Combining constraint network processing and pattern matching to describe and locate structured motifs in genomic sequences*

P. Thébault, S. de Givry, T. Schiex, C. Gaspin

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, no one 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 occur on more than one sequence and which are related together by possible hybridization. Together with pattern matching algorithms, constraint satisfaction techniques have been implemented in a prototype **MilPat** and applied to search for tRNA and archaeal H/ACA sRNA genes on genomic sequences. Results show that these combined techniques allow to efficiently search for interacting motifs in large genomic sequences and offer a simple and extensible framework to solve such problems. New and known sRNA are proposed to be H/ACA candidates in *Methanocaldococcus jannaschii*.
Availability: http://carlt.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. Firstly 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 and modification, development control and others (for review see [28]). Consequently, the systematic search of non-coding RNA (ncRNA) genes, which produce functional RNAs instead of proteins, represents an important challenge.

*This is a preprint of a forthcoming Bioinformatics paper entitled "Searching RNA motifs and their intermolecular contacts with constraint networks".

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, due 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 paper, 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 structural 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 (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 [30] that combines combinatorial optimization and pattern matching issues.

Traditionally, two types of 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 relatively time consuming dynamic programming based parsers. This is e.g. used in [25, 12] for RNA genes or in [7] for terminators. Another approach defines a signature as a set of interrelated motifs. Occurrences of the signature are sought using simple pattern-matching techniques and exhaustive tree search. Such programs include RnaMot [15], RnaBob [11]), PatScan [10], Palingol [6] and RnaMotif [20]. Although these programs allow pseudo-knots to be represented, they have very variable efficiencies. Both types of approaches are restricted to single RNA molecule signatures.

In this paper, 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 occurrences of such motifs.

2 Methods

The constraint network formalism [9] is a powerful and extensively used framework for describing combinatorial search problems 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 (not limited to numerical problems) and inter-relations are essentially arbitrary. This is usually well adapted to the definition of mathematical problems raised by molecular biology [14, 22, 2, 21, 5] and has been used to model the structured motif search problem in [13, 23].

2.1 Constraints network

A constraint network [9] P is defined as a triple $P = (V, D, C)$ where :

- $V = \{x_1, ...x_n\}$ is a set of n variables.
- $D = \{d_1, ..., d_n\}$ is a set of n finite domains, each containing the possible values for x_i. We denote d 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 constraint network may be represented by an undirected hyper-graph whose vertices are variables, and hyper-edges represent constraint scopes. When the constraint network involves only binary constraints, this hyper-graph becomes a graph. 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 combinations of values are used). When such a solution exists, the constraint network is said to be consistent. Proving consistency is an NP-complete problem. Most algorithms for solving constraint networks rely on systematic search through the possible assignments of values to variables and are guaranteed to find a solution, if one exists. The most common algorithm for performing systematic search is *backtracking*. In the backtracking method, variables are assigned values one after the other. If a forbidden combination of values is used at some point, backtracking goes back to the most recently assigned variable which still has alternative values available. It is easily described as a depth-first search of a tree whose root corresponds to the original network. The sons of a node are obtained by choosing one unassigned variable in the father problem and by giving it all possible values (yielding a d branching factor in the worst case). This naive algorithm runs in $O(d^n)$. Because of its practical inefficiency and of the associated NP-completeness for constraint network consistency, so-called *filtering algorithms* have been introduced. 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 values which do not participate in a solution are explicitly deleted. The most preeminent local consistency property is called *arc consistency*.

Arc consistency

We introduce the definition of arc consistency for binary constraints, but this property can easily be generalized to constraints of arbitrary arity. Given a variable x_i and a binary constraint $c_{\{x_i,x_j\}}$ involving variables x_i and x_j (with domains d_i and d_j respectively), x_i is said to be arc consistent wrt. $c_{\{x_i,x_j\}}$ iff any choice of value for x_i can be extended to a pair on x_j in such a way that the constraint $c_{\{x_i,x_j\}}$ is satisfied. More formally: $\forall v \in d_i, \exists w \in d_j$ *such that* $(v,w) \in c_{\{x_i,x_j\}}$. In this case, we say that w is a support for v on constraint $c_{\{x_i,x_j\}}$. A variable is arc consistent if it is arc consistent wrt to all the constraints involving it. A constraint network P is said to be arc-consistent if all its variables are arc consistent. Obviously, a value $v \in d_i$ with no support on $c_{\{x_i,x_j\}}$ cannot participate in a solution and it can be removed, yielding a simpler and equivalent problem. If the value v deleted was the only support for another value u (on another constraint) then this value u can also be deleted. The process of deletion continues until a unique fix point is reached. This process is also called *constraint propagation*. If the domain of any variable becomes empty during this process, then the problem is known to be inconsistent (no solution). Otherwise a smaller equivalent problem is obtained.

A tradeoff arises in the choice of the propagation which is used inside the tree-search: stronger propagation methods are more expensive but may detect inconsistencies earlier; weaker propagation methods are generally cheaper to execute but may increase the size of the explored part of the tree.

Based on pragmatical observations, the level of constraint propagation which is considered as the best compromise is usually arc consistency. This leads to the so-called MAC algorithm (Maintaining Arc-Consistency) [9] which is one of the most efficient algorithm for constraint network solving. In this algorithm, constraints are propagated to achieve arc-consistency at each node of the tree search. If an empty domain is obtained at some point, backtracking occurs. Although still worst-case exponential, this algorithm is able to solve larger problems than the naive backtracking algorithm. Its practical efficiency is largely influenced by the order of assignment of the variables. To choose the next variable to assign, the informal "first fail principle" indicates the most constrained variable (small domain, involved in many, tight constraints) as a good choice in order to reduce the search tree size.

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.

A possible occurrence of such a structured motif on genomic sequences can be described by the positions of the various elements on the (possibly different) sequence(s) which are correctly located relatively to each other.

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* $x_i, ..., x_j, y_l, ..., y_m$ involved in the constraint and possible extra *parameters* $p_1, ..., p_k$ that influence the actual combination of values that are authorized by the constraint. Such a constraint will be denoted as $\texttt{name}[p_1, ..., p_k](x_i, ..., x_j, y_l, ..., y_m)$. We now introduce the basic constraint types which are useful for RNA signature expression:

$\texttt{content}[\mathsf{word}, \mathsf{error}, \mathsf{type_{er}}](x_i)$

: this unary constraint is satisfied iff 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; error specifies the maximum number of tolerated mismatches between an occurrence and the specified string; $\mathsf{type_{er}}$ indicates if the error count is interpreted under the Hamming or Levenshtein distance metric ([27]). An example of possible use of this constraint is illustrated in Fig. 1(1) where variable x_1 is constrained to a position where the AGGGCUAGG pattern appears precisely. An occurrence is indicated by the arrow.

$\texttt{distance}[\mathsf{l_{min}}, \mathsf{l_{max}}](x_i, x_j)$

: this binary constraint is used to enforce the relative position of elements. It is satisfied iff $\mathsf{l_{min}} \leq x_j - x_i \leq \mathsf{l_{max}}$. The positive integer parameters $\mathsf{l_{min}}, \mathsf{l_{max}}$ specify the bounds for the difference between the two position variables.

$\texttt{helix}[\mathsf{rule}, \mathsf{error}, \mathsf{type_{er}}, \mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{b_{min}}, \mathsf{b_{max}}](x_i, x_j, x_k, x_l)$

:

this 4-ary 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 equality relation that defines when two characters are considered as "matching" in the palindrome.

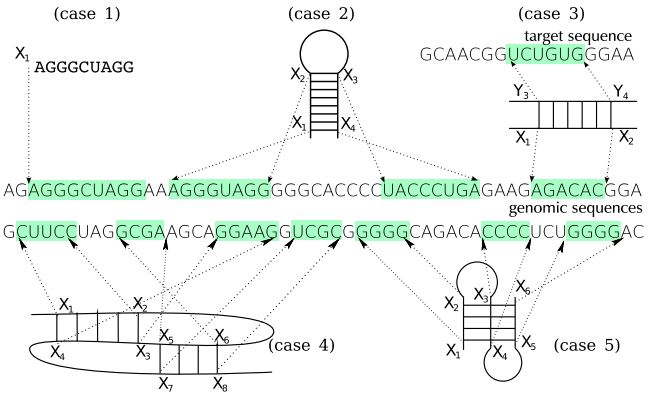

Figure 1: Basic constraints. (1): occurrence of a pattern at one position. The constraint graph contains one vertex with one unary constraint represented by an edge. (2): an hairpin loop defined by two related segments separated by specified lengths. The constraint graph contains four vertices, three implicit`distance` constraints represented by edges and one `helix` constraint represented by an hyper-edge connecting all four vertices with a rectangle in the middle. (3): a duplex composed of two independent substrings (from two sequences). The constraint graph is similar to the previous one (one `distance` constraint is removed). (4): two `helix` constraints can describe a pseudo-knot or (5) : a triple helix.

For an RNA helix one may use Watson-Crick (`A-U`, `G-C`) possibly extended with Wobble (`G-U`) pairing instead of equality relation. error gives the maximum number of tolerated mismatches between the two substrings. $\mathsf{type_{er}}$ gives the Hamming or Levenshtein distance metric for error counts. $\mathsf{l_{min}}, \mathsf{l_{max}}$ represents the interval specifying possible lengths of the two substrings. $\mathsf{b_{min}}, \mathsf{b_{max}}$ represents the interval specifying the possible distance between the two substrings (*i.e.*, $x_k - x_j$). This constraint is illustrated in Fig. 1(2), involving variables x_1, x_2, x_3 and x_4.

$\texttt{duplex}[\mathsf{l_{min}}, \mathsf{l_{max}}](x_i, x_j, y_k, y_l)$

: this 4-ary constraint is used to enforce the existence of a (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. Only Watson-Crick pairing is considered. This constraint is used to model RNA-RNA interactions between different molecules with $\mathsf{l_{min}}, \mathsf{l_{max}}$ representing the interval specifying possible lengths of the two substrings. The constraint is illustrated in Fig. 1(3) involving x_1, x_2 (on one sequence) and y_3, y_4 (on another sequence). An occurrence is indicated by the arrows.

Note that several such constraints can describe more complex structures like pseudo-knots (Fig. 1(4)) and triple helices (Fig. 1(5)).

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 Fig. 2(A). tRNA genes include four helices corresponding respectively to A-stem (7 base pairs), D-stem (from 3 to 4 base pairs), C-stem (5 base pairs) and T-stem (5 base pairs), six loops corresponding respectively to the single strand between A-stem and D-stem (sequence UN with U invariant), D-loop (4 to 14 bases), the single strand between D-stem and C-stem (one base), C-loop (6 to 60 bases), the single strand between C-stem and T-stem (also called V-loop, 2 to 22 bases), T-loop (NUC).

The corresponding constraint network is built from 16 variables Fig. 2(C)). the variable numbering follows the 5' $\rightarrow$ 3' orientation) 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) targets. They exhibit complementarity to specific sites within rRNA 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 Fig. 2(B) 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 [24]. 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

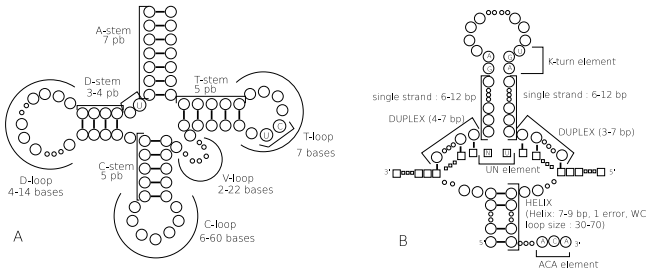

Figure 2: (A) Signature of tRNA and (B) H/ACA genes family. Circles represent bases ; edges represent interactions between two bases. (C) Modelisation by a constraint network hypergraph of tRNA and (D) H/ACA sRNA. Ellipses represent variables ; edges represent constraints. Dashed edges represent `content` constraints, thick hyperedges represent `helix` and `duplex` constraints. Other edges represent `distance` constraints.

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 [24], 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 (see Fig. 2(D)).

2.3 Algorithms and implementation

The problem is to find all solutions to the network. Compared to usual applications of the constraint network formalism, this one is characterized by the potential huge domain size (a genomic sequence is possibly a pseudo-molecule) and its specific constraint types (except for the `distance` constraint which is a usual arithmetic constraint).

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. Constraint propagation is done at each node in order to detect inconsistent problems earlier and limit the size of the tree explored.

Data structure and propagation algorithms

Traditional constraint propagation such as arc consistency enforcing can delete any value in any domain and has therefore a nd space complexity just to remember which values are deleted or not. In our case, this is impractical. A usual simple choice in this case is to describe domains as intervals $[lb, ub]$ (for lower bound and upper bound). This reduces the space complexity to a nice $2n$. The side-effect of this representation is that complete arc consistency cannot be enforced anymore in the general case. Indeed, if a value $a \in]lb, ub[$ has no support on a constraint, it cannot be deleted because there is no interval representation of the domain obtained. Thus only bounds without support can be deleted. This weakening form of arc consistency is called bound-consistency ([19]).

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 wrt. constraint $c_{\{x_i,x_j\}}$ involving variables x_i and x_j iff $\exists w_1, w_2 \in d_j$ 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 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 wrt. all the variables involving it. A constraint network P is bound-consistent if all its constraints are bound-consistent. For constraints of larger arities, the bounds must participate in at least one tuple that is authorized by the constraint and other domains. Using both an interval based representation for the domains and bound consistency saves space by representing each domain by only two integers and may also save time since existence of a support needs only to be enforced on the bounds. The basic operation used to perform bound consistency propagation 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. A classical example of this is the arithmetic `distance` constraint $\mathsf{I}_{\mathsf{min}} \leq x_j - x_i \leq \mathsf{I}_{\mathsf{max}}$ where the reduction operator for x_i is defined by $[a, b] = [lb_j - \mathsf{I}_{\mathsf{max}}, ub_j - \mathsf{I}_{\mathsf{min}}]$. In this specific case, it has been shown that bound consistency propagation is equivalent to arc consistency propagation and that it can be enforced in time $O(ed)$ [16]. For other types of constraint, dedicated reduction operators are written.

Dedicated constraint propagation

For each type of constraint, we developed specific reduction operators using appropriate pattern matching algorithms. Each type of constraint may be

considered as a black box isolated from the constraint network system itself. Then it can be simply described by a specific algorithm implementing reduction operator of the encoded 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 obviously 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 [3, 31] is used. This algorithm is based on a boolean representation of the search state and exploits the intrinsic parallelism of bitwise logical operations in modern CPU.

- `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. This makes it impossible to use string matching algorithms relying on a preprocessing of the substring searched. 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 $\mathsf{b}_{\mathsf{min}}$ and $\mathsf{b}_{\mathsf{max}}$. Together with parameters $\mathsf{l}_{\mathsf{min}}$ and $\mathsf{l}_{\mathsf{max}}$, this makes the complexity of the naive approach linear in the text length.

- `duplex`[...](x_i, x_j, y_k, y_l): this constraint differs from the previous one by the precise fact that there is no possible $\mathsf{b}_{\mathsf{min}}$ and $\mathsf{b}_{\mathsf{max}}$ parameters since the two words interacting 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 [29] that captures occurrences of patterns of bounded length. This data structure, called a *k-factor tree* [1] allows to perform string search in time linear in the length of the pattern searched (independently of the text length). The data structureis built once and for all at the initialization of the constraint network, in space and time linear in the length of the text [29]. 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 constraint propagations are quite expensive compared to 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 **MilPat** : **M**otifs and **I**nter-mo**L**ecular motifs searching tool using cs**P** form**A**lism and solving **T**echniques. MilPat is written in C++. The implementation of reduction operators as constraint propagation makes the architecture of MilPat suitable for the simple addition of new constraint types. Indeed, since (i) the propagation of a constraint requires information only on its own variables and has effect only on them and (ii) 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. A minimal language written in *lex* and *yacc* allows to describe motifs. It is possible to run MilPat by using a web interface at http://carlit.toulouse.inra.fr/MilPaT/MilPat.pl.

We tested our approach on different RNA gene search problems in order to assess its efficiency and modeling capacities.

3.1 tRNAs

The tRNA structure and sequence profiles 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. We 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 *Saccharomyces cerevisiae*).

We compared the time execution of MilPat with three other general purpose programs. The tRNA signature used in our comparisons is from Gautheret and *al.* [15]. It includes four helices constraints, 15 distance constraints and 2 content constraints (see Fig. 2). The results of this comparison are shown in Table 1. For each genome search test, every program gives the same number of solutions with a sensitivity close to 100% (in both cases, descriptors miss one tRNA : (i) the selenocysteine tRNA for *E. coli* and (ii) an arginine tRNA for *S. cerevisiae*).

This general descriptor was not optimized to provide the best correct hits or lowest false-positive rates across both organisms. In fact, more specific rules for mispairs, sequence conservation, and lengths of regions could conceivably be included to provide a better specificity. However, this simple descriptor provides a straightforward signature of the structural motif we are looking for, and is a good comparison basis for the tools as the motif can be identically represented in any language (specific to each software).

On the computing efficiency basis, three groups may be formed from the slowest to the fastest: (i) RnaMot and RnaMotif, (ii) Patscan and Milpat with variable order A, and (iii) MilPat with variable order B. It is well known

Software	*E. coli* (4,6 Mb)	*S. cerevisiae* (12,07 Mb)
PatScan	1 min. 32	1 h 40
RnaMotif	4 s.	8h40
RnaMot	2 min.	92 h
MilPat (order A)	39 s	1 h52
MilPat (order B)	39 s	20 min.

Table 1: Comparison of the time efficiency: 545 solutions are found for the *E. coli genome* and 849982 for the *S. cerevisiae* genome. The huge number is due to the allowed flexibility of the descriptors (for example, the specification of an intron for the *S. cerevisiae* genome). A way to get a better specificity for tRNAs search, is to observe that : (i) the total number of allowed mismatches (5 for 4 helices) is higher than the number usually found in tRNAs (2 mismatches), (ii) each helix can contain one (or 2 for one of them) mismatch(es).

that variable assignment order may have a significant influence on efficiency. The order A used by MilPat consists in ordering variables according to the topological order of the elements in the structured motif, from 5' to 3'. Order B is an optimized order following the first fail principle: most constrained variables are chosen first by the backtrack algorithm. Without this 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.

3.2 H/ACA sRNAs

Several studies have recently identified new H/ACA sRNA genes [17, 24, 26, 8, 4]. In order to test the ability of MilPat to model interactions between different molecules, we performed a computational screen of the archaeal *M. Jannashii*, *P. abyssi*, *P. furiosus* and *P. horikoschii* genomes for H/ACA sRNA. Results are described in Table 2. In *Pyrococcus* genomes, we have tested the ability of MilPat to locate known H/ACA sRNA. In *M. Jannashii*, we have selected only GC-richest candidate sequences.

H/ACA sRNA in *Pyrococcus*

In all the *Pyrococcus* genomes, for each known H/ACA sRNA, we have found one or more hairpins according to the signature used. *Pab91*, identified and annotated as an H/ACA sRNA in [8], appears to be the homologue of *Pf4* [17]. Other hairpins of H/ACA known sRNA were not detected because of a strict description of both the K-turn motif and the duplex. This latter case is to relate to the suffix tree modelisation of the `duplex` constraint which, in the current version of MilPat, is able to build only consecutive Watson-Crick base pairs.

Name (known as)	Nb HP MilPat /Nb known	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. horikoschii (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. jannashii (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]

Table 2: H/ACA sRNA candidates.: 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 litterature (first column). The second column gives the number of hairpins found by MilPat relatively 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 [24, 26, 17, 4, 8]

H/ACA sRNA in M. jannaschi

We identified five potential H/ACA sRNA in *M. jannaschi* (see Figure 3(a)), all having several possible targets (see target candidates via the web interface). Several secondary structures would be possible according to the selected targets. Only one secondary structure is represented for each candidate.

Mj-H/ACA-1, the homologue sequence of *Pf7* was found as a new sRNA. Remarkably, this sRNA is reported neither in [17] or [26]. After looking at the

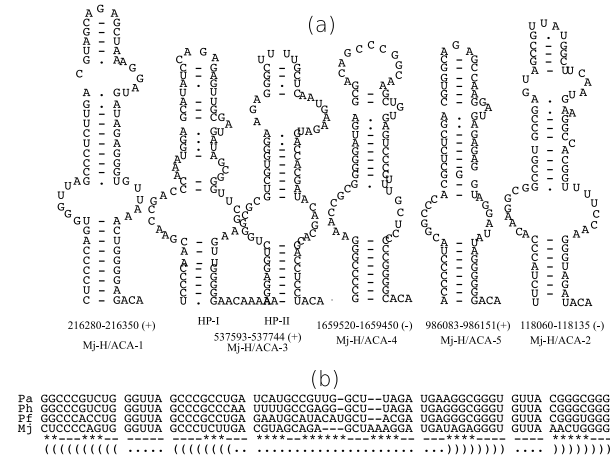

Figure 3: (a) Secondary structures of proposed H/ACA sRNAs for *M. jannaschi*. (b) Alignment of Mj-H/ACA-1 with homologue sequences in the Pyrococcus genomes.

5' and 3' ends of the sequence, it appears that Mj-H/ACA-1 contains only one hairpin, homologue to the HP-III hairpin of *Pf7* (see Figure 3(c)).

Other candidates were already identified as sRNA but not annotated as H/ACA sRNA. Mj-H/ACA-2 was computationnally identified as a sRNA in [26, 17] and experimentally identified in [17]. 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 *cnr7* and *cnr13*) in [26] but were not experimentally identified. Mj-H/ACA-4, identified partially in [26], computationnally and experimentally identified in [17] also presents all the characteristics of an H/ACA archaeal sRNA. Homologue sequences are found in the three Pyrococcus (see Figure 3(b)). They correspond to the *Pf9* H/ACA sRNA gene identified in [4].

In our search, the signature was not optimized to provide all the available hits. More specific rules for a K-turn degenerated motif, mispairs in the helix and duplex, possible bulges and different lengths of regions could be included to provide better sensitivity. However, the simple signature we used provides a good example of a quick and efficient modelisation of interacting molecules. Processing one of the archaeal genomes and one rRNA target with the H/ACA signature typically takes less than two minutes on a 700Mhz Athlon.

4 Conclusion

The main aim of our work is to offer a way of describing new generations of RNA patterns, including the specification of complexes formed by antisense

interactions between different regions of a genome. The combination of constraint network methodology together with efficient pattern matching data structures and algorithms provides 1) an increased efficiency, 2) extended modeling capabilities for intermolecular interactions and 3) an easily extensible framework.

Beyond this ability to describe inter and intra-molecular interactions with a great flexibility, a number of evolutions are possible to improve MilPat efficiency and modeling capabilities, including the ability to describe optional or alternative motifs. Within the framework of the biological application, these possibilities are essential to be closer to the structural reality of the molecules.

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 the deal of such information would require the use of more complex weighted constraint network algorithms [18].

References

[1] J. Allali and M-F. Sagot. The at most k-deep factor tree. Technical Report #2004-03, Institut Gaspard Monge, Université de Marne la Vallée, 2004.

[2] RB Altman, B Weiser, and HF Noller. 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*, pages 10–18, 1994.

[3] R Baeza-Yaltes and GH Gonnet. A new approach to text searching. In *Communications of the ACM*, volume 35, pages 74–82, 1992.

[4] DL Baker, OA Youssef, MIR Chastkofsky, DA Dy, RM Terns, and MP Terns. RNA-guided RNA modification: functional organization of the archaeal H/ACA RNP. *Genes & development*, 19:1238–1248, 2005.

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

[6] B Billoud, M Kontic, and A Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res*, 24(8):1395–403, 1996.

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

[8] B Charpentier, S Muller, and C Branlant. Reconstitution of archaeal H/ACA small ribonucleoprotein complexes active in pseudourydilation. *Nucleic Acids Res.*, 33(10):3133–3144, 2005.

[9] R Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[10] M Dsouza, N Larsen, and R Overbeek. Searching for patterns in genomic data. *Trends Genet*, 13(12):497–8, 1997.

[11] SR Eddy. Rnabob: a program to search for RNA secondary structure motifs in sequence databases. Manual, 1996.

[12] SR Eddy and R Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Res*, 22(11):2079–88, 1994.

[13] I Eidhammer, D Gilbert, I Jonassen, and M Ratnayake. A constraint based structure description language for biosequences. *Constraints*, 6:173–200, 2001.

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

[15] D Gautheret, F Major, and R Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Comput Appl Biosci*, 6(4):325–31, 1990.

[16] V Hentenryck, P, Y Deville, and M Teng, C. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2-3):291–321, 1992.

[17] RJ Klein, Z Misulovin, and SR Eddy. Noncoding RNA genes identified in AT-rich hyperthermophiles. *PNAS*, 99:7542–7547, 2002.

[18] J. Larrosa and T. Schiex. Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence*, 159:1–26, 2004.

[19] O Lhomme. Consistency techniques for numeric CSPs. In *International Joint Conference in Artificial Intelligence*, pages 232–238, 1993.

[20] TJ Macke, DJ Ecker, RR Gutell, D Gautheret, DA Case, and R Sampath. RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Res*, 29(22):4724–35, 2001.

[21] F Major, M Turcotte, D Gautheret, G Lapalme, E Fillion, and R Cedergren. The combination of symbolic and numerical computation for three-dimensional modeling of RNA. *Science*, 253:1255–1260, 1991.

[22] G Muller, C Gaspin, A Etienne, and E Westhof. Automatic display of RNA secondary structures. *Cabios*, 9(275):551–561, 1993.

[23] A Policriti, N Vitacolonna, M Morgante, and A Zuccolo. Structured motifs search. In *RECOMB2004*, pages 133–139. ACM Press, 2004.

[24] TS Rozhdestvensky, TH Tang, IV Tchirkova, J Brosius, JP Bachellerie, and A Httenhofer. 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(3):869–877, 2003.

[25] Y Sakakibara, M Brown, R Hughey, IS Mian, K jölander, RC Underwood, and D Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *CPM '94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 289–306. Springer-Verlag, 1994.

[26] P Schattner. Searching for RNA genes using base-composition statistics. *Nucleic Acids Res.*, 30(9):2076–2082, 2002.

[27] TF Smith and MS Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1), 1981.

[28] G Storz. An expanding universe of noncoding RNAs. *Science*, 296(5571):1259, 2002.

[29] E Ukkonen. Constructing suffix-trees on-line in linear time. In *Algorithms, Software, Architecture: Information Processing 92*, pages 484–492, 1992.

[30] S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theor. Comput. Sci.*, 312(2-3):223–249, 2004.

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