A Constraint Model for Constrained Hidden Markov Models: a first Biological Application*

Henning Christiansen, Christian Theil Have,
Ole Torp Lassen, and Matthieu Petit

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
{henning, cth, otl, petit@ruc.dk}

Abstract. A Hidden Markov Model (HMM) is a common statistical model which is widely used for analysis of biological sequence data and other sequential phenomena In the present paper we extend HMMs with constraints and show how the familiar Viterbi algorithm can be generalized, based on constraint solving methods. HMMs with constraints have advantages over traditional ones in terms of more compact expressions as well as opportunities for pruning during Viterbi computations. We exemplify this by an enhancement of simple prokaryote gene finder given by an HMM.

1 Introduction

Hidden Markov Models (HMMs) are one of the most popular models for analysis of sequential processes taking place in a random way, where "randomness" may also be an abstraction covering the fact that a detailed analytical model for the internal matters are unavailable. Such a sequential process can be observed from outside by its emission sequence (letters, sounds, measures of features, all kinds of signals) produced over time, and a HMM postulates a hypothesis about the internal machinery in terms of a finite state automaton equipped with probabilities for the different state transitions and single emissions. Decoding or prediction for a given observed sequence means to compute the most probably state transitions that the HMM can go through to produce the sequence, and thus this represents a best hypothesis for the internal structure or "content" of the sequence. HMMs are widely used in speech recognition and biological sequence analysis [8, 1].

Gene prediction aims at algorithmically identifying stretches of a DNA sequence that are biologically functional, in particular protein-coding genes but also other functional elements such as RNA genes [16]. Several HMM based gene finders have been proposed for gene prediction, including [3, 6, 5]. Decoding

* This work is supported by the project "Logic-statistic modeling and analysis of biological sequence data" funded by the NABIIT program under the Danish Strategic Research Council.

an "observed" DNA sequence using an HMM produces a state sequence, that appears as an annotation that identifies regions of genes and non-genes. The automaton defines the regular language for these annotations [12].

With the usual decoding algorithms, such as the Viterbi algorithm [14], it is difficult to add prior knowledge to an HMM about, say, verified coding regions in a specific sequence, or other side-constraints (e.g., this-and-this subsequence cannot occur in a coding region). For instance, fixing a known coding region at a given position n would require to modify the HMM so it is guaranteed to in this state after n iterations. This HMM transformation may require exponentially many new states.

In this paper, we focus on an extension of HMMs, called Constrained HMMs (CHMMs). The concept of CHMMs seems to introduced by Sato et al. in [10], although earlier and unrelated systems have used the same or similar names (commented on below). CHMMs restrict the set of allowed state and emission sequences (runs) by the addition of constraints to a standard HMM. The contribution of this paper is to introduce this framework into Constraint Programming. A constraint model is proposed to represent the allowed "runs". With this model, decoding essentially becomes a constraint optimization problem. We adapt the Viterbi algorithm to take into account such constraints using constraint solving techniques, and we exemplified it for enhancing an HMM based prokaryote gene finder by constraints that state existence of already known genes,

The paper is organized as follows: section 2 describes the background on HMM required to understand the rest of the paper and exemplifies it with a simple gene finder. In section 3, the constraint model for CHMM is described. Finally, section 4 presents related works and our plans for further work.

2 Background

Here we present Hidden Markov Models (HMMs) and the Viterbi algorithm so we can adapt them later with constraints.

2.1 Hidden Markov Model

For simplicity of the technical definitions, we limit ourselves to discrete first order HMMs with a distinguished initial state and no explicit final state (i.e., any state is final); the generalization to more initial states is straightforward.

Definition 1. *A* Hidden Markov Model *(HMM) is a 4-tuple* $\langle S, A, T, E \rangle$*, where*

- S *is a set of* states *which includes an* initial *state referred to as* $s^{(0)}$*;*
- A *is a finite set of emission symbols, individually denoted* e_i*;*
- T *is a set of* transition probabilities $\{p(s_i; s_j)\}_{s_i \in S}$ *representing the probability to transit from one state to another. For such* s_i*,* $\sum_{s_j \in S \setminus \{s_0\}} p(s_i; s_j) = 1$*.*
- E *is a set of* emission probabilities $\{p(s_i; e_j)\}_{s_i \in S \setminus \{s_0\}}$ *representing the probability of emitting symbol from a state. For such* s_i*,* $\sum_{e_j \in E} p(s_i; e_j) = 1$*.*

A run *of a HMM is defined as a pair consisting of a sequence of states* $s^{(0)} s^{(1)} \dots s^{(n)}$*, called a* path *and a corresponding sequence of emissions* $e^{(1)} \dots e^{(n)}$ *an called* observation*, such that*

– $\forall i > 0, p(s^{(i)}; s^{(i+1)}) > 0$ *(probability to transit from* $s^{(i)}$ *to* $s^{(i+1)}$ *);*
– $\forall i > 0, p(s^{(i)}; e^{(i)}) > 0$ *(probability to emit* $e^{(i)}$ *from* $s^{(i)}$*).*

The probability *of such a run is defined as* $\prod_{i=1..n} p(s^{(i-1)}; s^{(i)}) \cdot p(s^{(i)}; e^{(i)})$.

HMMs are commonly used to find the probability of a given observation, decoding the path corresponding to an observation and finally finding the model probabilities that maximizes the likelihood of generating a given set of observations. In this paper, we will only explain the decoding computation using the Viterbi algorithm [14].

2.2 The Viterbi Algorithm

The Viterbi algorithm [14] is a dynamic programming algorithm for finding a most probable path corresponding to a given observation. The algorithm keeps track of, for each prefix of an observed emission sequence, the most probable (partial) path leading to each possible state, and extends those step by step into longer paths, eventually covering the entire emission sequence.

We present the algorithm here as a rewriting system on a set of 4-tuples Σ, each representing a (potentially most probable) path for such prefixes; a fixed emission sequence $e^{(1)} \cdots e^{(n)}$ is assumed to be given. Each such 4-tuple is of form $\langle s, i, p, \pi \rangle$ where π is a partial path ending in state s and representing a path corresponding to the emission sequence prefix $e^{(1)} \cdots e^{(i)}$; p is the collected probability for the emissions and transitions applied in the construction of π.

The algorithm can be described by the two rewriting rules given by **Fig.** 1. The *trans* rule expands an existing partial path one step in each possible

$$
\begin{array}{ll}
\mathit{trans}: & \Sigma := \Sigma \cup \{\langle s', i{+}1, p \cdot p(s; s') \cdot p(s'; e^{(i+1)}), \pi\, s' \rangle\} \\
& \text{whenever } \langle s, i, p, \pi \rangle \in \Sigma \text{ and } p(s; s'), p(s'; e^{(i+1)}) > 0 \\
& \text{and } \mathit{prune} \text{ does not apply.} \\
\\
\mathit{prune}: & \Sigma := \Sigma \setminus \{\langle s, i{+}1, p', \pi' \rangle\} \\
& \text{whenever } \langle s, i{+}1, p, \pi \rangle, \langle s, i{+}1, p', \pi' \rangle \in \Sigma, \\
& p \geq p'.
\end{array}
$$

Fig. 1. Rewriting rules for the Viterbi algorithm for traditional HMMs

direction whereas *prune* removes those that are not optimal up to a given state; the condition that *trans* cannot apply in case *prune* is possible, ensures that no non-optimal partial path is expanded. The rules are expected to execute as long as possible, except that *trans* is only applied when it adds a 4-tuple to Σ that has not been added before. We take the following correctness property for granted.

Proposition 1. *Assume a HMM H with the notation as above and an observation* $Obs = e^{(1)} \cdots e^{(n)}$*. When the Viterbi algorithm* FIG. 1 *is executed from*

an initial set of 4-tuples $\{\langle s_0, 0, 1, \epsilon\rangle\}$, ϵ *being the empty path and* s_0 *the initial state of* H*, it terminates with a set of 4-tuples* Σ_{final}*. It holds that*

- *For any* $\langle s, n, p, \pi\rangle \in \Sigma_{final}$, π *is a most probable path for Obs ending in* s *and with probability* p*.*
- *Whenever there exists a path for Obs ending in* s, Σ_{final} *includes a 4-tuple of the form* $\langle s, n, p, \pi\rangle$*.*

The algorithm can run in time linear in the length of the given emission sequence times a quadratic factor of the number of states in the HMM; the latter is thus constant for a specific HMM.

2.3 An example HMM: a simple gene finder

An an example of an HMM that we later extend with constraints, we consider the problem of identifying protein coding genes in prokaryotes. A DNA sequence is composed of molecules, called *nucleotides*, represented by the four letters `a`, `c`, `t` and `g`. Some parts of a DNA sequence code for genes, called *coding regions*, while other parts do not and are called *non coding regions*. Coding regions contain a number of *codons*, triplets of nucleotides, each coding for an amino acid in a protein (to be produced by the gene). For prokaryotes, a coding region is contiguous, and it begins with a specific *start codon*, which is often `atg`, and ends with a *stop codon*, which is one of `taa`, `tga` or `tag`.

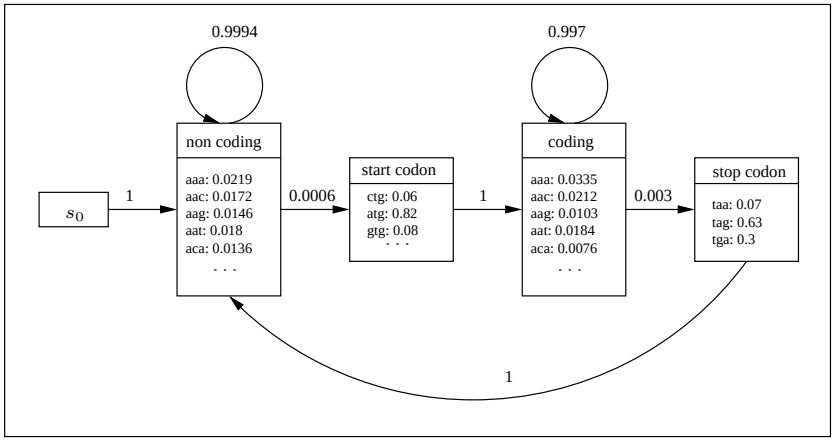

Fig. 2. A simple HMM for Prokaryote genes prediction

Fig. 2 shows a simple HMM for prediction of genes; more advances HMMs are used in successful gene finders that have been reported in the literature, e.g., Genemark.HMM [6] and EasyGene [5], and they can also be handled by our approach. The emission symbols of this HMM are codons, thus three letters form one symbol. It has four states: **start codon, non coding, coding, stop codon** From **non coding**, any codon can be emitted. From state **start codon**,

only start codons ata, atg, att, ctg, gtg and ctg can be emitted. From **coding**, any codon can be emitted except stop codons, taa, tga and tag. From state **stop codon**, only stop codons taa, tga or tag can be emitted. A consequence of the simplification of only emitting entire codons and not individual letters in this HMM is that we restrict to non coding regions whose length measured in codons is divisible, which is not the case in reality. Transition probabilities have been computed from an already annotated genome, Escherichia coli, K-12 substr. MG1655 (Genbank access NC_000913).

We can illustrate the annotation process as follows; we consider a small piece of E.coli from position 115 to 255.

ctt agg tca cta aat act tta acc aat ata ggc ata gcg cac aga cag ata aaa att aca gag tac aca aca tcc atg aaa cgc att agc acc acc att acc acc acc atc acc att acc aca ggt aac ggt gcg ggc tga,

The Viterbi algorithm computes the most probable path which is indicated as follows:

ctt	···	tcc	atg	aaa	···	ggc	tga
non coding	···	**non coding**	**start codon**	**coding**	···	**coding**	**stop codon**

From the indicated path, we can extract an annotation that states a non coding region from position 115 to 189 and a coding region from position 190 to 255.

3 CHMMS and Constraint Models for prediction

In this section, we give a formal definition of CHMMs and propose a constraint model for CHMM runs which is employed in an extended Viterbi algorithm.

3.1 Constrained Hidden Markov Model

A CHMM restricts the behavior of an HMM by constraints that must hold on paths that are considered.

Definition 2. *A* constrained HMM (CHMM) *is defined by a 5-tuple* $\langle S, A, T, E, C\rangle$ *where* $\langle S, A, T, E\rangle$ *is an HMM and* C *is a set of constraints, each of which is a mapping from HMM runs into* $\{true, false\}$.

A run *of a such a CHMM,* $\langle path, obs\rangle$ *is a run of the corresponding HMM for which* $C(path, observation)$ *is true (understood as the conjunction of the individual constraints in* C*).*

Notice that we defined constraints in a highly abstract way, independently of any specific constraint language. However, the Markov processes considered in this paper are discrete, and in the following we will apply constraints over finite domains [13]. In [11], constraints were expressed using in Prolog using tests and failure, in a way that do not invite to using constraint solving techniques.

3.2 A Constraint model for runs of CHMM

The constraint model represents runs of a CHMM. A run corresponds to a solution of the constraint model.

Let $\langle S, A, T, E, C\rangle$ be a CHMM and n an integer value that represents run length. The constraint model is described by the following syntax:

$$run([s^{(0)}, S_1, \ldots, S_n], [E_1, \ldots, E_n])$$

where each variable S_i and E_i represents the state and respectively the emission at the step i. The domain of S_i and E_i, noted $\mathrm{dom}(S_i)$ and respectively $\mathrm{dom}(E_i)$, is $S \setminus \{s^{(0)}\}$ and respectively E.

$run([s^{(0)}, S_1, \ldots, S_n], [E_1, \ldots, E_n])$ is *true* iff

$$\begin{aligned}
&\exists s^{(1)} \in \mathrm{dom}(S_1), \ldots, \exists s^{(n)} \in \mathrm{dom}(S_n) \text{ and} \\
&\qquad \exists e^{(1)} \in \mathrm{dom}(E_1), \ldots, \exists e^{(n)} \in \mathrm{dom}(E_n), \\
&\qquad C(s^{(0)}s^{(1)}\ldots s^{(n)}, e^{(1)}\ldots e^{(n)}) \text{ is true and} \\
&\qquad p(s^{(0)}; s^{(1)}) \cdot p(s^{(1)}; e^{(1)}) \ldots p(s^{(n-1)}; s^{(n)}) \cdot p(s^{(n)}; e^{(n)}) > 0 \quad (1).
\end{aligned}$$

By definition of a CHMM, variables S_i and E_i are constrained to satisfy by C. Formula (1) states that $s^{(0)}s^{(1)}\ldots s^{(n)}$ and $e^{(1)}\ldots e^{(n)}$ is a run the HMM. From this formula, restrictions on the domain of the variables S_i and E_i can be added.

The formula (1) is composed of a product of probabilities. Then, its value is positive iff the value of all the transition or emission probabilities are positive. We used this property to establish a (local) relationship between S_i and S_{i+1} and S_i and E_i. Indeed, valuation of S_i to $s^{(i)}$ and S_{i+1} to $s^{(i+1)}$ can be part of a solution of the constraint model whenever $p(s^{(i)}; s^{(i+1)}) > 0$. These relationships between variables of *run/2* are modeled by the following constraints added on them:

$$trans(S_i, S_{i+1}) \text{ and } emit(S_i, E_i)$$

where S_i, S_{i+1} and E_i are variables of *run/2*.

$trans(S_i, S_{i+1})$ is *true* iff

$$\exists s^{(i)} \in \mathrm{dom}(S_i) \text{ and } s^{(i+1)} \in \mathrm{dom}(S_{i+1}), p(s^{(i)}; s^{(i+1)}) \neq 0.$$

$emit(S_i, E_i)$ is *true* iff

$$\exists s^{(i)} \in \mathrm{dom}(S_i) \text{ and } e^{(i)} \in \mathrm{dom}(E_i), p(s^{(i)}; e^{(i)}) \neq 0.$$

These constraints are used for domain pruning during constraint propagation. For example, let us suppose that emission *taa* is observed at the step i of a run of the simple gene finder. During constraint propagation, $emit(S_i, E_i)$ prunes the domain of S_i to $\{\mathbf{non\,coding}, \mathbf{stop\,codon}\}$.

Constraints C of a CHMM are simply added on the variables of *run/2*. In the following, an example of C is defined to include prior knowledge on the simple gene finder.

3.3 A constrained gene finder

We illustrate the constraint model on the simple gene finder presented subsection 2.3. The HMM associated with the simple gene finder is constrained to be in certain states at given positions. For instance, this CHMM allows the inclusion of information about known coding regions during the Viterbi computation.

Consider

$$run([s^{(0)}, S_1, \ldots, S_n], [e^{(1)}, \ldots, e^{(n)}])$$

the constraint model associated with the simple gene finder where $e^{(1)}, \ldots, e^{(n)}$ is a sequence of n codons. A set of variables S_i is constrained to be equal to $State$ with the following constraint:

$$fix(State, Position_1, Position_2)$$

where $State \in S \setminus \{s^{(0)}\}$, $Position_1 \in \{1, \ldots, n\}$, $Position_2 \in \{1, \ldots, n\}$ and $Position_1 \leq Position_2$.

$fix(State, Position_1, Position_2)$ is *true* iff

$\exists k \in \mathrm{dom}(Position_1)$ and $\exists l \in \mathrm{dom}(Position_2), \forall i, k \leq i \leq l, S_i = State$.

For example, information of the position of a coding region can be expressed as the conjunction of

$$fix(\textbf{start codon}, P_1, P_1) \wedge P_1 + 1 = P_2 \wedge$$
$$fix(\textbf{coding}, P_2, P_3) \wedge P_3 + 1 = P_4 \wedge fix(\textbf{stop codon}, P_4, P_4).$$

These constraints on the simple gene finder oblige runs to be in a coding region between the position P_1 and P_4.

3.4 Viterbi Computation for a CHMM

Consider a CHMM $\langle S, A, T, E, C \rangle$, an observation $e^{(1)} \ldots e^{(n)}$ and a constraint model

$$run([s^{(0)}, S_1, \ldots, S_n], [e^{(1)}, \ldots, e^{(n)}]).$$

The most probable path is computed by finding the solution $s^{(1)}, \ldots, s^{(n)}$ of the constraint model that maximizes the objective function: run probability.

Viterbi computation for CHMM is expressed as a rewriting system on a set of 5-tuples Σ. Each such 5-tuple is of form $\langle s, i, p, \pi, \sigma \rangle$ where π is a partial path ending in state s and representing a path corresponding to the emission sequence prefix $e^{(1)} \cdots e^{(i)}$; p is the collected probability for the emissions and transitions applied in the construction of π and σ is the current constraint store, a conjunction of constraints. Solutions of the constraint store are denoted by $\mathrm{sol}(\sigma)$.

The two rules of the classical Viterbi algorithm are adapted for CHMM (see. **Fig.** 3). Viterbi computation is executed from an initial set of 5-tuples $\{\langle s^{(0)}, 0, 1, \epsilon, C \wedge \bigwedge_{i>0} trans(S_i, S_{i+1}) \wedge \bigwedge_{i>0} emit(S_i, E_i)) \rangle\}$. The $trans\_ctr$ rule expands an existing partial path one step in a restricted number of directions

$$
\begin{array}{ll}
\textit{trans\_ctr}: & \Sigma := \Sigma \cup \{\langle s', i{+}1, p \cdot p(s;s') \cdot p(s';e^{(i+1)}), \pi\, s', \sigma \wedge S_{i+1} = s' \rangle\} \\
& \text{whenever } \langle s, i, p, \pi, \sigma \rangle \in \Sigma,\ p(s;s'), p(s';e^{(i+1)}) > 0 \\
& \text{check\_sat}(\sigma \wedge S_{i+1} = s') \text{ and } \textit{prune\_ctr} \text{ does not apply.} \\
\\
\textit{prune\_ctr}: & \Sigma := \Sigma \setminus \{\langle s, i{+}1, p', \pi', \sigma' \rangle\} \\
& \text{whenever } \langle s, i{+}1, p, \pi, \sigma \rangle, \langle s, i{+}1, p', \pi', \sigma' \rangle \in \Sigma, \\
& p \geq p' \text{ and } \mathrm{sol}(\sigma') \subseteq \mathrm{sol}(\sigma).
\end{array}
$$

Fig. 3. Rewriting rules for the Viterbi algorithm for CHMM

that satisfy the constraint store. This satisfiability checking of the constraint store is denoted by check_sat. *prune_ctr* removes partial paths that are non optimal solutions. This is the case when two conditions are satisfied: the probability to reach s is not optimum and solutions of σ' are also solutions of σ. The second condition avoids removing partial paths that could be part of an optimal solution. If $\mathrm{sol}(\sigma')$ and $\mathrm{sol}(\sigma)$ can not be compared, we can not conclude that the partial path π' is not part the optimal solution.

Correctness property is argued in the previous paragraph. *prune_ctr* rule allows us to remove only partial paths detected as part of a non optimal solution. Unlike the classical Viterbi algorithm, this algorithm can run in time exponential in the length of the given emission sequence. Indeed, potentially all the extended partial paths need to be kept in Σ. However, with an efficient *check_sat*, partial paths that can not lead to the solution of the constraint model will be discarded as soon as possible. Size reductions of Σ due to *prune_ctr* is also important. The size for Σ will stay reasonable if the two constraint stores can easily be compared. That is the case for the fix constraint. Indeed, this constraint constrains only a restricted set of variables of the path. Outside this set, the remaining variables are not constrained. Then, the comparison is straightforward.

4 Discussion and future work

The term "Constrained HMM" is used in [9, 4], refers to restrictions on the finite automaton associated with a HMM but not as constraint on HMM runs. In [11], CHMMs were introduced to exemplify an EM algorithm for models with possible derivation failures. Our approach differs since we take care about constraints on the HMM during Viterbi computation whereas they do that during the learning process.

In constraint modeling, we can denote two recent works with a similar approach of ours [15, 17]. In [15], Will et al. propose a constraint model to represent pairwise alignment problem which integrates constraints related to non-coding RNA. In [17], Zytnicki et al. describe a weighted CSP model for locating motifs of non-coding RNA.

In this paper, a constraint model that represents runs of constrained HMMs is defined. In this framework, Viterbi computation is expressed as an optimization problem and conditions for an efficient computation are presented. Finally, a gene

finder that includes prior knowledge is presented along the paper to exemplify a usage of CHMM.

A first implementation based on PRISM allows us to perform a first experiment on fixing a coding region for the simple gene finder. PRISM build-ins give us for free a Viterbi computation for CHMM. To improve the efficiency of the *pruning_ctr* rule to reduce the search, we work on an implementation of the proposed algorithm in CHR [2], a multiset based rewriting system, that closely follows rewriting systems described in this paper.

We also work on an automatic generation of the constraint model given a HMM. This implementation will be based on our Probabilistic Choice Constraints library [7] which allow the simulation of partially defined probabilistic choices in Constraint Programming. In the constraint model, probabilistic choice is partially known when origin state of an transition or emission is not known. This framework will facilitate the definition of other kinds of constraints which can be combined with existing gene finding methods, improving flexibility and the quality of the results.

References

1. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambrigde University Press, 1998.
2. T. Fruhwirth. *Constraint Handling Rules*. Cambridge University Press, 2009.
3. A. Krogh, I.S. Mian, and D. Haussler. A hidden markov model that finds genes in e.coli dna. *Nucleic Acids Research*, 22(22):4768–4778, 1994.
4. N. Landwehr, T. Mielikinen, L. Eronen, H. Toivonen, and H. Mannila. Constrained hidden markov models for population-based haplotyping. *BMC Bioinformatics*, 8, 2007.
5. T.S. Larsen and A. Krogh. Easygene - a prokaryotic gene finder that ranks orfs by statistical significance. *BMC Bioinformatics*, 4(21), 2003.
6. A.V. Lukashin and M. Bordovsky. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, February 1998.
7. M. Petit and A. Gotlieb. Boosting probabilistic choice operators. In *Proceedings of the International Conference on Principles and Pratice of Constraint Programming*, LNCS, pages 559–573, Providence, USA, September 2007.
8. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognitation. *IEEE*, 77(2):257–286, February 1989.
9. S.T. Roweis. Constraint hidden markov models. In *Prof. of the International Conference of Advances in Neural Information Processing System*, pages 782–788, Denver, USA, Dec. 1999.
10. T. Sato, T. Kameya, and N.F. Zhou. Generative modeling with failure in PRISM. In *Proceedings of International Joint Conference on Aritificial Intelligence*, pages 847–852, Edinburgh, Scotland, July 2005.
11. T. Sato and Y. Kameya. New advances in logic-based probabilistic by PRISM. In *Probabilistic Inductive Logic Programming*, LNCS, pages 118–155. Springer, 2008.
12. D.B. Searls. *Artificial intelligence and molecular biology*, chapter The computational linguistics of biological sequences, pages 47–120. AAAI, 1993.
13. P. Van Hentenryck, V.A. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(fd). *Constraint Programming*, 910:293–316, May 1995.

14. Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, April 1967.
15. S. Will, A. Bush, and R. Backofen. Efficient sequence alignment with side-constraints by cluster tree elimination. *Constraints*, 13(1–2):110–129, 2008.
16. J. Xiong. *Essential Bioinformatics*. Cambridge Unversity Press, 2006.
17. M. Zytnicki, C. Gaspin, and T. Schiex. DARN! A weighted constraint solver for RNA motif localization. *Constraints*, 13(1–2):91–109, 2008.