BIOINFORMATICS

Vol. 22 no. 14 2006, pages e557–e565
doi:10.1093/bioinformatics/btl232

A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements

Shaojie Zhang[1,*], Ilya Borovok[2], Yair Aharonowitz[2], Roded Sharan[3] and Vineet Bafna[1]

[1]Department of Computer Science and Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA, [2]Department of Molecular Microbiology and Biotechnology and [3]School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel

ABSTRACT

Motivation: Recent studies have uncovered an "RNA world", in which non coding RNA (ncRNA) sequences play a central role in the regulation of gene expression. Computational studies on ncRNA have been directed toward developing detection methods for ncRNAs. State-of-the-art methods for the problem, like covariance models, suffer from high computational cost, underscoring the need for efficient filtering approaches that can identify promising sequence segments and speedup the detection process.

Results: In this paper we make several contributions toward this goal. First, we formalize the concept of a filter and provide figures of merit that allow comparison between filters. Second, we design efficient sequence based filters that dominate the current state-of-the-art HMM filters. Third, we provide a new formulation of the covariance model that allows speeding up RNA alignment. We demonstrate the power of our approach on both synthetic data and real bacterial genomes. We then apply our algorithm to the detection of novel riboswitch elements from the whole bacterial and archaeal genomes. Our results point to a number of novel riboswitch candidates, and include genomes that were not previously known to contain riboswitches.

Availability: The program is available upon request from the authors.

Contact: shzhang@cs.ucsd.edu

1 INTRODUCTION

A *database filter* is a computational procedure that takes a database as input, and outputs a subset of the database. The goal is to ensure that the object being searched for remains in the database after filtering, the filtered database is significantly smaller, and the filtering operation is very fast. Filters have played a central role in bioinformatics. BLAST is the prototypical example, with a keyword match filter greatly improving the search for remote homologs. Indeed, improving the filters for sequence similarity search remains an intensively researched area, with many recent publications. Filtering is also being applied in other bioinformatics domains, including structural genomics (Leibowitz *et al*., 1999), proteomics (mass-spectrometry) (Frank *et al*., 2005; Tanner *et al*., 2005), and non coding RNA (ncRNA) (Weinberg and Ruzzo, 2004a,b; Zhang *et al*., 2005). Here, we revisit the notion of filtering, focusing on applications to detecting ncRNAs.

ncRNAs are genomic sequences that are transcribed, but not translated, and function as RNA molecules. Recent discoveries of many novel families and sub-families of ncRNA have underscored their importance, and hint at an RNA world, where coding and non-coding genes play equally important roles (Eddy, 2001; Storz, 2002; Vitreschak *et al*., 2004). The signal for ncRNA is considerably weaker than that for protein coding genes and *de novo* approaches that look for secondary structure or transcriptional initializing signal do not work well (Rivas and Eddy, 2000). Therefore, comparative approaches are more popular with two major directions. One way is to look at compensatory mutations (or consensus folds) in pre-aligned orthologous regions (Rivas and Eddy, 2001; Washietl *et al*., 2005; Pedersen *et al*., 2006). However, the success of this method relies on a good ''structural'' alignment which is difficult to get (Bafna *et al*., 2006). The other comparative approach to discovering novel homologs of a query ncRNA is also increasing in importance, much like BLAST is often used to identify novel homologs of coding genes. While viable, this approach poses a technical challenge since the known algorithms for aligning ncRNA are at least an order of magnitude slower than sequence alignment (Klein and Eddy, 2003; Zhang *et al*., 2005), and even slower when other secondary structures (such as pseudoknots) are allowed (Dost *et al*., 2006). Indeed, using a search based on a covariance model (CM) (Durbin *et al*., 1998), it would take 54 hours to query two bacterial genomes: *E. coli* K12 and *Staphylococcus aureus* MW2 (7.5 Mb) for a sub-family such as the FMN riboswitch (145 bp). This makes the filtering problem both easier and harder. On the one hand, the alignment is so expensive (cubic time), that even a computationally intensive filter (quadratic time) could be useful. At the same time, since the alignment is so expensive, the filtering itself must be very *efficient* in removing a large portion of the database while retaining the true hits. For example, a filter that removes 50% of the database is still not sufficient to make CM searches tractable for large genomic sequences.

Algorithms that align ncRNA are expensive because they score for both sequence and structure conservation, and the latter task is computationally intensive. Filtering for RNA was systematically explored by Weinberg and Ruzzo (2004a, b) who used a pigeonhole argument to show that it is enough to scan for sequence similarity,

*To whom correspondence should be adderessed.

expressed by a hidden Markov model, leaving the more expensive structural alignment for the filtered sequence. Henceforth, we refer to their filter as *HMM-filter*. Subsequently, they, and independently, some of us also used partial structure conservation for the filtering (Weinberg and Ruzzo, 2004b; Zhang *et al*., 2005). Even after applying these filters, the problem remains computationally expensive, and it is worthwhile to ask if one can do better.

Here, we make several contributions in this regard. First, we formalize the concept of a filter and provide figures of merit that allow comparison between filters. Second, we design novel filters and show that they dominate the HMM filters of Weinberg and Ruzzo (2004a) (we defer a formal definition of the notion of notion of dominance to Section 2). In practice, this leads to 1-2 orders of magnitude decrease in search time. However, our main point is not that we can build better filters, but that it is relatively easy to do so. Indeed, the filters we design are very simple conceptually, indicating perhaps that we have only scratched the surface on this problem. The main contribution of this paper is a principled approach to combining filters that have different performance characteristics to achieve dominance (Section 3).

We also revisit the issue of alignment by aligning an RNA-profile to a filtered substring. We emphasize that there is a strong (practically, 1-1) correspondence with CMs in both the alignment algorithm, and the observed results. Indeed, the advantage of the CMs is that their parameters can be trained using the same formalization. However, our reformulation helps us take advantage of simple tricks like banding and others which help speed up the alignment without appreciable loss in accuracy (Section 4). Similar extensions would require a departure from the formalism of stochastic context free grammars that support CMs. This also has an impact on filtering. Unlike previous approaches, we do not tie the accuracy of our filtering procedure to the accuracy of an existing alignment procedure. Thus, it is relatively easy to use our filtering procedure in conjunction with other different alignment algorithms. For example, in recent work, we used the filtering to search genomes for pseudoknotted RNA (Dost *et al*., 2006).

Within ncRNA, we focus our attention on Riboswitches. Riboswitches are ncRNA elements that often occur in the 5' Untranslated Region (UTR) regions of genes (Mandal *et al*., 2004; Nahvi *et al*., 2003; Rodionov *et al*., 2003a; Sudarsan *et al*., 2003; Vitreschak *et al*., 2003, 2004). The riboswitches have a mode of action that one normally associates with proteins: they directly sense the levels of specific metabolites with a structurally conserved aptamer domain to regulate expression of downstream genes. Riboswitches respond to a wide range of metabolites including coenzymes, purines, amino acids and some others. Most riboswitches are predicted to be within UTRs of mRNAs that encode biosynthetic enzymes or metabolite and metal transporters. Novel members are continuously being discovered. The Rfam database (Griffiths-Jones *et al*., 2005), version 7.0, has members from 12 sub-families of riboswitches. Due to their widespread and exclusive occurrence in bacteria, they are attractive antimicrobial targets. Our results point to a number of novel candidates for each of these sub-families, and include genomes that were not previously known to contain riboswitches.

2 FORMALIZING NCRNA FILTERS

Covariance Models (CMs) are probabilistic context-free grammar models that describe both structure and sequence information of an RNA family (Durbin *et al*., 1998; Eddy, 2002). The score of an RNA sequence t against a CM model $\mathcal{M}$ is roughly the sum of two components: its sequence similarity to the modeled family, measured using a position specific scoring matrix (PSSM) of nucleotides, and its structural similarity, measured against the distribution of nucleotide pairs in aligned positions. Formally,

$$S(\mathcal{M},t) \sim \textsc{SeqScore}(\mathcal{M},t) + \textsc{StructScore}(\mathcal{M},t)$$

where SeqScrcore is the score of the PSSM part of $\mathcal{M}$ against t. For ungapped alignments, this would simply be the sum over all columns

$$\textsc{SeqScore}(\mathcal{M},t) = \sum_j \textsc{SeqScore}(\mathcal{M}_j, t_j).$$

If gaps are allowed, we must compute an alignment that optimizes $S[\mathcal{M}, t]$. The SeqScore computation is an order of magnitude faster than an optimum StructScore computation. Weinberg and Ruzzo (2004a) use this as the basis of their sequence based HMM filter[1]. For a given threshold T for $\mathcal{M}$, they compute a threshold T_{ps} as

$$T_{ps} = \min\{\textsc{SeqScore}(\mathcal{M},t) : S(\mathcal{M},t) \geq T\}.$$

This choice of T_{ps} ensures that each 'true homolog' ($S(\mathcal{M}, t) \geq T$) will pass the filter. Moreover, much of the database will be rejected by this filter, and will not undergo the more expensive CM alignment.

In order to improve upon this filter, we start with formalizing the definitions of a filter and its quality. A *filter F* takes a sequence as input and outputs sub-sequences. We assume the operating parameters (such as a threshold) as part of the filter definition. To make the notion of performance independent of the database, we measure it on a suitably defined random database sequence D, with a set of true sequences A embedded in D. The performance of the filter is measured with the following:

(1) **Running Time:** The running time $T_F(|D|, n)$ is a function of query length n, and database length $|D|$.

(2) **Efficiency:** Let $O_F(D)$ be the output of filter F. Define efficiency as $e_F = \frac{|O_F(D)|}{|D|}$. The lower the better.

(3) **Accuracy:** Let A_F denote the subset of true sequences that are accepted by the filter. Then accuracy is defined as $A_F = \frac{|A_F|}{|A|}$. The higher the better.

Filter F_1 *dominates* F_2 if it is faster, more accurate, and more efficient than F_2. Often, filters perform well in one or two but not all of these aspects. In many cases, they can be combined for further improvement. The two obvious ways to combine filters are:

- **Union F_1+F_2**: in which $O_{F_1+F_2}(D) = O_{F_1}(D) \cup O_{F_2}(D)$. Union helps if both F_1 and F_2 are fast and efficient, but not accurate.
- **Composition F_1·F_2**: $O_{F_1 \cdot F_2}(D) = O_{F_2}(O_{F_1}(D))$. Composition helps when the two filters are accurate but not very efficient,

[1]They use HMMs (not PSSMs) to describe the filter, but that technical difference does not change the argument.

and F_1 is faster than F_2. Note that composition is always better than intersection, as the running time $T_{F_1}(D,n) + T_{F_2}(O_{F_1}(D),n)$ is better than T_{F_2} with identical accuracy.

We will use both of these operations in designing better filters. The following result shows that it is not essential to be able to compute efficiency directly in order to prove dominance.

THEOREM 1. *Filter F can be dominated if there exists a filter F_1 with $A_F \subseteq A_{F_1}$ and $T_{F_1}(D,n)/T_F(D,n) \leq 1 - e_{F_1}$.*

PROOF. We simply use the composition $F_1 \cdot F$ as the filter. Clearly, it has better accuracy and is more efficient. For running time, we note that

$$\begin{aligned} T_{F_1}(D,n) + T_F(O_{F_1}(D),n) &\leq T_{F_1}(D,n) + e_{F_1}T_F(D,n) \\ &\leq (1 - e_{F_1})T_F(D,n) + e_{F_1}T_F(D,n) \\ &\leq T_F(D,n). \end{aligned}$$

While self-evident, Theorem 1 is useful because instead of trying to compute efficiency exactly we can look for a constant θ such that $T_{F_1}(D,n)/(T_F(D,n)) \leq \theta$, and $e_{F_1} \leq 1 - \theta$. As an application of the theorem, we can think of the CM itself as a filter F. F is very accurate (gets all the true hits) and efficient (random sequences do not score high), but slow ($T_F(D, n) = \Omega(|D|n^2)$) (Klein and Eddy, 2003; Zhang *et al*., 2005). On the other hand, the HMM filter F_1 is accurate ($A_F = A_{F_1}$), and an order of magnitude faster ($T_{F_1}(D,n) = O(|D|n)$), but not as efficient. Can the composite filter dominate? Note that $T_{F_1}(D,n)/(T_F(D,n)) \leq 1/n$. From Theorem 1, the composite filter $F_1 \cdot F$ dominates F if $e_{F_1} \leq (n-1)/n$. As this condition is relatively easy to achieve, Weinberg and Ruzzo show improvements for most families (Weinberg and Ruzzo, 2004a). In the following, we will describe sequence based filters that run in time $c|D|$, where c is a small constant. By the previous argument, we only need to show marginal efficiency $(n-c)/n$ to dominate. Thus, the filters we design will dominate the HMM filters of Weinberg and Ruzzo (2004a).

3 SEQUENCE FILTERS

Let F_P denote a sequence based filter, which computes a gapped SEQSCORE, and uses a threshold T, chosen so that the accuracy of F_P is identical to the CM. We will define a sequence based filter F_s that matches the accuracy of F_P, but is faster. The idea is based on an application of the pigeonhole principle, and the fact that text search using a *dictionary* of words is fast. For a sequence to score T against a profile of length L, each column must score T/L on the average. In fact, every sequence that scores T against the profile contains an l-mer w that scores Tl/L or better against the profile. F_S proceeds by computing all subsequences that match at least one keyword in T. We use the following procedure:

(1) Generate a set of keywords K, each of length l (for a fixed parameter l), by selecting all words that score Tl/L in an ungapped region of the profile. Label each such keyword w so that LABEL(w) is the profile position where it occurs.
(2) Search D for exact matches to keywords from K.
(3) For each position i that matches a keyword with label p, identify $D[i-p, \ldots, i-p+L]$ as a candidate sequence.
(4) Merge significantly overlapping candidate sequences.

By the pigeonhole principle, the accuracy of F_S is high ($A_{F_P} \subseteq A_{F_S}$). The filtering can be done in $O(|D|)$ time through the use of Aho-Corasick tries, or hashing, so the filter time is an order of magnitude faster. It remains to evaluate the efficiency of this filter. For any position i to be selected, either of the keywords in K must match at a specific position (given by their label) relative to i. Therefore, assuming a uniform distribution of words along the sequence, the efficiency of this filter is given by $(\frac{|K|}{4^l})$. By Theorem 1 , we only require $\frac{|K|}{4^l} < \frac{n-1}{n}$ for dominance, and can often find single keyword filters that suffice. In the following we improve upon this simple filter by considering multiple keywords.

3.1 Multiple keyword (chain) filtering

We define an (l, m, δ, K)-*chain filter* as follows: sequence $D[i, \ldots, i+L]$ is accepted by an (l, m, δ, K)-chain filter if m words $w_1, w_2, \ldots, w_m \in K$, each of length l match at positions $i+i_1, i+i_2, \ldots, i+i_m$, s.t. for all j, $i_j \geq i_{j-1} + l$ (i.e., words are ordered and non-overlapping) and $|i_j - \text{LABEL}(w_j)| \leq \delta$. For ungapped alignments, $\delta = 0$, but otherwise, δ must be chosen carefully to maximize accuracy. We have the following result:

THEOREM 2. *Consider an (l, m, δ, s_K)-chain filter. If s_K is the maximum number of keywords with an identical label in K then the efficiency on a uniform random database is given by*

$$e_F(l,m,\delta,s_K) = \binom{L - m(l-1)}{m} \left(\frac{2\delta s_K}{4^l}\right)^m. \qquad (1)$$

PROOF. Consider a random position i in the database D. By definition,

$$e_F(l,m,\delta,s_K) = Pr[D[i, \ldots, i+L] \text{ is accepted}].$$

Define a *configuration* w.r.t. a position i as an m-tuple $C(i) = (i_1, i_2, \ldots, i_m)$, such that $i \leq i_1 \leq i_2 \ldots \leq i_m \leq i + L$ and $i_j \geq i_{j-1} + l$ for all j. Then i is accepted by the filter if there exists a configuration $C(i)$ such that for all $i_j \in C$, $D[i_j, \ldots, i_j + l - 1] = w_j$ for some $w_j \in K$ with $|\text{LABEL}(w_j) - i_j| \leq \delta$. Thus, the probability for i_j to match up by chance is $\frac{2\delta s_K}{4^l}$. It follows that the efficiency of the (l, m, δ, K)-chain filter is $C_m(2\delta s_K/4^l)^m$, where C_m is the number of possible configurations. To compute this number, consider a binary string b with exactly m ones and $L - lm$ zeros. For $1 \leq j \leq m$, let b_j be the position of the j-th '1' from the left. Define $i_j = b_j + (j-1)l$. Then each binary string corresponds to a unique m-tuple $(i_1, i_2, \ldots, i_m)$, and $i_{j+1} - i_j = b_{j+1} + b_j + l \geq l$ for all $j < m$. The number of configurations is equal to the number of distinct binary strings, given by $C_m = \binom{L - m(l-1)}{m}$.

Figure 1 shows (as expected) that the efficiency of a chain filter F_C decreases exponentially with increasing m. The slightly faster than exponential decay is due to the fact that $L - ml$ also decreases with increasing m. Likewise, higher values of s_K decrease the rate of decay. However, for multiple keywords, selecting the set K of keywords becomes a challenging problem. The pigeonhole principle guarantees the existence of m words that score at least mTl/L, but does not bound the minimum score on any single word. If we were to choose K to be the set of all keywords, s_K could be prohibitively large. On the other hand, any choice of a lower bound will reduce Accuracy ($A_{F_P} \not\subseteq A_{F_C}$). In practice, there are many reasonable choices that ensure that the accuracy remains 1 and high efficiency is maintained. Currently, the features we deploy use empirically chosen cut-offs for keyword scores. However, there is a principled

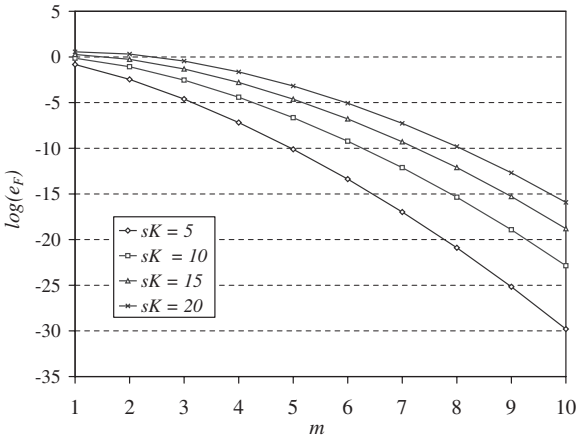

Fig. 1. A plot of $\log(e_F)$ versus m, when $L = 150$, $l = 8$ and $\delta = 20$. Different lines correspond to different values of s_K.

way to get around this obstacle by using an appropriately chosen *union* of filters.

3.2 Accuracy of chain filters

To control the accuracy of chain filters we extend their definition to allow a score threshold S, such that a sequence is accepted by the filter if in addition to satisfying the above conditions the total score of the matched keywords exceeds S. Let $\theta = Tl/L$. We are interested in computing a chain of words that score $m\theta$. We illustrate the approach using a parameter $\theta' = \theta/2$. Any subsequence that is accepted by the chain-filter must have some k ($1 \leq k \leq m$) words $w_1, \ldots, w_k$ that each score at least θ'. Let $w_{k+1}, \ldots, w_m$ denote the remaining words in the chain filter. We have

$$m\theta \leq \sum_{j=1}^{k} \text{SCORE}(w_j) + \sum_{j=k+1}^{m} \text{SCORE}(w_j)$$
$$\leq \sum_{j=1}^{k} \text{SCORE}(w_j) + (m-k)\theta'.$$

Thus $\sum_{j=1}^{k} \text{SCORE}(w_j) \geq m\theta - (m-k)\theta' = (m+k)\theta/2$.

For all $1 \leq k \leq m$, define an extended chain filter F_k of k words in which each word scores at least $\theta/2$, and the chain must score at least $(m+k)\theta/2$. Observe that $F_1 + \ldots + F_m$ accepts every chain that scores above $m\theta$, implying that $A_{F_C} \subseteq A_{F_1+\ldots+F_m}$. In the next section, we show that chain filters can be computed efficiently, in time that is often $o(|D|n)$. The search time of the union filter grows linearly with m, and so an efficiency/speed trade-off must be considered in selecting an appropriate m. Once again, Theorem 1 can be used to ensure dominance, but we must do it in an empirical setting as the running time depends upon the score distribution of keywords in K, which in turn, depends upon the alignment. Our results in Section 5 show that dominating filters are easy to find.

3.3 Implementing chain filters

We wish to filter substrings that match an extended (l, m, δ, K, S)-chain filter (where S is the score threshold). Our goal is to improve upon the profile search time of $O(L|D|)$. As chain filters are based on matches with l-mers, we can improve the speed by using string matching techniques. The algorithm is as follows:

(1) Build an Aho-Corasick Trie T_K with K (alternatively, if l is small, construct a hash table for occurrences of l-mers in D).

(2) Initialize a set of *active* intervals $\mathcal{I} = \phi$.

(3) Scan D with T_K. For each hit of word $w \in K$ at position i, add the interval $\pi = [i - \text{LABEL}(w) - \delta, i - \text{LABEL}(w) + \delta]$ to $\mathcal{I}$. The score of the interval $\text{SC}[\pi]$ is set to the score of w against the profile. Also, set the position as $\text{POS}[\pi] = i$.

(4) For each position $j \in D$, let $\mathcal{I}_j = \{\pi \mid j \in \pi\}$ be the subset of intervals that overlap with j. For most choices of parameters, $|\mathcal{I}_j| << L$. Select position j if there exist m intervals that are disjoint and have net score better than m. This is done as follows:

  (1) Sort the intervals in $\mathcal{I}_j$ according to $\text{POS}[\pi]$. For each $\pi \in \mathcal{I}_j$, let $p_1(\pi)$ be the largest interval with $\text{POS}[\pi] - \text{POS}[p_1(\pi)] > l$, and $p_2(\pi)$ be the predecessor of π.

  (2) for all intervals $\pi \in \mathcal{I}_j$ $\text{SCORE}[j, \pi] = \max\{\text{SC}[\pi] + \text{SCORE}[j, p_1(\pi)], \text{SCORE}[j, p_2(\pi)]\}$. Output j, if $\text{SCORE}[j, \pi]$ exceeds the threshold.

The entire computation takes time $\sum_j |\mathcal{I}_j| = o(L|D|)$. Also, the computation is done only if the depth of coverage at position j exceeds a threshold. The depth of coverage can be computed in linear time. This discussion hides an important problem. Insertions and deletions make the profile length significantly longer than any sequence. For example, the average length of cobalamin riboswitches is 200 , while the profile length is closer to 600 . A simple way around this is to discard columns with many gap characters, but that entails deciding which columns are dominated by gaps. Instead, we revise the definition of the LABEL of a position. Recall that LABEL of a keyword is its position in the profile, and should match its position in the query sequence. Instead, define the LABEL as the expected position in the query sequence. Let p_i denote the probability that the i-th position of the profile is not a gap (in other words, $p_i = P[i, A] + P[i, C] + P[i, G] + P[i, T]$). Then define

$$\text{label}_i = \begin{cases} p_i & \text{if } i = 1, \\ \text{label}_{i-1} + p_i & \text{otherwise.} \end{cases}$$

Each keyword that appears at position i in the profile is assigned label_i as its label.

4 RNA-PROFILE SCORING AND ALIGNMENT

In this section we describe our algorithm for scoring a sequence against a structural alignment of an RNA family, where we score for conservation of both sequence and structure. The algorithm is very similar to Covariance Model (Durbin *et al*., 1998; Eddy, 2002). However, we provide our own implementation to allow for faster banded scoring. Also, our filter design can be more effectively tied to the scoring. Formally, we treat the RNA-profile alignment as a filter, and compose it with the chain filter. Finally, our algorithm can be extended to include more complex RNA models, such as pseudoknots, which will be explored in future work.

The structural alignment of an RNA family is a (gapped) multiple alignment R of its sequences with structure described by a set M of pairs of positions (i, j), such that for a majority of sequences in the family, the nucleotides aligning to these positions form base-pairs.

procedure PAln

(*M is the set of base-pairs in RNA profile R. M' is the augmented set. *)

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

if $v \in M$

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

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

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

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

$$A[i,j,v] = \max_{i \le k \le j} \{A[i, k-1, \text{left\_child}[v]] + A[k, j, \text{right\_child}[v]]\}$$

end if

end for

Fig. 2. An algorithm for aligning an RNA profile R with m columns against a database string t of length n. The query consensus structure M has been *Binarized* to get M'. Each node v in the tree corresponds to a base-pair $(l_v, r_v) \in M'$.

The alignment of the RNA family against a target sequence t is described by a $2 \times m$ matrix A, in which row 1 contains *column positions* of the profile interspersed with spaces (insertion of aligned sequence), and row 2 contains the *sequence*, also interspersed with spaces (deletion of profile columns). For all columns j, $A[1, j] \neq' -'$ or $A[2, j] \neq' -'$. For $r \in \{1, 2\}$, define $\rho_r[j] = j - |\{l < i \text{ s.t. } A[r, l] =' -'\}|$. In other words, if $A[1, j] \neq' -'$, it contains the position $\rho_1[i]$ of R. The score of alignment A is given by

$$\sum_j \gamma(A[1,j], A[2,j]) + \sum_{(\rho_1[i], \rho_1[j]) \in M} \delta(\rho_1[i], \rho_1[j], \rho_2[i], \rho_2[j]).$$

The function γ scores for sequence similarity, and δ scores for structure conservation. Our goal is to find an alignment that maximizes this score. While this formulation encodes a linear gap penalty, we note here that alignments of RNA molecules may contain large gaps, particularly in the loop regions, and we implement affine penalties for gaps (details omitted).

4.1 Choosing the scoring functions

Consider an alignment of n RNA sequences from a family. Let $n_i(a)$ be the number of sequences with $a \in \{A, C, G, U,' -'\}$ in the i-th column of the multiple alignment. The probability of observing a in the i-th position can be estimated by

$$P_i(a) = \frac{C_a + n_i(a)}{\sum_{a'} C_{a'} + n}$$

where C_a are pseudo-counts, chosen so that $p_a = C_a / (\Sigma_{a'} C_{a'})$, where p_a is the probability of occurrence of a in the family. These probabilities are used to construct a position specific scoring matrix. Then for all positions i, and al symbols $a \in \{A, C, G, U,' -'\}$

$$\gamma(i, a) = \sum_{a' \in \{A, C, G, T, -\}} S(a', a) \times P_j(a') \qquad (2)$$

where $S(a', a)$ is the score of substituting a' with a. We use a nucleotide substitution scoring matrix (Klein and Eddy, 2003). We model insertions and deletions with the gap penalties $\gamma('-', a)$, and $\gamma(i,'-')$, respectively.

Likewise, to score for structure conservation we look at the probabilities of specific base-pairs that occur in each pair of positions. For each $(i, j) \in M$, let $n_{i,j}(a, b)$ describe the number of sequences in the alignment that contain a in position i, and b in position j. As before,

$$P_{i,j}(a, b) = \frac{C_{a,b} + n_{i,j}(a, b)}{\sum_{a', b' \in \{A, C, G, U, '-'\}} C_{a', b'} + n}$$

and the score for conserved structure is given by

$$\delta(i, j, a, b) = \sum_{a', b' \in \{A, C, G, U\}} P_{i,j}(a', b') \times S_p(a', b', a, b) \qquad (3)$$
$$\forall (i, j) \in M, a, b \in \{A, C, G, U\}$$

where S_p is scoring matrix for substituting (a', b') with (a, b), and rewards both sequence and structure conservation. Note that δ is only defined when $(i, j) \in M$, and $a, b \in \{A, C, G, U\}$. In other cases, the structure is obviously not conserved, and the appropriate score is given by γ.

4.2 The alignment procedure

We make the assumption that the base-pairs are non-crossing. For each base-pair $(i, j) \in M$, there is a unique (*parent*) base-pair (i', j') such that $i' < i < j < j'$, and there is no base-pair (i'', j'') such that $i < i'' < i'$, or $j < j'' < j'$. Thus the alignment can be done by recursing on the nodes of the tree. However, the tree can have high degree and not all columns of the profile participate in it. To this end we binarize the tree using the procedure given in Zhang *et al.* (2005). Specifically, we add spurious nodes (base-pairs) to the tree so that every column participates as a tree node, the degree of any node is at most 3 , and the number of nodes is $O(m)$, where m is the number of columns in the profile. Further, a node corresponding to a true base-pair $(i, j) \in M$ has at most one child.

Table 1. Riboswitch sub-families in the Rfam database (version 7.0)

Rfam Id	Name	Average length	%id	#seed	#total
RF00050	FMN	145	66	48	136
RF00059	TPP	110	52	237	382
RF00080	yybP-ykoY	128	45	74	127
RF00162	SAM	110	67	71	219
RF00167	Purine	100	56	37	100
RF00168	Lysine	182	49	60	98
RF00174	Cobalamin	204	46	171	249
RF00234	glmS	184	58	14	37
RF00379	ydaO-yuaA	158	54	35	74
RF00380	ykoK	168	60	39	53
RF00442	ykkC-yxkD	106	62	16	21
RF00504	gcvT	101	51	117	163

Average length and ''%identity'' are based on the information in the Rfam database. '#seed' is the number of sequences in the seed alignment. '#total' is the number of full family sequences.

Figure 2 describes a dynamic programming algorithm for aligning a sequence to an RNA profile. The RNA profile is described by a tree. Each node v in the tree either corresponds to a base-pair $(l_v, r_v) \in M'$ of the profile, where M' is the augmented list of base-pairs. The alignment of the sequence to the RNA profile is done by recursing on the tree like structure of RNA. Each node in the binarized tree either represents a base-pair/unpaired base (and has its own PSSM), or represents a branching point in a pair of parallel loops. The algorithm maintains the sequence interval being aligned and the current node in the structure tree.

5 EXPERIMENTAL RESULTS

We implemented the chain filtering and the profile alignment algorithms as described above. All tests reported herein were performed on a 2.8 GHz Intel PC (genomic searches were done on 1.6GHz AMD Opteron grid). For chain filtering, we chose the parameters l, m, δ and score threshold (affects s_K) so as to optimize efficiency while maintaining optimal accuracy. The chain filtering was also composed with HMM filtering (from RAVENNA package (Weinberg and Ruzzo, 2004a)) to further improve the filtering efficiency. For the alignment of the filtered sequences to an RNA model we used both our profile alignment tool and the CMsearch tool from the INFERNAL suite (http://infernal.wustl.edu) Eddy (2002); Griffiths-Jones *et al*. (2003). Both the HMM filters (using expended HMM filters) and CMsearch were applied in the following with their default parameters or recommended parameters from the Rfam database website.

We applied these algorithms to search for riboswitch elements. We chose to focus on riboswitches both due to their importance and due to their unique properties that make them an ideal test case: many ncRNA families show strong sequence similarity, which makes sequence based filtering very efficient, and relatively trivial. In contrast, the riboswitches, with 12 distinct sub-families (and new sub-families being continuously discovered) are quite diverse, and relatively difficult to filter. Table 1 summarizes known riboswitches from the Rfam database, version 7.0 (Griffiths-Jones *et al*., 2003, 2005).

5.1 Filter efficiency and accuracy

To systematically test our filters, we downloaded data on 12 riboswitch sub-families from the Rfam database, version 7.0 (Griffiths-Jones *et al*., 2003, 2005). These data contain for each family a 'seed' alignment, which is a hand-curated alignment of known members, and a 'full' collection of family sequences, which contains known and predicted (by CMsearch) members. In the following we refer to a member of the seed alignment as *seed sequence*, and to a member of the full collection as *family sequence*.

Synthetic databases: As a first test of our method we synthesized several test sequences. For each sub-family, we created a random genomic sequence of size 1 Mb with G+C content of 0.5, and randomly planted all the family sequences therein. We tested the filter's performance on the composite sequence. Table 2 summarizes the results of the chain filter (CF) in comparison to the HMM filters and to a combined filter. In addition to the efficiency measure we also report a second measure *efficiency*2, which is computed exclusively on the random sequence. While the actual genomic sequence will have some true hits as well, it is unlikely to have more than a few members per Mb, so efficiency2 is a better approximation to the true efficiency.

Recall from Theorem 1 that high gains in filter speed at the cost of efficiency is desirable because filter composition can be used to achieve dominance. Thus, the key statistic in Table 2 is search time. The sequence based chain filter is much faster (on average, 9 sec/Mb) than the HMM filter (71 sec/Mb). Interestingly, even the efficiency of CF filter remains very high on the average (0.036) while maintaining optimal accuracy. The faster speed and the optimal accuracy of the CF filter makes the composite filter (CF·HMM), which applies CF filter first and HMM filter later on the database, dominate the HMM filter. In Table 2, CF·HMM further improves the efficiency significantly (0.029), and it is still much faster (on average, 14 sec/Mb) than the HMM filter. The filtering is followed by alignment with RNA-Profile. We also include a direct comparison between profile alignment and the CM approach. As can be seen from Table 3, profile alignment attains very similar accuracies but is much faster.

Genomic sequences: Next, we tested the performance of our filter on two genomes with biased G+C content, previously used by Weinberg and Ruzzo (2004)a: *E. coli* K12 and *Staphylococcus aureus* MW2. We searched for the 12 riboswitch families on these genomes whose total length is 7.5 Mb. Table 4 presents a comparison to the HMM filter. As expected, the chain filter is much faster. On the average, its efficiency is also very high (0.017), outperforming that of the HMM filter (0.34). Note that all true hits in these two genomes were recovered by every filtering method with the corresponding alignment algorithm. Obviously, the composite filter, CF·HMM, still provides the fastest filtering solution.

5.2 Discovering novel riboswitches

We applied our sequence based filters, coupled with profile alignment, to search all bacterial and archaeal genomes for the twelve riboswitch families. A total of 254 genomes spanning 818 Mb were searched. Of these, 179 have some ncRNA annotations. Table 5 summarizes the search results. In total we identified 463 novel (putative) riboswitches based on a P-value cutoff 0.04. Interestingly, 413 of these predictions were within 500 bp upstream of an annotated gene. These predictions include hits to

Table 2. Filtering performance of chain filters (CF), HMM filters (HMM), and composite filters (CF·HMM) on synthetic sequences

Family	CF				HMM				CF·HMM		
	eff.	eff2.	acc	time(m:s)	eff.	eff2.	acc.	time(m:s)	eff.	eff2.	time(m:s)
FMN	1.3e-2	0	1	0:10	2.8e-2	0	1	1:10	1.3e-2	0	0:11
TPP	6.3e-2	3.4e-2	1	0:07	1	1	1	0:59	5.8e-2	3.1e-2	0:14
yybP-ykoY	1.5e-1	1.4e-1	1	0:08	1	1	1	1:07	1.4e-1	1.3e-1	0:28
SAM	1.8e-2	2.1e-3	1	0:07	5.9e-2	4.0e-4	1	0:55	1.7e-2	0	0:09
Purine	3.8e-2	3.1e-2	0.99∗	0:7	1.1e-2	1.5e-4	1	0:52	7.4e-3	5.9e-5	0:10
Lysine	1.5e-2	3.9e-3	0.99∗	0:10	1	1	1	1:34	1.5e-2	3.8e-3	0:13
Cobalamin	6.3e-2	3.4e-2	1	0:13	1	1	1	1:42	6.2e-2	3.3e-2	0:26
glmS	1.3e-2	9.1e-3	1	0:14	7.7e-3	3.0e-4	0.97	1:25	2.4e-3	0	0:17
ydaO-yuaA	1.2e-2	4.9e-3	1	0:08	1.9e-2	1.0e-3	1	1:11	6.9e-3	7.5e-5	0:10
ykoK	1.2e-2	6.0e-3	1	0:10	1.2e-2	1.2e-4	1	1:32	5.9e-3	0	0:12
ykkC-yxkD	1.7e-3	0	1	0:07	2.4e-3	0	1	0:53	1.7e-3	0	0:07
gcvT	3.7e-2	2.5e-2	1	0:07	1.9e-1	1.6e-1	1	0:51	1.6e-2	4.3e-3	0:10
Average	**0.036**	**0.024**	**1**	**0:09**	**0.361**	**0.347**	**1**	**1:11**	**0.029**	**0.017**	**0:14**

'eff.' is the efficiency on synthetic sequences, 'eff2.' is the efficiency on exclusively random sequences, 'acc.' is the accuracy on synthetic sequences, and 'time' is the running time on synthetic sequences. (∗) Note that these filters only miss one hit.

Table 3. Comparison of RNA profile alignment (PAln) and CMsearch (CM) on synthetic sequences

Family	PAln #TP	PAln #true	PAln retrieval rate	CF· PAln time (m:s)	CM #TP	CM #true	CM retrieval rate	HMM· CM time (h:m:s)
FMN	136	136	1	1:29	136	136	1	13:24
TPP	373	382	0.98	6:06	382	382	1	7:06:47
yybP-ykoY	119	127	0.94	14:43	127	127	1	4:11:31
SAM	218	219	1	2:23	219	219	1	12:03
Purine	99	99	1	2:17	100	100	1	2:05
Lysine	97	98	0.99	3:16	98	98	1	13:57:59
Cobalamin	242	249	0.97	14:58	248	249	1	27:39:27
glmS	36	37	0.97	2:36	35	37	0.95	6:53
ydaO-yuaA	73	74	0.99	3:15	73	74	0.99	13:16
ykoK	52	53	0.98	1:22	53	53	1	8:39
ykkC-yxkD	21	21	1	0:30	21	21	1	1:56
gcvT	138	163	0.85	3:15	163	163	1	37:48

RNA profile alignment uses p-value cut-off 0:05 to get the top ranking hits (one hit in cobalamin family is marginal), and CMsearch use the same cutoff bits score from Rfam data website. 'retrieval rate' is defined as the percentage of true positive (#TP) hits over the possible true hits (#true) after filtering (either chain filtering (CF) or HMM filtering (HMM)).

genomes that had previously been annotated for ncRNA in Rfam. For cobalamin riboswitch (as an example), most of the predictions are, indeed, in 5' UTRs of cobalamin-related or cobalamin-associated genes (Rodionov *et al*., 2003b; Vitreschak *et al*., 2003) (B12 synthesis, cobalt transporters and alternative cobalamin-independent enzymes). One of the predicted cobalamin riboswitches has been experimentally tested and confirmed (data not shown). In the gcvT (glycine-dependent riboswitch) family, we found 28 novel hits, of which 12 occur as proximal pairs, which is known a preferred mechanism of action for this family (Mandal *et al*., 2004). Detailed information on these discoveries is presented in supplementary data (http://www.cse.ucsd.edu/~shzhang/paper/ISMB2006).

6 CONCLUSIONS

We reiterate that the main contribution of this paper is not simply to provide improved filtering, but to formalize the filtering problem, and demonstrate that a simple approach based on combining filters is useful. While our results improve the state-of-the-art and are likely to be useful in discovering novel ncRNAs, many questions remain unanswered. Some of the open problems are directly related to our analysis. First, can we give theoretical bounds on the efficiency vs. speed trade-off for the union filters? This will probably entail some assumptions on the distribution of keyword scores. Second, can we design optimal chain filters, which provably dominate all other sequence based filters? Indeed the bulk of the results presented here are presented on filters that are fast, but not perhaps as efficient as could be. On the other hand, HMMs are efficient, but not always fast, which indicates that there is room for more filters in between. Examples of such filters include subsets of profiles (choose a subset of contiguous conserved columns, and filter based on those), or a hierarchy of compositions instead of a single one. Finally, for the most diverse families, it is likely that sequence based filters will not be efficient. Fast filters based on structure considerations have been shown to be effective (Weinberg and

Table 4. Filtering performance of chain filters (CF), HMM filters (HMM) and composite filters (CF·HMM) on two real genomes with alignment performance of profile alignment (PAln) and CMsearch (CM)

Family	CF eff.	CF time (m:s)	'CF· PAln' time (m:s)	HMM eff.	HMM time (m:s)	'HMM· CM' time (m:s)	'CF· HMM· CM' time (m:s)	'CF· HMM· PAln' time (m:s)	CM· time estimated (hours)
FMN	1.2e-4	1:28	2:24	9.1e-6	15:40	16:10	1:32	2:24	54h
TPP	2.5e-2	1:10	10:39	9.9e-1	12:03	45h	16:10	7:55	40h
yybP-ykoY	6.7e-2	1:20	42:23	1	13:55	44h	53:04	36:59	38h
SAM	4.5e-4	1:11	2:01	8.2e-4	11:34	12:39	1:15	1:51	31h
Purine	2.8e-2	1:08	10:38	2.7e-3	10:51	12:36	1:34	1:49	18h
Lysine	3.8e-3	1:41	6:06	1	19:23	100h	6:28	5:46	85h
Cobalamin	4.0e-2	1:55	44:56	9.6e-1	20:31	177h	69:42	39:40	166h
glmS	1.9e-2	2:00	13:09	2.1e-3	17:53	18:28	2:24	3:32	80h
ydaO-yuaA	3.8e-3	1:20	6:27	3.5e-4	11:00	11:47	1:22	3:10	99h
ykoK	3.0e-3	1:21	5:32	1.0e-3	114:00	16:19	1:26	3:00	61h
ykkC-yxkD	0	1:00	1:29	0	8:19	8:23	1:00	1:29	23h
gcvT	1.5e-2	0:54	6:49	1.5e-1	7:45	2.4h	2:01	2:37	33h
Average	**0.017**	**1:22**	**12:42**	**0.34**	**14:05**	**31h**	**13:10**	**9:11**	**61h**

'eff.' is the filtering efficiency, and 'time' is the running time of the corresponding filters. Note that when HMM filter efficiency is close to 1, the computation time for HMM · CM (from RAVENNA (Weinberg and Ruzzo, 2004*a*) package) is longer than the computation time for CMsearch (from Infernal package (Griffiths-Jones *et al*., 2003)). This is because of the differences between the C++ and the C compiler.

Table 5. Summary of searching riboswitches against the whole bacterial and archaeal genomes

Family	#known	#TP	#new	#new*	CF eff.	CF·PAln time (hours)	CM time estimated (days)
FMN	103	92	34	2	8.5e-4	4.8	236.9
TPP	305	235	89	6	7.9e-3	6.7	232.4
yybP-ykoY	109	74	65	25	7.7e-2	63.7	166.5
SAM	204	182	80	3	6.7e-4	3.4	136.0
Purine	82	72	31	10	5.7e-2	34.3	82.6
Lysine	82	61	23	5	5.7e-3	12.6	405.8
Cobalamin	189	141	70	15	3.6e-2	65.1	794.0
glmS	24	23	8	1	1.4e-3	6.9	372.1
ydaO-yuaA	68	62	17	57	2.3e-2	36.9	470.2
ykoK	44	39	7	2	3.9e-3	10.5	266.7
ykkC-yxkD	14	14	11	1	1.4e-5	2.8	98.7
gcvT	148	98	28	1	4.2e-2	27.2	136.8

'#known' is the number of known riboswitches in the whole bacterial and archaeal genomes, '#TP' is the number of predicted known hits, '#new' is the number of new predictions in these genomes, and '#new∗' is the number of new predictions from the genomes that had previously been annotated for ncRNA in Rfam.

Ruzzo, 2004b; Zhang *et al*., 2005), but have been completely ignored in the present study. It is an important open problem to formalize their efficiency and speed, and to study their combination with sequence based filters. We hope that these and related challenges will spur the development of filters, and ultimately lead to better tools for mining biomolecular databases.

ACKNOWLEDGEMENTS

This work is supported by a grant from the National Science Foundation (NSF-DBI:0516440) (S.Z. and V.B) and by an Alon Fellowship (R.S.). This research is also supported in part by the UCSD FWGrid Project (NSF Research Infrastructure Grant Number EIA-0303622).

REFERENCES

Bafna,V., Tang,H. and Zhang,S. (2006) Consensus folding of unaligned RNA sequences revisited. *J. Comput. Biol.*, **13** (2), 283–295.

Dost,B., Han,B., Zhang,S. and Bafna,V. (2006) Structural alignment of pseudoknotted RNA. In *RECOMB '06: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology,* Springer New York, NY, USA, 143–158.

Durbin,R., Eddy,S.R., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis*. Cambridge University Press.

Eddy,S.R. (2001) Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet*., **2**, 919–929.

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

Frank,A., Tanner,S., Bafna,V. and Pevzner,P. (2005) Peptide sequence tags for fast database search in mass-spectrometry. *J. Proteome Res*., **4** (4), 1287–1295.

Griffiths-Jones,S., Bateman,A., Marshall,M., Khanna,A. and Eddy,S.R. (2003) Rfam: an RNA family database. *Nucleic Acids Res*., **31**(1), 439–441.

Griffiths-Jones,S., Moxon,S., Marshall,M., Khanna,A., Eddy,S.R. and Bateman,A. (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res*., **33** (Database issue), 121–124.

Klein,R.J. and Eddy,S.R. (2003) Rsearch: finding homologs of single structured rna sequences. *BMC Bioinformatics*, **4** (1), 44.

Leibowitz,N., Fligelman,Z.Y., Nussinov,R. and Wolfson,H.J. (1999) Multiple structural alignment and core detection by geometric hashing. In *Proc. Int. Conf. Intell. Syst. Mol. Biol*. pp. 169–177.

Mandal,M., Lee,M., Barrick,J.E., Weinberg,Z., Emilsson,G.M., Ruzzo,W.L. and Breaker,R.R. (2004) A glycine-dependent riboswitch that uses cooperative binding to control gene expression. *Science*, **306** (5694), 275–279.

Nahvi,A., Sudarshan,N., Ebert,M., Zou,X., Brown,K. and Breaker,R. (2003) Genetic control by a metabolite binding mRNA. *Chem. Biol*., **9**, 1043–1049.

Pedersen,J.S., Bejerano,G., Siepel,A., Rosenbloom,K., Lindblad-Toh,K., Lander,E.S., Kent,J., Miller,W. and Haussler,D. (2006) Identification and Classification of Conserved RNA Secondary Structures in the Human Genome. *PLoS Comput. Biol*., **2**(4), e33.

Rivas,E. and Eddy,S. (2000) Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, **16**(7), 583–605.

Rivas,E. and Eddy,S.R. (2001) Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, **2**, 8.

Rodionov,D.A., Vitreschak,A.G., Mironov,A.A. and Gelfand,M.S. (2003a) Regulation of lysine biosynthesis and transport genes in bacteria: yet another RNA riboswitch? *Nucleic Acids Res*., **31** (23), 6748–6757.

Rodionov,D.A., Vitreschak,A.G., Mironov,A.A. and Gelfand,M.S. (2003b) Comparative genomics of the vitamin B12 metabolism and regulation in prokaryotes. *J. Biol. Chem*., **278** (42), 41148–41159.

Storz,G. (2002) An expanding universe of noncoding RNAs. *Science*, **296** (5571), 1260–1263.

Sudarsan,N., Wickiser,J.K., Nakamura,S., Ebert,M.S. and Breaker,R.R. (2003) An mRNA structure in bacteria that controls gene expression by binding lysine. *Genes Dev*., **17** (21), 2688–2697.

Tanner,S., Shu,H., Frank,A., Wang,L.C., Zandi,E., Mumby,M., Pevzner,P.A. and Bafna,V. (2005) InsPecT: identification of posttranslationally modified peptides from tandem mass spectra. *Anal. Chem*., **77** (14), 4626–4639.

Vitreschak,A.G., Rodionov,D.A., Mironov,A.A. and Gelfand,M.S. (2003) Regulation of the vitamin B12 metabolism and transport in bacteria by a conserved RNA structural element. *RNA*, **9** (9), 1084–1097.

Vitreschak,A.G., Rodionov,D.A., Mironov,A.A. and Gelfand,M.S. (2004) Riboswitches: the oldest mechanism for the regulation of gene expression? *Trends Genet*., **20** (1), 44–50.

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

Weinberg,Z. and Ruzzo,W.L. (2004a) Faster genome annotation of non-coding rna families without loss of accuracy. In: *RECOMB '04: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, ACM Press New York, NY, USA, pp. 243–251.

Weinberg,Z. and Ruzzo,W.L. (2004b) Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, **20** (Suppl 1), I334–I341.

Zhang,S., Hass,B., Eskin,E. and Bafna,V. (2005) Searching genomes for non-coding rna using fastr. *IEEE/ACM Trans. on Comput. Biol. and Bioinformatics*, **2** (4), 366–379.