

FastR: Fast database search tool for non-coding RNA

Vineet Bafna and Shaojie Zhang
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114
{vbafna, shzhang}@cs.ucsd.edu

Abstract

The discovery of novel non-coding RNAs has been among the most exciting recent developments in Biology. Yet, many more remain undiscovered. It has been hypothesized that there is in fact an abundance of functional non-coding RNA (ncRNA) with various catalytic and regulatory functions. Computational methods tailored specifically for ncRNA are being actively developed. As the inherent signal for ncRNA is weaker than that for protein coding genes, comparative methods offer the most promising approach, and are the subject of our research. We consider the following problem: Given an RNA sequence with a known secondary structure, efficiently compute all structural homologs (computed as a function of sequence and structural similarity) in a genomic database. Our approach, based on structural filters that eliminate a large portion of the database, while retaining the true homologs allows us to search a typical bacterial database in minutes on a standard PC, with high sensitivity and specificity. This is two orders of magnitude better than current available software for the problem.

Keywords: non-coding RNA genes, RNA, database search, filtration, dynamic programming, sequence alignment.

1. Introduction

“Structural genes encode proteins, and regulatory genes produce ncRNA” - Jacob F. & Monod J., 1961.

The recent publications of the genomic sequence of large eukaryotes [11, 20] are a landmark for Molecular Biology. The whole genome sequence is a natural starting point for many investigations, and these papers contain a tantalizing glimpse into the biology of the future. One of the more surprising discoveries of this project was the relatively small

number of genes that were found. Many explanations have been given, but the one that intrigues us is the notion that most of the genes that we were looking for were *protein coding* genes. Could it be that many cellular functions are carried out by non-coding genes (genes that are transcribed into functional RNA, which is not translated [6])?

The idea is not without merit. Novel families of functional RNA are continually being discovered. The discovery of endogenous small interfering RNA (RNAi) and other newer families of RNA has generated a lot of excitement in the scientific community. Targeted search for other non-coding RNA (ncRNA) [1, 13, 14] has led to startling discoveries of novel sub-families of ncRNA. It is clear that the prevalence of ncRNA genes has been grossly underestimated. It is now being hypothesized that there is in fact an *abundance* of undiscovered, functional ncRNA with various catalytic and regulatory functions (The Modern RNA world [6]). This is understandable, given that computational tools for finding nc-genes are not as advanced as those for protein coding genes. Building effective tools for discovering ncRNA will fulfill an *unmet need* for biology.

Various computational approaches to detecting noncoding genes are being tried. Some of these are attempts at *de novo* prediction, looking for signals that might suggest a functional RNA in the molecule. The most promising approach seemed to be the use of secondary structure as a signal [4, 12] to discover RNA. However, recent reports [16, 21] have concluded that the signal is not strong enough to detect RNA, and random sequence with a high GC composition, or with a *di-nucleotide* composition similar to a true RNA sequence usually allow folding into energetically favorable secondary structures. Other *de novo* approaches including looking for the transcription start and similar signals, but have had limited success. The consensus is that the ncRNA signals in a genome are not as strong as the signals for protein coding genes. Therefore, the natural approach to the problem is based on comparative methods. Methods based on similarity have been published earlier [15]. These are usually tailored to finding homologs of a

specific RNA, such as tRNA. Recently, Klein and Eddy [10] developed a tool, RSEARCH, for searching a database with a query RNA molecule. The method depends upon existing algorithms for computing alignments between an RNA sequence and substrings of a database, where the alignment score is a function of sequence and structural similarity. Known algorithms for computing such alignments are computationally intensive (approximately $O(mw^2n)$, where m is the length of the query sequence, n is the length of the database sequence, and w is the maximum length of a database substring that is aligned to the query). The running time is expressed better by parameterizing on the elements of the RNA secondary structure, to be described in section 3. Not surprisingly, RSEARCH is slow to use. For a test run on an Intel/linux PC with 2.8GHz, 1Gb memory, a microbial database of size 1.67M, and a query 5S rRNA sequence, the program took over 6.5 hrs. to run. This makes it impractical when either the query or the database is large.

In this paper, we describe *FastR*, an efficient database search tool for ncRNA. An analogy can be drawn from fast search tools (BLAST/FASTA) for DNA and Protein sequences that has made database searching practical. The speed and effectiveness of BLAST in particular has contributed in large measure to the exponential growth of sequence databases, and the use of database search as an accepted method for finding novel DNA/protein homologs. By proposing *FastR*, which includes novel ideas for RNA structure filtering and alignment, we hope to do the same for ncRNA. As an example, *FastR* reduces the compute time of the previously mentioned query to 103s.

2. Designing Filters

Key to our efficient searching for homologs is the notion of *filters* for RNA structure that allow a large fraction of the database to be discarded while retaining all of the homologs. The obvious question is whether sequence similarity with the query string is sufficient to get an initial set of candidate regions. To test this, we queried the whole genome of *A. pernix* (GenBank NC_000854.1) with an Asn-tRNA sequence. With default parameters, Blastn selected 4 hits with an E-value < 0.001 , and 24 hits with E-value < 10 . 3 of the 4, and 10 of the 24 matched the 43 hits produced by RSEARCH. Also, most of the computed alignments were less than 20bp in length. From this, and similar tests not included here, we do not anticipate a tool based on sequence similarity to be effective in finding RNA homologs. Therefore, we turn to the secondary structure of the query RNA sequence as the basis for our filter design. We will continue to use sequence similarity in computing the final alignments.

Instead of working with the standard formalisms such as Correlation Models, and Stochastic Context Free Gram-

mars for RNA structure [5], we turn to a loose description, but one that still captures the essential details. As shown in Fig. 1, the secondary structure has a tree like shape and can be decomposed into various loops (Interior loops, bulges, multi-loops) and stack regions. Each stem in this tree contains energetically favorable stacked base-pairs. In the 'stretched' view (see Fig. 1(b)), a stack of length k corresponds to a pair of complementary strings. These stacked base-pair units form the basis of our filters, described formally below.

Consider a nucleotide string s , with $|s| = n$. A pair of indices (i, j) , $i < j$ forms a (k, w) -bp_unit, if $(j - i) \leq w$, and $s[i \dots i + k - 1]$, and $s[j \dots j + k - 1]$ can form a base-pair stack (see Fig. 1(b)). Recall that $s[i \dots i + k - 1]$, and $s[j \dots j + k - 1]$ form a base-pair stack if $s[i + u]$ and $s[j + k - 1 - u]$ are *complementary* for all $0 \leq u < k$. A simple filter choice for an RNA structure is the set of all starting positions i which contain a (k, w) -bp_unit for appropriately chosen k and w . Let p be the probability that a pair of randomly chosen bases is complementary. The probability that a pair of indices (i, j) with $(j - i) \leq w$ forms a (k, w) -bp_unit is p^k . Define $X_{i,j}$ as the indicator variable with $X_{i,j} = 1$ if and only if (i, j) forms a (k, w) -bp_unit. Using linearity of expectation, the expected number of hits in a random string of length n is

$$E\left(\sum_{i=1}^n \sum_{j=i+k}^{i+w} X_{i,j}\right) = \sum_{i=1}^n \sum_{j=i+k}^{i+w} E(X_{i,j}) \leq nwp^k$$

See Table 1 for the expected number of hits per starting position ($\approx wp^k$). Obviously, for large k and small w , even this simple filter can be quite powerful. We assume that in an RNA structure, the base-pairing is limited to the Watson-Crick base pairs ($A \leftrightarrow U$, $C \leftrightarrow G$), and the wobble base-pair ($G \leftrightarrow U$). For a randomly and uniformly chosen pair of bases, the probability p of pairing is $p = \frac{3}{8}$. As an example, typical tRNA structures have a clover-leaf shape with the outermost stem having a 7 base-pair stack separated by about 70 bases. The $(7, 70)$ filter would eliminate over 90% of the starting positions from consideration. In fact, we can do better as this base-pair unit is in fact separated by at least 50 bases in all tRNA, therefore making w effectively 20 ($50 \leq w \leq 70$), eliminating 98% of the starting positions.

Design Criteria for Filters

We will use the (k, w) -bp_unit as the basis for our filter design. However, we need to design more sophisticated filters as indels may sometimes disrupt base-pair stacks (decreasing the effective value of k), and variability in separation may increase the effective value of w . As many different filters are possible, we set forth some design criteria that

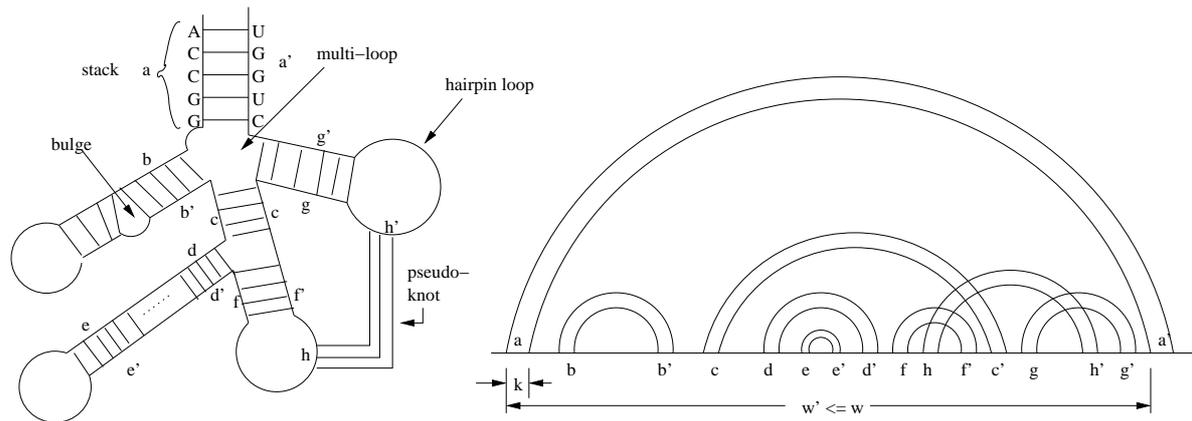


Figure 1. (a) An RNA structure with various structural elements including stacked base-pairs, bulges, hairpin, and multi-loops. (b) An alternative view. The set of bases in (a, a') forms a (k, w) -bp_unit.

$w \setminus k$	4	5	6	7	8	9	10
20	0.3955	0.1483	0.0556	0.0208	0.0078	0.0029	0.001
40	0.791	0.2966	0.1112	0.0417	0.0156	0.0058	0.0021
60	1.1865	0.4449	0.1668	0.0625	0.0234	0.0087	0.0032
80	1.582	0.5932	0.2224	0.0834	0.0312	0.0117	0.0043
100	1.9775	0.7415	0.278	0.1042	0.0391	0.0146	0.0054

Table 1. Expected number of hits in a random string in a (k, w) -filter.

should be used to evaluate different designs. In the following sub-section we describe a few filters that are effective, but we hope that these design criteria will help spur research in this area. Any good filter should meet the following criteria:

Generality: The filters must be general, and *simply* described, so as to be applicable (with appropriate parameter tuning) to every ncRNA family.

Efficiency: The time to filter should be no more than the time to align and score the filtered hits, and preferably as small as possible.

Sensitivity: Sensitivity is described as the fraction of all members of the ncRNA family that is admitted by the filter, and should be as close to 1 as possible. It may be acceptable to work with lower sensitivities, for example, to look for members in a sub-family.

Specificity: Defined here as the expected number of base-pairs per hit in a random database. It should be as large as possible.

We propose the following *Nested and Multiloop* filters:

(k, w, d) -bp_units: These are positions (i, j) such $j - i \leq w$, and there exists a string s' , such that the edit dis-

tance between s' and $s[i \dots i + k - 1]$ is $\leq d$, and s' forms a base-pair stack with $s[j \dots j + k - 1]$. By definition, a $(k, w, 0)$ -bp_unit is the same as (k, w) -bp_unit. In practice, we will work with low values of d , perhaps $d = 1$. For ease of exposition, we will drop the parameter d in the rest of the proposal. Unless stated otherwise, the arguments should work for various values of d . Another way to restrict bp_units is to limit the number of wobble base-pairs in a unit. In our filters, we restrict that number to 2 which seems like an appropriate specificity/sensitivity trade-off. See Section 4 for details.

Nested bp_units: Considering the RNA secondary structure as a tree, and going depth first down a path (see Fig. 1), we have many nested (k, w) -bp_units. The term (k, w, l) -nested_bp_unit is used to define l (k, w) -bp_units in a nested configuration. For example, in Fig. 1, the configuration $(a, a'), (c, c'), (d, d'), (e, e')$, is a $(k, w, 4)$ -nested_bp_unit. A (k, w, l) -nested_bp_unit may be *overlapping* if the different (k, w) -bp_units are allowed to overlap, and *non-overlapping* otherwise. Overlapping nested bp_units are another useful

way of dealing with indels in the homolog structure.

Parallel and Multiloop bp_units: Yet another way of looking at RNA structural elements is to locate non-nested, non-overlapping (k, w) -bp_units. We define a (k, w, l) parallel_bp_unit as a configuration of l non-nested, non-overlapping (k, w) -bp_units. This definition can be extended to a multiloop_bp_unit. A (k, w, l) -multiloop_bp_unit is a configuration in which a $(k, w, l - 1)$ -parallel_bp_unit is enclosed by a (k, w) -bp_unit. The units (b, b') , (d, d') , (f, f') , and (g, g') in Fig. 1 form a $(k, w, 4)$ -parallel_bp_unit. Correspondingly, $\{(a, a'), (b, b'), (d, d'), (f, f'), (g, g')\}$ is a $(k, w, 5)$ -multiloop_bp_unit.

The nested, parallel, and multiloop bp_units are all generalizations of the (k, w) -bp_unit, and therefore applicable to all families of ncRNA. There are conserved structural elements in every ncRNA family that enforce the correct folding, so it should be possible to find multiloop and nested structures with high sensitivity. Also, the simple description allows us to compute specificity using combinatorial arguments. To increase the specificity of these filters, we need to extend the design to include distance constraints (number of base-pairs) in between the various (k, w) bp_units. For a filter with l (k, w) bp_units, there are $2l$ substrings of length k each with $2l - 1$ distances between adjacent substrings. To this, we add an additional distance between the first and the last substring, and we have a vector of $2l$ distances. We constrain the distances by a $2l$ -dimensional vector \vec{w} containing the allowed ranges for each of these distances. Choose w_0 to be the range of distances between the first and last substring, and $w_j, j > 1$ to be the range of distances in the substrings ordered from left to right. A (multiloop/nested) filter satisfying these constraints is a (k, \vec{w}, l) -filter. Note that (k, w, l) -multiloop_bp_unit can be redefined by choosing \vec{w} such that $w_j = (0, w)$ for all j . $(4, \vec{w}, 4)$ -multiloop_bp_unit for tRNA with appropriate distance constraints is shown in Fig. 2.

Specificity and Sensitivity

Consider the earlier model of a randomly generated database with all nucleotides being equally likely. In order to understand specificity of a filter, we pose the following combinatorial problem: *What is the probability that an arbitrary position in the random database is the start of a (k, \vec{w}, l) -multiloop_bp_unit?* This question is not easy because of the various dependencies between overlapping units, so we approach it indirectly. Consider a $2l$ dimensional vector \vec{v} . If the distances in \vec{v} are within the range specified by \vec{w} , then \vec{v} denotes a configuration of a (k, \vec{w}, l) -multiloop_bp_unit obtained by fixing the $2l$ positions of the

l (k, w) -bp_units using distances in \vec{v} . The probability of occurrence of an arbitrary configuration is exactly p^{kl} . For an arbitrary starting position, and a configuration \vec{v} , define an indicator variable

$$X_{\vec{v}} = \begin{cases} 1 & \text{if a } (k, \vec{w}, l)\text{-multiloop_bp_unit occurs} \\ & \text{with configuration } \vec{v}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Let $Y = \sum_{\vec{v}} X_{\vec{v}}$. We are interested in computing $Pr[Y > 0]$. By linearity of expectation, $E(Y) = \sum_{\vec{v}} E(X_{\vec{v}}) = n_{k, \vec{w}, l} p^{kl}$, where $n_{k, \vec{w}, l}$ is number of possible configurations of a (k, \vec{w}, l) -multiloop_bp_unit. $n_{k, \vec{w}, l}$ can be computed using standard combinatorial arguments. We consider the following extreme cases for multiloop_bp_unit. Similar arguments will work for the nested bp_units.

Theorem 1: Let $0 \leq w_j \leq w$ for all j . Then, $n_{k, \vec{w}, l} = \binom{w - 2(k-1)l - 1}{2l - 1}$.

Proof:

In this extreme case all w_j ($i \leq j \leq l$) have the same constraints. We know that every distinct configuration of a (k, \vec{w}, l) -multiloop_bp_unit is obtained by selecting a start position of each of the $2l$ k -mers. Let each (k, w) bp_unit's two substrings (which form a base-pair stack) represented by two '1's, and let all unpaired bases represented by '0's. We can see there are $2l$ '1's and $(w - 2kl)$ '0's. The number of possible configurations of a (k, \vec{w}, l) -multiloop_bp_unit is equivalent to the number of configurations of a string formed by $2l$ '1's and $(w - 2kl)$ '0's with first letter being '1' (since the starting position must be part of a (k, w) bp_unit). Therefore, the number of possible configurations is $\binom{w - 2kl + 2l - 1}{2l - 1}$. \square

Theorem 2: Let $0 \leq w_0 \leq \infty$, and for all $j > 0$, let $0 \leq w_j < x$. Then, $n_{k, \vec{w}, l} = x^{2l - 1}$.

Proof:

In this extreme case, we only have $(2l - 1)$ distance constraints between any two adjacent substrings. Since the starting position of first substring is fixed and for all other $(2l - 1)$ starting positions each has x choices, The total number of possible configurations of a (k, \vec{w}, l) -multiloop_bp_unit is $x^{2l - 1}$. \square

Ideally, we choose the distance constraints so that $n_{k, \vec{w}, l} p^{kl} \ll 1$. For those values, we can use the Markov inequality ($Pr[Y > 0] < E(Y)$) to get the desired bound. For higher values of $E(Y)$, we need other techniques to bound the probability, which are not discussed in the paper. However, it is clear from Theorems 1 and 2 that in order to filter with high specificity, we will need to limit the number of possible configurations by tightening the distance constraints. Further increases in specificity are obtained by using intersections of nested and multiloop_bp_units. In section 4, we

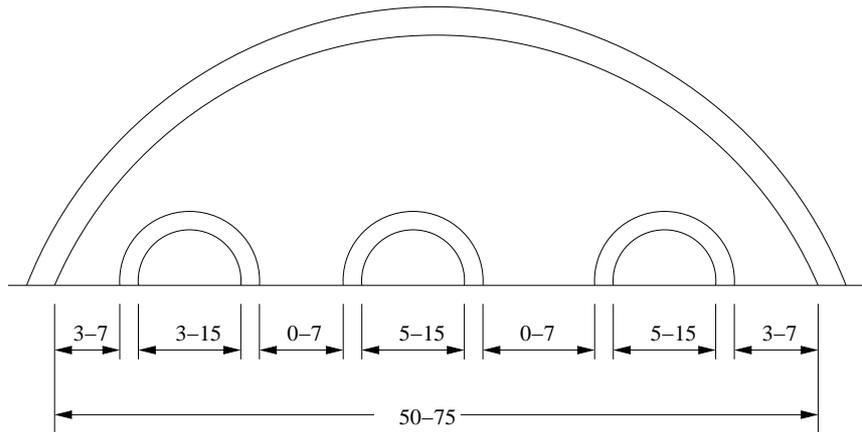


Figure 2. A multiloop filter for tRNA with distance constraints.

describe our filtering results on various test cases. Informally, making a filter restrictive increases specificity at the cost of sensitivity. One problem that we do not address in this paper is optimal filter design. Formally:

Optimal multiloop_bp_unit filter design:

Input: A family \mathcal{R} of ncRNA sequences with known/inferred structure, and parameter $0 < \alpha < 1$.

Output: Choose a k, \vec{w}, l -multiloop_bp_unit that maximizes $kl - \log_{\frac{1}{p}} n_{k, \vec{w}, l}$ (minimizes $n_{k, \vec{w}, l} p^{kl}$) while admitting an α fraction of \mathcal{R} .

In this article, we choose optimal filters empirically by trying different parameters. Algorithms for choosing optimal filters will be described elsewhere.

Efficiency

What fraction of the computation time is devoted to filtering? We use a combination of string matching and dynamic programming techniques [8] to compute filters efficiently.

1. **Hash:** Build a hash table to compute all k -mer positions in the database. The time taken is $O(m)$, where m is the size of the database.
2. **(k, w) -bp_units:** Let s_i denote the k -mer at an arbitrary position i in the database. For each s_i , compute a neighborhood $N(s_i)$ of all 'complementary' k -mers. In general, any k -mer that can form an energetically favorable stack with s should be in $N(s_k)$. In our current implementation, we do not allow indels, and allow at most 2 G-U pairs. This will be generalized after some experimentation. Using the hash table compute all positions j such that $s_j \in N(s_i)$, and $j - i$ satisfies distance constraints. Add (i, j) to a list of (k, w) -bp_units.

With appropriate data structures, the time taken is linear in the number of (k, w) -bp_units.

3. **Filters:** Note that multiloop and nested filters are combinations of (k, w) -bp_units. We scan the database with a moving window of size w . An 'active' list of (k, w) -bp_units within the window is maintained, and a dynamic programming technique is used to compute filters from this list. The total computation is bounded by $O(m_k w)$, where m_k is the number of (k, w) -bp_units. Typically $m_k < \frac{m}{w}$.

Note that we use a general definition of Neighborhood. While our current implementation does not allow gaps in pairing, they can be included simply by modifying the Neighborhood computation and hashing functions. We will incorporate them after some tests to see their effect on accuracy and efficiency. In Section 4, we show the current filter time is a few seconds per Mb of sequence, which is easily dominated by the time for computing alignments. Also, the filters are very effective in eliminating a large fraction of the database while retaining most of the true hits.

3. Computing RNA Sequence Alignment

The filtered database substrings must be structurally aligned to the query to identify true homologs. We focus on both, correctness (distinguishing true hits from false positives), and on efficiency. The problem has been well studied in the literature, with scoring based on a Nussinov like counting model [2, 18] and probabilistic models such as covariance models and stochastic context free grammars for RNA [5, 17]. It is also possible to extend the Zuker-Turner thermodynamic model [9, 22] for scoring sequence structure alignments, but that has not yet been done. Here, we extend the approach from [2] to include a new binarizing procedure, and banded alignment

for efficient computation. We use the Bayesian scoring matrix (RIBOSUM) from Klein and Eddy [10] and empirically generated affine gap penalties to score the alignments.

Similar to sequence alignment, an alignment \mathcal{A} of two RNA strings s and t (with lengths m and n respectively) can be described by a matrix of two rows. The first row $\mathcal{A}[1, *]$ contains the string s with gaps, and the second row $\mathcal{A}[2, *]$ contains the string t with some interspersed gaps. Each column has at most one gap in it. We score for both sequence and structure using the function $\delta(\mathcal{A}[1, i], \mathcal{A}[1, j], \mathcal{A}[2, i], \mathcal{A}[2, j])$. In this function $\mathcal{A}[1, i], \mathcal{A}[1, j]$ (likewise $\mathcal{A}[2, i], \mathcal{A}[2, j]$) are scored for complementarity to preserve the base-pairing, and the pairs $(\mathcal{A}[1, i], \mathcal{A}[2, i])$ and $(\mathcal{A}[1, j], \mathcal{A}[2, j])$ are scored for sequence conservation.

Additionally, we score each column that does not participate in base-pairing, by a function $\gamma(a, b)$ that measures sequence conservation, for all $a, b \in \{A, C, G, U, -\}$. Alignments are scored by summing up the contributions of sequence and structural alignments. A schematic algorithm for aligning an RNA query against a sequence is given in Fig. 4, which is an extension of ideas in [2]. Note that this algorithm uses linear gap penalties. In our implementation, we use a slightly more sophisticated affine gap function (omitted in Figure 4 for exposition).

A naive algorithm would iterate over all pairs of intervals in s , and t . We can do better by exploiting the structure of s . The structure of an RNA sequence s is a set S of base-pairs, where $(i, j) \in S$ implies that $s[i]$ bonds with $s[j]$. Ignoring pseudo-knots, each base-pair has a unique enclosing base-pair, thus S can be shown to be a tree with each node denoting a base-pair, and the obvious parent-child relation. First, we augment the tree (See algorithm and an illustration in Fig. 3) by adding spurious base-pairs so that each nucleotide (originally base-paired or not) is in some base-pair, each node has at most two children, and the number of nodes is $O(m)$, where $|s| = m$. Additionally, each node $v \in S$ has at most one child in the augmented structure, denoted by S' .

We limit the intervals in s to nodes $v \in S'$, which are bounded by $O(m)$. Let m_1 , and $m_2 = m - m_1$ denote the number of nodes with 1 and 2 children respectively. The complexity of *alignRNA*, with a query of length m , and a target of length n and n is $O(n^2 m_1 + n^3 m_2)$. This parametrization is useful because in typical structures $m_2 \ll m$. In our case, the complexity can be further reduced. The sequence pairs that need to be aligned have been filtered for an underlying sub-structure. The preliminary alignment obtained by this filter allows us to limit the nodes in S' that can be applied to a position i in t , based on the left end-point of v , and the width. This *banding* reduces the number of nodes to a constant, effectively mak-

ing the complexity $O(n^2 \delta_m^2)$, where $\delta_m \ll m$ is the size of the banded region. The banding forces a tradeoff. Overlapping hits from the filter can either be aligned independently with a tight band, or merged and aligned once with larger band size. Due to lack of space, we will not discuss this issue further.

P-value

For an effective database search, we need to have p -values for the probability that a hit was obtained by chance. Klein and Eddy make the argument that the distribution of scores of RNA structural alignments follow the Gumbel distribution. As this is a strong assumption, and determination of a true p -value is a challenging research problem. Therefore, we choose to express the p -value by using the non-parametric Chebyshev's inequality [3] as a conservative estimate. To obtain the mean and variance, the query is aligned against randomly generated sequence with a similar GC content as the database after each query. We have found that a cut-off of 0.05 – 0.1 is a reasonable value in practice.

4. Results and Discussions

The efficiency and quality of search depends on the design of filters as well as the (appropriately banded) alignment algorithm. We describe the results on filtering and alignment independently before giving combined results. To test our algorithms, we worked with arbitrary ncRNA subfamilies of known/predicted structure from the RFAM [7] and the 5S Ribosomal RNA database [19]. Four sub-families are considered here, tRNA, 5S rRNA, a Purine Riboswitch, and the Hammerhead Ribozyme. A description of these sub-families is deferred to the final version. For every sub-family, we chose some members arbitrarily, and inserted them in a random database of 1Mb, and tested our algorithms on the composite sequence. For completeness, we have also described results on some real genomes. The probability of finding bp_units at random depends on the GC content, so the random database was created by first choosing the GC-content, and subsequently generating bases with appropriate fixed probability. $G + C$ probabilities of 0.35, 0.5, and 0.75 were chosen to study the effect of GC-content. All experiments were performed on an Intel PC (2.8GHz, 1Gb RAM), running Linux.

Filtering for ncRNA

Table 2 describes results of applying various filters. As expected, as the filters become more stringent (higher k, l ,

```

procedure Binarize( $i,j$ ) (* Binarize the interval  $(i,j)$ . *)
if ( $i = j$ )
    return (create_node( $i,j$ ,dotted,Nil)); (* A dotted node with 0 child. *)
if ( $(i,j) \in S$ )
     $v =$  Binarize( $i+1,j-1$ );
    return (create_node( $i,j$ ,solid, $v$ )); (* A solid node with 1 child  $v$ . *)
if ( $(k,j) \in S$  for some  $i < k < j$ )
     $vl =$  Binarize( $1,k-1$ );
     $vr =$  Binarize( $k,j$ );
    return (create_node( $i,j$ ,dotted, $vl,vr$ )); (* A dotted node with 2 children,  $vl$  and  $vr$ . *)
if ( $i < j$ )
     $v =$  Binarize( $i,j-1$ );
    return (create_node( $i,j$ ,dotted, $v$ )); (* A dotted node with 1 child  $v$ . *)
end if

```

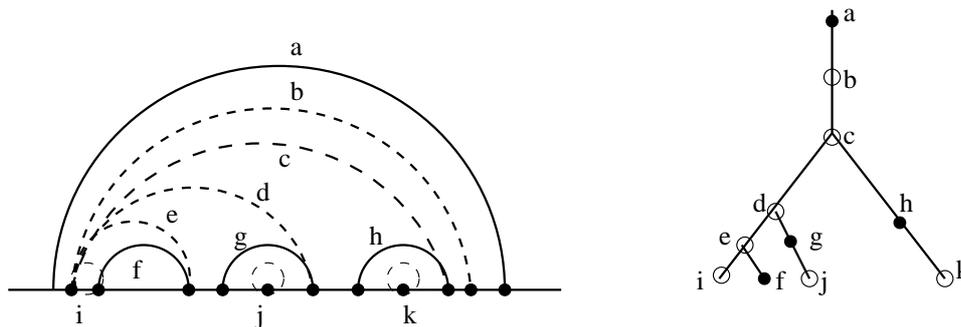


Figure 3. Procedure to create a Binary tree with $O(m)$ nodes such that each node has at most 2 children. The solid nodes correspond to pairs in S , while the dotted nodes correspond to augmented spurious edges. An illustration of the Binarize procedure.

less variable distances), the number of false negatives increases. However, for each family, there exist appropriate filters that filter out a large portion of the database while retaining most of the members of the family. Also, as the GC-content is biased away from 0.5, the number of false hits increases. W.r.t False Negatives, many occur simply because the proposed structure contains unconventional base-pairs which we do not allow. For example, 10/100 tRNA sequences contain $A - G$, $A - A$, or $A - C$ base-pairs, and would be difficult to find with any ncRNA discovery tool. In ongoing work, we plan to change the neighborhood computation to include all k -mers that form low energy stacks according to Zuker-Turner thermodynamic considerations.

The running time for filtering increases linearly with the size of the database (data not shown). Clearly, the time to filter is small enough to be dominated by the alignment, so in principle, one could try more complex filters. However, the advantage of the simpler nested and multi-loop filters is their universality over various families of ncRNA. Designing an appropriate filter for a sub-family, is then just a matter of choosing appropriate values for k , l , and \vec{w} .

Alignment

To test alignment quality, we computed alignments on a randomly generated 300Kb database sequence with 100 tRNAs inserted in it. No filtering was used for FastR. Fig. 5 shows ROC plots for the two alignment algorithms. The two are comparable, with RSEARCH performance better for distant homologs. The time taken for FastR (banded) alignment is 3m48s, compared to 20m42s for RSEARCH. In ongoing research, we plan to improve the quality of FastR alignment, while maintaining compute times.

Finally, we evaluate FastR after combining filtering and alignment. Table 3 summarizes the results of our search. As the two p -values for FastR and RSEARCH are not directly comparable, we use a p -value 0.1 for FastR and a p -value 1 for RSEARCH to get the results are comparable. It can be seen that FastR is close to two orders of magnitude faster than RSEARCH while maintaining comparable sensitivity. Note that there is little or no change in sensitivity due to banding. As seen in the previous section, FastR alignments and scores are good for the high quality hits, but decrease thereafter leading to a loss of sensitivity. Ongoing

procedure alignRNA

(*S is the set of base-pairs in RNA structure of s . S' is the augmented set. *)

for all intervals (i, j) , $1 \leq i < j \leq n$, all nodes $v \in S'$

if $v \in S$

$$A[i, j, v] = \max \begin{cases} A[i + 1, j - 1, \text{child}(v)] + \delta(t[i], t[j], s[l_v], s[r_v]), \\ A[i, j - 1, v] + \gamma(' - ', t[j]), \\ A[i + 1, j, v] + \gamma(' - ', t[i]), \\ A[i + 1, j, \text{child}[v]] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], ' - '), \\ A[i, j - 1, \text{child}[v]] + \gamma(s[l_v], ' - ') + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[l_v], ' - ') + \gamma(s[r_v], ' - '), \end{cases}$$

else if $v \in S' - S$, and v has one child

$$A[i, j, v] = \max \begin{cases} A[i, j - 1, \text{child}[v]] + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[r_v], ' - '), \\ A[i, j - 1, v] + \gamma(' - ', t[j]), \\ A[i + 1, j, v] + \gamma(' - ', t[i]), \end{cases}$$

else if $v \in S' - S$, and v has two children

$$A[i, j, v] = \max_{i \leq k \leq j} \{A[i, k - 1, \text{left_child}[v]] + A[k, j, \text{right_child}[v]]\}$$

end if

end for

Figure 4. An algorithm for aligning a query RNA s of length m with a database string t of length n . $A[i, j, v]$ is the maximal alignment score of subsequence $t[i...j]$ against the subtree (subsequence of $s[1...m]$) rooted at v . The query sequence $s[1...m]$ and structure S have been *Binarized* to get S' . In our implementation, we use affine gap penalty for stacks and loops (omitted for exposition).

ncRNA	Filter	GC	k	l	w ₀	#Hits (/Mb)	False Neg.	Speed (s/Mb)
tRNA	M-fs-4	0.50	4	3	40 - 70	21120	11/100	2.2
tRNA	M-fs-4	0.35	4	3	40 - 70	29379	11/100	1.6
tRNA	M-fs-4	0.7	4	3	40 - 70	37208	11/100	2.4
tRNA	M-fs-1	0.5	5	2	40 - 70	8237	18/100	2.2
tRNA	M-fs-1	0.35	5	2	40 - 70	12496	18/100	1.7
tRNA	M-fs-1	0.7	5	2	40 - 70	16773	18/100	2.4
5S rRNA	M-rrna-2	0.7	5	2	95 - 112	7502	20/100	1.3
5S rRNA	M-rrna-2	0.5	5	2	95 - 112	3307	20/100	1.2
Purine-Rs	M-pur-1	0.5	5	2	50 - 65	2499	10/37	1.1
Purine-Rs	M-pur-3	0.5	4	2	50 - 65	41687	0/37	0.7
Hammerhead	M-ham-3	0.5	4(*)	2	145 - 180	6250	7/57	1.8
Hammerhead	M-ham-2	0.5	4	2	145 - 180	7078	10/57	1.1

Table 2. The results of applying nested and multiloop filters (with various parameters) to random databases. As the filters become more stringent, the number of hits decrease, and the number of false-negatives increase. In all but (*) cases, at most 2 G-U base-pairs are allowed in a bp_unit. (*) refers to cases where only 1 G-U base-pair is allowed.

	Query	Filter	Hits (TP/Tot)	Possible Hits	Time
RSEARCH	Asn-tRNA(AE001087.1/4936-5008)	-	85/93	100	3411s
FastR-U	"	M-fs-1	79/89	82	502s
FastR-B	"	M-fs-1	77/93	82	75s
RSEARCH	5S rRNA (AE016770.1/210436-210555)	-	97/97	100	14939s
FastR-U	"	M-rrna-2	80/80	80	870s
FastR-B	"	M-rrna-2	80/80	80	69s
RSEARCH	Purine-Rs (AE010606.1/4680-4581)	-	33/39	37	9215s
FastR-U	"	M-pur-1	26/38	27	344s
FastR-B	"	M-pur-1	27/38	27	47s
RSEARCH	Hammerhead (M83545.1/56-3)	-	50/58	50	2741s
FastR-U	"	M-ham-3	47/47	47	122s
FastR-B	"	M-ham-3	47/47	47	35s

Table 3. Comparison of FastR and RSEARCH. FastR-B (FastR-U) stands for banded (unbanded). For FastR, approximately the same number of top hits as RSEARCH have been chosen. FastR searches were conducted on the top strand and the time shown has been doubled. Possible hits refers to the number of TPs that are not filtered out by FastR.

Query	Genome	FastR hits (hits/TP/FN)	RSEARCH hits (E-val \leq 10)	FastR time	RSEARCH time
Asn tRNA	<i>A. pernix</i> (NC_000854.1)	25/24/9	57/31/2	2m57s	146m22s
5S rRNA	<i>A. pernix</i> (NC_000854.1)	9/1/1	2/1/1	1m43s	390m7s

Table 4. Comparison of RSEARCH and FastR results on querying the 1.67Mb *A. pernix* genome. The true positives are obtained from known annotations. For False Negatives, we do not consider tRNAs with introns.

research will focus on optimizing the score function to improve sensitivity, with no increase in compute time. As expected, some of the loss of sensitivity can be attributed to filtering. For 5s rRNA, the filter M-rrna-2 allows only 80 of the 100 true positives, which are almost completely retrieved by FastR. In contrast, RSEARCH gets the top 97 but takes over 4 hours compared to the time of 69s for FastR. We have also tested FastR on real genomes, where it is difficult to distinguish true hits. As shown in Table 4, querying the 1.67 Mb *A. pernix* genome yielded comparable results. FastR could not detect the 14 intron containing tRNAs, but detected 24 out of the remaining 33. For 5S rRNA, the single known annotation was the top hit, but there were other alignments of similar quality, indicative of novel 5S rRNAs. In the other two cases (Hammerhead and Purine-Rioboswitch), RSEARCH did not return any signif-

icant hit, and no annotations were available, hence no comparison could be made. FastR dominates again in speed.

5. Conclusion and Future Work

The development of fast filtering and searching tools for ncRNA is a natural area of research, analogous to the development of sequence similarity tools like BLAST, and Fasta. However, as the discussion above shows, the underlying structure and diversity of ncRNA makes this problem quite different in character. The proposed tool *FastR* effectively uses filters to provide a large speedup while retaining high sensitivity. Our software is research quality, but is available to all academic researchers. We are also working to deploy it on the web for use by biologists.

The ideas presented here open many lines of research,

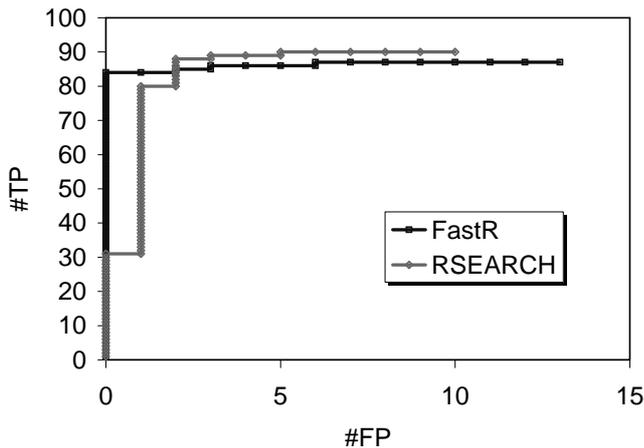


Figure 5. ROC plots for the alignments generated by RSEARCH and FastR.

which we are actively pursuing. The first is the design of optimal (multi-loop and nested) filters for an ncRNA family. Next, we can extend these ideas to compute multiple alignments of RNA, and to search databases with multiple alignments. Finally, our application of FastR on various families of interest including miRNA has resulted in discovery of several novel ncRNA (data not shown). We plan to collaborate with experimental scientists in validating these discoveries.

References

- [1] L. Argaman et al. Novel small RNA-encoding genes in the intergenic regions of *Escherichia coli*. *Curr. Biol.*, 11:941–950, 2001.
- [2] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In Z. Galil and E. Ukkonen, editors, *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, number 937, pages 1–16, Espoo, Finland, 1995. Springer-Verlag, Berlin.
- [3] C. Blom. *Probability and statistics: theory and applications*. Springer-Verlag, 1980.
- [4] J.-H. Chen, S.-Y. Lee, and B. Shapiro. A computational procedure for assessing the significance of RNA secondary structure. *CABIOS*, 6:7–18, 1990.
- [5] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*, chapter 10.3 Covariance models: SCFG-based RNA profiles. Cambridge University Press, 1998.
- [6] S. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews in Genetics*, 2:919–929, 2001.
- [7] S. Griffiths-Jones, A. bateman, M. Marshall, A. Khanna, and S. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 2003.
- [8] D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.
- [9] J. Jaeger, D. Turner, and M. Zuker. Improved prediction of secondary structures for RNA. *Proceedings of the National Academy of Sciences*, 86:7706–7710, 1989.
- [10] R. Klein and S. Eddy. Rsearch: Finding homologs of single structured rna sequences. *BMC Bioinformatics*, 4(1):44, 2003.
- [11] E. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.
- [12] S.-Y. Le, J.-H. Chen, and J. Maizel. *Structure and Methods: Human Genome Initiative and DNA Recombination*, volume 1, chapter Efficient searches for unusual folding regions in RNA sequences, pages 127–136. Adenine Press, 1990.
- [13] R. Lee and V. Ambros. An extensive class of small RNAs in *Caenorhabditis elegans*. *Science*, 294:862–864, 2001.
- [14] L. Lim, N. Lau, E. Weinstein, A. Abdelhakim, S. Yekta, M. W. Rhoades, C. Burge, and D. Bartel. The microRNAs of *Caenorhabditis elegans*. *Genes & Development*, 17:991–1008, 2003.
- [15] T. Lowe and S. Eddy. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research*, 25:955–964, 1997.
- [16] E. Rivas and S. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16(7):583–605, 2000.
- [17] Y. Sakakibara, M. Brown, R. Hughey, I. Mian, K. Sjölander, R. Underwood, and D. Haussler. Recent methods for RNA modeling using Stochastic Context Free Grammars. In *Combinatorial Pattern Matching Conference. Lecture Notes in Computer Science*, volume 807, 1994.
- [18] D. Sankoff. Simulations solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.
- [19] M. Szymanski, M. Barciszewska, V. Erdmann, and J. Barciszewski. 5S ribosomal RNA database. *Nucleic Acids Research*, 28(1):166–167, 2002.
- [20] J. Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–51, 2001.
- [21] C. Workman and A. Krogh. No evidence that mRNA have lower folding free energy than random sequences with the same dinucleotide distribution. *Nucleic Acids Research*, 27(24):4816–4822, 1999.
- [22] M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bull. Math. Biol.*, 46:591–621, 1984.