

# Query-Dependent Banding (QDB) for Faster RNA Similarity Searches

Eric P. Nawrocki and Sean R. Eddy  
HHMI Janelia Farm Research Campus  
19700 Helix Drive  
Ashburn VA 20176  
<http://selab.janelia.org/>

November 30, 2006

**Running head:** Faster RNA Similarity Searches

## Abstract

When searching sequence databases for RNAs, it is desirable to score both primary sequence and RNA secondary structure similarity. Covariance models (CMs) are probabilistic models well suited for RNA similarity search applications. However, the computational complexity of CM dynamic programming alignment algorithms has limited their practical application. Here we describe an acceleration method called *query-dependent banding* (QDB), which uses the probabilistic query CM to precalculate regions of the dynamic programming lattice that have negligible probability, independently of the target database. We have implemented QDB in the freely available *INFERNAL* software package. QDB reduces the average-case time complexity of CM alignment from  $LN^{2.4}$  to  $LN^{1.3}$  for a query RNA of  $N$  residues and a target database of  $L$  residues, resulting in a four-fold speedup for typical RNA queries. Combined with other improvements to *INFERNAL*, including informative mixture Dirichlet priors on model parameters, benchmarks also show increased sensitivity and specificity resulting from improved parameterization.

## Introduction

Many functional RNAs conserve a base-paired secondary structure. Conserved RNA secondary structure induces long-distance pairwise correlations in homologous RNA sequences. When performing database searches to identify homologous structural RNAs, it is desirable for RNA similarity search programs to score a combination of secondary structure and primary sequence conservation.

A variety of approaches for RNA similarity searching have been described. There are specialized programs for identifying one particular RNA family or motif, such as programs that identify tRNAs [1, 2], snoRNAs [3, 4], microRNAs [5, 6], SRP RNAs [7], and rho-independent transcription terminators [8]. There are also pattern-matching algorithms that rely on expertly designed query patterns [9, 10]. However, the most generally useful approaches are those that take any RNA (or any multiple RNA alignment) as a query, and use an appropriate scoring system to search a sequence database and rank high-scoring similarities, just as programs like BLAST do for linear sequence comparison [11].

In a general search program, one wants to score a combination of RNA sequence and structural conservation in a principled rather than *ad hoc* manner. A satisfactory solution to this problem is known, using probabilistic models called *stochastic context-free grammars* (SCFGs). SCFGs readily capture both primary sequence and (non-pseudoknotted) RNA secondary structure conservation [12, 13]. Just as hidden Markov models (HMMs) are useful for many different linear sequence modeling applications, including gene finding, multiple alignment, motif finding, and similarity search [13], SCFGs are a generally useful paradigm for probabilistic RNA sequence/structure analysis, with applications including secondary structure prediction and gene finding. A particular SCFG architecture called *covariance models* (CMs) was developed specifically for the RNA similarity search problem [14]. CMs are profile SCFGs, analogous to the use of profile HMMs in sequence analysis [14, 15]. The Rfam database of RNA families [16] is based on CM software (INFERNAL) in much the same way that the Pfam database of protein families is based on profile HMM software (HMMER) [17, 18].

The most serious problem with using CMs has been their computational complexity. Applying standard SCFG dynamic programming alignment algorithms to the particular case of CMs results in algorithms that require  $O(N^3)$  memory and  $O(LN^3)$  time for a query of length  $N$  residues (or consensus alignment columns) and a target database sequence of length  $L$ . The memory complexity problem has essentially been solved, by extending divide-and-conquer dynamic programming methods (the Hirshberg or Myers/Miller algorithm) to the case of CMs [15], but the time complexity problem still stands.

Weinberg and Ruzzo have described two filtering methods for accelerating CM searches [19–21]. Both methods work by scoring a target sequence first by a linear sequence comparison method using a profile HMM specially constructed from the query CM, and passing only the subset of hits above a threshold for rescoring with the more expensive CM alignment algorithm. For most current Rfam models, the Weinberg/Ruzzo filters give about a hundred-fold speed up relative to a full CM based search at little or no cost to sensitivity and specificity. However, because the Weinberg/Ruzzo filters depend on primary sequence conservation alone, they can be relatively ineffective for RNA families that exhibit poor sequence conservation – unfortunately, precisely the RNAs that benefit the most from SCFG-based search methods. Indeed, in this respect, we are concerned that the overall performance of these filters on the current Rfam database may be somewhat misleading. Rfam currently uses a crude BLAST-based filtering method to accelerate the CM searches used in curating the database. This step introduces a bias towards high primary sequence similarity in current Rfam alignments. As Rfam improves and incorporates more diverse structural homologs, the effectiveness of sequence-based filters will decrease.

Here, we describe a method for accelerating CM searches using a banded dynamic programming (DP) strategy. In banded DP, one uses some fast method to identify a band through the DP matrix where the optimal alignment is likely to lie, and then calculates computationally expensive DP recursions only within that band. In most cases, including our approach, banded DP is a heuristic that sacrifices guaranteed alignment optimality. Banding is a standard approach in many areas of sequence analysis. Gapped BLAST uses banded DP to convert ungapped high-scoring pairs (HSPs) to full gapped alignments [11]. LAGAN and MULTILAGAN use banded dynamic programming (referred to as limited-area dynamic programming) to stitch together alignments between anchored sequences when aligning long genomic sequences [22]. Banding has also been applied to profile SCFGs by Michael Brown in his RNACAD program by using information from a profile HMM alignment to define bands for the expensive SCFG alignment [23]. The key to developing a banded DP strategy is in deciding how the bands are identified. Usually, including all the examples just mentioned, banded DP involves performing some sort of rapid approximate sequence alignment between the query and the target.

In contrast, the method we describe here, called *query-dependent banding* (QDB), takes advantage of specific properties of CMs in order to predefine bands that are independent of any target sequence. QDB depends on the consensus secondary structure of the query, so it is complementary to acceleration methods like the Weinberg/Ruzzo filters that rely on sequence but not structure.

## Results

Briefly, the key idea is the following. Each base pair and each single-stranded residue in the query RNA is represented in a CM by a *state*. States are arranged in a tree-like structure that mirrors the secondary structure of the RNA, along with additional states to model insertions and deletions. The standard CM dynamic programming alignment algorithm works by calculating the probability that a substructure of the query rooted at state  $v$  aligns to a subsequence  $i..j$  in the target sequence. The calculation is recursive, starting at the leaves of the CM (ends of hairpin loops) and subsequences of length 0, and working upwards in larger substructures of the CM, and outwards in longer and longer subsequences.

To guarantee optimality, at each  $v$ , the DP algorithm must score all possible  $i..j$  subsequences in the target sequence. However, most of these subsequences are obviously too long or short, when one considers the size of the query substructure under state  $v$ . For example, when state  $v$  models the closing base pair of a consensus four-base loop, only  $i..j$  subsequences of length six are likely to occur in any optimal alignment to state  $v$  ( $i = j - 5$ ,  $j$  being the base pair, and  $j - 4..j - 1$  being the four bases of the hairpin loop). Likewise, the optimal subsequence aligned to the next consensus base pair in that stem is almost certainly of length eight.

Because insertions and deletions may occur in the target sequence, no subsequence length is known with certainty, but because the CM is a probabilistic model, a probability distribution for subsequence lengths under each state (including the probability of insertions and deletions) can be analytically derived from the query CM. These distributions can be used to determine a band of subsequence lengths that captures all but a negligible amount of the probability mass. A CM dynamic programming algorithm can then look not at all subsequences  $i, j$  for each state  $v$ , but only those  $i$  within a band of minimum and maximum distance relative to each  $j$ .

To formalize this idea, we start with a description of CMs, followed by the QDB algorithms for calculating the subsequence length distributions, using these length distributions to determine bands, and using the bands in a banded CM dynamic programming alignment algorithm. Calculation of the bands is sensitive to transition parameter estimation, so we describe INFERNAL's new implementation of informative Dirichlet priors for CM parameter estimation. Finally, we present results from a benchmark that suggest the sensitivity and specificity of a QDB-accelerated search is negligibly different from a non-banded search.

## Covariance models

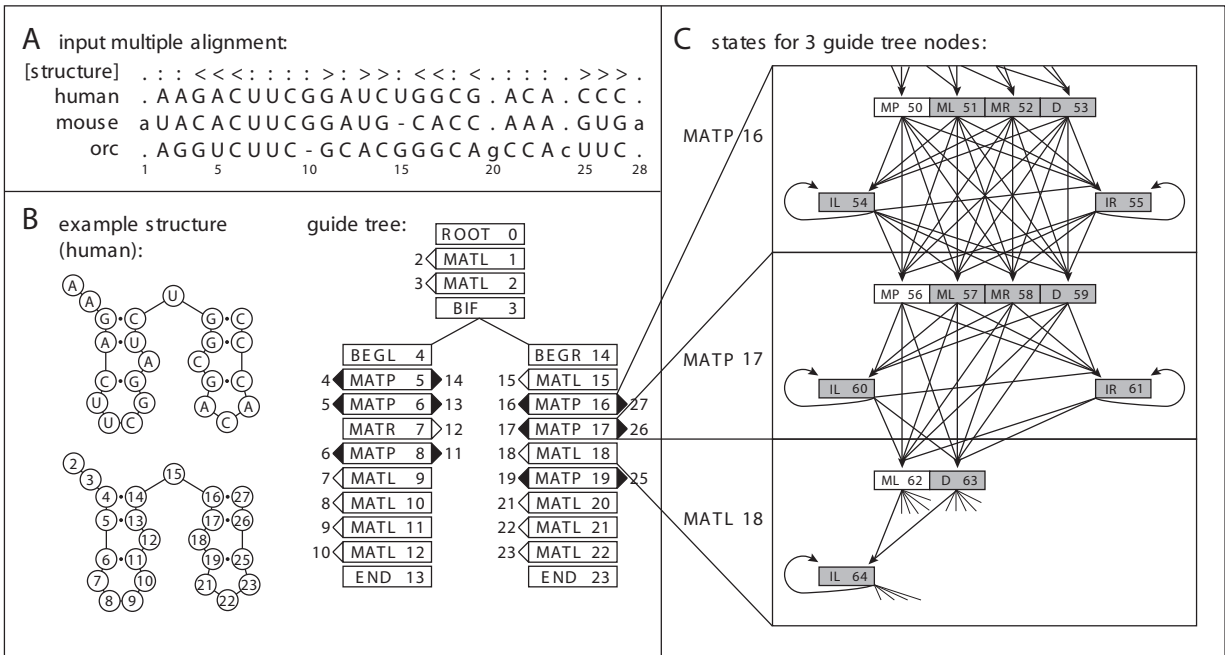
CMs are a convention for mapping an RNA secondary structure into a tree-like, directed graph of SCFG states and state transitions (or equivalently, SCFG nonterminals and production rules). The CM is organized by a binary tree of *nodes* representing base pairs and single-stranded residues in the query’s structure. Each node contains a number of *states*, where one state represents the consensus alignment to the query, and the others represent insertions and deletions relative to the query. Figure 1 shows an example of converting a consensus structure to the guide tree of nodes, and part of the expansion of those guide tree nodes into the CM’s state graph. Here we will only concentrate on the aspects of CMs necessary to understand QDB, and a subset of our usual notation. For full details on CM construction, see [15,24].

A guide tree consists of eight types of nodes. MATP nodes represent consensus base pairs. MATL and MATR nodes represent consensus single-stranded residues (emitted to the left or right with respect to a stem). BIF nodes represent bifurcations in the secondary structure of the family, to deal with multiple stem-loops. A ROOT node represents the start of the model. BEGL and BEGR nodes represent the beginnings of a branch on the left and right side of a bifurcation. END nodes end each branch.

The CM is composed of seven different types of states, each with a corresponding form of production rule, with notation defined as follows:

State type	Description	Production	$\Delta_v^L$	$\Delta_v^R$	Emission	Transition
P	(pair emitting)	$P \rightarrow aYb$	1	1	$e_v(a, b)$	$t_v(Y)$
L	(left emitting)	$L \rightarrow aY$	1	0	$e_v(a)$	$t_v(Y)$
R	(right emitting)	$R \rightarrow Ya$	0	1	$e_v(a)$	$t_v(Y)$
B	(bifurcation)	$B \rightarrow SS$	0	0	1	1
D	(delete)	$D \rightarrow Y$	0	0	1	$t_v(Y)$
S	(start)	$S \rightarrow Y$	0	0	1	$t_v(Y)$
E	(end)	$E \rightarrow \epsilon$	0	0	1	1

That is, for instance, if state  $v$  is a pair state, it produces (aligns to and scores) two correlated residues  $a$  and  $b$  and moves to some new state  $Y$ . The probability that it produces a residue pair  $a, b$  is given by an *emission probability*  $e_v(a, b)$ . The probability that it moves to a particular state  $Y$  is given by a *transition probability*  $t_v(Y)$ . The set of possible states  $Y$  that  $v$  may transit to is limited to the states in the next (lower) node in the guide tree (and insert states in the current node); the set of possible children states  $Y$  is called  $C_v$ , for “children of  $v$ ”. The indicators  $\Delta_v^L$  and  $\Delta_v^R$  are used to simplify notation in CM dynamic programming



**Figure 1: An example RNA family and corresponding CM. (A):** A toy multiple alignment of three RNA sequences, with 28 total columns, 24 of which will be modeled as consensus positions. The [structure] line annotates the consensus secondary structure: < and > symbols mark base pairs, :’s mark consensus single stranded positions, and .’s mark “insert” columns that will not be considered part of the consensus model because more than half the sequences in these columns contain gaps. **(B):** The structure of one sequence from (A), the same structure with positions numbered according to alignment columns, and the guide tree of nodes corresponding to that structure, with alignment column indices assigned to nodes (for example, node 5, a MATP “match-pair” node, will model the consensus base pair between columns 4 and 14). **(C):** The state topology of three selected nodes of the CM, for two MATP nodes and one consensus “leftwise” single residue bulge node (MATL, “match-left”). The consensus pair and singlet states (two MPs and one ML) are white, and the insertion/deletion states are grey. State transitions are indicated by arrows.

algorithms. They are the residues emitted to the left and right of state  $v$ , respectively. Bifurcation rules are special, in that they always transition to two particular start (S) states, at the root of subtrees in the guide tree, with probability 1.0.

These state types essentially define a “normal form” for SCFG models of RNA, akin to SCFGs in Chomsky normal form where all productions are in one of two forms,  $Y \rightarrow a$  or  $Y \rightarrow YY$ . We describe CM algorithms (including QDB) in terms of this normal form. CMs define a specific way that nodes in the guide tree are expanded into states, and how those states are connected within each node and to states in the next node in the guide tree. For example, a MATP node that deals with a consensus base pair contains six states called MATP\_MP (a P state for matching the base pair), MATP\_ML and MATP\_MR (a L and R state for matching only the leftmost or rightmost base and deleting the right or left one, respectively), MATP\_D (a D state for deleting the base pair), and MATP\_IL and MATP\_IR (L and R states with self-transitions, for inserting one or more residues to the left and/or right before going to the next node).

Thus a CM is a generative probabilistic model of homologous RNAs. A sequence is emitted starting at the root, moving downwards from state to state according to state transition probabilities, emitting residues and residue pairs according to emission probabilities, and bifurcating into substructures at bifurcation states. An important property of a CM is the states can be numbered from  $0..M - 1$  (from root to leaves) such that for any state  $v$ , the states  $y$  that it can transit to must have indices  $y \geq v$ . There are no cycles in a CM, other than self-transitions on insert states. This is the property that enables the recursive calculations that both CM DP alignment algorithms and QDB rely on.

Without any change in the above description, CMs apply to either global or local alignment, and to either pairwise alignment to single RNA queries or profile alignment to a consensus query structure of a multiple RNA sequence alignment. CMs for single RNA queries are derived identically to profiles of a consensus structure, differing only in the parameterization method [25]. Local structural alignment to substructures and truncated structures (as opposed to requiring a global alignment to the whole RNA structural model) is achieved by adding state transitions from the ROOT that permit entering the model at any internal consensus state with some probability, and state transitions from any internal consensus state to an END with some probability [24, 25].

## QDB algorithm

Observe that for any state  $v$ , we could enumerate all possible paths down the model from  $v$  to the END(s). Each path has a certain probability (the product of the transition probabilities used by the path), and it will emit a certain number  $n$  of residues (2 per P state, 1 per L or R state in the path). The sum of these path probabilities for each  $n$  defines a probability distribution  $\gamma_v(n)$ , the probability that the CM subgraph rooted at  $v$  will generate a subsequence of length  $n$ . Given a finite limit  $Z$  on maximum subsequence length (defined later), we can calculate  $\gamma_v(n)$  by an efficient recursive algorithm, working from the leaves of the CM towards the root, and from smallest subsequences to largest:

for  $v = M - 1$  down to 0:

$$\begin{array}{l}
 v = \text{end state } (E): \\
 \\
 v = \text{bifurcation } (B): \\
 \\
 \text{else } (v = S, P, L, R):
 \end{array}
 \left|
 \begin{array}{l}
 \gamma_v(0) = 1 \\
 \gamma_v(d) = 0 \\
 \\
 \gamma_v(d) = \sum_{n=0}^d \gamma_y(n) * \gamma_z(d - n) \\
 \\
 \gamma_v(d) = 0 \\
 \gamma_v(d) = \sum_{y \in C_v} \gamma_y(d - (\Delta_v^L + \Delta_v^R)) * t_v(y)
 \end{array}
 \right.
 \begin{array}{l}
 \text{for } d = 1 \text{ to } Z \\
 \\
 \text{for } d = 0 \text{ to } Z \\
 \\
 \text{for } d = 0 \text{ to } (\Delta_v^L + \Delta_v^R - 1) \\
 \text{for } d = (\Delta_v^L + \Delta_v^R) \text{ to } Z
 \end{array}$$

For example, if we are calculating  $\gamma_v(d)$  where  $v$  is a pair state, we know that  $v$  must emit a pair of residues then transit to a new state  $y$  (one of its possible transitions  $C_v$ ), and then a subgraph rooted at  $y$  will have to account for the rest of the subsequence of length  $d - 2$ . Therefore,  $\gamma_v(d)$  must be the sum, over all possible states  $y$  in  $C_v$ , of the transition probability  $t_v(y)$  times the probability that the subtree rooted at  $y$  generates a subsequence of length  $d - 2$  – which is  $\gamma_y(d - 2)$ , guaranteed to have already been calculated by the recursion. For the  $B$  state (bifurcation) calculation, indices  $y$  and  $z$  indicate the left and right S (start) state that bifurcation state  $v$  must connect to.

A band  $\text{dmin}(v)..\text{dmax}(v)$  of subsequence lengths that will be allowed for each state  $v$  is then defined as follows. A parameter  $\beta$  defines the threshold for the negligible probability mass that we are willing to allow outside the band. (The default value of  $\beta$  is set to  $10^{-7}$ , as described later.) We define  $\text{dmin}(v)$  and  $\text{dmax}(v)$  such that the cumulative left and right tails of  $\gamma_v(n)$  contain less than a probability  $\frac{\beta}{2}$ :

$$\sum_{d=0}^{\text{dmin}(v)-1} \gamma_v(d) < \frac{\beta}{2},$$

$$\sum_{d=\text{dmax}(v)+1}^Z \gamma_v(d) < \frac{\beta}{2}.$$

Larger values of  $\beta$  produce tighter bands and faster alignments, but at a cost of increased risk of missing the optimal alignment.  $\beta$  is the only free parameter that must be specified to QDB.

Because CMs have emitting self-loops (i.e. insert states), there is no finite limit on subsequence lengths. However, we must impose a finite limit  $Z$  to obtain a finite calculation.  $Z$  can be chosen to be sufficiently large that it does not affect  $\text{dmax}(v)$  for any state  $v$ . On a digital computer with floating point precision  $\epsilon$  (the largest value for which  $1 + \epsilon = 1$ ), it suffices to guarantee that, for all  $v$ :

$$\frac{\sum_{d=Z+1}^{\infty} \gamma_v(d)}{\sum_{d'=\text{dmax}(v)+1}^Z \gamma_v(d')} \leq \epsilon$$

Empirically, we observe that the tails of the  $\gamma_v(d)$  densities decrease approximately geometrically. We can estimate the mass remaining in the unseen tail by fitting a geometric tail to the observed density  $\gamma_v(d)$ . Our implementation starts with a reasonable guess at  $Z$  and verifies that the above condition is true for each  $v$ , assuming these geometrically decreasing tails; if it is not,  $Z$  is increased and bands are recalculated until it is.

A QDB calculation only needs to be performed once per query CM to set the bands. Overall, a QDB calculation requires  $\Theta(MZ)$  in time and space; or equivalently, because both  $M$  and  $Z$  scale roughly linearly with the length  $L$  in residues of the query RNA,  $\Theta(L^2)$ . The time and space requirement is negligible compared to the requirements of a typical CM search.

## Banded CYK database search algorithm for CMs

A standard algorithm for obtaining the maximum likelihood alignment (parse tree) of an SCFG to a target sequence is the Cocke-Younger-Kasami (CYK) dynamic programming algorithm [26–28]. Formally, CYK applies to SCFGs reduced to Chomsky normal form, and it aligns to the complete sequence. The CM database search algorithm is a CYK variant, specialized for the “normal form” of our seven types of RNA production rules, and for scanning long genomic sequences for high-scoring subsequences (hits) [13].

The CM search algorithm recursively calculates  $\alpha_v(j, d)$ , the log odds score of the most likely CM parse subtree rooted at state  $v$  that generates (aligns to) the length  $d$  subsequence  $x_{j-d+1}..x_j$  that ends at position  $j$  of target sequence  $x$  [13, 14]. This calculation initializes at the smallest subgraphs ( $E$  states)

and shortest subsequences ( $d = 0$ ) and iterates upwards and outwards to progressively larger subtrees and longer subsequences up to a preset window size  $W$ . The outermost loop iterates over the end position  $j$  on the target sequence, enabling an efficient scan across a long target like a chromosome sequence. Banding is achieved simply by limiting all loops over possible subsequence lengths  $d$  to the bounds  $\text{dmin}(v)..\text{dmax}(v)$  derived in the band calculation algorithm, rather than all possible lengths  $0..W$ . The banded version of the algorithm is as follows:

Initialization (impose bands): for  $j = 0$  to  $L$ ,  $v = M - 1$  down to 0:

for  $d = 0$  to  $\min((\text{dmin}(v) - 1), j)$   $\alpha_v(j, d) = -\infty$ ;

for  $d = (\text{dmax}(v) + 1)$  down to  $j$   $\alpha_v(j, d) = -\infty$ .

Initialization at  $d = 0$ : for  $j = 0$  to  $L$ ,  $v = M - 1$  down to 0:

$v = \text{end state } (E)$ :  $\alpha_v(j, 0) = 0$ ;

$v = \text{bifurcation } (B)$ :  $\alpha_v(j, 0) = \alpha_y(j, 0) + \alpha_z(j, 0)$ ;

$v = \text{delete or start } (D, S)$ :  $\alpha_v(j, 0) = \max_{y \in C_v} [\alpha_y(j, 0) + \log t_v(y)]$ ;

else ( $v = P, L, R$ ):  $\alpha_v(j, 0) = -\infty$ .

Recursion: for  $j = 1$  to  $L$ ,  $d = \max(1, \text{dmin}(v))$  to  $\min(\text{dmax}(v), j)$ ,  $v = M - 1$  down to 0

$v = \text{end state } (E)$ :  $\alpha_v(j, d) = -\infty$ ;

$v = \text{bifurcation } (B)$ :  $\text{kmin} = \max(\text{dmin}(z), (d - \text{dmax}(y)))$ ,

$\text{kmax} = \min(\text{dmax}(z), (d - \text{dmin}(y)))$ ,

$\alpha_v(j, d) = \max_{\text{kmin} \leq k \leq \text{kmax}} [\alpha_y(j - k, d - k) + \alpha_z(j, k)]$ ;

$v = \text{delete or start } (D, S)$ :  $\alpha_v(j, d) = \max_{y \in C_v} [\alpha_y(j, d) + \log t_v(y)]$ ;

else ( $v = P, L, R$ ):  $\alpha_v(j, d) = \max_{y \in C_v} [\alpha_y(j - \Delta_v^R, d - (\Delta_v^L + \Delta_v^R)) + \log t_v(y)]$   
 $+ \log e_v(x_i, x_j)$

For example, if we are calculating  $\alpha_v(j, d)$  and  $v$  is a pair state ( $P$ ),  $v$  will generate the basepair  $x_{j-d+1}, x_j$  and transit to a new state  $y$  (one of its possible transitions  $C_v$ ) which then will have to account for the smaller subsequence  $x_{j-d+2}..x_{j-1}$ . The log odds score for a particular choice of next state  $y$  is the sum of three terms: an emission term  $\log e_v(x_{j-d+1}, x_j)$ , a transition term  $\log t_v(y)$ , and an already calculated solution for the smaller optimal parse tree rooted at  $y$ ,  $\alpha_y(j - 1, d - 2)$ . The value assigned to  $\alpha_v(j, d)$  is the maximum over all possible choices of child states  $y$  that  $v$  can transit to.

The  $W$  parameter defines the maximum size of a potential hit to a model. Previous INFERNAL implementations required an *ad hoc* guess at a reasonable  $W$ . The band calculation algorithm delivers a probabilistically derived  $W$  for database search in  $\text{dmax}(0)$ , the upper bound on the length of the entire sequence (the sequence generated from the root state of the CM).

QDB does not reduce the asymptotic computational complexity of the CM search algorithm. Both the banded algorithm and the original algorithm are  $O(MW + BW^2)$  memory and  $O(L(MW + BW^2))$  time, for a model of  $M$  states containing  $B$  bifurcation states, window size  $W$  of residues, and target database length  $L$ .  $M$ ,  $B$ , and  $W$  all scale with the query RNA length  $N$ , so roughly speaking, worst-case asymptotic time complexity is  $O(LN^3)$ .

### **Informative Dirichlet priors**

The subsequence length distributions calculated by QDB depend on the CM’s transition probabilities. Transition probability parameter estimation is therefore crucial for obtaining predicted subsequence length bands that reflect real subsequence lengths in homologous RNA targets. Transition parameters in INFERNAL are mean posterior estimates, combining (*ad hoc* weighted) observed counts from an input RNA alignment with a Dirichlet prior [24]. Previous to this work, INFERNAL used an uninformative uniform Dirichlet transition prior, equivalent to the use of Laplace “plus-1” pseudo-counts. However, we found that transition parameters derived under a uniform prior inaccurately predict target subsequence lengths, as shown in an example in Figure 2. The problem is exacerbated when there are few sequences in the query alignment, when the choice of prior has more impact on mean posterior estimation. To alleviate this problem, we estimated informative single component Dirichlet prior densities for CM transition parameters, as follows.

The training data for transition priors consisted of the 381 seed alignments in the Rfam database, version 6.1 [16]. For each alignment, we built CM structures by INFERNAL’s default procedure, and collected weighted counts of observed transitions in the implied parse trees of the training sequences. Considering all possible combinations of pairs of adjacent node types, there are 73 possible distinct types of transition probability distributions in CMs. To reduce this parameter space, we tied these 73 distributions into 36 groups by assuming that certain distributions were effectively equivalent. 36 Dirichlet densities were then estimated from these pooled counts by maximum likelihood as described in [29], with the exception that we optimize by conjugate gradient descent [30] rather than by expectation-maximization (EM). The results, including the Dirichlet parameters, are given in Table 1. Using these priors for transition probability pa-

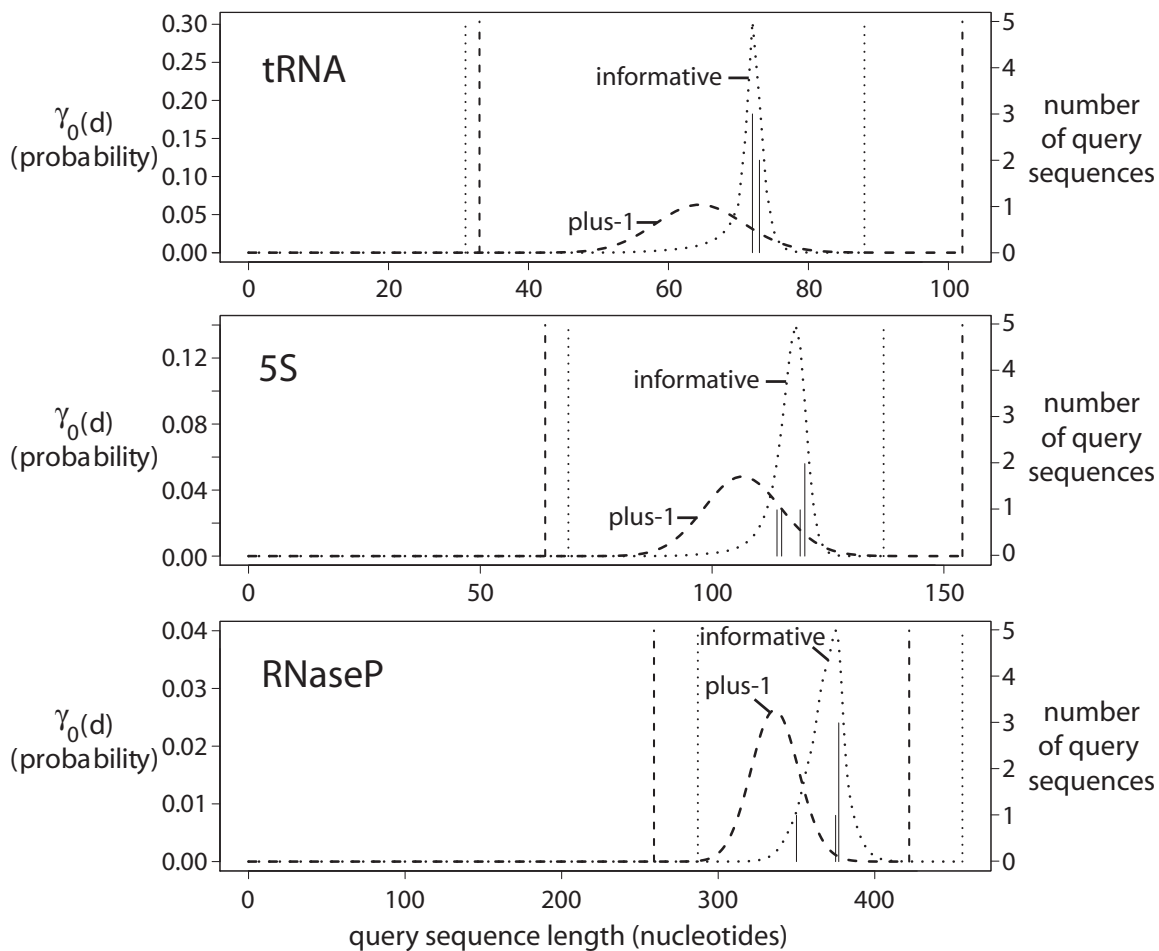


Figure 2: **Effect of transition priors on band calculation.** Predicted and actual target lengths are shown for three CMs built from alignments of five tRNA, 5S rRNA, and RNaseP sequences, which are about 75, 120, and 380 residues long, respectively. Solid vertical lines are histogram bars of the actual lengths of the query sequences in each alignment, corresponding with the right vertical axis labels. Dashed and dotted curves show QDB calculations for  $\gamma_0(d)$  for the root state of each model, for uninformative versus informative Dirichlet priors, respectively. Dashed and dotted vertical lines show the band bounds ( $d_{\min}(0)$  (left) and  $d_{\max}(0)$  (right)) derived from the  $\gamma_0(d)$  distributions using  $\beta = 10^{-7}$ . The uninformative plus-one prior results in consistent underprediction of target sequence lengths, with a broad distribution. The new informative priors produce tighter distributions that are centered on the actual subsequence lengths. We observe the same result for all other states (data not shown).

parameter estimation results in an improvement in the utility of QDB calculations, often yielding tighter, yet accurate subsequence length distributions, as illustrated by anecdotal example in Figure 2.

We also estimated informative mixture Dirichlet density priors for emission probabilities. Emission probabilities have no effect on QDB, but informative emission priors should improve sensitivity and specificity of CM searches, as they do for profile hidden Markov models [29, 31]. We collected filtered counts of aligned single-stranded residues and base-pairs from annotated ribosomal RNA alignments from four alignments in the 2002 version of the European Ribosomal RNA Database [32, 33]: large subunit rRNA (LSU), bacterial/archaeal/plastid small subunit rRNA (SSU-bap), eukaryotic SSU rRNA (SSU-euk), and mitochondrial SSU rRNA (SSU-mito). These alignments were filtered, removing sequences in which either less than 40% of the base paired positions are present or more than 5% of the nucleotides are ambiguous, and removing selected sequences based on single-linkage-clustering such that no two sequences in a filtered alignment were greater than 80% identical (in order to remove closely related sequences). Summary statistics for the filtered alignments and collected counts in the training dataset are given in Table 2. These data were used to estimate a nine-component Dirichlet mixture prior for base pairs, and an eight-component Dirichlet mixture prior for single stranded residues. (The decision to use nine and eight components was empirical, after benchmarking different choices.) The base pair prior is given in Table 3, and the singlet residue prior is given in Table 4.

The reason to use two different datasets to estimate transition versus emission priors is the following. Rfam provides many different structural RNA alignments, but of uneven quality and varying depth (number of sequences). The European rRNA database provides a small number of different RNA alignments, but of high quality and great depth. A transition prior training set should be maximally diverse, so as not to bias any transition types toward any particular RNA structure, so we used the 381 different Rfam alignments for transitions. Emission prior estimation, in contrast, improves with alignment depth and accuracy, but does not require broad structural diversity per se, so we used rRNA data for emissions.

Inspection of the Dirichlet  $\alpha$  parameters shows sensible trends. In the transition priors, transitions between main (consensus) states are now favored (higher  $\alpha$  values) relative to insertions and deletions. In the base pair emission mixture prior, all components favor Watson-Crick and G-U pairs, with different components preferring different proportions of pairs in a particular covarying aligned column (for instance, component 1 likes all four Watson-Crick pairs, component 2 describes covarying conservation of CG,UA,UG pairs, and component 3 specifically likes conserved CG pairs), and the marginal (mean)  $\alpha$  parameters pre-



		# filtered	# aln	# consensus	# consensus	base pair	SS
alignment	# seqs	seqs	columns	base pairs	SS columns	counts	counts
LSU	1551	139	7270	601	1532	65229	180558
SSU bap	12773	254	2653	421	680	97834	153565
SSU euk	7151	207	4558	407	959	72521	174260
SSU mito	1039	107	3791	216	524	19803	56510

Table 2: **Summary statistics for the dataset used for emission prior estimation.** “SS” = single-stranded.

for GC/CG pairs over AU/UA pairs. In the singlet emission mixture prior, some components are capturing strongly conserved residues (component 1 favors conserved U’s, for example) while other components favor more variation (components 4 and 5, for example), and the marginal  $\alpha$  parameters show a strong A bias, reflecting the known bias for adenine in single-stranded positions of structural RNAs (especially ribosomal RNAs).

In another step to increase sensitivity and specificity of the program, we adopted the “entropy weighting” technique described for profile HMMs [34] for estimating the total effective sequence number for an input query alignment. This is an *ad hoc* method for reducing the information content per position of a model, which helps a model that has been trained on closely related sequences to recognize distantly related homologues [35]. In entropy weighting, one reduces the total effective sequence number (which would normally be the actual number of sequences in the input alignment), thereby increasing the influence of the Dirichlet priors, flattening the transition and emission distributions, and reducing the overall information content, until a desired target entropy is achieved. We approximate a model’s entropy as the mean match state entropy, the summed entropy of all consensus match states (MATP\_MP, MATL\_ML, and MATR\_MR) divided by the consensus query length. The target entropy for INFERNAL is a free parameter, which we optimized on the benchmark described below to identify our default value of 1.46 bits.

## Benchmarking

To assess the effect of QDB, informative priors, and entropy weighting on the speed, sensitivity, and specificity of RNA similarity searches, we designed a benchmark based on the Rfam database [16]. The benchmark was designed so that we would test many RNA query/target pairs, with each query consisting of a given RNA sequence alignment, and each target consisting of a distantly related RNA homolog buried in a

component $i$	1	2	3	4	5	6	7	8	9
$q_i$	0.0305	0.0703	0.1185	0.1810	0.1888	0.1576	0.0417	0.0959	0.1156
$ \alpha $	14.3744	2.9920	26.2757	0.5342	4.2716	13.3232	33.8619	22.2258	33.1991

$ab$	mean $\alpha$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$	$\frac{\alpha_{ab}}{ \alpha }$
AA	0.0063	0.0398	0.0390	0.0011	0.0017	0.0005	0.0062	0.0064	0.0058	0.0002
AC	0.0092	0.0421	0.0176	0.0009	0.0152	0.0018	0.0125	0.0115	0.0051	0.0046
AG	0.0052	0.0381	0.0226	0.0046	0.0034	0.0008	0.0032	0.0040	0.0053	0.0001
AU	<b>0.1663</b>	<b>0.1092</b>	0.0864	0.0194	<b>0.2138</b>	<b>0.1464</b>	<b>0.2563</b>	<b>0.7360</b>	<b>0.1295</b>	0.0404
CA	0.0086	0.0412	0.0510	0.0054	0.0027	0.0044	0.0018	0.0030	0.0138	0.0002
CC	0.0038	0.0327	0.0115	0.0030	0.0001	0.0003	0.0036	0.0039	0.0035	0.0041
CG	<b>0.2412</b>	<b>0.1007</b>	<b>0.1392</b>	<b>0.8310</b>	<b>0.1359</b>	<b>0.3211</b>	0.0889	0.0340	<b>0.2870</b>	0.0147
CU	0.0066	0.0418	0.0172	0.0027	0.0104	0.0019	0.0045	0.0076	0.0052	0.0003
GA	0.0061	0.0362	0.0266	0.0002	0.0074	0.0002	0.0058	0.0045	0.0042	0.0021
GC	<b>0.2547</b>	<b>0.1299</b>	0.0544	0.0206	<b>0.1786</b>	<b>0.1613</b>	<b>0.4079</b>	0.0945	<b>0.1155</b>	<b>0.8858</b>
GG	0.0063	0.0327	0.0142	0.0045	0.0091	0.0005	0.0072	0.0023	0.0044	0.0030
GU	0.0567	0.0811	0.0412	0.0049	<b>0.1355</b>	0.0451	0.0668	0.0303	0.0356	0.0218
UA	<b>0.1571</b>	<b>0.1063</b>	<b>0.3085</b>	0.0672	<b>0.1856</b>	<b>0.2293</b>	0.0902	0.0363	<b>0.3108</b>	0.0151
UC	0.0063	0.0477	0.0263	0.0006	0.0048	0.0002	0.0056	0.0042	0.0060	0.0038
UG	0.0543	0.0746	<b>0.1054</b>	0.0317	0.0807	0.0814	0.0299	0.0120	0.0551	0.0032
UU	0.0114	0.0459	0.0389	0.0022	0.0151	0.0048	0.0098	0.0095	0.0133	0.0008

Table 3: **Parameters of the 9 component Dirichlet mixture emission prior for base pairs.**  $q_i$  = mixture coefficient for component  $i$ . Normalized  $\alpha$  values  $> 0.10$  are in bold face.

component $i$	1	2	3	4	5	6	7	8
$q_i$	0.0851	0.0159	0.1020	0.4160	0.0745	0.0554	0.1184	0.1327
$ \alpha $	15.4467	154.4640	180.2862	5.4562	0.2199	16.4089	13.4592	19.9059

$a$	mean $\alpha$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$	$\frac{\alpha_a}{ \alpha }$
A	<b>0.3951</b>	0.0373	<b>0.9961</b>	<b>0.9787</b>	<b>0.3109</b>	<b>0.3383</b>	0.0375	0.0864	<b>0.8247</b>
C	<b>0.1635</b>	0.0490	0.0015	0.0052	<b>0.2067</b>	<b>0.1782</b>	<b>0.8916</b>	0.0303	0.0493
G	<b>0.2041</b>	0.0220	0.0023	0.0072	<b>0.1751</b>	<b>0.2905</b>	0.0182	<b>0.8313</b>	0.0569
U	<b>0.2372</b>	<b>0.8917</b>	0.0000	0.0090	<b>0.3073</b>	<b>0.1930</b>	0.0527	0.0519	0.0691

Table 4: **Parameters of the 8 component Dirichlet mixture emission prior for singlets.**  $q_i$  = mixture coefficient for component  $i$ . Normalized  $\alpha$  values  $> 0.10$  are in bold faced type.

context of a random genome-like background sequence.

We started with seed alignments from Rfam version 7.0. In each alignment, sequences shorter than 70% the median length were removed. We clustered the sequences in each family by single-linkage-clustering by % identity (as calculated from the given Rfam alignment), then split the clusters such that the training set and test sequences satisfied three conditions: 1) no training/test sequence pair is more than 60% identical; 2) no test sequence pair is greater than 70% identical; 3) at least 5 sequences are in the training set. Fifty-one families satisfy these criteria (listed in Table 5), giving us 51 different query alignments (containing 5 to 1080 sequences each) and 450 total test sequences (from 1 to 66 per query). We embedded the test sequences in a one megabase “pseudo-genome” consisting of twenty 50 kilobase “chromosomes”, generated as independent, identically distributed (iid) random sequences with uniform base frequencies. The 450 test sequences were embedded into this sequence by replacement, by randomly choosing a chromosome, orientation, and start position, and disallowing overlaps between test sequences. The total length of the 450 test sequences is 101,855 nt, leaving 898,145 nt of random background sequence.

The benchmark proceeds by first building a CM for each query alignment, then searching the pseudo-genome with each CM in local alignment mode. All hits above a threshold of 8.0 in raw bit score for each of the 51 queries were sorted by score into 51 ranked family specific lists, as well as one ranked master list of all 51 sets of scores. Each hit is classified into one of three categories, “positive”, “ignore”, or “negative”. A “positive” is a hit that significantly overlaps with a true test sequence from the same family as the query. An “ignore” is a hit that significantly overlaps with a test sequence from a different family,

Rfam 7.0 ID	family name	# query	# test	avg len query	W	non-banded time	QDB ( $\beta = 10^{-7}$ )		QDB ( $\beta = 10^{-7}$ )			
							time	spd up	MER	FP	FN	thr
RF00177	SSU_rRNA_5	145	21	593	690	96.55	7.74	12.48	0	0	0	9.90
RF00024	Telomerase-vert	20	11	436	505	51.16	4.51	11.35	0	0	0	11.30
RF00011	RNaseP_bact.lb	30	1	366	441	33.36	3.77	8.86	0	0	0	11.31
RF00018	CsrB	8	1	351	403	28.75	2.28	12.61	0	0	0	12.98
RF00040	rne5	6	1	338	368	19.73	2.61	7.55	0	0	0	11.79
RF00023	tmRNA	19	40	334	463	24.53	2.51	9.77	11	0	11	11.20
RF00010	RNaseP_bact.a	233	1	332	514	33.84	3.32	10.19	0	0	0	12.61
RF00009	RNaseP_nuc	26	21	320	530	27.89	7.24	3.85	19	0	19	11.67
RF00017	SRP_euk.arch	28	21	303	328	14.77	2.75	5.38	6	0	6	10.40
RF00028	Intron_gpI	5	24	300	375	17.22	3.33	5.17	20	0	20	12.25
RF00373	RNaseP.arch	20	13	290	337	15.04	3.18	4.73	0	0	0	12.23
RF00030	RNase_MRP	18	3	284	394	19.77	3.16	6.26	3	0	3	12.46
RF00101	SraC_RyeA	6	1	250	278	9.49	1.55	6.12	0	0	0	11.88
RF00230	T-box	10	35	244	298	8.20	1.83	4.48	1	0	1	12.34
RF00448	IRES_EBNA	7	1	213	238	7.25	1.29	5.62	1	0	1	11.99
RF00012	U3	6	5	212	240	7.17	1.64	4.37	2	0	2	13.02
RF00174	Cobalamin	87	66	203	326	9.44	2.90	3.26	0	0	0	11.28
RF00004	U2	76	1	184	215	5.94	1.15	5.18	0	0	0	10.01
RF00234	glmS	8	3	181	303	7.98	1.87	4.27	0	0	0	11.22
RF00168	Lysine	33	17	180	223	5.71	1.48	3.87	0	0	0	15.98
RF00380	ykoK	35	3	168	192	4.33	1.25	3.45	0	0	0	13.10
RF00003	U1	46	6	159	184	4.14	0.91	4.56	0	0	0	11.24
RF00025	Telomerase-cil	10	2	157	188	3.88	1.02	3.79	2	0	2	13.97
RF00002	5_8S_rRNA	62	1	151	183	3.45	0.99	3.49	0	0	0	11.28
RF00379	ydaO_yuaA	31	4	147	227	4.38	1.66	2.64	0	0	0	12.25
RF00067	U15	9	3	146	178	2.69	1.00	2.69	0	0	0	11.11
RF00029	Intron_gpII	7	11	141	276	4.74	1.35	3.51	1	0	1	11.08
RF00015	U4	25	1	141	187	3.67	1.05	3.48	1	0	1	13.46
RF00096	U8	5	1	135	177	2.98	0.95	3.14	0	0	0	11.56
RF00080	yybP-ykoY	20	33	129	173	3.05	1.16	2.63	1	0	1	10.78
RF00114	S15	10	1	117	138	1.78	0.60	2.94	0	0	0	13.12
RF00020	U5	29	3	115	139	2.06	0.75	2.73	0	0	0	13.64
RF00059	THI	228	8	109	222	3.33	1.49	2.24	0	0	0	13.66
RF00504	gecT	109	5	102	199	2.36	1.44	1.64	0	0	0	13.40
RF00167	Purine	33	4	99	119	1.48	0.58	2.57	0	0	0	13.02
RF00169	SRP_bact	46	15	96	120	1.46	0.67	2.19	0	0	0	11.58
RF00055	snoZ37	5	1	94	117	1.14	0.54	2.10	1	0	1	13.96
RF00019	Y	15	1	94	128	1.41	0.75	1.89	1	0	1	14.25
RF00033	MicF	8	1	93	114	1.27	0.52	2.45	0	0	0	13.18
RF00213	snoR38	7	3	88	147	1.35	0.71	1.89	0	0	0	16.07
RF00054	U25	5	1	87	107	0.95	0.47	2.04	1	0	1	16.66
RF00206	U54	12	1	81	115	0.93	0.54	1.72	1	0	1	15.80
RF00104	mir-10	9	2	73	94	0.85	0.53	1.60	2	0	2	16.13
RF00005	tRNA	1080	19	73	127	1.35	0.49	2.77	5	1	4	12.62
RF00170	msr	5	3	70	112	0.85	0.46	1.88	3	0	3	13.49
RF00163	Hammerhead_1	65	1	68	232	1.59	0.87	1.82	0	0	0	16.16
RF00031	SECIS	11	24	64	87	0.69	0.43	1.58	13	2	11	14.58
RF00165	Corona_pk3	10	1	63	80	0.55	0.32	1.73	1	0	1	14.72
RF00066	U7	28	2	62	85	0.58	0.35	1.68	0	0	0	14.23
RF00008	Hammerhead_3	82	1	55	101	0.71	0.45	1.58	0	0	0	14.71
RF00037	IRE	36	1	28	45	0.17	0.12	1.38	1	0	1	14.98
MER statistics summed across all families									97	3	94	N/A
Summary MER statistics (using one threshold for all families)									113	2	111	16.38
average timing statistics						9.96	1.66	4.14				
total timing statistics						507.99	84.51	6.01				

Table 5: **Rfam benchmark families with timing and MER statistics.** “W” = window length, maximum size of a hit per family, calculated as  $\text{dmax}(0)$ . Running times for standard (non-banded) and QDB ( $\beta = 10^{-7}$ ) searches are given for each family, in CPU-hours per Mb. The “MER” score is the minimum number of false positives (“FP”) plus false negatives (“FN”) at any threshold; the “thr” column shows that score threshold for each family, in bits. In the row labeled “Summary MER statistics”, these are derived from a single score threshold in a ranked list of all hits across all families. All statistics are for INFERNAL version 0.71 in local alignment mode.

where “significantly overlap” means that the length of overlap between two sequences (either two hits, or one hit and one test sequence embedded in the pseudo-genome) is more than 50% the length of the shorter sequence. (Although it would be desirable to measure the false positive rate on nonhomologous structural RNAs, we cannot be sure that any given pair of Rfam families is truly nonhomologous. Like most sequence family databases, Rfam is clustered computationally, and more sensitive methods will reveal previously unsuspected relationships that should not be benchmarked as “false positives”.) A “negative” is a hit that is not a positive or an ignore. For any two negatives that significantly overlap, only the one with the better score is counted.

The minimum error rate (MER) (“equivalence score”) [36] was used as a measure of benchmark performance. The MER score is defined as the minimum sum of the false positives (negative hits above the threshold) and false negatives (true test sequences which have no positive hit above the threshold), at all possible choices of score threshold. The MER score is a combined measure of sensitivity and specificity, where a lower MER score is better. We calculate two kinds of MER scores. For a *family-specific* MER score, we choose a different optimal threshold in each of the 51 ranked lists, and for a *summary* MER score, we choose a single optimal threshold in the master list of all hits. The summary MER score is the more relevant measure of our current performance, because it demands a single query-independent bit score threshold for significance. A family-specific MER score reflects the performance that could be achieved if INFERNAL provided P-values (currently it reports only raw bit scores).

For comparison, BLASTN was also benchmarked on these data using a family-pairwise-search (FPS) procedure [37]. For each query alignment, each training sequence is used as a query sequence to search the pseudo-genome, all hits with an E-value of less than 1.0 were sorted by increasing E-value, and the lowest E-value positive hit to a given test sequence is counted.

Using this benchmark, we addressed several questions about QDB’s performance.

What is the best setting of the single QDB free parameter,  $\beta$ , which specifies how much probability mass to sacrifice? Figure 3 shows the average speedup per family and summary MER score as a function of varying  $\beta$ . There is no clear choice. The choice of  $\beta$  is a tradeoff of accuracy for speed. We chose a default of  $\beta = 10^{-7}$  as a reasonable value that obtains a modest speedup with minimal loss of accuracy.

How well does QDB accelerate CM searches? Figure 4 shows the time required for searching the 1 Mb benchmark target sequence with each of the 51 models, as a function of the average query RNA length. QDB reduces the average-case running time complexity of the CM search algorithm from  $LN^{2.36}$

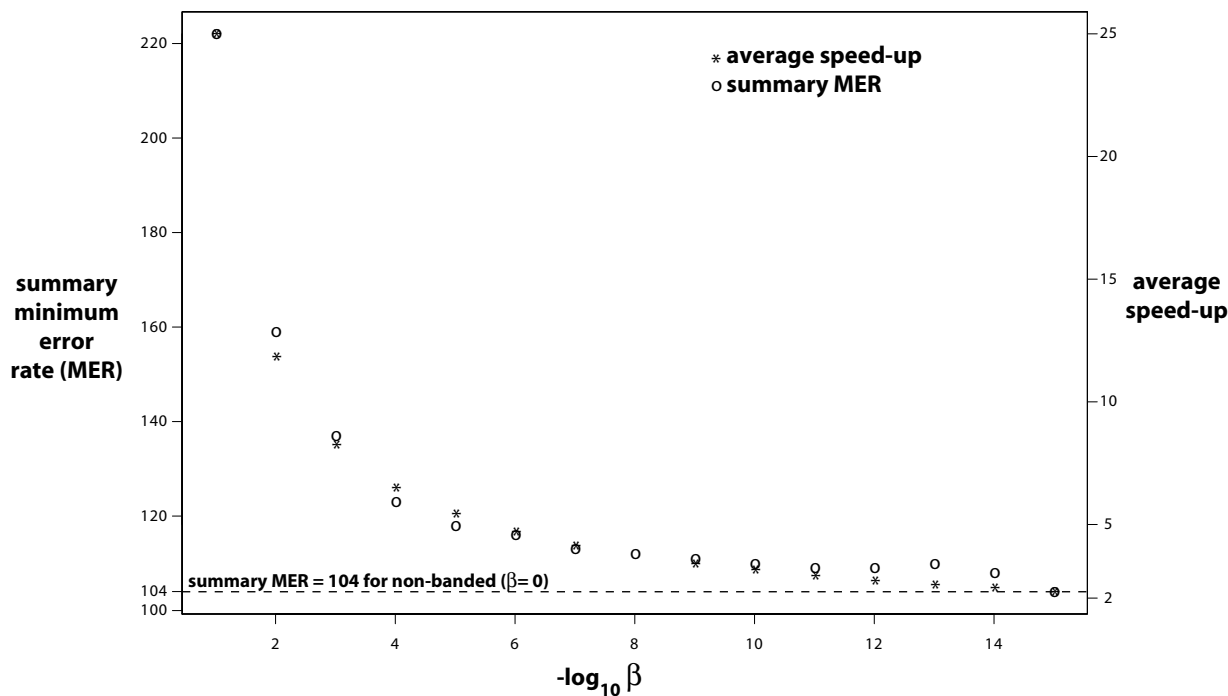


Figure 3: Effect of varying the  $\beta$  parameter on sensitivity, specificity, and speedup.

to  $LN^{1.32}$ . Observed accelerations relative to the standard algorithm range from 1.4-fold (for the IRE, iron response element) to 12.6-fold (for CsrB RNA), with an average speed-up per family of 4.1-fold. In total search time for the benchmark (sum of all 51 searches), the acceleration is six-fold, because large queries have disproportionate effect on the total time.

How much does QDB impact sensitivity and specificity? Optimal alignments are not guaranteed to lie within QDB's high-probability bands. This is expected to compromise sensitivity. The hope is that QDB's bands are sufficiently wide and accurate that the loss is negligible. Figure 5 shows ROC plots (sensitivity versus false positive rate) on the benchmark for the new version of INFERNAL (version 0.71) in standard versus QDB mode. These plots are nearly superposed, showing that the loss in accuracy is small at the default QDB setting of  $\beta = 10^{-7}$ .

How much do our changes in parameterization (the addition of informative Dirichlet priors and entropy weighting) improve sensitivity and specificity? Figure 5 shows that the new INFERNAL 0.71 is a large improvement over the previous INFERNAL version 0.55, independent of QDB. (On average, in this benchmark, INFERNAL 0.55 is no better than a family-pairwise-search with BLASTN.) Table 6 breaks this result down in more detail, showing summary and family-specific MER scores for a variety of combinations of

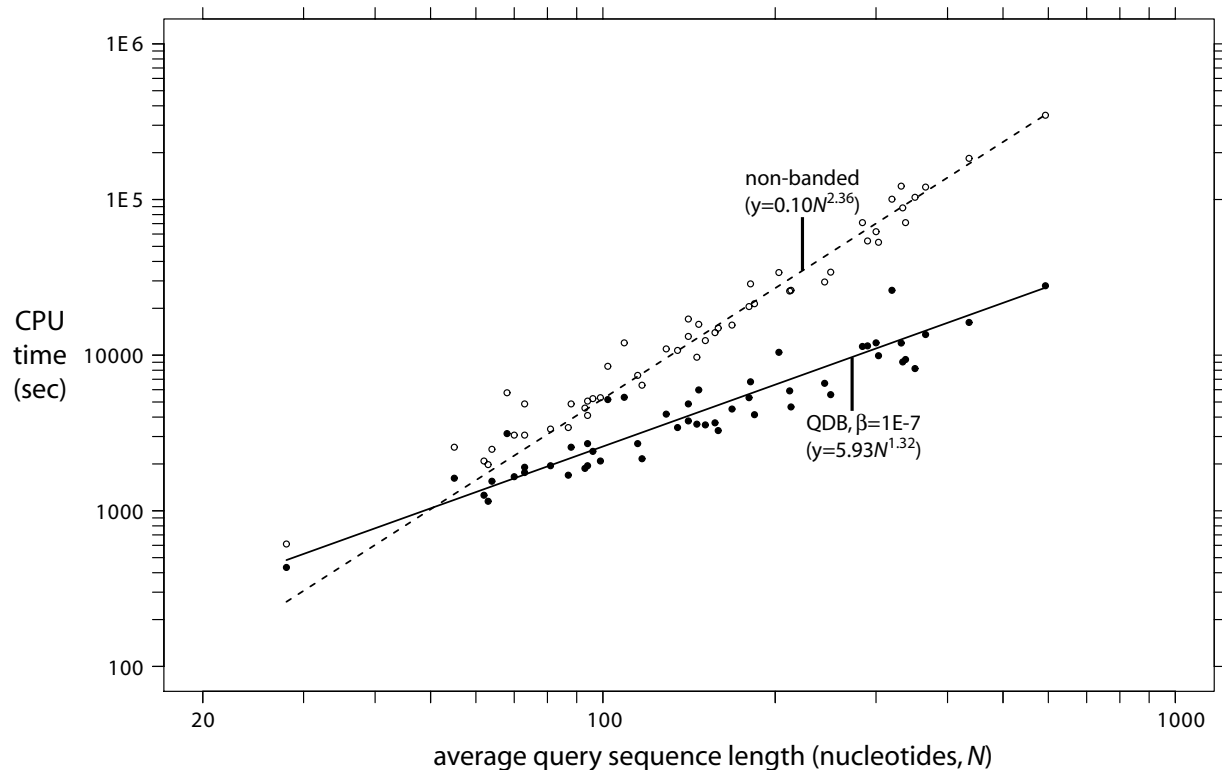


Figure 4: **CPU time required by CM searches with and without QDB.** The time required for searching the 1 Mb target pseudogenome with each of the 51 benchmark models is shown as a point, plotted on a log-log graph as a function of the average length of the RNA sequences in the query alignment; open circles are without QDB, and filled circles are with QDB (with the default  $\beta = 10^{-7}$ ). Lines represent fits to a power law ( $aN^b$ ), showing that for a fixed  $L = 1$  Mb target database size, the standard CYK algorithm empirically scales as  $N^{2.36}$  and the QDB algorithm scales as  $N^{1.32}$ . The apparent intersection of the linear fitted lines is deceptive. At small query lengths, run time is dominated by factors other than the CM alignment computation, such as i/o. QDB searches are always faster than non-banded searches even for synthetic tiny queries of less than 10 nt (data not shown).

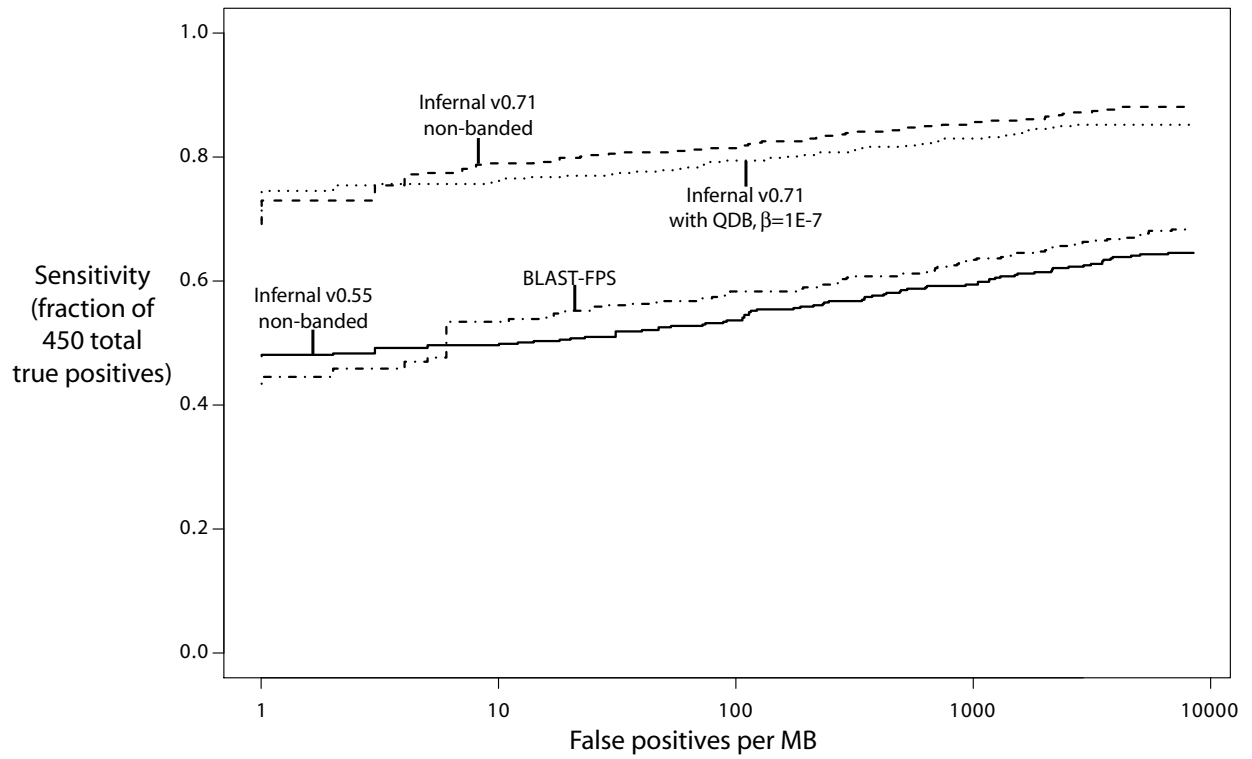


Figure 5: **ROC curves for the benchmark.** Plots are shown for the new INFERNAL 0.71 with and without QDB, for the old INFERNAL 0.55, and for family-pairwise-searches (FPS) with BLASTN.

prior, entropy weighting, and QDB. These results show that both informative priors and entropy weighting individually contributed large improvements in sensitivity and specificity.

program	prior	entropy (bits)	$\beta$	summary	family-specific
				MER	MER
BLASTN	-	-	-	216	188
INFERNAL 0.55	plus-1	-	-	232	180
INFERNAL 0.71	plus-1	-	-	216	190
INFERNAL 0.71	plus-1	1.46	-	208	191
INFERNAL 0.71	informative	-	-	179	163
INFERNAL 0.71	informative	1.46	-	104	89
INFERNAL 0.71	informative	1.46	$10^{-7}$	113	97

Table 6: **Rfam benchmark MER summary statistics.** “prior”: “plus-1” if uninformative Laplace plus-1 priors were used; “informative” if new Dirichlet priors were used. “entropy”: target model entropy in bits for entropy weighting; “-” if entropy weighting was not used. “ $\beta$ ”: tail probability loss for banded calculation used; “-” if search was non-banded. “summary MER”: MER across 51 benchmark families; “family-specific MER”: MER for each family, summed over all 51 families. Program versions: Row 1: WU-BLASTN-2.0MP -kap -W=7. For INFERNAL version 0.55, window length values ( $W$ ) were preset as calculated in version 0.71 with plus-1 priors.

## Discussion

CM searches take a long time, and this is the most limiting factor in using the INFERNAL software to identify RNA similarities. Prior to this work, INFERNAL 0.55 required 508 CPU-hours to search 51 models against just 1 megabase of sequence in our benchmarks (Table 5). Using QDB with  $\beta$  banding cutoffs that do not appreciably compromise sensitivity and specificity, INFERNAL 0.71 offers a six-fold speedup, performing the benchmark in 85 hours. Our eventual goal is to enable routine genome annotation of structural RNAs: to be able to search thousands of RNA models against complete genome sequences. A search of all 503 Rfam 7.0 models against the 3 GB human genome with INFERNAL 0.71 in QDB mode would take on the order of 300 CPU-years (down from 1800 with INFERNAL 0.55). We need to be able to do it in at most a few days, so we still need to increase CM search speed by five orders of magnitude. Thus, the QDB algorithm is a partial but certainly not complete solution to the problem. However, QDB combines synergistically with other acceleration techniques. Parallelization, on large clusters (though prohibitively expensive for all but a few centers) could give us further acceleration of three orders of magnitude. Software improvement (code optimization) will also contribute, but probably only about two-fold. Hardware improvements will contribute about two-fold per year or so so long as Moore’s law continues. Finally, QDB is complementary

to the filtering methods recently described by Weinberg and Ruzzo [19–21]. We view QDB as part of a growing suite of approaches that we can combine to accelerate INFERNAL.

Is it really worth burning all this CPU time in the first place? Do CM searches identify structural RNA homologies that other methods miss? Obviously we think so, but one would like to see convincing results. For large, diverse RNA families like tRNA, where a CM can be trained on over a thousand well-aligned sequences with a well-conserved consensus secondary structure, CM approaches have been quite powerful. The state of the art in large scale tRNA gene identification remains the CM-based program TRNASCANSE [1], and CMs were also used, for example, to discover the divergent tRNA for pyrrolysine, the “22nd amino acid” [38]. But Figure 5 shows that on average, over 51 more or less “typical” RNA families of various sizes and alignment quality, INFERNAL 0.55 was actually no better than doing a family-pairwise-search with BLASTN. Until recently, we have spent relatively little effort on how INFERNAL parameterizes its models, and relatively more on reducing its computational requirements [15], so previous versions of INFERNAL have performed best where naive parameterization works best: on very large, high-quality alignments of hundreds of sequences, which are atypical of many interesting homology search problems.

In this work, partly because the level of acceleration achieved by QDB is sensitive to transition parameterization, we have brought INFERNAL parameterization close to the state of the art in profile HMMs, by introducing mixture Dirichlet priors [29] and entropy weighting [34]. This resulted in a large improvement in the sensitivity and specificity of searches, as judged by our benchmark (Figure 5). The difference between INFERNAL and family-pairwise BLASTN now appears pronounced for average-case behavior, not just best-case behavior. However, while we trust our benchmarking to tell us when we have improved INFERNAL relative to previous versions of itself, we prefer to see independent benchmarks to gauge our performance relative to other software. Benchmarks by tool developers are notoriously biased, and however honest we may try to be, it is difficult to entirely avoid such biases. For one thing, establishing an internal benchmark for ongoing code development creates an insidious form of training on the test set, because we accept code changes that improve benchmark performance. Currently we are aware of only one independent benchmark to date, BRaliBase III (Freyhult, Bollback, and Gardner, unpublished; available at <http://www.binf.ku.dk/~pgardner/bralibase/bralibase3/>). BRaliBase III consists of many different query alignments of 5 or 20 RNA sequences, drawn from three different RNA families (U5, 5S rRNA, and tRNA). These authors’ results broadly confirm our internal observations: while INFERNAL 0.55 showed mediocre performance compared to BLASTN and several other tools, a recent version of

INFERNAL stood out as a superior method for RNA similarity search.

Nonetheless, though INFERNAL 0.71 shows large improvements in speed, sensitivity, and specificity over previous versions, there are numerous areas where we need to improve further.

A significant gap in our current implementation is that INFERNAL reports only raw bit scores, and does not yet report expectation values (E-values). CM local alignment scores empirically follow a Gumbel (extreme value) distribution [25], just as local sequence alignment scores do [39], so there are no technical hurdles in implementing E-values. This will be an immediate focus for the next version of INFERNAL. E-value calculations not only have the effect of reporting statistical significance (more meaningful to a user than a raw bit score), but they also normalize each family's score distribution into a more consistent overall rank order, because different query models exhibit different null distributions (particularly in the location parameter of the Gumbel distribution). We therefore expect E-values to contribute a large increase in performance whenever a single family-independent threshold is set. Table 6 roughly illustrates the expected gain, by showing the large difference between summary MER scores and family-specific MER scores.

Parameterization of both CMs and profile HMMs remains problematic, because these methods continue to assume that training sequences are statistically independent, when in fact they are related (often strongly so) by phylogeny. Methods like sequence weighting and entropy weighting do help, but they are *ad hoc* hacks: unsatisfying and unlikely to be optimal. Even mixture Dirichlet priors, though they appear to be mathematically sophisticated, fundamentally assume that observed counts are drawn as independent multinomial samples, and therefore the use of Dirichlet priors is fundamentally flawed. Probabilistic phylogenetic inference methodology needs to be integrated with profile search methods, but this area continues to be stymied by a lack of understanding of how to model insertions and deletions efficiently in a phylogenetic probability model.

Finally, QDB is not the only algorithmic acceleration method we can envision. Michael Brown described a complementary banding method to accelerate his SCFG-based RNACAD ribosomal RNA alignment software [23], in which he uses profile HMM based sequence alignment to the target to determine bands where the more rigorous SCFG-based alignment should fall (because some regions of the alignment are well-determined based solely on sequence alignment). The gapped BLAST algorithm (seed word hits, ungapped hit extension, and banded dynamic programming) can conceivably be extended from two-dimensional sequence alignment to three-dimensional CM dynamic programming lattices. Developing such algorithms – and incorporating them into a widely useful, freely available codebase – are priorities for us.

## **Materials and Methods**

The version and options used for BLAST in our benchmark are `WU-BLASTN-2.0MP --kap -W=7`. For INFERNAL, versions 0.55 and 0.71 were used as indicated. The complete INFERNAL software package, including documentation and the Rfam-based benchmark described here, may be downloaded from <http://infernal.janelia.org>. It is developed on GNU/Linux operating systems but should be portable to any POSIX-compliant operating system, including Mac OS/X. It is freely licensed under the GNU General Public License.

## **Acknowledgements**

We thank Elena Rivas for critical comments on the manuscript.

## **Funding**

EPN gratefully acknowledges financial support from an NIH NHGRI Institutional Training Grant in Genomic Science, T32-HG000045. Work in SRE's lab in Washington University in St. Louis, where this work was initiated, was supported by NIH NHGRI RO1-HG01363, HHMI, and Alvin Goldfarb.

## References

- [1] Lowe TM, Eddy SR (1997) tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucl Acids Res* 25:955–964.
- [2] Laslett D, Canback B (2004) ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. *Nucl Acids Res* 32:11–16.
- [3] Lowe TM, Eddy SR (1999) A computational screen for methylation guide snoRNAs in yeast. *Science* 283:1168–1171.
- [4] Schattner P, Barberan-Soler S, Lowe TM (2006) A computational screen for mammalian pseudouridylation guide H/ACA RNAs. *RNA* 12:15–25.
- [5] Lai EC, Tomancak P, Williams RW, Rubin GM (2003) Computational identification of drosophila microRNA genes. *Genome Biol* 4:R42.
- [6] Lim LP, Glasner ME, Yekta S, Burge CB, Bartel DP (2003) Vertebrate microRNA genes. *Science* 299:1540.
- [7] Regalia M, Rosenblad MA, Samuelsson T (2002) Prediction of signal recognition particle RNA genes. *Nucl Acids Res* 30:3368–3377.
- [8] Ermolaeva MD, Khalak HG, White O, Smith HO, Salzberg SL (2000) Prediction of transcription terminators in bacterial genomes. *J Mol Biol* 301:27–33.
- [9] Macke TJ, Ecker DJ, Gutell RR, Gautheret D, Case DA, et al. (2001) RNAMotif, an RNA secondary structure definition and search algorithm. *NAR* 29:4724–4735.
- [10] Gautheret D, Lambert A (2001) Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles. *J Mol Biol* 313:1003–1011.
- [11] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, et al. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl Acids Res* 25:3389–3402.
- [12] Sakakibara Y, Brown M, Hughey R, Mian IS, Sjolander K, et al. (1994) Stochastic context-free grammars for tRNA modeling. *Nucl Acids Res* 22:5112–5120.

- [13] Durbin R, Eddy SR, Krogh A, Mitchison GJ (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge UK: Cambridge University Press.
- [14] Eddy SR, Durbin R (1994) RNA sequence analysis using covariance models. *Nucl Acids Res* 22:2079–2088.
- [15] Eddy SR (2002) A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics* 3:18.
- [16] Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy SR, et al. (2005) Rfam: Annotating non-coding RNAs in complete genomes. *Nucl Acids Res* 33:D121–D141.
- [17] Sonnhammer ELL, Eddy SR, Birney E, Bateman A, Durbin R (1998) Pfam: Multiple sequence alignments and HMM-profiles of protein domains. *Nucl Acids Res* 26:320–322.
- [18] Finn RD, Mistry J, Schuster-Bockler B, Griffiths-Jones S, Hollich V, et al. (2006) Pfam: Clans, web tools and services. *Nucl Acids Res* 34:D247–D251.
- [19] Weinberg Z, Ruzzo WL (2004) Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics* 20 Suppl. 1:I334–I341.
- [20] Weinberg Z, Ruzzo WL (2004) Faster genome annotation of non-coding rna families without loss of accuracy. *RECOMB '04* :243–251.
- [21] Weinberg Z, Ruzzo WL (2006) Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics* 22:35–39.
- [22] Brudno M, Do CB, Cooper GM, Kim MF, Green EDDENCSP, et al. (2003) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res* 13:721–731.
- [23] Brown MP (2000) Small subunit ribosomal RNA modeling using stochastic context-free grammars. *Proc Int Conf on Intelligent Systems in Molecular Biology* 8:57–66.
- [24] Eddy SR (2003). The Infernal user's guide. [<http://infernal.wustl.edu/>].
- [25] Klein RJ, Eddy SR (2003) RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics* 4:44.

- [26] 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.
- [27] Younger DH (1967) Recognition and parsing of context-free languages in time  $n^3$ . *Information and Control* 10:189–208.
- [28] Hopcroft JE, Ullman JD (1979) *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.
- [29] Sjölander K, Karplus K, Brown M, Hughey R, Krogh A, et al. (1996) Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *Comput Applic Biosci* 12:327–345.
- [30] Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1993) *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.
- [31] Brown M, Hughey R, Krogh A, Mian IS, Sjolander K, et al. (1993) Using dirichlet mixture priors to derive hidden Markov models for protein families. In: Hunter L, Searls D, Shavlik J, editors, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*. Menlo Park, CA: AAAI, pp. 47–55.
- [32] Wuyts J, Rijk PD, de Peer YV, Winkelmanns T, Wachter RD (2001) The European large subunit ribosomal RNA database. *Nucl Acids Res* 29:175–177.
- [33] Wuyts J, de Peer YV, Winkelmanns T, Wachter RD (2002) The European database on small subunit ribosomal RNA. *Nucl Acids Res* 30:183–185.
- [34] Karplus K, Barrett C, Hughey R (1998) Hidden Markov models for detecting remote protein homologies. *Bioinformatics* 14:846–856.
- [35] Altschul SF (1991) Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol* 219:555–565.
- [36] Pearson WR (1995) Comparison of methods for searching protein sequence databases. *Protein Sci* 4:1145–1160.
- [37] Grundy WN (1998) Homology detection via family pairwise search. *J Comput Biol* 5:479–491.

- [38] Srinivasan G, James CM, Krzycki JA (2002) Pyrrolysine encoded by UAG in archaea: Charging of a UAG-decoding specialized tRNA. *Science* 296:1459–1462.
- [39] Karlin S, Altschul SF (1993) Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc Natl Acad Sci USA* 90:5873–5877.