BIOINFORMATICS

Vol. 00 no. 00 2008
Pages 1–8

Local RNA structure alignment with incomplete sequence

Diana L. Kolbe [1] and Sean R. Eddy [1*]

[1]HHMI Janelia Farm Research Campus, Ashburn VA 20147, USA

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Accuracy of automated structural RNA alignment is improved by using models that consider not only primary sequence but also secondary structure information. However, current RNA structural alignment approaches tend to perform poorly on incomplete sequence fragments, such as single reads from metagenomic environmental surveys, because nucleotides that are expected to be base paired are missing.

Results: We present a local RNA structural alignment algorithm, trCYK, for aligning and scoring incomplete sequences under a model using primary sequence conservation and secondary structure information when possible. The trCYK algorithm improves alignment accuracy and coverage of sequence fragments of structural RNAs in simulated metagenomic shotgun datasets.

Availability: The source code for Infernal 1.0, which includes trCYK, is available at http://infernal.janelia.org

Contact: {kolbed,eddys}@janelia.hhmi.org

1 INTRODUCTION

Sequence alignment approaches may be broadly divided into *global alignment* methods, where sequences are assumed to be homologous and alignable over their entire lengths, and *local alignment* methods, where only part of each sequence is assumed to be homologous and alignable (Gusfield, 1997; Durbin *et al.*, 1998). Local alignment is more widely used because there are many biological and technical reasons why sequences may not be globally alignable. For example, many protein sequences have arisen by accretion of common protein domains in different combinations (Vogel *et al.*, 2004), and some high-throughput sequencing strategies such as metagenomic shotgun sampling generate fragmentary sequence data (Schloss and Handelsman, 2005).

For local alignment of primary sequences (Smith and Waterman, 1981; Pearson and Lipman, 1988; Altschul *et al.*, 1990), one is merely looking for an alignment of contiguous linear subsequences of a query and a target. At this level there is little informative difference between local alignments that arise by biological evolution versus incomplete data. However, the nature of local alignment can be markedly different when we adopt more realistic and complex sequence alignment models that capture evolutionary constraints at a higher level than primary sequence alone. For example, in comparing three-dimensional protein structures, which often share structural similarity in only part of their overall fold (Chothia and Lesk, 1986), it is advantageous to adopt local structural alignment algorithms that allow alignment of spatially local units of three-dimensional structure that may not be composed of contiguous colinear residues in the primary sequence (Gibrat *et al.*, 1996).

Here we are concerned with alignment of structural RNAs, using models that consider both primary sequence and secondary structure constraints. Evolution of a structural RNA is constrained by its secondary structure. Base pairing tends to be conserved even as the sequence changes, and aligned sequences exhibit correlated substitutions in which base pairs are substituted by compensatory base pairs. Computational methods for aligning structural RNAs under a combined primary sequence and secondary structure scoring model have been developed (Sakakibara *et al.*, 1994; Backofen and Will, 2004) including "covariance model" (profile stochastic context-free grammar) methods (Eddy and Durbin, 1994; Durbin *et al.*, 1998; Eddy, 2002; Nawrocki and Eddy, 2007). These models represent a given RNA consensus secondary structure as a binary tree, with individual nodes representing and scoring individual base pairs and single-stranded residues.

Local RNA secondary structural alignment has been implemented by allowing an alignment to start or end at any internal node in the tree (Eddy, 2002; Klein and Eddy, 2003; Backofen and Will, 2004), much as local primary sequence alignment allows starting and ending at any residue in the linear sequence. This *subtree method* of local RNA alignment can include or exclude any subtree of the RNA, corresponding well to secondary structure domains. Biologically, this serves as a reasonable approximation of some important evolutionary constraints on RNA secondary structure alignment, although it neglects higher-order constraints, including pseudoknots and tertiary structure.

However, defining locality by subtrees is a poor model of local structural RNA alignment when locality arises for technical rather than biological causes. A shotgun sequencing strategy will truncate at the linear sequence level without respect for the conserved base-paired structure, and residues involved in base pairs may be missing in the observed sequence, as illustrated in Figure 1. In this case, we do not want to score the missing residues as deletions of conserved base pairs, but neither do we want to leave the homologous observed residues unaligned if we are trying to get the most information from fragmentary sequence data.

Here we will focus specifically on the *homology search and alignment* problem. We have a given RNA sequence and secondary structure as a query, and the task is to search a sequence database for homologous sequences and/or align target sequences to the query. This is directly analogous to the use of the Smith/Waterman local alignment algorithm for primary sequence analysis (Smith and Waterman, 1981), and it is the problem addressed by our Infernal

*to whom correspondence should be addressed

software package (Nawrocki *et al.*, 2009), for instance using Rfam models of known RNA families to infer and annotate homologous RNAs in genome sequence (Gardner *et al.*, 2009). It should not be confused, for example, with the related problem of *de novo motif identification*, which arises in RNA analysis when the input data consist of two or more sequences that are presumed to share an *unknown* structural motif in common, and the task is to produce a local structural alignment that identifies the common motif and infers its common structure. De novo motif identification requires a means of inferring the unknown structural consensus in addition to a local alignment algorithm. Although we expect that *de novo* motif identification approaches such as CMFinder (Yao *et al.*, 2006) or other approaches for inferring locally conserved RNA structure such as LocARNA (Will *et al.*, 2007) would be able to incorporate the local alignment algorithm we will describe, for the purposes of introducing our local alignment algorithm in the present paper, we will not discuss the *de novo* motif identification problem further.

An important example of the local RNA alignment problem in homology search and alignment arises in metagenomic shotgun survey sequencing (Schloss and Handelsman, 2005; Chen and Pachter, 2005), particularly when assembly is incomplete or not possible. Structural RNA sequence alignments (particularly of small subunit ribosomal RNA) are important in analyzing the phylogenetic diversity of metagenomic samples, but a single shotgun read (often of only about 200-400 bp) will fall more or less randomly into the consensus alignment of an RNA, generally leaving unsatisfied consensus base pairs because of the incomplete nature of the sampled sequences, and it may also include extraneous genomic sequence.

To deal with this *truncated sequence* type of locality we want to align the observed sequence, or a subsequence of it, to a contiguous subsequence of the yield of the model's tree: the linear consensus sequence, as read counterclockwise around the tree's leaves. We want to use secondary structure information wherever we have both residues in a base pair, but revert to primary sequence alignment when we are missing sequence data. If we magically knew *a priori* the endpoints of the correct alignment of an observed sequence read with respect to the yield of the RNA model, we could derive a new model that used base pair states where we had both residues, and converted pair states to appropriately marginalized single-residue states where the pairing partner was missing. The problem is that these endpoints must be inferred when we align the observed sequence to the model. We describe an optimal recursive dynamic programming solution for this problem, and evaluate the algorithm's utility in accurate alignment of simulated datasets of unassembled metagenomic sequence.

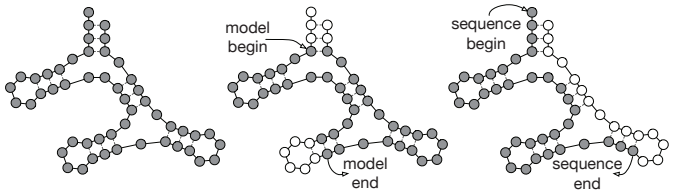

Fig. 1. Comparison of local alignment types. Left: global alignment; filled circles indicate observed residues in an RNA structure, which can be thought of as a binary tree. **Center:** subtree method of local RNA structural alignment. Whole domains of the RNA structure may be skipped (open circles indicate consensus positions without aligned sequence residues), but the observed alignment satisfies all expected structural constraints: if a residue is aligned to a pair state, another residue will be aligned to form a base pair. **Right:** truncated sequence method of local RNA structural alignment, where the observed sequence may begin and end anywhere with respect to the consensus RNA structure. Aligned residues may be base-paired to positions that are missing from the alignment.

2 APPROACH

2.1 Local alignment as a missing data problem

We frame the alignment of truncated sequences as a missing data inference problem (Rubin, 1976). We specify two probabilistic processes: one that generates complete data (our existing probabilistic model of global alignment), and one that generates observed fragments from the complete data (by random sequence truncation). The joint probability of observed sequence fragments and their local alignment to the model will then be an appropriate marginal sum over global alignments. We will identify the optimal local alignment for the observed sequence by maximizing this joint probability.

We will describe the essence of the approach (and two approximations we make) in general terms with respect to binary trees, deferring the specific notation we use for profile stochastic context-free grammars (covariance models, CMs). In a CM, both the consensus structure of the model and a particular structural alignment of the model to an individual RNA sequence are binary trees. (A binary tree suffices to capture all nested base-pairing correlations, but non-nested interactions such as pseudoknots and higher order interactions such as base triples are neglected (Durbin *et al.*, 1998).) Construction of a CM starts by representing the RNA consensus structure as a **guide tree**, with **nodes** representing consensus base pairs and consensus unpaired positions. Each consensus node is then stereotypically expanded into one or more stochastic context-free grammar (SCFG) **states**, with one state representing the consensus ("match") behavior and additional states and state transitions representing the probability of insertions and deletions relative to consensus. A CM is a special case of SCFGs, with all its states and state transitions arranged in a directed graph following the branching pattern of a consensus RNA structure's binary tree. An alignment of the CM to a particular sequence is represented as an SCFG *parse tree*, a state path through the consensus guide tree, using match, insert, and delete states to account for alignment positions, and start, bifurcation, and end states to account for the branching tree structure itself.

A parameterized RNA CM θ specifies a joint probability distribution $P(\hat{\mathbf{x}}, \hat{\pi} \mid \theta)$ over *complete* sequences $\hat{\mathbf{x}}$ and parse trees $\hat{\pi}$: i.e. over *global* alignments.

A missing data process $P(\mathbf{x}, \pi \mid \hat{\mathbf{x}}, \hat{\pi}, \theta)$ specifies how a complete sequence $\hat{\mathbf{x}}$ with length $\hat{\mathbf{L}}$ is truncated to an observed sequence fragment $\mathbf{x}$ of length $\mathbf{L}$, and correspondingly, how the global parse tree $\hat{\pi}$ is truncated to a notion of a local parse tree π. (We will solidify our definition of a local parse tree shortly.) Because we are imagining a complete sequence randomly truncated to a sequence fragment, the missing data process would ideally be conditionally independent of the model and the parse tree. For instance, we could sample each possible sequence fragment from a complete sequence of length $\hat{L}$ with uniform probability $\frac{2}{\hat{L}(\hat{L}+1)}$. However, under this missing data process, we would need to marginalize (sum over) all possible complete sequences of all possible lengths $\hat{L} = 0 \ldots \infty$.

This $\frac{2}{\hat{L}(\hat{L}+1)}$ term becomes problematic in the recursive dynamic programming optimization framework we describe below.

Instead we will make what should be a reasonable approximation of the truncation process. We assume that a truncation Δ^{gh} is done by selecting a fragment $g \ldots h$ relative to the positions in the fixed-length *consensus yield* as defined by the *model* (the consensus sequence positions defined by the CM's consensus guide tree nodes). This truncation is then conditionally independent of both the parse tree and the sequence. This approximation should be reasonable because high-probability complete sequences $\hat{x}$ will generally have lengths similar to the consensus length. It means that local alignments will only begin and end at consensus positions, never at sequence insertions. For a model with W consensus positions, the probability of choosing any particular fragment $g \ldots h$ with respect to the complete yield $1..W$ is $P(\Delta^{gh} \mid \theta) = \frac{2}{W(W+1)}$. This term is now a constant with respect to the necessary summation over complete data.

Now we define what we mean by a local parse tree fragment π. Choose two positions g, h on the consensus yield of the model: these consensus sequence positions correspond unambiguously to states s_g and s_h in parse trees (the states used by the parse tree to account for how the endpoints of a particular sequence align to a model consensus position: either a consensus match, or a deletion). A **local parse tree** π^{gh} (equivalent to what we have called just π until now) is defined as the minimal (smallest) subtree of a complete parse tree $\hat{\pi}$ that contains s_g and s_h. Usually this is a parse subtree rooted at either s_g or s_h, but s_g and s_h may also be on opposing sides of a bifurcation, with the minimal subtree rooted at the bifurcation state.

Truncation of a complete parse tree $\hat{\pi}$ to a local parse tree π^{gh} defines two different sorts of missing data. Outside the local parse tree, we are missing (and will sum over) both sequence residues and parse tree states that were in the complete parse tree; let this missing data be represented by $\mathbf{x}', \pi'$. Inside the local parse tree, we may have states with unsatisfied, missing sequence residues, such as base pairs where only one residue is in the observed sequence: here we will be summing only over the missing sequence residues, denoted as $\mathbf{x}''$. The combination of the observed data $(\mathbf{x}, \pi^{gh})$ and the unobserved data $(\mathbf{x}'', \mathbf{x}', \pi')$ together uniquely determine the complete alignment $\hat{\mathbf{x}}, \hat{\pi}$.

The desired joint probability may then be written as a summation over the two types of missing data defined by a local parse tree:

$$P(\mathbf{x}, \pi^{gh} \mid \theta) = \sum_{\mathbf{x}''} \sum_{\mathbf{x}', \pi'} P(\Delta^{gh} | \hat{\mathbf{x}}, \hat{\pi}, \theta) P(\hat{\mathbf{x}}, \hat{\pi} | \theta).$$

Summation over missing data $\mathbf{x}', \pi'$ results in two terms. The first is a term $P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta)$ that represents the marginal probability that a complete parse tree truncated at g, h has states s_g, s_h assigned to the endpoints of the truncation; this is just the fraction of complete parse trees that contain states s_g and s_h. The second term is $P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta)$ for the local parse tree and its associated sequence residues (both observed and unobserved) conditional on local parse tree endpoints at states s_g, s_h. Thus:

$$\begin{aligned} P(\mathbf{x}, \pi^{gh} \mid \theta) &= \frac{2}{W(W+1)} P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta) \\ &\quad \times \sum_{\mathbf{x}''} P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta) \end{aligned}$$

Although it is straightforward to calculate $P(\pi_{x_1} = s_g, \pi_{x_L} = s_h \mid \theta)$, the term becomes problematic in the dynamic programming recursion we define. One or both of the optimal truncation endpoints s_g, s_h are undetermined until the dynamic programming recursion is complete and a traceback is performed. We therefore make our second approximation here, approximating this term as 1.0 when s_g, s_h are consensus match states and 0.0 when they are not. This corresponds to an assumption that all probability mass flows through the consensus match states at the endpoints g, h, neglecting the probability that an SCFG deletion state could be used at one of these consensus positions. Local alignments will therefore be forced to start and end with consensus match positions (just as in standard Smith/Waterman local sequence alignment). This leaves:

$$\begin{aligned} &P(\mathbf{x}, \pi^{gh} \mid \theta) \simeq \\ &\quad \frac{2}{W(W+1)} \sum_{\mathbf{x}''} P(\mathbf{x}, \mathbf{x}'', \pi^{gh} | \pi_{x_1} = s_g, \pi_{x_L} = s_h, \theta) \end{aligned}$$

In the next section, we show there is an efficient dynamic programming algorithm for finding the parse tree π^{gh} that performs the necessary summation over missing data and maximizes this joint probability for a given observed sequence fragment $\mathbf{x}$.

2.2 Description of the trCYK algorithm

The Cocke-Younger-Kasami (CYK) algorithm is a standard algorithm for calculating the maximum likelihood SCFG parse tree for a given sequence (Kasami, 1965; Younger, 1967; Hopcroft and Ullman, 1979; Durbin *et al.*, 1998). CYK recursively calculates terms $\alpha_v(i, j)$ representing the log probability of the optimal parse subtree rooted at state v that accounts for a subsequence $x_i \ldots x_j$, initializing at the smallest subtrees and subsequences (model end states aligned to subsequences of length 0) and iteratively building larger subtrees accounting for longer subsequences. At termination, the score $\alpha_0(1, L)$ is the log probability of a parse tree rooted at the model's start state 0 accounting for the complete sequence $x_1 \ldots x_L$. The optimal parse tree is then recovered by a traceback of the dynamic programming matrix. When applied specifically to a CM of M consensus nodes and a sequence of length L, the CYK algorithm requires $O(L^2 M)$ memory and $O(L^3 M)$ time (Eddy and Durbin, 1994). A more complex divide-and-conquer variant of the CM CYK algorithm requires $O(L^2 \log M)$ memory (Eddy, 2002).

Previously, we implemented subtree-based local RNA structure alignment by a minor adaptation of the CM's generative model that required no substantive alteration of the CYK algorithm. Specifically, we allowed a start transition from the model's root state 0 to *any* of the consensus states in the model, and we allowed an end transition from any consensus state in the model to a "local end" state (EL) that emits zero or more nonhomologous residues with a self-transition loop. The start transition allows the model to align to any model subtree and not just the complete model, and the end

transition allows it to replace any subtree with a nonhomologous insertion.

The truncated sequence local alignment algorithm we describe here, for finding an optimal local parse tree π^{gh} that accounts for a linear sequence fragment, does require a substantial modification of the CYK algorithm because it needs to identify the optimal endpoints g, h. The two approaches to local alignment are not mutually exclusive. We retain the local end transition to an EL state to model nonhomologous replacement of structural elements inside a local parse subtree.

The key property of local parse trees π^{gh} that enables a recursive CYK-style algorithm can be summarized as "once marginal, always marginal", as illustrated in Figure 2. As the CYK calculation builds larger and larger subtrees – climbing "up" the model – it will usually grow by adding appropriate (v, i, j) triplets that deal with complete (joint) emission of any base paired residues (upper left panel of Figure 2). At some point it may need to decide that the sequence is truncated at either the right or left endpoint of the optimal parse subtree (upper middle and upper right panels of Figure 2, respectively), in which case only the left residue x_i or the right residue x_j (respectively) will be added to the growing parse subtree, and scored as the marginal probability of generating the observed residue at state v summed over all possible identities of the missing residue. We refer to this as **joint mode** versus left and right **marginal modes** for a growing alignment. The switch from joint mode to a marginal mode identifies one of the endpoints (h or g, respectively). Once switched, the alignment must stay in that marginal mode until the root state of the optimal local parse subtree is identified. Left marginal mode alignments may only be extended by aligning left residues x_i (central panel of Figure 2), and right marginal mode alignments may only be extended by aligning right residues x_j (center right panel of Figure 2). In marginal modes, residue emission probabilities involving missing data are the appropriate marginal summation of the state's emission probabilities.

In order to recursively calculate the optimal local parse tree, including these optimal switch points from joint to marginal modes, we extend the CYK algorithm to treat the different modes separately (essentially as an additional layer of hidden state information), and calculate separate matrices for each mode: α^J for the standard case (joint mode), α^L for extension only at the 5' end (emissions are marginalized to the left), and α^R for extension only at the 3' end (emissions are marginalized to the right). Each column in Figure 2 illustrates the main cases that have to be examined : for example, the calculation of $\alpha_v^L(i,j)$ for a base-pair emitting state v would examine each of its transition-connected states y and consider both the possibility of reaching (v, i, j) by extension of a previously calculated left-marginal $\alpha_y^L(i-1,j)$, and the possibility of reaching it by switching from a previously calculated joint $\alpha_y^J(i-1,j)$.

The calculation at bifurcation states requires special consideration, as illustrated in Figure 3. Only combinations of modes for left and right branches that would form a contiguous subsequence $x_i \ldots x_j$ aligned to a local parse tree rooted at bifurcation state v are allowed. Cases in which an entire branch is missing data must also be considered (shown as $\emptyset$ in Figure 3). There is a unique case when both branches of the bifurcation have marginal alignments (bottom of Figure 3), and the resulting join cannot be extended further. For convenience, we call the score of this case α^T, noting that it is only defined when v is a bifurcation state and that because it is a terminal case, it does not have to be stored in the recursion.

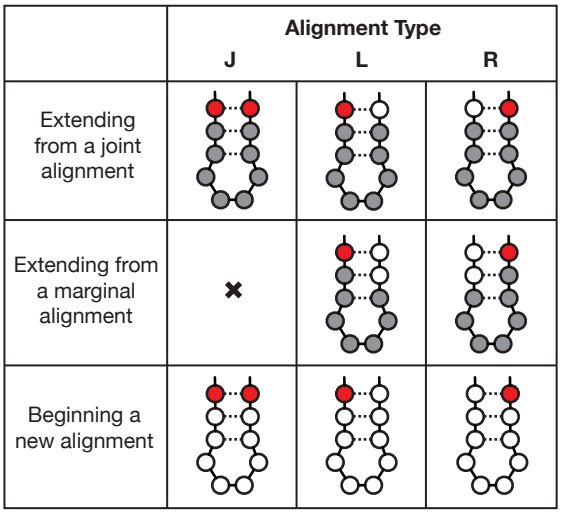

Fig. 2. Extension possibilities for building alignments. Extension may generally be joint (J), left marginal (L), or right marginal (R). **Top:** an existing joint alignment may use any type of extension. Grey circles indicate the previously existing alignment, with the new residues added in red, and open circles for when no residue is aligned. Transitioning from joint to marginal alignment sets an endpoint of the alignment. **Center:** an alignment that is already marginal may only continue with marginal extension on the same side. **Bottom:** a new alignment may be started in any of the three modes. The joint alignment here is shown skipping a portion of the subtree, but that need not be the case. Initiating an alignment in marginal mode also sets one of the alignment endpoints.

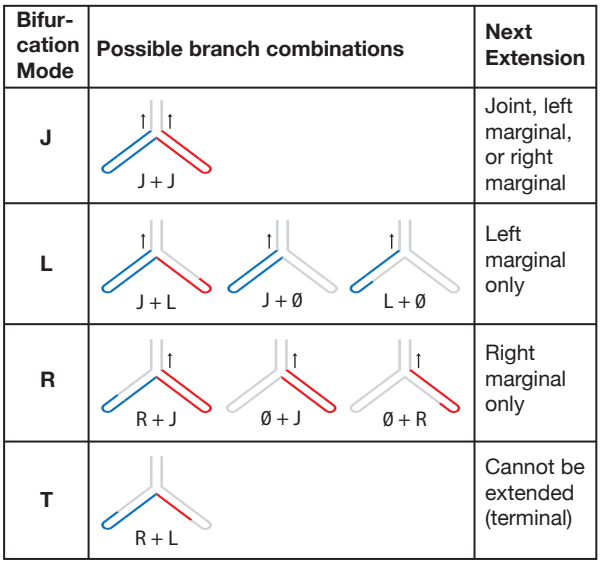

Fig. 3. Extension possibilities at bifurcations. A bifurcation state joins two subtrees, one 5' (left, blue) and one 3' (right, red). Each subtree has its own alignment mode, either joint (J), left marginal (L), right marginal (R), or empty ($\emptyset$). The subtree modes together must give a continuous subsequence, and all valid combinations are shown. The combination determines the mode of the bifurcation state, which can subsequently be extended like any other state, except for the terminal case T. (Arrows show possibilities for later extension.)

The score of the optimal local parse tree aligned to a subsequence $x_i \ldots x_j$ is the combination of consensus state v, sequence positions i, j, and mode $X \in \{J, L, R, T\}$ that maximizes $\alpha_v^X(i,j)$. Alternatively, the entire observed sequence $x_1 \ldots x_L$ can be forced into an optimal local alignment to the model by choosing v, X that maximize $\alpha_v^X(1, L)$.

2.3 The trCYK algorithm

The following description of the truncated sequence CYK dynamic programming algorithm (trCYK) assumes familiarity with notation and conventions used in previously published descriptions of CMs (Durbin *et al.*, 1998; Eddy, 2002). Briefly, sequence positions are indexed by i, j and k; x_i is the residue at position i; and d refers to the length of a subsequence $x_i \ldots x_j$ where $d = j - i + 1$. The main parameters of a CM are the emission and transition probabilities of its states. These states are indexed by v, y, z, ranging from 1 to M, the total number of model states. $\mathcal{C}_v$ lists all the *children* y of state v; the transition probability for moving from v to y is $t_v(y)$. (A bifurcation state v splits to y, z with probability 1.0). $\mathcal{S}_v$ is the *type* of state v; possible values are S (start), P (pair emit), L (left emit), R (right emit), D (deletion), B (bifurcation), and E (end). e_v represents emission probabilities at state v, which (depending on the state type) may emit either one or two residues, $e_v(x_i)$ or $e_v(x_i, x_j)$. Emission probabilities marginalized over a missing residue are indicated by a $*$ for the missing residue; for example $e_v(x_i, *) = \sum_a e_v(x_i, a)$.

Initialization:
best_score $= -\infty$
for $j = 0$ **to** L, $d = 0$ **to** j
  $i = j - d + 1$
  $\alpha_{\mathrm{EL}}^J(i,j) = d * \log t_{\mathrm{EL}}(\mathrm{EL})$
  $\alpha_{\mathrm{EL}}^{\{L,R\}}(i,j) = -\infty$
for $v = \mathrm{M}$ **down to** 1, $j = 0$ **to** L
  if $\mathcal{S}_v = \mathrm{D}$ or S
    $\alpha_v^J(j+1,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(j+1,j) + \log t_v(y)\right]$
    $\alpha_v^{\{L,R\}}(j+1,j) = -\infty$
  else if $\mathcal{S}_v = \mathrm{P}$ or L or R
    $\alpha_v^{\{J,L,R\}}(j+1,j) = -\infty$
  else if $\mathcal{S}_v = \mathrm{B}$
    $\alpha_v^J(j+1,j) = \alpha_y^J(j+1,j) + \alpha_z^J(j+1,j)$
    $\alpha_v^{\{L,R,T\}}(j+1,j) = -\infty$
  else if $\mathcal{S}_v = \mathrm{E}$
    $\alpha_v^{\{J,L,R\}}(j+1,j) = 0$
Recursion:
for $v = \mathrm{M}$ **down to** 1, $j = 1$ **to** L, $d = 1$ **to** j
  $i = j - d + 1$
  if $\mathcal{S}_v = \mathrm{D}$ or S
    $\alpha_v^J(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i,j) + \log t_v(y)\right]$
    $\alpha_v^L(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^L(i,j) + \log t_v(y)\right]$
    $\alpha_v^R(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^R(i,j) + \log t_v(y)\right]$
  else if $\mathcal{S}_v = \mathrm{P}$
    if $d \geq 2$
      $\alpha_v^J(i,j) = \log e_v(x_i, x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i+1,j-1) + \log t_v(y)\right]$
      $\alpha_v^L(i,j) = \log e_v(x_i, *) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,L\}}(i+1,j) + \log t_v(y)\right]$
      $\alpha_v^R(i,j) = \log e_v(*, x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,R\}}(i,j-1) + \log t_v(y)\right]$
    else $\alpha_v^J(i,j) = -\infty$
      $\alpha_v^L(i,j) = \log e_v(x_i, *)$
      $\alpha_v^R(i,j) = \log e_v(*, x_j)$
    if $\alpha_v^{\{J,L,R\}}(i,j) >$ *best_score*
      store *best_score*, $v, i, j,$ *mode*
  else if $\mathcal{S}_v = \mathrm{L}$
    $\alpha_v^J(i,j) = \log e_v(x_i) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i+1,j) + \log t_v(y)\right]$
    if $d \geq 2$ $\alpha_v^L(i,j) = \log e_v(x_i) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^L(i+1,j) + \log t_v(y)\right]$
    else $\alpha_v^L(i,j) = \log e_v(x_i)$
    $\alpha_v^R(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,R\}}(i,j) + \log t_v(y)\right]$
    if $\alpha_v^{\{J,L\}}(i,j) >$ *best_score*
      store *best_score*, $v, i, j,$ *mode*
  else if $\mathcal{S}_v = \mathrm{R}$
    $\alpha_v^J(i,j) = \log e_v(x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^J(i,j-1) + \log t_v(y)\right]$
    $\alpha_v^L(i,j) = \max_{y \in \mathcal{C}_v} \left[\alpha_y^{\{J,L\}}(i,j) + \log t_v(y)\right]$
    if $d \geq 2$ $\alpha_v^R(i,j) = \log e_v(x_j) + \max_{y \in \mathcal{C}_v} \left[\alpha_y^R(i,j-1) + \log t_v(y)\right]$
    else $\alpha_v^R(i,j) = \log e_v(x_j)$
    if $\alpha_v^{\{J,R\}}(i,j) >$ *best_score*
      store *best_score*, $v, i, j,$ *mode*
  else if $\mathcal{S}_v = \mathrm{B}$
    $(y, z) = \mathcal{C}_v$
    $\alpha_v^J(i,j) = \max_{i-1 \leq k \leq j} \left[\alpha_y^J(i,k) + \alpha_z^J(k+1,j)\right]$
    $\alpha_v^L(i,j) = \max_{i-1 \leq k \leq j} \left[\alpha_y^J(i,k) + \alpha_z^L(k+1,j)\right]$
    $\alpha_v^R(i,j) = \max_{i-1 \leq k \leq j} \left[\alpha_y^R(i,k) + \alpha_z^J(k+1,j)\right]$
    $\alpha_v^T(i,j) = \max_{i \leq k \leq j-1} \left[\alpha_y^R(i,k) + \alpha_z^L(k+1,j)\right]$
    if $\alpha_v^{\{J,L,R,T\}}(i,j) >$ *best_score*
      store *best_score*, $v, i, j,$ *mode*
    $\alpha_v^L(i,j) = \max\left\{\alpha_v^L(i,j), \alpha_y^{\{J,L\}}(i,j)\right\}$
    $\alpha_v^R(i,j) = \max\left\{\alpha_v^R(i,j), \alpha_z^{\{J,R\}}(i,j)\right\}$
  else if $\mathcal{S}_v = \mathrm{E}$
    $\alpha_v^{\{J,L,R\}}(i,j) = -\infty$
Termination:
  score = *best_score* $+ \log \frac{2}{W(W+1)}$

After this recursion is completed, the optimal local parse tree may be recovered by traceback from the best scoring $\alpha_v^X(i,j)$. To facilitate this, it is helpful to store traceback pointers during the dynamic programming recursion; for clarity, these are not indicated in the algorithm description above. In order to avoid parsing ambiguity, any ties in the traceback are resolved in favor of joint mode over marginal modes. Thus marginal mode is only used when required to account for one or more missing residues in the local parse tree.

It is worth noting that an additional kind of structural alignment locality is dealt with by the state structure of a covariance model, rather than by the trCYK algorithm. The alignable subsequence (as identified by trCYK) may also be subject to large internal deletions and insertions relative to the consensus RNA structure. CMs accomodate large structural insertions and deletions by allowing any consensus state to transition to a special "EL" (end-local) state which has a self-transition loop, thereby allowing any structural domain to be truncated anywhere and replaced by zero or more non-homologous residues. The EL state appears in the recursion above,

and its use and rationale for accomodating local structural variation are more fully explained elsewhere (Klein and Eddy, 2003).

3 IMPLEMENTATION

The algorithm described above requires $O(L^2M)$ memory to store traceback pointers for recovering an optimal local parse tree. In order to be able to align large RNAs, we also implemented an extension of the divide and conquer approach described in (Eddy, 2002) to trCYK, reducing the memory requirement to $O(L^2 \log M)$ at the expense of a small increase in computation time. The divide and conquer version was used to obtain the results described below. Both versions are provided in the ANSI C source code of Infernal in the supplementary material.

trCYK has an upper bound time complexity of $O(L^3M)$, the same as standard CYK. trCYK's additional calculations and three matrices in place of one contribute a constant multiplier. Empirically, trCYK runs about five-fold slower than standard CYK on the same problem size. For example, a single RNase P alignment for the results in Figure 2 (283 nodes and 1119 states in the model; sequence length of 400) took 40 seconds for `trcyk` vs. 9.4 seconds for Infernal `cmsearch` with equivalent settings, on a single 3.0GHz Intel Xeon processor.

4 EVALUATION

We compared the effectiveness of the trCYK method for local structural RNA alignment to the previous subtree method, by measuring how accurately and completely the two methods align single shotgun sequence reads to structural RNA consensus models. To do this, we constructed a synthetic test of realigning simulated reads generated by sampling sequence fragments from trusted (presumed correct) alignments. We chose to use simulated data instead of real data because we are interested in conducting a controlled comparison of the two algorithms against known correct answers. Because alignment quality and (in particular) local alignment coverage are strongly affected by parameterization, in order to isolate the algorithm's effect, we used the same profile SCFGs as parameterized by the same implementation (Infernal), and compared Infernal's default subtree alignment method versus its newly implemented trCYK option. To put this comparison in context, we also test two other primary sequence methods: pairwise alignment with BLASTN, and sequence profile alignment with HMMER.

We used multiple sequence alignments of two well known structural RNA genes, small subunit ribosomal RNA (SSU rRNA) and RNase P. SSU rRNA is in general highly conserved, so in many regions of the RNA consensus and for all but the most outlying taxa, SSU rRNA is usually not a particularly challenging sequence alignment problem. RNase P sequences, in contrast, tend to be highly divergent at the primary sequence level. For a trusted RNase P multiple alignment, we used the bacterial class A seed alignment from Rfam 8.1 (Brown, 1999; Griffiths-Jones *et al.*, 2005), and our trusted SSU rRNA alignment was adapted from the bacterial seed alignment from the Comparative RNA Web Site, CRW (Cannone *et al.*, 2002). Due to the large number of sequences, the SSU alignment was filtered to remove sequences such that no aligned pair was more than 92% pairwise identical. It was also edited slightly towards a consensus structural alignment in preference to an evolutionarily correct alignment where there was ambiguity between structural conservation and homology. Our SSU rRNA alignment is provided in the supplementary material.

The sequences in the trusted alignment were clustered by single linkage by pairwise identity (as defined by the original alignment) and split into a smaller training alignment subset and a testing set, so as to minimize pairwise identity between training and test data and create more challenging alignment tests. For SSU, this gave 101 training sequences and 51 testing sequences, with maximum identity between sets of 82%. The smaller RNase P family has 28 training sequences and 15 testing sequences, with maximum identity of 60%.

We simulated a genomic context for each test sequence, consisting of randomly generated sequence of the same monoresidue composition, and then sampled a random subsequence of length 800 (SSU rRNA) or 400 (RNase P) that contained at least 100 nucleotides of the RNA. Five fragments were sampled for each SSU rRNA test sequence, and ten for each RNase P test sequence, for a total of 255 SSU test fragments and 150 RNase P test fragments. The 800 nt length of SSU rRNA test fragments roughly corresponds to the average single read length in recent metagenomic sequencing surveys with Sanger sequencing technology (Venter *et al.*, 2004; Rusch *et al.*, 2007). RNase P is a shorter RNA (average length just 310 nt) so shorter fragments of 400 bases were used, roughly according to the current capabilities of newer 454 sequencing technology. The training alignments and test fragments are provided in the supplementary material.

We aligned each test sequence fragment to the training alignment using four different local alignment methods. For BLAST local sequence alignment, we used NCBI blastn (Altschul *et al.*, 1997), with default parameters except for a word length of W=6 and an E-value cut-off of 1.0, and used the pairwise alignment with the lowest E-value to identify the nearest neighbor among any of the individual training set sequences. All alignments to that nearest-neighbor, including lower scoring ones, were considered as portions of the complete alignment. For profile alignment, we used HMMER 2.3.2 (Eddy, 2008) to build a profile HMM from the training alignment subset with `hmmbuild` using the `-f` option to build local alignment models, and aligned each test fragment to the profile (thereby adding it to the multiple alignment with the training sequences) with `hmmalign` using default parameters. For the subtree-based local alignment method, we used Infernal version 1.0 to build a CM of the training alignment subset with `cmbuild` with the `--enone` option to shut off entropy weighting. (We have observed that Infernal's entropy weighting option (Nawrocki and Eddy, 2007) is appropriate for maximal sensitivity in remote homology search, but not for alignment accuracy; DLK, SRE, and Eric Nawrocki, unpublished observations.) We aligned each test fragment to the CM with `cmsearch` using default parameters. Finally, for trCYK, we used the `trcyk` program included in Infernal 1.0 to align test fragments to the same CM used for `cmsearch`.

To evaluate the alignments, they are mapped to the reference alignment using an intermediary sequence; for blastn, this is the nearest-neighbor sequence it was aligned to, and for the profile methods it is the consensus sequence of the model. If a residue in the test sequence was aligned to a non-gap position in the reference alignment, it is correct in the output alignment if it is aligned to that same position, and incorrect otherwise. If the residue was originally aligned to a gap position, it is judged to be correct if, in the output alignment, it is between the same two consensus positions that bordered the original gap. Misaligned residues include both residues that should be aligned but are incorrect, and residues that should

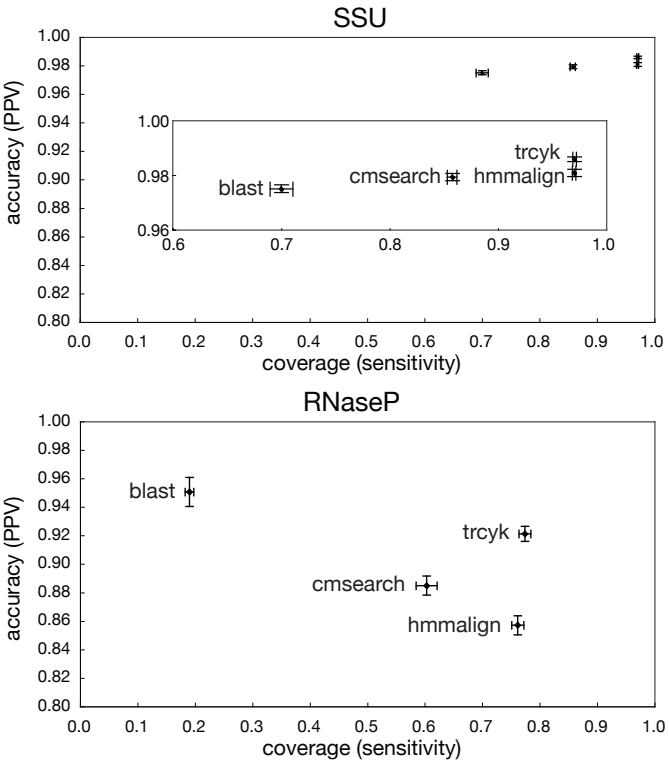

Fig. 4. Per-residue accuracy of alignment methods. Alignment of simulated metagenomic reads compared against a reference alignment for four alignment methods: primary sequence (blastn), primary sequence profile (hmmalign), CM with subtree-based local alignment (cmsearch), and CM with truncated sequence model (trcyk). Means and standard deviations for sensitivity and PPV are plotted. Top: alignment of 800nt fragments to the bacterial small subunit ribosomal RNA. Bottom: alignment of 400nt fragments to bacterial RNase P.

not be aligned at all (part of the surrounding "genomic" sequence). We measured both the accuracy of the resulting alignments (positive predictive value, PPV: the fraction of aligned positions that are also found in the trusted alignment) and the coverage (sensitivity: the fraction of aligned positions in the trusted alignment that are found in the calculated local alignment).

The results, mean and standard deviations for sensitivity and PPV for each method, are shown in Figure 4. BLAST generally returns highly accurate alignments, but has low coverage, corresponding to a tendency to pick out only the most highly conserved portions of the true alignments. (The default NCBI BLAST scoring scheme is tuned for high sequence identity. In principle we should be able to improve the coverage somewhat by a different choice of scoring matrix.) Profile HMMs (hmmalign) achieve both high accuracy and coverage. CMs with subtree-based local alignment (cmsearch) show poor coverage relative to HMMs, illustrating the issue that motivated this work. The new method, trCYK, matches the coverage of profile HMM sequence alignment, while providing higher accuracy. The improvement is not large, but even small increases in coverage and accuracy are important when the alignment is to be used in downstream phylogenetic analyses that are sensitive to both.

5 DISCUSSION

The trCYK algorithm performs local structural RNA alignment in a manner that uses secondary structural information (correlated base pairs) where possible, and reverts to sequence alignment when a truncation has removed sequence that would be base paired. Alignment coverage of sequence fragments (such as single reads from metagenomic shotgun sequencing) is maximized, while still retaining the accuracy of CM-based RNA structural alignment methods. The trCYK algorithm rests on good theoretical ground by viewing the sequence truncation problem formally as a missing data inference problem, and it makes only two minor assumptions to simplify the missing data inference problem to one that can be solved by a relatively efficient dynamic programming recursion.

The disadvantage of trCYK is that unlike local primary sequence alignment, which is as computationally efficient as global sequence alignment, it needs to track the three possible structural alignment modes (joint, left marginal, and right marginal) in the dynamic programming recursion. This imposes about a three-fold increase in memory and five-fold increase in CPU time required relative to previous CM alignment implementations. This cost is unfortunate, as the use of CM-based approaches is already limited by their relatively high computational complexity. We expect to be able to accelerate trCYK with the same approaches we are developing for standard CYK using the subtree-based alignment model (Nawrocki and Eddy, 2007). We additionally expect it will be feasible to develop simple accelerated heuristics for identifying optimal or near-optimal switch points from joint to marginal alignment modes, in order to bypass the need for full dynamic programming. For example, we should be able to use fast primary sequence alignment to determine likely endpoints of the alignment on the consensus yield of the structural model, and from that derive a CM with an appropriately marginalized partial structure. We therefore envision trCYK's future role as a rigorous baseline against which more heuristic local RNA structural alignment methods may be compared.

FUNDING

This work was supported by a National Science Foundation Graduate Research Fellowship to DLK, and by the Howard Hughes Medical Institute.

REFERENCES

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.*, **25**, 3389–3402.

Backofen, R. and Will, S. (2004). Local sequence-structure motifs in RNA. *J Bioinform Comput Biol*, **2**, 681–698.

Brown, J. W. (1999). The ribonuclease P database. *Nucl. Acids Res.*, **27**, 314.

Cannone, J. J., Subramanian, S., Schnare, M. N., Collett, J. R., D'Souza, L. M., Du, Y., Feng, B., Lin, N., Madabusi, L. V., Mller, K. M., Pande, N., Shang, Z., Yu, N., and Gutell, R. R. (2002). The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics.*, **3**, 2.

Chen, K. and Pachter, L. (2005). Bioinformatics for whole-genome shotgun sequencing of microbial communities. *PLoS Comput Biol.*, **1**, 106–112.

Chothia, C. and Lesk, A. (1986). The relation between the divergence of sequence and structure in proteins. *EMBO J.*, **5**, 823–826.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. J. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK.

Eddy, S. R. (2002). A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, **3**, 18.

Eddy, S. R. (2008). HMMER - biosequence analysis using profile hidden Markov models. [http://hmmer.janelia.org/].

Eddy, S. R. and Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucl. Acids Res.*, **22**, 2079–2088.

Gardner, P. P., Daub, J., Tate, J. G., Nawrocki, E. P., Kolbe, D. L., Lindgreen, S., Wilkinson, A. C., Finn, R. D., Griffiths-Jones, S., Eddy, S. R., and Bateman, A. (2009). Rfam: Updates to the RNA families database. NAR, in press.

Gibrat, J. F., Madej, T., and Bryant, S. H. (1996). Surprising similarities in structure comparison. *Curr Opin Struct Biol.*, **6**, 377–385.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R., and Bateman, A. (2005). Rfam: Annotating non-coding RNAs in complete genomes. *Nucl. Acids Res.*, **33**, D121–D141.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free algorithms. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, Mass.

Klein, R. J. and Eddy, S. R. (2003). RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, **4**, 44.

Nawrocki, E. P. and Eddy, S. R. (2007). Query-dependent banding (QDB) for faster RNA similarity searches. *PLoS Comput. Biol.*, **3**, e56.

Nawrocki, E. P., Kolbe, D. L., and Eddy, S. R. (2009). Infernal 1.0: Inference of RNA alignments. manuscript submitted.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, **85**, 2444–2448.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, **63**, 581–592.

Rusch, D. B., Halpern, A. L., Sutton, G., Heidelberg, K. B., Williamson, S., Yooseph, S., Wu, D., Eisen, J. A., Hoffman, J. M., Remington, K., Beeson, K., Tran, B., Smith, H., Baden-Tillson, H., Stewart, C., Thorpe, J., Freeman, J., Andrews-Pfannkoch, C., Venter, J. E., Li, K., Kravitz, S., Heidelberg, J. F., Utterback, T., Rogers, Y. H., Falcon, L. I., Souza, V., Bonilla-Rosso, G., Eguiarte, L. E., Karl, D. M., Sathyendranath, S., Platt, T., Bermingham, E., Gallardo, V., Tamayo-Castillo, G., Ferrari, M. R., Strausberg, R. L., Nealson, K., Friedman, R., Frazier, M., and Venter, J. C. (2007). The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLoS Biol.*, **5**, e77.

Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjölander, K., Underwood, R. C., and Haussler, D. (1994). Stochastic context-free grammars for tRNA modeling. *Nucl. Acids Res.*, **22**, 5112–5120.

Schloss, P. D. and Handelsman, J. (2005). Metagenomics for studying unculturable microorganisms: Cutting the gordian knot. *Genome Biol.*, **6**, 229.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Venter, J. C., Remington, K., Heidelberg, J. F., Halpern, A. L., Rusch, D., Eisen, J. A., Wu, D., Paulsen, I., Nelson, K. E., Nelson, W., Fouts, D. E., Levy, S., Knap, A. H., Lomas, M. W., Nealson, K., White, O., Peterson, J., Hoffman, J., Parsons, R., Baden-Tillson, H., Pfannkoch, C., Rogers, Y. H., and Smith, H. O. (2004). Environmental genome shotgun sequencing of the Sargasso Sea. *Science.*, **304**, 66–74.

Vogel, C., Bashton, M., Kerrison, N. D., Chothia, C., and Teichmann, S. A. (2004). Structure, function and evolution of multidomain proteins. *Curr. Opin. Struct. Biol.*, **14**, 208–216.

Will, S., Reiche, K., Hofacker, I. L., Stadler, P. F., and Backofen, R. (2007). Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput Biol.*, **3**, e65.

Yao, Z., Weinberg, Z., and Ruzzo, W. L. (2006). CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics.*, **22**, 445–452.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, **10**, 189–208.