HAL
open science

Seed-Based Exclusion Method for Non-coding RNA Gene Search

Jean-Eudes Duchesne, Mathieu Giraud, Nadia El-Mabrouk

To cite this version:

Jean-Eudes Duchesne, Mathieu Giraud, Nadia El-Mabrouk. Seed-Based Exclusion Method for Non-coding RNA Gene Search. 13th Annual International Computing and Combinatorics Conference (COCOON 2007), Jul 2007, Banff, Canada. pp.27-39, 10.1007/978-3-540-73545-8_6 . inria-00271642

HAL Id: inria-00271642

https://inria.hal.science/inria-00271642

Submitted on 9 Apr 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Seed-Based Exclusion Method for Non-coding RNA Gene Search

Jean-Eudes Duchesne[1], Mathieu Giraud[2], and Nadia El-Mabrouk[1]

[1] DIRO – Université de Montréal – H3C 3J7 – Canada
{duchesnj,mabrouk}@iro.umontreal.ca
[2] Bioinfo/Sequoia – LIFL/CNRS, Université de Lille 1 – France
giraud@lifl.fr

Abstract. Given an RNA family characterized by conserved sequences and folding constraints, the problem is to search for all the instances of the RNA family in a genomic database. As seed-based heuristics have been proved very efficient to accelerate the classical homology based search methods such as BLAST, we use a similar idea for RNA structures. We present an exclusion method for RNA search allowing for possible nucleotide insertion, deletion and substitution. It is based on a partition of the RNA stem-loops into consecutive seeds and a preprocessing of the target database. This algorithm can be used to improve time efficiency of current methods, and is guaranteed to find all occurrences that contain at least one exact seed.

1 Introduction

The last 20 years have seen an explosion in the quantity of data available for genomic analysis. Much work has been devoted to speeding up data mining of proteins or gene coding DNA, but these sequences account for only a fraction of the genome. In addition, many non-coding RNA genes (ncRNAs) are known to play key roles in cellular expression, yet few efforts have been made to facilitate their search in large scale databases. Classical homology based search methods like Blast [1] often fail when searching for non-coding genes since the input is stripped from structural information down to its bare sequence. Searching algorithms that permits inputs with structural information should yield better results.

Historically, the first computer scientists to interest themselves with ncRNAs have created tailor made algorithms for specific RNA families such as tRNAs [6,4,9]. Other more general search tools where created to give control of the biological context to the user [3,12,7]. Still these tools lacked the capacity to efficiently parse large genomic databases. Klein and Eddy provided a database specialized search tool [8] for ncRNA including structural information, but is self admittedly slow for large scale databases. More recently, Zhang and Bafna presented a method to efficiently filter databases with a set of strings matching a profile to specific parameters [2]. Their experimentation gave rise to specialized filters for specific RNA families. As such this strategy would require prior

G. Lin (Ed.): COCOON 2007, LNCS 4598, pp. 27–39, 2007.
© Springer-Verlag Berlin Heidelberg 2007

knowledge on the RNA families of interest when generating database, this can become restrictive in some experimental contexts which would benefit from an all-purpose filtering method for ncRNA. Although this can be offset by combining filters with different parameters in an attempt to maximize efficiency and accuracy.

In addition to the capacity of parsing large genomic databases, as sequence and structure constraints are established from a restricted set of an RNA family representatives, any search method should account for a certain flexibility and deviation from the original consensus, allowing for possible mismatches and insertion/deletion (indel) of nucleotides. In particular RNAMotif [12] (one of the most popular and time efficient tool for RNA search), does not explicitly allow for base pair indels. In [5], we have considered a more general representation of folding constraints and developed an approximate matching algorithm allowing for for both mismatches and indels of base pairs. The major drawback of the method was its time inefficiency.

In this paper, we develop a seed-based exclusion method allowing for mismatches and indels, able to speed up existing RNA search methods. Similar heuristics have been proved very efficient to accelerate the classical homology based methods. In particular, PatternHunter [11,10] based on multiple spaced seeds has become one of the most popular method for sequence search at a genomic scale. Recently, Zhang et al.[16] proposed a formalization of the filtering problem and a demonstration that the combination of several filters can improve the search of ncRNAs. Here, we develop a new seed-based heuristic for RNA search, using seeds with distance and folding constraints. It is based on a partition of the RNA stem-loops into consecutive seeds and a preprocessing of the target database storing the occurrences of all possible seeds in a hash table. The search phase then reports, in constant time, the position lists of all seeds of the query stem-loops, and uses extension rules to account for possible errors. The heuristic is guaranteed to find all occurrences containing at least one exact seed.

The rest of the paper is organized as follows. Section 2 presents the basic concepts and definitions, and introduces the general idea of the Sagot-Viari algorithm [15] that will be used in our algorithm's search phase. Section 3 describes our new exclusion method. In Section 4, we study the choice of seeds and anchor elements. Finally, we present our experimental results in Section 5, and show how our method can be used in conjunction with RNAMotif to improve its running time.

2 Preliminary Definitions

2.1 RNA Structures

An RNA *primary structure* is a strand of consecutive nucleotides linked by phosphodiester bonds: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). When transcribed from DNA to RNA, thymine is substituted into uracil (U). As

such, U and T are considered synonymous for most purposes. We denote Σ_{DNA} the alphabet of nucleotides $\{A, C, G, T\}$.

Considering that an individual nucleotide's main biological property is to form a structural bond with other nucleotides, primary structure alone ill-defines ncRNAs. An RNA *secondary structure* is represented by a series of base pairings, the most frequent ones being the canonical Watson-Crick A-T and C-G. The secondary structure is organized in a set of nested stems and loops, where a stem is a sequence of paired and unpaired nucleotides, and a loop is a sequence of unpaired nucleotides. A stem followed by a loop is called a *stem-loop* (Figure 1.(a)).

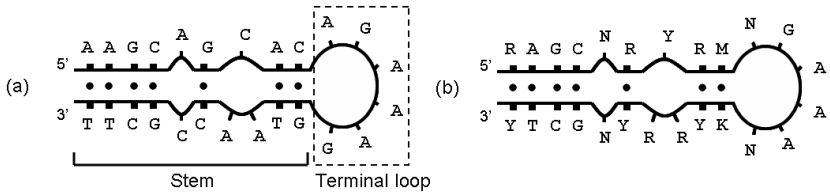

Fig. 1. (a) A stem-loop with canonical base pairings represented as dots; (b) A stem-loop descriptor. (a) is an occurrence of (b).

It is well documented that the functional properties of an RNA molecule is dependent on its final structure obtained by additional foldings over its secondary structure. Our work relies on the hypothesis that there is enough signal in the primary and secondary structure to find the overall molecule with this simplified view.

2.2 Descriptors

Descriptors are user-defined sets of conserved elements of a specific molecule's primary and secondary structure. They are often obtained from multiple alignments of different instances of the same molecule's sequence from various species, but how a good descriptor is obtained is beyond the scope of this current work.

In [5], we have introduced a rigorous and very flexible representation of folding constraints in term of "secondary expressions". In this paper, we focus on a more restrictive descriptor form, though allowing to represent most of the RNA families found in the literature. The considered constraints are:

1. Positions characterized by a possible subset of nucleotides and represented by a degenerate alphabet, the IUPAC code, over all possible substitutions (Table 1). For example, N allows for any nucleotide at the observed position.
2. Correlated constraints due to canonical base pairings. For example, in Figure 1.(b), the left-most pairing (R, Y) means that the upper nucleotide can be either A or G, but if it is A (respec. G) then the opposite nucleotide should be T (respec. C).
3. Bounded range of possible lengths for unpaired parts of the structure.

Table 1. The standard IUPAC code defines symbols for sets of nucleotides

A : A	K : G | T	B : C | G | T
C : C	M : A | C	D : A | G | T
G : G	R : A | G (purine)	V : A | C | G
T : T	S : C | G	H : A | C | T
	W : A | T	
	Y : C | T (pyrimidine)	N : A | C | G | T

2.3 The Sagot-Viari Notations

The Sagot-Viari algorithm [15] is designed to search for all stem-loops in a genomic sequence, allowing for possible mispairings. More precisely, given four parameters s, e, d_{min} and d_{max}, the algorithm finds all possible stem-loops in the genome G characterized by a maximum stem length s, a loop of size d with $d_{min} \leq d \leq d_{max}$, and a maximum number of e mispairings and nucleotide insertion and deletion (indels).

The interesting design feature of their method was to keep separate the two complementary parts of the stems until the final reconstruction step. Another way to look at their method is to consider that they filter a complete genome for sequences that can potentially form a stem-loop structure but differ the actual verification until the sequences have been extended to the full length of the pattern.

We first introduce some basic notations. Given a sequence $u = u_1 u_2 \ldots u_n$, we denote by $u_{i,j}$ the subsequence $u_{i,j} = u_i u_{i+1} \ldots u_j$. The sequence $\overline{u}$ is the complementary inverse of u. For example, if $u =$ AATGC, then $\overline{u} =$ GCATT. Given a sequence u of size k on Σ_{DNA}, we denote by $Oc(u)$ the list of positions of all occurrences of u in the genomic sequence G, eventually within a threshold of error e. The occurrences list of a stem-loop described by u is:

$$S_{(u,\overline{u})} = \{(p,q) \,|\, p \in Oc(u), q \in Oc(\overline{u}), \mathsf{good}(p,q)\}$$

The predicate $\mathsf{good}(p,q)$ checks the distance ($d_{\min} \leq q - p \leq d_{\max}$) and the error constraints.

The algorithm proceeds by successive *extensions* and *filtering* steps, starting from sets $Oc(\alpha)$ for each $\alpha \in \Sigma_{\mathrm{DNA}}$. Each set $Oc(u_{i,j})$ could be constructed by extending $Oc(u_{i+1,j})$ and $Oc(u_{i,j-1})$. However, a majority of the positions in $Oc(u)$ can be eliminated before the final filtering. In fact, the algorithm never computes any Oc list beyond the initial step. It considers only *possible occurrences* (of the stem-loop) position lists:

$$POc(u_{i,j}) = \{p \in Oc(u_{i,j}) \,|\, \exists q \in Oc(\overline{u_{i,j}}), \mathsf{good}(p,q)\}$$

The lists $POc(u_{i+1,j})$ and $POc(u_{i,j-1})$ are extended and merged into one list $POc'(u_{i,j})$. Filtering that list for the distance and error constraints give rise to

the list $POc(u_{i,j})$. At the end, the solution set $S_{(u,\overline{u})}$ is obtained by a (quadratic) filtering between $POc(u)$ and $POc(\overline{u})$.

The POc and POc' lists are represented through stacks, and all the extension and filtering operations are done in linear time relative to the size of the stacks.

3 An Exclusion Method for RNA Search

Given an RNA descriptor D and a genomic sequence (or database) G, the goal is to find the position list S_D of the occurrences of D in G, possibly with an error threshold e. We propose to search the descriptor D starting with a set of n anchor sequences extracted from the descriptor's stem-loops. A heuristic based on an exclusion method is developed for an efficient search of anchor sequences: each anchor is partitioned into consecutive (and overlapping) seeds of a given size, and a preliminary step consists in building a seed database over the genomic sequence G. In section 4, we discuss the choice of appropriate "constraining" anchors allowing a good speed-up with a convenient sensibility.

A high level sketch of the exclusion method is given below and is schematized in Figure 2. Details are in the following subsections.

1. **Preprocessing phase:** Build a seed database over the genomic sequence G.
2. **Partition phase:** Choose a set of anchor sequences from D (with their relative distance constraints) and a set of seed-shapes, and partition the anchor sequences into consecutive seeds.
3. **Anchor search phase:** Query the database for the seeds, giving lists of occurrences Oc. Then extend occurrences and filter them while checking length, error and folding constraints.
4. **Check phase:** Check whether each RNA candidate verifies the descriptor constraints that were not used as anchors in the search phase.

3.1 The Preprocessing Phase

The genomic database G is first processed to output all elementary motifs of a given size. The preprocessing phase is designed to allow for a constant time access to the position list of all occurrences of elementary motifs, represented by seeds. Rigorous definitions follow.

A *seed of size* k or k*-seed* is a sequence of size k on the alphabet Σ_{DNA}. To allow the possibility of spaced seeds, we define two types of characters: `#` and `-`, where `-` denotes the don't care character. A k*-seed-shape* is a sequence of k elements from the alphabet $\{\texttt{\#}, \texttt{-}\}$.

Given a set of seed-shapes, the preprocessing phase builds a hash table containing an entry for each set of sequences with the same `#` positions. For example, for seed-shape `##-#`, $AGAC$ et $AGTC$ are stored at the same position. We discuss the choice of appropriate seed-shapes and lengths in Section 4.

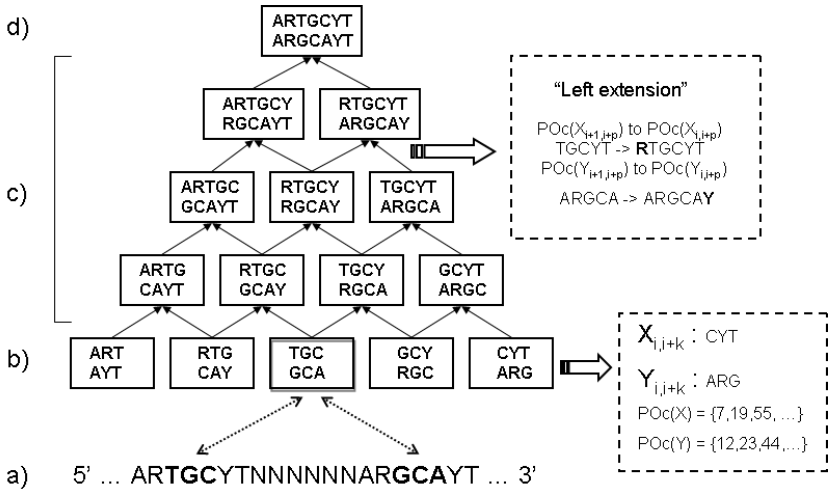

Fig. 2. (a) A specific descriptor sequence with the set of anchors $\mathcal{A} = \{ARTGCYT, ARGCAYT\}$ of common length $m = 7$. The distance constraints are $d^{1,2}_{min} = d^{1,2}_{max} = 6$. Anchors are partitioned into consecutive 3-patterns. The elements in bold represent a single pair of seeds (seed shape ###); (b) The initial step of the search phase is to query the database for all positions of the selected seeds. In the figure, boxes are labeled by their implicit sequences. Their actual data is the lists of positions of these sequences, as illustrated by the rightmost box; (c) Next, the algorithm iterates over a series of extensions and merges to filter the seeds that cannot possibly extend into the desired motif. Each level represents a single iteration. A single box receives incoming extensions from two sources, hence the need for merging sets of positions into a single set. One of these extension is shown in greater detail; (d) After the final iteration, the algorithm returns a list of candidate positions for the full anchor sequences. Each position needs to be validated to confirm the presence or absence of the desired motif at the given position in the genome.

3.2 The Partition Phase

The RNA descriptor is first parsed to extract a given number of *anchors* that are ordered in a priority search list (see section 4). More precisely an anchor $\mathcal{A}$ is a set of sequences $\{\mathcal{A}^1, \cdots, \mathcal{A}^l\}$ on the IUPAC alphabet, with a set of distance constraints $\{(d^{i,j}_{min}, d^{i,j}_{max})\}$. Anchor sequences can be related with complementary relations, but that is not mandatory.

A sequence of size k over the IUPAC alphabet is called a *k-pattern*. For a given length k, each anchor sequence $\mathcal{A}^i$ is partitioned into its consecutive k-patterns $\mathcal{A}^i_{1,k}, \mathcal{A}^i_{2,k+1}, \cdots \mathcal{A}^i_{m-k+1,m}$. For a given k-seed-shape sh, we then report the set of seeds corresponding to each k-pattern. A formal definition follows.

Definition 1. *Let $u = u_1 \cdots u_k$ be a k-pattern and $sh = sh_1 \cdots sh_k$ be a k-seed-shape. We say that a seed $s = s_1 \cdots s_k$ is a representative of u with respect to sh iff, for any i such that $sh_i = \#$, $s_i \in u_i$.*

Given a k-seed-shape sh and a k-pattern u, we denote by $L(u)$ the list of seed representative of u with respect to sh. For example, if $u = ARYC$ and sh=`##-#` we have $L(u) = \{AAAC, AACC, AAGC, AATC, AGAC, AGCC, AGGC, AGTC\}$. We also denote by $L(\mathcal{A}^i)$ the list of representative of all k-patterns of $\mathcal{A}^i$. The partition phase reports the lists $L(u)$ of each k-pattern of each anchor sequence $\mathcal{A}^i$.

A final definition is required for the following section. Given a genomic database G and a k-pattern u, the list of all occurrence positions of $L(u)$ in G is denoted by $Oc(u)$. For example, if $G = TAGACTAAAC$ and u is the k-pattern introduced above, then $Oc(u) = \{2, 7\}$.

3.3 The Anchor Search Phase

For clarity of presentation, we describe the search phase for an anchor with two anchor sequences of the same length, and a unique seed-shape of size k. Generalization to anchors of different lengths only requires a final step to extend the longest anchor sequence. Generalization to anchors with more than two sequences requires to consider one POc and POc' list per sequence. Anchors with a single sequence are usually inefficient to consider during the search phase of an RNA descriptor. Generalization to multiple seed-shapes is straightforward.

Let $\mathcal{A} = \{X, Y\}$ be the considered anchor, with the distance constraint (d_{min}, d_{max}), and m be the common length of X and Y. Let k be the size of the considered seed-shape, and the consecutive k-patterns of each anchor sequence be $X_{1,k} \cdots X_{m-k+1,m}$ (respec. $Y_{1,k} \cdots Y_{m-k+1,m}$).

The initialization step consists in computing $m - k + 1$ pairs of lists $(Oc(X_{i,i+k-1}), Oc(Y_{i,i+k-1}))$ with respect to the genomic sequence G. Following the partition phase, each seed is an entry in the hash table and accessed in constant time. Following the Sagot-Viari methodology (Section 2.3), the two lists are then traversed and filtered with respect to the distance constraints (d_{min}, d_{max}). The list's elements are of the form (*pos*, *num_errors*), where *pos* represents a position in the genome and *num_errors* is the minimum number of errors between the G subsequence at position p and the considered k-pattern with respect to the seed-shape (errors are computed on the `#` positions of the seed-shape).

The following $m - k$ steps extend the consecutive k-seed surviving lists to $k + 1$-seeds, then $k + 2$-seeds, until the m-seeds surviving lists representing the complete anchor. As allowed seed lengths vary from k to m, we will number the following steps from k to m.

Step $\mathbf{p}$*, for* $\mathbf{k} \leq \mathbf{p} < \mathbf{m}$*:*
For each i, $1 \leq i < m - p + 1$ do:

1. **Extend left** $POc(X_{i+1,i+p})$ to $POc(X_{i,i+p})$ and respectively $POc(Y_{i+1,i+p})$ to $POc(Y_{i,i+p})$ iff $1 \leq i$. To do so we use the Sagot-Viari rules of model construction (extension by the character X_i or Y_i) with the exception that elements of both $POc(X_{i,i+p})$ and $POc(Y_{i,i+p})$) need to satisfy one condition out of the match, mismatch, insertion and deletion. This is because we allow

for errors in both X and Y with respect to the descriptor while the original Sagot-Viari algorithm did not have that restriction.

2. **Extend right** $POc(X_{i,i+p-1})$ to $POc(X_{i,i+p})$ and respectively $POc(Y_{i,i+p-1})$ to $POc(Y_{i,i+p})$ iff $i+p \leq m$. This extension mirrors the previous step but uses equivalent symmetric extensions to add characters X_{i+p} and Y_{i+p} respectively.
3. **Merge** the two resulting lists into a new pair of lists $POc'(X_{i,i+p})$ and $POc'(Y_{i,i+p})$. If the resulting lists contain consecutive elements representing the same position but with different numbers of errors, we keep a single copy with the minimum number of errors.
4. **Filter** $POc'(X_{i,i+p})$ and $POc'(Y_{i,i+p})$ with respect to the distance, folding and error bound constraints. The resulting lists are $POc(X_{i,i+p})$ and $POc(Y_{i,i+p})$.

In contrast with the original Sagot-Viari algorithm, errors should be allowed for both anchor sequences. The filtering step should then account, not only for the distance constraint, but also for the combined error constraints. Moreover if X and Y are two strands of a given stem, then folding constraints must be checked.

At the end of the search phase, the two remaining lists $POc(X) = POc(X_{1,m})$ and $POc(Y) = POc(Y_{1,m})$ contain all possible occurrences of both anchor sequences. The last step is then to return all occurrence pairs $S_{X,Y}$ respecting the distance, error and folding constraints.

3.4 The Check Phase

The rest of the descriptor D should finally be validated against the positions of the anchor $\mathcal{A}$. For this purpose any existing RNA search method can be used, such as BioSmatch [5] or RNAMotif if indels are not allowed.

4 Choosing the Anchor Sequences and Seed Shapes

The first idea is to choose the most constraining anchor sequences (those that are likely to give rise to the minimum number of occurrences in the database), that is those with the lowest p-value. Statistical work on structured motifs of form $X\, x(\ell, \ell{+}\delta)\, Y$, where X and Y are correlated by secondary structure constraints, have been done in [14]. The difficulty arise from the overlapping structure of the patterns. The p-value can be computed by brute enumeration or by sampling.

However, the most constraining anchors are not necessarily the easiest to parse. Indeed, degenerated symbols (representing sets with more than one nucleotide) can give rise to a large list of seed representative in the partition phase (see section 3.2). More precisely, anchor sequences of the same size and with the same occurrence probability may give rise to different lists of seeds representatives depending on the distribution of their degenerated positions. For example, in Table 2, though both anchor sequences A**R**TGC**Y**T and ACT**N**CAT have the same occurrence probability of $1/4^6$ under the Bernoulli model, the second

Table 2. Size of the lists involved in each stage of the extension phases, for an exact search with the seed-shape `###` on a 10M test database (from E. coli. and B. subtilis)

Anchor $\mathcal{A}$	Number of seeds in $L(\mathcal{A})$	seed occurrences in the database	seeds remaining after extensions			
			step 3	step 4	step 5	step 6
ACTGCAT	5	856408	17846	871	42	4
A**R**TGC**Y**T	8	1427873	35303	2380	206	10
ACT**N**CAT	14	2130755	60553	3599	167	6

sequence gives rise to a larger list of seed representative leading to a much larger list of occurrences in the database. As the first extension phase of our algorithm is the most time-consuming phase, a good estimation of the total time comes from the number of seed representative.

This is further illustrated in Figure 3 where all possible anchors represented by pairs of 5-patterns from the 5S RNA helix III (see Figure 3) are searched in a database of bacterial genomes. Not surprisingly, the anchors with the fewest seeds are significantly faster. Therefore, among the most constraining anchor sequences (those with the lowest p-value), we choose those that give rise to the shortest list of seed representative $L(\mathcal{A})$.

Finally, Table 3 shows the sensibility and the speed obtained with different seed-shapes. It appears that longer shapes lead to a smaller execution time, but at the cost of a lower sensibility: some sequences are missed. Here the best compromise is to use the spaced seed-shape `##-#` : the parsing time is more than 40% smaller than the time needed by RNAMotif and the sensibility remains at 99%.

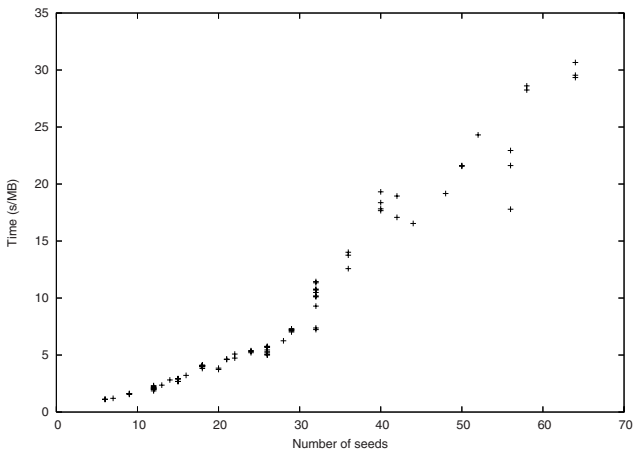

Fig. 3. Relation between the speed of the exclusion method and the number of seed representative of the anchor. We tested each possible anchor represented by pairs of 5-patterns from the 5S RNA helix III, with the seed-shape `####`. The horizontal axis gives the number of seeds corresponding to each anchor pair, and the vertical axis the time taken for the search on a 130 MB bacterial database.

Table 3. Speed and sensitivity of the Exclusion method. The descriptor is an helix with 6-base stems and a loop $x(10, 50)$, searched with 1 error. It occurs 2110 times in on a 10M test database (sequences from E. coli. and B. subtilis). Sensibilities of our method are lowered by 1% due to an additional heuristic in stack transversal during the search phase. The time ratios are against the time for RNAMotif.

	RNAMotif	Exclusion			
	v. 3.0.4	###	##-#	####	##-##
sensibility	100%	97%	99%	68%	67%
preprocessing time (ms)	–	684	927	1068	1262
parsing time (ms)	2709	4144 (153%)	1512 (56 %)	352 (13%)	135 (5%)
total time (ms)	2709	4828 (178%)	2439 (90 %)	1420 (51%)	1407 (45%)

5 Testing on RNA Stem-Loops

Here, we tested our new method for both quality and speed, by comparing with RNAMotif. Indeed though RNAMotif has the limitation of ignoring possible nucleotide insertions and deletions, it is an exact method thus giving a good benchmark to test our heuristic's sensitivity. Moreover it is the fastest RNA search method developed so far.

We considered three RNA families: 5S rRNAs and RNase P RNAs as in [5] as well as group II introns. In each case, the most conserved region was considered, namely helix III for 5S rRNAs, P4 region for RNase P RNAs and domain V for group II introns. The tests are performed exclusively on stem-loop signatures because of technical limitations in our current implementation. This will be extended to full structures of ncRNAs in the near future. We used a database containing 25 randomly selected microbial genomes from GenBank representing a total of more than 75 million base pairs. All tests were performed on an intel Pentium 4 PC with a 2800MHz processor, 2 GB of memory and running Fedora Core 2. The stem-loop signatures were chosen to represent various testing conditions and parameters (stem and loop size).

We considered the seed-shape ####, and two anchor sequences of size 5. Since computational time rises exponentially with the number of seed representative generated by anchor pairs, a cutoff value was selected to avoid anchors likely

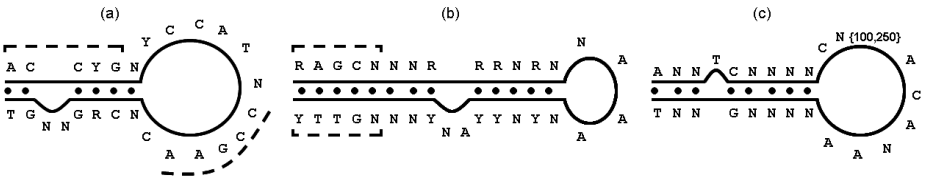

Fig. 4. The stem-loop signature used for (a) 5S rRNAs helix III, (b) group II intron domain V and (c) RNase P RNA P4 region. The dotted lines represent the specific anchor sequences selected for searching.

to generate large initial sets of occurrences from the database. It was set to 16 seed representative, based on experimental results (Figure 3). This cutoff could be raised or lowered on execution to influence speed (lower cutoff) or sensitivity (higher cutoff), but 16 has been a good compromise thus far. Both the 5s rRNA and the Intron group II consensus had several anchor pairs of size 5 falling under that cutoff. The chosen anchors are illustrated in Figure 4. The anchor sequences used for the Intron group II consensus are related by folding constraints, where as those used for the 5s rRNA are only related by distance constraints. Unfortunately, no suitable pair of anchor sequences was found for the RNAse P. This is more likely a limitation of the current implementation rather than the method since the whole consensus structure could have yielded for adequate anchors which were not present in the P4 region. However, this illustrates that certain ncRNA might not have sufficient conserved regions to select adequate anchor sequences.

We tested the ability of our exclusion method to speed up RNAMotif, in other words, the check phase was completed by using RNAMotif. Running times (Table 4, third column) are clearly improved for both 5S rRNAs and group II intron domain V. This clearly shows that the exclusion method can shave off significant amount of computational time for ncRNA searching methods. Finally, we used our method not as a filtering strategy but rather as a stand alone algorithm. In other words the exclusion method was used over the full ncRNA stem-loop signatures (Table 4, last column). As expected from the many degenerated positions in the structure consensus, the execution times are fairly slow for this setup. Here we can clearly see the relationship between conservation and execution times with the most conserved consensus structure (5S rRNAs) being significantly faster to search than the other candidates.

Table 4. Computation times obtained by running our exclusion algorithm and RNAMotif v3.0.0 on the stem-loop signature considered for each structured motif family on a database of bacterial genomes. For each method we show the times obtained when the full helix is searched and when only the most conserved subsets are searched. No suitable anchor subset was available for the RNase P RNA P4 region.

	RNAMotif	RNAMotif with Exclusion method	Exclusion on full helix
5S RNA, helix III	2.6 s/Mb	1.1 s/Mb	12.7 s/Mb
Intron group II, domain V	3.3 s/Mb	2.7 s/Mb	31.0 s/Mb
RNase P RNA P4 region	3.1 s/Mb	–	59.1 s/Mb

The database contained 71 annotated sequences of the 5S rRNA and 17 sequences of the group II intron. Of these 89 annotated ncRNA genes, only 2 weren't found by RNAMotif, both of the 5S rRNA variety. The exact same results were found by the exclusion method in combination with RNAMotif. In the case of the RNase P RNA, although we couldn't find a suitable pair of anchor sequences, using the exclusion method as a stand alone algorithm did provide

the same predictions as RNAMotif, where 36 of 39 annotated sequences were found. In other words, no loss in sensitivity was observed over the tested data when compared to an exhaustive method like RNAMotif.

6 Conclusion

We have developed an exclusion method allowing for nucleotide mismatches and indels, that can be used in combination with other existing RNA search methods to speed up the search. We have shown that given sub-motifs with small degeneracy values, a hashing method built on the preprocessing of the target database can significantly improve search times. The idea is to select in the descriptor anchors which yield the least computation. That being the case, it's not given that any descriptor contains enough consecutive conservations to permit sublinear filtering. By using distance constraints we can significantly reduce the number of needed consecutive conserved positions by introducing gaps between pairs of anchors.

Furthermore, we have shown that restricting these features to the helical structures alone is not an efficient method to filter a database. This result concords with previous literature on the subject of finding signals in secondary structure alone [13]. Generalizing the problem to seeds with distance constraints without considering the secondary structures yields the best results as it takes into account signal in both secondary and primary sequence.

This filtering method is still in its early stage as we can explore many other features. It is evident from our current results that there is a bias for selecting small elements and using the largest possible seed to gain the greatest speed increase. In [16], Zhang et al. present a more robust way to select target anchors from a pattern by creative use of the pigeonhole principle. In this paper we address the same sensitivity issue through the use of "don't care" characters which gives added flexibility in choosing the anchor sequences for seeding the search. We have not yet determined if both approaches are compatible and can be defined into a single model. In any case, we plan to incorporate Zhang's filter definition into our future work to facilitate the comparison and/or addition of our parameters. Furthermore, we plan to generalize the method to an arbitrary number of anchors separated by constraint distances. This could be a viable avenue to limit the number of initial candidates to process and further lower computational times.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology 215, 403–410 (1990)
2. Bafna, V., Zhang, S.: FastR: Fast database search tool for non-coding RNA. In: Proceedings of IEEE Computational Systems Bioinformatics (CSB) Conference, pp. 52-61 (2004)

3. Eddy, S.R.: RNABOB: a program to search for RNA secondary structure motifs in sequence databases (1992) http://bioweb.pasteur.fr/docs/man/man/rnabob.1.html#toc1
4. El-Mabrouk, N., Lisacek, F.: Very fast identification of RNA motifs in genomic DNA. Application to tRNA search in the yeast genome. Journal of Molecular Biology 264, 46–55 (1996)
5. El-Mabrouk, N., Raffinot, M., Duchesne, J.E., Lajoie, M., Luc, N.: Approximate matching of structured motifs in DNA sequences. J. Bioinformatics and Computational Biology 3(2), 317–342 (2005)
6. Fichant, G.A., Burks, C.: Identifying potential tRNA genes in genomic DNA sequences. Journal of Molecular Biology 220, 659–671 (1991)
7. Gautheret, D., Major, F., Cedergren, R.: Pattern searching/alignment with RNA primary and secondary structures. Comput. Appl. Biosci. 6(4), 325–331 (1990)
8. Klein, R., Eddy, S.: RSEARCH: Finding homologs of single structured RNA sequences (2003)
9. Laslett, D., Canback, B.: ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. Nucleic Acids Research 32, 11–16 (2004)
10. Li, M., Ma, B., Kisman, D., Tromp, J.: PatternHunter II: Highly Sensitive and Fast Homology Search. Journal of Bioinformatics and Computational Biology 2(3), 417–439 (2004) Early version in GIW 2003
11. Ma, B., Tromp, J., Li, M.: PatternHunter: faster and more sensitive homology search. Bioinformatics 18(3), 440–445 (2002)
12. Macke, T., Ecker, D., Gutell, R., Gautheret, D., Case, D.A., Sampath, R.: RNAmotif – a new RNA secondary structure definition and discovery algorithm. Nucleic Acids Research 29, 4724–4735 (2001)
13. Rivas, E., Eddy, S.R.: Secondary Structure Alone is Generally Not Statistically Significant for the Detection of Noncoding RNAs. Bioinformatics 16(7), 583–605 (2000)
14. Robin, S., Daudin, J.-J., Richard, H., Sagot, M.-F., Schbath, S.: Occurrence probability of structured motifs in random sequences. J. Comp. Biol. 9, 761–773 (2002)
15. Sagot, M.F., Viari, A.: Flexible identification of structural objects in nucleic acid sequences: palindromes, mirror repeats, pseudoknots and triple helices. In: Hein, J., Apostolico, A. (eds.) Combinatorial Pattern Matching. LNCS, vol. 1264, pp. 224–246. Springer, Heidelberg (1997)
16. Zhang, S., Borovok, I., Aharonovitz, Y., Sharan, R., Bafna, V.: A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements. Bioinformatics 22(14), e557–e565 (2006)