Searching RNA motifs and their intermolecular contacts with constraint networks.

Patricia Thebault, Simon de Givry, Thomas Schiex, Christine Gaspin

BIOINFORMATICS ORIGINAL PAPER Vol. 22 no. 17 2006, pages 2074–2080
doi:10.1093/bioinformatics/btl354

Genome analysis

Searching RNA motifs and their intermolecular contacts with constraint networks

P. Thébault[1,2], S. de Givry[1], T. Schiex[1] and C. Gaspin[1,*]

[1]Unité de Biométrie & Intelligence Artificielle INRA, Chemin de Borde Rouge, Auzeville, BP 52627, 31326 Castanet-Tolosan, France and [2]Plateforme Bioinformatique, INRA, Chemin de Borde Rouge, Auzeville, BP 52627, 31326 Castanet-Tolosan, France

Received on February 1, 2006; revised and accepted on June 23, 2006
Advance Access publication July 4, 2006
Associate Editor: Alex Bateman

ABSTRACT

Motivation: Searching RNA gene occurrences in genomic sequences is a task whose importance has been renewed by the recent discovery of numerous functional RNA, often interacting with other ligands. Even if several programs exist for RNA motif search, none exists that can represent and solve the problem of searching for occurrences of RNA motifs *in interaction* with other molecules.

Results: We present a constraint network formulation of this problem. RNA are represented as structured motifs that can occur on more than one sequence and which are related together by possible hybridization. The implemented tool **MilPat** is used to search for several sRNA families in genomic sequences. Results show that MilPat allows to efficiently search for interacting motifs in large genomic sequences and offers a simple and extensible framework to solve such problems. New and known sRNA are identified as H/ACA candidates in *Methanocaldococcus jannaschii*.

Availability: http://carlit.toulouse.inra.fr/MilPaT/MilPat.pl

Contact: milpat@toulouse.inra.fr

1 INTRODUCTION

Our understanding of the role of RNA has changed in recent years. Initially considered as being simply the messenger that converts genetic information from DNA into proteins, RNA is now seen as a key regulatory factor in many of the cell's crucial functions, affecting a large variety of processes including plasmid replication, phage development, bacterial virulence, chromosome structure, DNA transcription, RNA processing, development control and others [for review see Storz (2002)]. Consequently, the systematic search of non-coding RNA (ncRNA) genes, which produce functional RNAs instead of proteins, represents an important challenge.

Unlike double-stranded DNA, RNA is a single-stranded molecule. This characteristic allows different regions of the same RNA strand (or of several RNA strands) to fold together via base pair interactions to build structures that are essential for the biological function. The level of organization relevant for biological function corresponds to the spatial organization of the entire nucleotides chain and is called the tertiary structure. However, owing to the difficulty of determining high-order RNA structures, the RNA secondary structure is viewed as a simplified model of the RNA tertiary structure. In this article, we define the secondary structure of an RNA gene as the set of base-paired nucleotides which appear in the folded RNA, including possible bindings with other RNA molecules. This extends the usual definition which is often limited to planar structures (therefore excluding pseudo-knots and multiple helices) and is restricted to intra-sequence interactions. For functional RNA molecules, the secondary structure is generally conserved among members of a given family. Thus common structural characteristics can be captured by a signature that represents the elements which are conserved inside a set of related RNA molecules. We focus here on the problem of searching for new members of a gene family given their common signature. Solving this problem requires (1) to be able to formalize what a signature is and what it means for such a signature to occur in a sequence and (2) to design algorithms and data-structures that can efficiently look for such occurrences in large sequences. For sufficiently general signatures, this is an NP-complete problem (Vialette, 2004).

Traditionally, two approaches have been used for this problem: signatures can be modelled as stochastic context free grammars (excluding pseudo-knots or complex structures) and then searched using dynamic programming based parsers. This is, for example, used in Sakakibara *et al*. (1994) and Eddy and Durbin (1994) for RNA genes or in Bockhorst and Craven (2001) for terminators. Another approach defines a signature as a set of interrelated motifs. Occurrences of the signature are sought using pattern-matching techniques and exhaustive tree search. Such programs include RnaMot (Gautheret *et al*., 1990), RnaBob (Eddy, 1996), PatScan (Dsouza *et al*., 1997), Palingol (Billoud *et al*., 1996) and RnaMotif (Macke *et al*., 2001). Although these programs allow pseudo-knots to be represented, they have very variable efficiencies. Both types of approaches are restricted to single RNA signatures.

In this article, we clearly separate the combinatorial aspects from the pattern matching aspects by modelling a signature as a constraint network. This model captures the combinatorial features of the problem while the constraints use pattern matching techniques to enhance their efficiency. This combination offers an elegant and simple way to describe several RNA motifs in interaction and a general purpose efficient algorithm to search for all the occurrences of such motifs.

2 METHODS

The constraint network formalism (Rossi *et al*., 2006) is a powerful and extensively used framework for describing combinatorial search problems

*To whom correspondence should be addressed.

in artificial intelligence and operations research. Constraint networks allow to represent a problem as a set of inter-related variables in a very flexible way. Variables may have arbitrary domains and inter-relations are essentially arbitrary. This is usually well adapted to the definition of mathematical problems raised by molecular biology (Gaspin and Westhof, 1995; Muller *et al*., 1993; Altman *et al*., 1994; Major *et al*., 1991; Barahona and Krippahl, 2002) and has been used to model the structured motif search problem in Eidhammer *et al*. (2001), Policriti *et al*. (2004). The main differences with our approach lies in the type of constraints modelled [only distance constraints are considered in Policriti *et al*. (2004)] and in the propagation algorithms used [i.e. Eidhammer *et al*. (2001) used a much weaker propagation].

2.1 Constraint network

A constraint network (Rossi *et al*., 2003) is a triple (V, D, C):

- $V = \{x_1, \ldots, x_n\}$ is a set of n variables.
- $D = \{d_1, \ldots, d_n\}$ is a set of n domains, each containing the possible values for x_i. d denotes the size of the largest domain.
- C is a set of e constraints. Each constraint c_s is a relation over a subset $s \subset V$ of variables (called its scope) which defines the combinations of values (or tuples) that these variables may take. If the scope involves one, two or k variables, the constraint is said respectively unary, binary or k-ary.

A solution to a constraint network is an assignment of values from their domain to every variable, in such a way that every constraint is satisfied (only authorized value combinations are used). When a solution exists, the constraint network is said to be consistent.

2.2 Structured motifs as constraint networks

The elements that may characterize an RNA gene family are usually described in terms of the gene sequence itself (eg. it must contain some possibly degenerated pattern), the structures the sequence creates (loops, helices, hairpins and possible duplexes with other molecules) and by specifying how these various elements are positioned relatively to each other. An occurrence of such a structured motif on genomic sequences is defined by the positions of the various elements on the (possibly different) sequence(s), with correct relative positions.

A natural constraint network model emerges from this description: the variables of the network will represent the positions on the genomic sequence(s) of the elements of the description. More formally, each variable $x_i \in V$ ($y_m \in V$ is also used for clarity in order to distinguish between two interacting molecules) will represent a position on an associated sequence. The initial domain of variable x_i (y_m), unless otherwise stated, will therefore be equal to the size of the associated sequence. In order to represent information on required patterns, structures, and on relative positions of these elements, constraints will be used. To describe a constraint we separate the variables $x_i, \ldots, x_j, y_l, \ldots, y_m$ involved in the constraint and extra parameters $p_1, \ldots, p_k$ that influence the combinations of values authorized by the constraint. We now introduce the basic constraint types which are useful for RNA signature expression.

`content` [*word*, *error*, $type_{er}$](x_i). This constraint is satisfied if and only if some given pattern occurs at position x_i on the associated sequence. The pattern that must occur is specified by the following constraint parameters: *word* is a word on the IUPAC alphabet (http://www.iupac.org/dhtml_home.html); *error* specifies the maximum number of tolerated mismatches between an occurrence and the specified string; $type_{er}$ indicates if the error count is interpreted under the Hamming or edit distance metric (Smith and Waterman, 1981). An example of possible use of this constraint is illustrated in Figure 1 (case 1) where variable x_1 is constrained to a position where the `AGGGCUAGG` pattern appears precisely. An arrow indicates one occurrence.

`distance` [$l_{\min}$, $l_{\max}$](x_i,x_j). This binary constraint is used to enforce the relative position of elements. It is satisfied if and only if $l_{\min} \leq x_j - x_i \leq l_{\max}$.

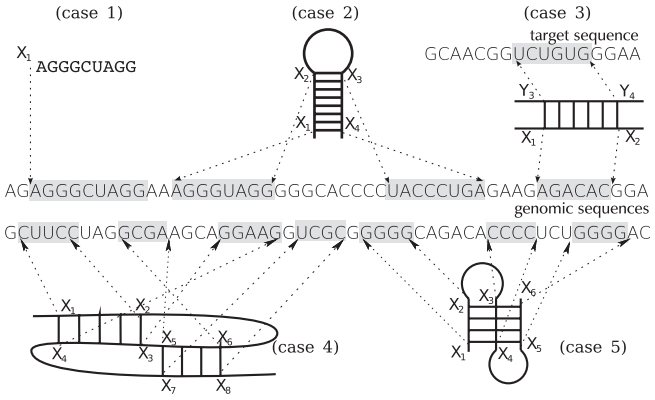

Fig. 1. Basic constraints. (case 1): occurrence of a pattern at one position. (case 2): an `helix` defined by two related segments separated by specified lengths. (case 3): a `duplex` composed of two independent substrings (from two sequences). (case 4): two `helix` constraints can describe a pseudo-knot or (case 5) a triple helix.

The positive integer parameters $l_{\min}$, $l_{\max}$ specify the bounds for the difference between the two variables.

`helix` [*rule*, *err*, $type_{er}$, $l_{\min}$, $l_{\max}$, $b_{\min}$, $b_{\max}$] (x_i, x_j, x_k, x_l). This constraint is used to enforce the existence of a generalized palindrome between the substrings delimited by $[x_i, x_j]$ and $[x_k, x_l]$ assuming that the four variables are related to the same sequence. Length and distances are also constrained. The constraint accepts the following parameters: *rule* is a binary relation on the RNA alphabet generalizing the usual pairing relation that defines when two characters are 'matching'. For an RNA helix one may use Watson–Crick (`A`–`U`, `G`–`C`) possibly extended with Wobble (`G`–`U`) pairing instead of equality relation. *err* gives the maximum number of tolerated mismatches between the two substrings. $type_{er}$ gives the Hamming or edit distance metric for error counts. $l_{\min}$, $l_{\max}$ represents the interval specifying possible lengths of the two substrings. $b_{\min}$, $b_{\max}$ represents the interval specifying the possible distance between the two substrings (i.e. $x_k - x_j$). This constraint is illustrated in Figure 1 (case 2), involving variables x_1, x_2, x_3 and x_4. Note that several such constraints can describe more complex structures like pseudo-knots [Figure 1 (case 4)] and triple helices [Figure 1 (case 5)].

`duplex` [$l_{\min}$, $l_{\max}$](x_i, x_j, y_k, y_l). This constraint enforces the existence of a (purely Watson–Crick based) duplex between the substrings delimited by $[x_i, x_j]$ and $[y_k, y_l]$ but without assuming that the two substrings belong to the same sequence. This constraint is used to model RNA–RNA interactions between different molecules with $l_{\min}$, $l_{\max}$ representing the interval specifying possible lengths of the two substrings. The constraint is illustrated in Figure 1 (case 3) involving x_1, x_2 (on one sequence) and y_3, y_4 (on another sequence). Arrows point to an occcurrence.

The flexibility of the constraint network representation using simply the four previous basic constraints can be illustrated on famous RNA gene families. A **tRNA** signature is represented in Figure 2A. tRNA genes include four helices corresponding respectively to A-stem (7 bp), D-stem (from 3 to 4 bp), C-stem (5 bp) and T-stem (5 bp), six loops corresponding respectively to the single strand between A-stem and D-stem (sequence UN with U invariant), D-loop (4–14 bases), the single strand between D-stem and C-stem (one base), C-loop (6–60 bases), the single strand between C-stem and T-stem (also called V-loop, 2–22 bases), T-loop (NUC). The corresponding constraint network is built from variables with 15 `distance` constraints (one constraint between each successive pair of variables), 2 `content` constraints and 4 `helix` constraints.

The same process can be applied to archaeal **H/ACA sRNA** signature. H/ACA sRNA are involved in a type of site-specific modification, the pseudouridylation (conversion of uridine into pseudouridine), within ribosomal RNA (rRNA). They exhibit complementarity to specific sites within rRNA

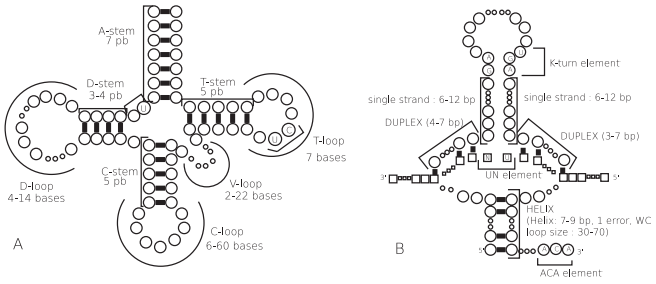

Fig. 2. (**A**) Signature of tRNA and (**B**) H/ACA genes family. Circles represent bases (small circles consolidate several bases); squares represent bases in the target rRNA; edges represent interactions between two bases.

sequences, thereby determining the site of modification. Archaeal H/ACA sRNA contain one or more hairpins connected by single stranded regions, all having similar characteristics. We have built a signature depicted in Figure 2B of such a consensus hairpin on the basis of known available H/ACA sRNA secondary structures in *Pyrococcus furiosus*, *Pyrococcus abyssi*, *Pyrococcus horikoshii* and *Archaeoglobus fulgidus* genomes (Rozhdestvensky *et al*., 2003). The signature identifies simultaneously a sequence representing one sRNA hairpin containing all the characteristics of both the sRNA and the sequence of the target able to form the interaction. The sRNA is described as containing the lower stem of the hairpin, an internal loop from which the two single stranded regions are able to form a duplex with the target, two single stranded regions corresponding to an irregular upper stem, a K-turn motif (Rozhdestvensky *et al*., 2003), a single strand corresponding to the loop of the hairpin and, at the 3′ end, an ACA box element. The target is described as containing two regions separated by 'UN' (U being the uridine which will be converted into a pseudouridine) and able to pair with H/ACA regions.

The corresponding constraint network is built from 16 variables (12 variables for the sRNA motif, 4 variables for the target motif) one `helix` constraint, two `duplex` constraints, 15 `distance` constraints and 4 `content` constraints.

2.3 Algorithms and implementation

The central problem on constraint networks is to prove the existence of a solution (consistency). This is an NP-complete problem usually solved by exhibiting one solution. This problem is most often tackled using sophisticated variants of *backtrack search*. In naive backtrack search, variables are assigned values one after the other. If at some point an unauthorized combination of values is used, backtracking goes back to the most recently assigned variable which still has alternative values available. With at most d values per variable, this means that backtrack explores a tree of size $O(d^n)$. For practical efficiency, *filtering algorithms* are used at each node of the tree. A filtering algorithm transforms a constraint network into an equivalent network (with the same set of solutions) which satisfies an additional *local consistency property*. Typically, a local consistency property ensures that some values (or combinations of) which do not participate in a solution are explicitly deleted. If the filtered problem has an empty domain, it has no solution and backtracking can occur more rapidly than in naive backtrack. A trade-off arises in the strength of the filtering used since stronger filtering is more expensive but may detect inconsistency earlier. Finally, variable and value ordering heuristics may drastically improve search performance in practice.

2.3.1 Data structure and filtering algorithms Compared with usual applications of the constraint network formalism, this one is characterized by its specific constraint types (except for the `distance` constraint which is a usual arithmetic constraint) and by the potentially huge domain size (the sequences handled can be complete genomic sequences of several million base pairs).

With such large domains, the naive backtracking scheme, which successively tries all values of a variable, would lead to a branching factor equal to the domain size and thereby would be difficult to manage in practice. Instead, our algorithm explores a binary tree. The root of the tree corresponds to the initial problem with no variable assigned. The two sons of a node are obtained by (left) assigning to one variable the first value of its domain and by (right) removing this value from the domain.

Another issue is the filtering algorithm used at each node. Traditional filtering algorithms such as arc consistency (Rossi *et al*., 2006) can delete any value in the domain of any variable which requires an $O(nd)$ space complexity to describe domains. This makes them unsuitable for problems with very large domains. A usual choice in this case is to describe domains as intervals $[lb, ub]$ (for lower bound and upper bound) which reduces the space complexity to a nice $2n$. This leads to a property called *bound consistency*.

Bound consistency. For simplicity, we define bound consistency assuming binary constraints. A variable x_i with domain $d_i = [lb_i, ub_i]$ is bound consistent w.r.t. constraint $c_{\{x_i,x_j\}}$ involving variables x_i and x_j if and only if $\exists w_1, w_2 \in d_j$ such that $(lb_i, w_1) \in c_{\{x_i,x_j\}}$ and $(ub_i, w_2) \in c_{\{x_i,x_j\}}$. In this case, w_1, w_2 are called the supports for the bounds of x_i on constraint $c_{\{x_i,x_j\}}$. A constraint is bound-consistent if it is bound consistent w.r.t. all the variables involving it. A constraint network is bound-consistent if all its constraints are bound-consistent. For constraints of largersencodi arities, the bounds must participate in at least one tuple that is authorized by the constraint and other domains. Compared to usual local consistencies, using an interval based representation for the domains with bound consistency saves not only space but may also save time since existence of a support needs only to be enforced on the bounds. The basic operation used to enforce bound consistency consists in directly computing for a variable x_i, involved in a constraint c, the largest interval $R(c, x_i) = [a, b]$ such that a and b have a support on the constraint c. $R(c, x_i)$ is called the *reduction operator* for the constraint c and variable x_i. To enforce bound consistency, for every variable x_i and every constraint $c \in C$ involving x_i, the domain d_i is reduced to $d_i \cap R(c, x_i)$. This is done repeatedly until no modification occurs (a fix-point is reached). For specific types of constraints, this reduction operator can be computed more efficiently than by checking for the existence of supports for successive values.

The arithmetic `distance` constraint $l_{\min} \leq x_j - x_i \leq l_{\max}$ is a classical example for which the reduction operator for x_i is defined by $[a, b] = [lb_j - l_{\max}, ub_j - l_{\min}]$. In this case, bound consistency can be enforced in time $O(ed)$ (Hentenryck *et al*., 1992). For other types of constraint, dedicated reduction operators are needed.

2.3.2 Dedicated reduction operators For each type of constraint, we developed reduction operators using appropriate pattern matching algorithms. Each constraint implementation is independent from rest of the system and can be described by the algorithm implementing the reduction operator of the constraint's relation. The reduction operator for the `distance` constraint has been described before. We now describe the operator for other constraints.

`content` [...](x_i). The lower bound of the domain of x_i can be updated to the first occurrence of the pattern after lb_i in the associated text. To find this occurrence, the algorithm of Baeza-Yates and Manber (Baeza-Yaltes and Gonnet, 1992; Wu and Manber, 1991) is used.

`helix` [...](x_i, x_j, x_k, x_l). Imagine we want to reduce the domain of variable x_i. The problem is to find the first helix (a support) that satisfies the parameters of the constraint. By first we mean the helix with the smallest position of the 5′ extremity of the first strand (pointed by x_i). The problem for helices (which can be seen as two related substrings) is more complex than for `content` since the two substrings are initially unknown. A naive approach that successively tries all possible positions for the first and second substrings is obviously quadratic. However, in our case, the distance between the regions where the substrings may appear is constrained by the length

parameters $b_{\min}$ and $b_{\max}$. Together with parameters l_{min} and $l_{\max}$, this makes the complexity of the naive approach linear in the text length.

`duplex` $[\ldots](x_i, x_j, y_k, y_l)$. This constraint differs from the previous one by the fact that there is no possible $b_{\min}$ and b_{max} parameters since the two interacting words do not necessarily appear on the same sequence. The previous naive approach is therefore impractical. We decided to use a specialized version of suffix-trees (Ukkonnen, 1992) that captures occurrences of patterns of bounded length. This data structure, called a *k*-factor tree (Allali and Sagot, 2004), allows to perform string search in time linear in the length of the pattern searched (independently of the text length). The data structure is built once and for all at the initialization of the constraint network, in time and space linear in the length of the text (Ukkonen, 1992). The associated reduction operator is only used when one of the two variables x_i or x_j is assigned. All the occurrences of the Watson–Crick reverse complement can then be efficiently found in the *k*-factor tree and used to update the bounds of the other variables (taking the position of the first and last possible occurrences as new bounds).

These reduction operators are quite expensive compared with the operators of the simple distance constraint. In order to avoid repeated useless applications of the reduction operator, once a support is found it is memorized and reused until one of the value in the support is removed. In this case, the support is lost and a new search for a valid support must be performed.

3 RESULTS

This approach has been implemented in a general purpose program called **MilPat**. MilPat is written in C++. The implementation of filtering using reduction operators makes the architecture of MilPat suitable for the simple addition of new constraint types. Indeed, since (1) the application of reduction operators for a constraint requires information only on its own variables and has effect only on them and (2) all the constraints attached to a variable can be checked independently, one after the other, in any order, it suffices to just implement a new constraint type with associated reduction operators to extend MilPat. We have implemented a simple motif definition language making MilPat accessible through a web interface at http://carlit.toulouse.inra.fr/MilPaT/MilPat.pl. MilPat has been tested on different RNA gene search problems in order to assess its efficiency and its modeling capacities. All the signatures used in this section are available on the website.

3.1 Efficiency

The tRNA genes are perhaps the best studied among RNAs; hence, they are very appropriate for a first benchmarking. The signature of tRNAs used here is deliberately a simple one that can be modeled in all existing general purpose tools. We have concentrated on finding sequences that can adopt a cloverleaf-like secondary structure within given ranges of stem and loop lengths and searched the *Escherichia coli* and *Saccharomyces cerevisiae* genomes with two different tRNA descriptors. The main differences between both series of descriptors are the existence of the CCA arm in the case of prokaryotes and the existence of an intron in the loop of stem3 (for *S.cerevisiae*).

We compare the time execution of MilPat with three other general purpose programs. Results are shown in Table 1. For each genome search test, every program gives the same number of solutions with a sensitivity close to 100%. Considering computing efficiency, three groups may be formed from the slowest to the fastest: (1) RnaMot and RnaMotif, (2) Patscan and MilPat with variable order A and (3) MilPat with optimized variable order B. It is well-known that variable assignment order may have a significant influence on efficiency. Without the optimized order, MilPat already has an execution time close to the most efficient program, PatScan. Just changing the order leads to an earlier pruning of the search tree and a considerably improved execution time.

Table 1. Comparison of cpu-time efficiency

Software used	*E.coli* [4,6 Mb]	*S.cerevisiae* [1207 Mb]
tRNA search		
PatScan	1 min 32	1 h 40
RnaMotif	4 s	8 h 40
RnaMot	2 min	92 h
MilPat (order A)	39 s	1 h 52
MilPat (order B)	39 s	20 min
Artificial motif search		
PatScan	15 s	40 s
RnaMotif	45 s	1 min 58 s
RnaMot	3 min 15 s	8 min 48 s
MilPat	13 s	40 s

For tRNA search, the order A used by MilPat orders variables following the 5′ to 3′ order in the structured motif. Order B is an optimized order following the first fail principle: most constrained variables are chosen first by the backtrack algorithm. For artificial sequence, we have inserted in the genomes the sequence AUGCCAAAAAAAAGG-CAUAAAAAAAAAAAAAUGCCAAAAAAAAGGCAUAAAAAAAAAAA repeated five times so as it can fold into a structure containing 10 hairpin loops with a stem size varying from 5 to 10 and a loop of size 8.

Additional tests have been performed by inserting in the genomes of *E.coli* and *S.cerevisiae* an artificial sequence which can fold into a complex motif that was then searched for. CPU times are consistent with those obtained for tRNA search.

3.2 Modeling capacities

Other sRNA have been chosen to evaluate the modeling abilities.

Type A and B bacterial RNase P. Larger sequences than tRNA, such as type A and type B bacterial RNase P have been searched. In our tests, the signature uses type A and type B bacterial RNase P conserved elements as described in (Massire *et al*., 1998). This signature includes P1, P2, P4, P12 conserved helices and the ACAGNRA and GUGNAA consensus sequences. Table 2 gives results on a few bacterial genomes. Remarkably, results show very good sensitivity and specificity.

C/D box sRNAs. In Archaea C/D box sRNA can be described by C (RUGAUGA), D′ (CUGA), C′(UGAUGA) and D (CUGA) boxes, and one or two duplexes the size of which is from 8 to 12 base pairs (Gaspin *et al*., 2000) which are located just before the D or D′ box. We have built a signature (available on MilPat web site) including these elements, with 1 mismatch allowed in each box. Table 3 shows that taking into account both duplexes increases specificity while decreasing sensitivity. Considering two errors instead of one in both C and C′ boxes improves sensitivity (data not shown).

H/ACA box sRNAs. Several studies have recently identified new H/ACA sRNA genes in Archaea (Klein *et al*., 2002; Rozhdestvensky *et al*., 2003; Schattner, 2002; Charpentier *et al*., 2005; Baker *et al*., 2005) and in Yeast (Schattner *et al*., 2004).

Table 2. Type A and type B bacterial RNase P

Genome	Genome size	# candidates	RNase P	Time (s)
Escherichia coli	4.64 Mb	1	yes	32.0
Yersinia pestis	4.7 Mb	1	yes	33.36
Bacillus subtilis	4.21 Mb	3	yes	33.09
Clostridium acetobutylicum	4.13 Mb	2	yes	22.25
Salmonella enterica	4.94 Mb	1	yes	30.0
Ralstonia solanacearum	5.81 Mb	1	yes	13.10
Walbachia pipientis wMel	1.27 Mb	2	yes	12.05
Staphylococcus aureus	2.84 Mb	3	yes	15.0
Shigella dysenteriae	4.55 Mb	1	yes	32.23

Each genome contains one RNase P.

Table 3. C/D box sRNA candidates

Genome	Without duplex (TP)	With D and D′ duplex (TP)
Pyrococcus abyssi	201 (57)	64 (46)
Pyrococcus horikoshii	216 (52)	80 (48)
Pyrococcus furiosus	217 (53)	62 (42)
Methanococcus jannaschii	295 (8)	31 (6)

Each line of the table indicates the number of candidates found when the duplexes formed with the rRNA target is either ignored or explicitly modeled. TP are True Positives.

Table 4. H/ACA sRNA candidates in *S.cerevisiae* chromosome XV

Target	Nb. cand.	snR5	snR8	snR9	snR31	snR35	snR36	snR181
18S	874	N	N	Y	Y	Y	Y	Y
25S	1237	Y	Y	Y	N	N	N	Y

H/ACA candidates. The complete sequence of chromosome XV was screened with 18S and 25S sequences as possible target sequences. Every known sRNA is found by MilPat. Y indicates a possible target.

In order to test the ability of MilPat to model interactions between different molecules, we performed a computational screen of the yeast chromosome XV for which 7 H/ACA sRNA are known and archaeal *Methanococcus jannaschii*, *Pyrococcus abyssi*, *Pyrococcus furiosus* and *Pyrococcus horikoshii* genomes for H/ACA box sRNA. A specific tool was recently developed for H/ACA sRNA finding in *S.cerevisiae* (Schattner *et al*., 2004). A corresponding signature (available on MilPat website) has been designed for this problem. Its main characteristics is that it contains only one hairpin. The results appear in Table 4. As done in (Schattner *et al*., 2004), a further classification of candidates based on the position of the modification and the minimum free energy would immediately lead to improved specificity.

For archaea, results are described in Tables 5 and 6: Table 5 shows how interaction modeling can considerably increase specificity for archaeal genomes and Table 6 presents a GC% based selection of the candidates obtained in Table 5 using interaction modeling. Some of our candidates have been previously proposed and some have been experimentally validated as sRNAs.

Table 5. H/ACA sRNA candidates

Genome	With interaction	Without interaction
P.furiosus [1.91 Mb]	100	1155
P.abyssi [1.77 Mb]	89	765
P.horikoshii [1.74 Mb]	148	820
M.jannaschii [1.74 Mb]	118	1586

H/ACA candidates respectively with (without) the duplex. Signatures are available on the web site. For each organism, complete genomic sequences were screened with 16S and 23S sequences as possible targets sequences.

Table 6. H/ACA sRNA candidates

Name (known as)	# HP with MilPat/# known HP	Identified in
P.furiosus [1.91 Mb]		
Pf-H/ACA-1 (Pf1, Pfu-sR9, sR9)	1/1	[1],[3]
Pf-H/ACA-2 (Pf3, hgcE)	1/2	[1],[3]
Pf-H/ACA-3 (Pf4)	1/1	[3]
Pf-H/ACA-4 (Pf6, hgcF)	1/2	[1],[3]
Pf-H/ACA-5 (Pf7, hgcG)	1/3	[1],[3]
Pf-H/ACA-6 (Pf9)	1/1	[3], [4]
P.abyssi [1.77 Mb]		
Pa-H/ACA-1 (Pab-sR9)	1/1	[1], [3]
Pa-H/ACA-2	1/2	[1]
Pa-H/ACA-3 (Pab-91)	1/1	[3],[5]
Pa-H/ACA-4	1/2	[1], [3]
Pa-H/ACA-5	2/3	[1], [3]
Pa-H/ACA-6	1/1	[3]
P.horikoshii [1.74 Mb]		
Ph-H/ACA-1 (Pho-sR9)	1/1	[1], [3]
Ph-H/ACA-2	1/2	[1]
Ph-H/ACA-3	1/1	[3]
Ph-H/ACA-4	1/2	[1], [3]
Ph-H/ACA-5	2/3	[1], [3]
Ph-H/ACA-6	1/1	[3]
M.jannaschii [1.74 Mb]		
Mj-H/ACA-1	1/unknown	no
Mj-H/ACA-2 (Mj2, hgcA, cbr1)	1/unknown	[2], [3]
Mj-H/ACA-3 (cnr7)	2/unknown	[2]
Mj-H/ACA-4 (Mj4, Mj9, cnr9)	1/unknown	[2], [3]
Mj-H/ACA-5 (cnr13)	1/unknown	[2]

H/ACA sRNA identified by using the signature depicted in Fig. 2(B). Names between parenthesis refers respectively to the different names found in the literature (first column). The second column gives the number of hairpins found by MilPat relative to the number of hairpins in known H/ACA sRNA. In the last column, [1], [2], [3], [4] and [5] refer to the papers of respectively, Rozhdestvensky *et al*., (2003), Schattner (2002), Klein *et al*., (2002), Baker *et al*. (2005) Charpentier *et al*. (2005). For each organism, target sequences were complete sequences of 16S and 23S.

For all *Pyrococcus* genomes and each known H/ACA sRNA, we have found one or more hairpins. For the *M.jannaschii* genome, we identified five potential H/ACA sRNA (see Fig. 3a), all having several possible targets (see target candidates via the web interface). Several secondary structures are possible according to the selected

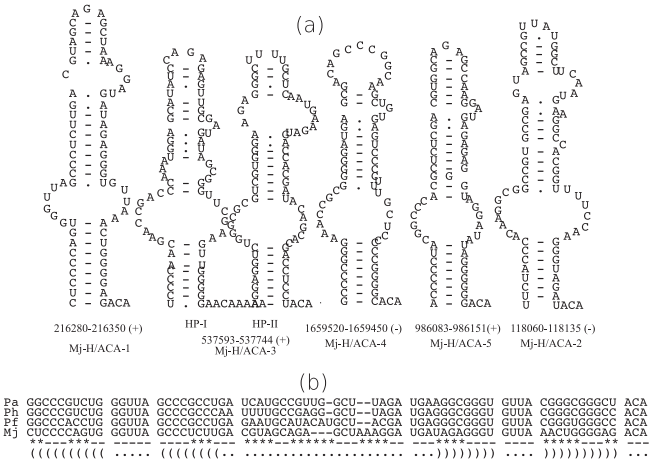

Fig. 3. (**a**) Secondary structures of proposed H/ACA sRNAs for *M.jannaschii*. (**b**) Alignment of Mj-H/ACA-1 with homologous sequences in the *Pyrococcus* genomes.

target. Only one secondary structure is represented for each candidate. Mj-H/ACA-1, was found as a new sRNA. Remarkably, this sRNA is reported neither in (Klein *et al*., 2002) or (Schattner, 2002). After looking at the 5′ and 3′ ends of the sequence, we found that Mj-H/ACA-1 contains only one hairpin which is homologous to the HP-III hairpin of *Pf*7 and is therefore a true positive (see Fig. 3B). Other candidates were already identified as sRNA but not annotated as H/ACA sRNA. Mj-H/ACA-2 was computationally identified in Schattner (2002) and Klein *et al*. (2002) and experimentally identified in Klein *et al*. (2002). It contains all the characteristics of an H/ACA archaeal sRNA. Mj-H/ACA-3 and Mj-H/ACA-5 also appear to be new H/ACA genes with respectively two hairpins and one hairpin. They are identified as sRNA (respectively as *cnr*7 and *cnr*13) in Schattner (2002), but were not experimentally identified. Mj-H/ACA-4, identified partially in Schattner (2002), computationally and experimentally identified in Klein *et al*. (2002) also presents all the characteristics of an H/ACA archaeal sRNA. Homologue sequences are found in the three *Pyrococcus* (Fig. 3C). They correspond to the *Pf* 9 H/ACA sRNA gene identified in Baker *et al*. (2005). The signatures we used provides a good example of a quick and efficient modeling of interacting molecules. One may note that processing one of the archaeal genomes and one rRNA target with the H/ACA signature typically takes <2# minutes on a 700 Mhz Athlon.

4 CONCLUSION

The aim of this work is to offer a way of describing new generations of RNA patterns, including the specification of complexes formed by interactions between different regions of a genome. The combination of constraint network methodology with pattern matching algorithms provides (1) an increased efficiency, (2) extended modeling capabilities for intermolecular interactions and (3) an easily extensible framework. First results show that MilPat is at least as efficient in CPU time as related tools and that, in archaea, modeling H/ACA sRNA motif in interaction with their target improves specificity without decreasing sensitivity. Beyond this, a number of evolutions are possible to improve MilPat efficiency and modeling capabilities, including the ability to describe optional or alternative motifs.

In its current version, MilPat provides all the true occurrences (satisfying all constraints). Future developments aim to offer a scoring system based on mismatches, thermodynamic or probabilistic parameters. Taking advantage of such information would require the use of more complex weighted constraint network algorithms (Larrosa and Schiex, 2004).

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their detailed reviews. The work of P.T. has been partially funded by Génoplante.

Conflict of Interest: none declared.

REFERENCES

Allali,J and Sagot,M.-F. (2004) The at most κ-deep factor tree. *Technical Report # 2004-03*, Institut Gaspard Monge, Université de Marne la Vallée.

Altman,R., Weiser,B. and Noller,H. (1994) Constraint satisfaction techniques for modeling large complexes: application to the central domain of 16s ribosomal RNA. In *Proceedings of the second international conference on Intelligent Systems for Molecular Biology,* AAAI Press, Menlo Park, pp. 10–18.

Baeza-Yaltes,R. and Gonnet,G. (1992) A new approach to text searching. *Communi. ACM'*, **35**, 74–82.

Baker,D. *et al.* (2005) RNA-guided RNA modification: functional organization of the archaeal H/ACA RNP. *Genes Develop*., **19**, 1238–1248.

Barahona,P. and Krippahl,L. (2002) PSICO: Solving protein structure with cconstraint programming and optimization. *Constraints*, **7**, 317–331.

Billoud,B. *et al.* (1996) Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res*., **24**, 1395–1403.

Bockhorst,J. and Craven,M. (2001) Refining the structure of a stochastic context-free grammar. In *Internation Conference in Artificial Intelligence,* pp. 1315–1322.

Charpentier,B. *et al*. (2005) Reconstitution of archaeal H/ACA small ribonucleoprotein complexes active in pseudourydilation. *Nucleic Acids Res*., **33**, 3133–3144.

Dsouza,M. *et al*. (1997) Searching for patterns in genomic data. *Trends Genet*., **13**, 497–8.

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

Eddy,S. (1996) Rnabob: a program to search for RNA secondary structure motifs in sequence databases. *Manual*. http://www.genetics.wustl.edu/eddy/software/#rnabob

Eidhammer,I. *et al*. (2001) A constraint based structure description language for biosequences. *Constraints*, **6**, 173–200.

Gaspin,C. and Westhof,E. (1995) An interactive framework for RNA secondary structure prediction with a dynamical treatment of constraints. *J. Mol. Biol*., **254**, 163–174.

Gaspin,C. *et al*. (2000) Archaeal homologs of eukaryotic methylation guide small nucleolar RNAs: lessons from the pyrococcus genomes. *J. Mol. Biol*., **297**, 895–906.

Gautheret,D. *et al*. (1990) Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Comput. Appl. Biosci*., **6**, 325–331.

Hentenryck,P. V. *et al*. (1992) A generic arc-consistency algorithm and its specializations. *Artif. Intell*., **57**, 291–321.

Klein,R. *et al*. (2002) Noncoding RNA genes identified in AT-rich hyperthermophiles. *Proc. Natl Acad. Sci. USA*, **99**, 7542–7547.

Larrosa,J. and Schiex,T. (2004) Solving weighted CSP by maintaining arc consistency. *Artif. Intell*., **159**, 1–26.

Macke,T. *et al*. (2001) RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Res*., **29**, 4724–4735.

Major,F. *et al*. (1991) The combination of symbolic and numerical computation for three-dimensional modeling of RNA. *Science*, **253**, 1255–1260.

Massire,C. *et al*. (1998) Derivation of the three-dimensional architecture of bacterial ribonuclease p RNAs from comparative sequence analysis. *J. Mol. Biol.*, **279**, 773–793.

Muller,G. *et al*. (1993) Automatic display of RNA secondary structures. *Cabios*, **9**, 551–561.

Policriti,A. *et al*. (2004) Structured motifs search. In *Proceedings RECOMB2004,* ACM Press, pp. 133–139.

Rossi,F., van Beek,P. and Walsh,T., (editors) (2006) *Handbook of Constraint Programming*. Elsevier.

Rozhdestvensky,T. *et al*. (2003) Binding of L7Ae protein to the K-turn of archaeal snoRNAs: a shared RNA binding motif for C/D and H/ACA box snoRNAs in Archaea. *Nucleic Acids Res.*, **31**, 869–877.

Sakakibara,Y. *et al*. (1994) Recent methods for RNA modeling using stochastic context-free grammars. Springer-Verlag, In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM '94)*, Springer-Verlag, pp. 289–306.

Schattner,P. *et al*. (2004) Genome-wide searching for pseudouridylation guide snornas: analysis of the *Saccharomyces cerevisiae* genome. *Nucleic Acids Res.*, **32**, 4281–4296.

Schattner,P. (2002) Searching for RNA genes using base-composition statistics. *Nucleic Acids Res.*, **30**, 2076–2082.

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

Storz,G. (2002) An expanding universe of noncoding RNAs. *Science*, **296**, 1259.

Ukkonen,E. (1992) Constructing suffix-trees on-line in linear time. Algorithms, Software, Architecture: Information Processing '92. Vol. 1, In *Proceedings of the IFIP 12th World Computer Congress*, Madrid, Spain, pp. 484–492.

Vialette,S. (2004) On the computational complexity of 2-interval pattern matching problems. *Theor. Comput. Sci.*, **312**, 223–249.

Wu,S. and Manber,U. (1991) Fast text searching with errors. *Report TR-91-11*, Department of Computer Science, University of Arizona, Tucson, AZ.