2)$. In stage 2, the computation of all $L_{i',j'}^p$ and $R_{i,j}^p$ tables is equivalent to the computation of all L^p and R^p tables in the previous algorithm, and thus this computation takes $O(dm^2 n \log n)$ time. The computation of all $M_{L,i',j'}^p$ and $M_{R,i,j}^p$ tables takes $O(dnm^2)$ time. It follows that the time complexity of the algorithm is $O(dm^2 n \log n)$.

$$R^p_{i,j}[i, i', j, j'] = \min \begin{cases} R^p_{i,j}[i'-1, j'] + \gamma_1(i') & \text{I} \\ R^p_{i,j}[i', j'-1] + \gamma_2(j') & \text{II} \\ R^p_{i,j}[i'-1, j'-1] + \beta(i', j') & \text{III} \\ \text{for all } p_1 = (i_0, i') \in P_1,\ p_2 = (j_0, j') \in P_2 \text{ with } i \leqslant i_0,\ j \leqslant j_0, & \\ \text{and hchild}(p) \text{ is nested in } p_1 & \text{IV} \\ \quad M^p_{R,i,j}[i_0 - 1, j_0 - 1] + A[p_1, p_2] + \alpha(p_1, p_2) & \\ \text{for all } p_1 = (i_0, i') \in P_1,\ p_2 = (j_0, j') \in P_2 \text{ with } i \leqslant i_0,\ j \leqslant j_0, & \\ \text{and hchild}(p) \text{ is not nested in } p_1 & \text{V} \\ \quad R^p_{i,j}[i_0 - 1, j_0 - 1] + A[p_1, p_2] + \alpha(p_1, p_2) & \end{cases}$$

Fig. 12. Recurrence for the table $R^p_{i,j}$.

5. Conclusion

We presented an algorithm that computes the optimal sequence structure alignment for a nested consensus structure and crossing input structures. In practice, crossing input structures can be used to represent several suboptimal structures simultaneously, from which the alignment effectively selects the most appropriate one. On the theoretical side, we generalized the optimizations developed by Klein [10] to crossing input structures.

References

[1] T. Akutsu, Approximation and exact algorithms for RNA secondary structure prediction and recognition of stochastic context-free languages, J. of Combinatorial Optimization 3 (1999) 321–336.

[2] A.F.B. Consortium, R. Backofen, S.H. Bernhart, C. Flamm, C. Fried, G. Fritzsch, J. Hackermuller, J. Hertel, I.L. Hofacker, K. Missal, A. Mosig, S.J. Prohaska, D. Rose, P.F. Stadler, A. Tanzer, S. Washietl, S. Will, RNAs everywhere: genome-wide annotation of structured RNAs, J. of Experimental Zoology Part B: Molecular and Developmental Evolution 308 (1) (2007) 1–25.

[3] E.D. Demaine, S. Mozes, B. Rossman, O. Weimann, An optimal decomposition algorithm for tree edit distance, in: Proc. 34th International Colloquium on Automata, Languages and Programming (ICALP), 2007, pp. 146–157.

[4] C.B. Do, D.A. Woods, S. Batzoglou, CONTRAfold: RNA secondary structure prediction without physics-based models, Bioinformatics 22 (14) (2006) e90–e98.

[5] P.A. Evans, Finding common RNA pseudoknot structures in polynomial time, in: Proc. 17th Symposium on Combinatorial Pattern Matching (CPM), 2006, pp. 223–232.

[6] P.A. Evans, Algorithms and complexity for annotated sequence analysis, PhD thesis, University of Alberta, 1999.

[7] P.P. Gardner, R. Giegerich, A comprehensive comparison of comparative RNA structure prediction approaches, BMC Bioinformatics 5 (2004) 140.

[8] J.H. Havgaard, R.B. Lyngso, G.D. Stormo, J. Gorodkin, Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%, Bioinformatics 21 (9) (2005) 1815–1824.

[9] T. Jiang, G. Lin, B. Ma, K. Zhang, A general edit distance between RNA structures, J. of Computational Biology 9 (2) (2002) 371–388.

[10] P. Klein, Computing the edit-distance between unrooted ordered trees, in: Proc. 6th European Symposium on Algorithms (ESA), 1998, pp. 91–102.

[11] D.H. Mathews, D.H. Turner, Dynalign: an algorithm for finding the secondary structure common to two RNA sequences, J. of Molecular Biology 317 (2) (2002) 191–203.

[12] R. Nussinov, A. Jacobson, Fast algorithm for predicting the secondary structure of single-stranded RNA, Proc. National Academy of Science 77 (11) (1980) 6309–6313.

[13] D. Sankoff, Simultaneous solution of the RNA folding, alignment and protosequence problems, SIAM J. on Applied Mathematics 45 (5) (1985) 810–825.

[14] S. Washietl, I.L. Hofacker, M. Lukasser, A. Huttenhofer, P.F. Stadler, Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome, Nature Biotechnology 23 (11) (2005) 1383–1390.

[15] M.S. Waterman, T.F. Smith, RNA secondary structure: A complete mathematical analysis, Mathematical Biosciences 42 (1978) 257–266.

[16] Y. Wexler, C. Zilberstein, M. Ziv-Ukelson, A study of accessible motifs and RNA folding complexity, J. of Computational Biology 14 (6) (2007) 856–872.

[17] S. Will, K. Reiche, I.L. Hofacker, P.F. Stadler, R. Backofen, Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering, PLOS Computational Biology 3 (4) (2007) e65.

[18] M. Ziv-Ukelson, I. Gat-Viks, Y. Wexler, R. Shamir, A faster algorithm for RNA co-folding, in: Proc. 8th Workshop on Algorithms in Bioinformatics (WABI), 2008, pp. 174–185.

[19] M. Zuker, P. Stiegler, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucleic Acids Research 9 (1) (1981) 133–148.