BIOINFORMATICS

Vol. 24 ECCB 2008, pages i160–i166
doi:10.1093/bioinformatics/btn282

Efficient representation and *P*-value computation for high-order Markov motifs

Paulo G. S. da Fonseca[1,2,*] Christian Gautier[2], Katia S. Guimarães[1] and Marie-France Sagot[2]

[1]Centro de Informática, Universidade Federal de Pernambuco, 50732-970, Recife, Brazil and [2]Université de Lyon, F-69000, Lyon; Université Lyon 1; INRIA Rhône-Alpes; CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, F-69622, Villeurbanne, France

ABSTRACT

Motivation: Position weight matrices (PWMs) have become a standard for representing biological sequence motifs. Their relative simplicity has favoured the development of efficient algorithms for diverse tasks such as motif identification, sequence scanning and statistical significance evaluation. Markov chain-based models generalize the PWM model by allowing for inter-position dependencies to be considered, at the cost of substantial computational overhead, which may limit their application.

Results: In this article, we consider two aspects regarding the use of higher order Markov models for biological sequence motifs, namely, the representation and the computation of *P*-values for motifs described by a set of occurrences. We propose an efficient representation based on the use of tries, from which empirical position-specific conditional base probabilities can be computed, and extend state-of-the-art PWM-based algorithms to allow for the computation of exact *P*-values for high-order Markov motif models.

Availability: The software is available in the form of a Java object-oriented library from http://www.cin.ufpe.br/∼paguso/kmarkov.

Contact: paguso@cin.ufpe.br

1 INTRODUCTION

Position weight matrices (PWMs) have become a *de facto* standard probabilistic model for biological sequence motifs, for instance, for representing transcription factor-binding sites (TFBS). Key to this success is their simplicity, which allows for immediate interpretation and easier analytical manipulation, and the consequent computational efficiency, which has also favoured the development of numerous techniques for the various tasks related to the analysis of sequence motifs such as motif discovery (see GuhaThakurta, 2006, for a recent survey), sequence scanning (Beckstette *et al.*, 2006; Pizzi *et al.*, 2007) and statistical significance assessment (Bejerano, 2003; Touzet and Varre, 2007; Zhang *et al.*, 2007). A strong hypothesis on which the PWM model is based is that of the independence of each position of the motif relative to other positions. This assumption, the practicality it confers to the model notwithstanding, comes at the cost of an alleged simplification of the actual biochemical interactions that take place at the regulatory sites.

Markov models have been used for biological sequence analysis in general (Durbin *et al.*, 1999), and in particular for representing binding site motifs (Ellrott *et al.*, 2002; Huang *et al.*, 2006; Zhao *et al.*, 2005). They represent a step up in terms of expressiveness over PWMs, since they allow for local dependencies between adjacent positions to be represented directly (which does not mean that non-adjacent positions are independent). Other more general models have been proposed which allow for more complex inter-position (in)dependence schemes to be considered. For example, Barash *et al.* (2003) have successfully used Bayesian network models to represent TFBS motifs. Unfortunately, however, the gain in terms of expressiveness is often accompanied by a substantial computational overhead, which may limit their application in practice. Moreover, we conjecture that no model will replace PWMs as a standard until it includes a reasonably complete algorithmic framework for the diverse tasks related to the analysis of sequence motifs mentioned above.

In this article, we address two aspects regarding the utilization of higher order Markov models for biological sequence motifs. First, we consider the representation of high-order Markov motifs described by a set of occurrences. We propose a convenient data structure based on tries, from which empirical position-specific conditional base probabilities can be computed as needed. Next, we consider the problem of computing exact *P*-values for putative occurrences of high-order Markov motifs. We propose non-trivial extensions of two state-of-the-art PWM-based algorithms for working with a Markov dependency model. We illustrate the use of the proposed model and algorithms on a set of TFBS motifs defined by collections of aligned sites extracted from the TRANSFAC database (Wingender *et al.*, 2000).

2 METHODS

2.1 The *K*-order Markov model

A motif $\mathbf{m}$ of length W is a probabilistic model that defines a multivariate discrete probability distribution $P_{\mathbf{m}}(\cdot)$ over the space of words of length W of a fixed alphabet $\mathcal{A}$. If $\mathbf{x} = x_1 \cdots x_W \in \mathcal{A}^W$, then $P_{\mathbf{m}}(\mathbf{x})$ is actually a shorthand for $P(X_1 = x_1, \ldots, X_W = x_W | \mathbf{m})$, where X_j is a r.v. corresponding to the position j of the pattern. Here, we consider sequence motifs that are described by discrete K-order non-homogeneous Markov chains. The likelihood of a particular word $\mathbf{x} = x_1 \cdots x_W$ under such a model $\mathbf{m}$ is given by

$$P(x_1 \cdots x_W | \mathbf{m}) = \prod_{j=1}^{W} P(x_j | x_{j-K} \cdots x_{j-1}, \mathbf{m}).$$

The model parameters are the position-specific transition probabilities

$$\Pr[X_j = a_0 \mid X_{j-K} = a_K, \ldots, X_{j-1} = a_1],$$

for all possible assignments of $a_0, \ldots, a_K$ in $\mathcal{A}$ and for each position $j = 1, \ldots, W$. If $K = 0$, then the model resumes to a PWM. In general, though, this model captures local relationships between individual letters that can vary according to their positions in the sequence.

*To whom correspondence should be addressed.

2.2 Representing a set of motif occurrences

In most practical cases, the parameters of a motif model are derived from a set of occurrences, even during a learning procedure. For example, if we examine the algorithms that learn motif models by EM (Bailey and Elkan, 1994), we see that the model parameters are updated so as to maximize the likelihood of the occurrences indicated by the current estimates of the hidden variables. The same goes for Gibbs sampling-based techniques (Lawrence *et al.*, 1993), where the model parameters are calculated from the last sampled occurrences. In other words, the model parameters summarize the relevant information contained in a set of occurrences. Motivated by these observations, we adopt an alternative approach that consists in representing a set of occurrences with a convenient data structure in such a way that the information necessary for the computation of word probabilities can be recovered as needed. Our proposed representation is based on the use of index structures known as *tries* (Fredkin, 1960; Knuth, 1998).

Definition 1 (Trie). *A trie over a fixed alphabet $\mathcal{A}$ is a rooted tree such that (i) every edge is labelled with a character in $\mathcal{A}$ and (ii) every node has, at most, one outgoing edge labelled with a, for any $a \in \mathcal{A}$.*

An important property resulting from the definition of tries is that we can then establish a 1 : 1 association between each node v and the word label(v) obtained by concatenating the labels of the edges on the path from the root to that node [by definition, label(root) = ε, the empty string]. We then define the set of words represented by a trie T by

$$\mathrm{str}(T) \overset{\mathrm{def}}{=} \{\mathrm{label}(l) \mid l \in \mathrm{leaves}(T)\}.$$

If no word of a set of words $\mathcal{X}$ is a proper prefix of another, then there exists one, and only one trie $T = T(\mathcal{X})$ s.t. $\mathrm{str}(T) = \mathcal{X}$. Since we can enforce this pre-condition on $\mathcal{X}$ by appending a 'sentinel' symbol $\$ \notin \mathcal{A}$ to each of its elements, we can assume w.l.o.g. that it holds. Moreover, we can generalize the definition of tries so that they can represent *multisets* of words: we store the number of copies of a word in the multiset in the corresponding leaf of the trie. We also attach to every internal node the sum of the counts of the leaves of the subtree rooted at that node. We will refer to these numbers generically as *leaf counts*. Figure 1 displays an example of an extended trie with its leaf counts. Two fundamental properties of leaf counts are:
P1. The leaf count of a node v of $T(\mathcal{X})$ corresponds to the number of words of $\mathcal{X}$ having label(v) as prefix.
P2. The leaf count of an internal node v equals the sum of leaf counts of its children.

The extended trie encodes the character frequencies at each position of the represented words as in a PWM. This fact is established by the following proposition, which follows immediately from the definition of extended tries.

Proposition 1. *Let $T = T(\mathcal{X})$ be a trie representing a multiset of words $\mathcal{X}$. The frequency of the character a at position j in $\mathcal{X}$ is obtained from T as the sum of the leaf counts of the nodes at height j (the root is assumed to be at height zero) whose incoming edges are labelled by a, divided by the total number of represented words, i.e. the leaf count of the root node.*

Consider the trie of Figure 1 for an example of the property above. The proportion of words whose second letter is a `t` in that set, $\Pr[x_2 = \texttt{t}]$,

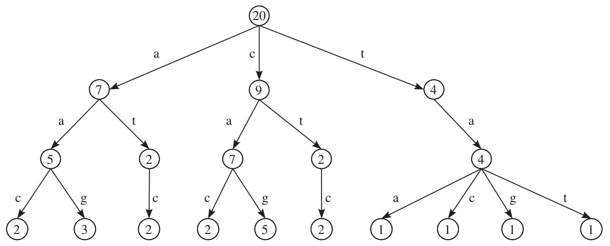

Fig. 1. Extended trie representing a multiset of 20 words of length 3.

is given by the sum of the leaf counts of the two nodes at height 2 whose incoming edge is labelled by `t`, divided by the total number of words, that is $\Pr[x_2 = \texttt{t}] = (2+2)/20 = 0.2$. However, tries are most useful when we consider connections between positions. Suppose, for instance, that we want to establish the probability of having a `c` in the first and third positions. By looking at the leaf count of the central node at height one, we see that nine words start with a `c`. By looking at the leaf counts of its children, we see that in seven out of those nine words, the initial `c` is followed by an `a`, whereas in the other two, the second letter is a `t`. Since we made no restrictions about the second letter, in principle all those words can interest us. However, by looking at leaf counts further down in the tree, we notice that although the two words that start with `ct` end by `c`, only two out of the seven words that start with `ca` end with a `c`. The two words of the latter case plus the other two of the former are then the only ones to match our requirements. The desired probability can thus be computed as $\Pr[x_1 = \texttt{c}, x_3 = \texttt{c}] = (2+2)/20 = 0.2$. This principle is generalized in the following proposition which is, again, a direct consequence of the definition of extended trie.

Proposition 2. *Let $T = T(\mathcal{X})$ be a trie representing a multiset of words $\mathcal{X}$, each of length W. Let $\mathbf{i} = (i_1, \ldots, i_L) \subset (1, \ldots, W)$ be a 'subvector' of positions, and $\mathbf{a} = (a_1, \ldots, a_L) \in \mathcal{A}^L$ be a vector of L characters. We denote by $\mathbf{x_i} = \mathbf{a}$ the event $[x_{i_1} = a_1, \ldots, x_{i_L} = a_L]$. Then the probability of $\mathbf{x_i} = \mathbf{a}$ in $\mathcal{X}$ is given by the sum of the leaf counts of all nodes of height i_L whose ingoing edges are labelled by a_L, and such that the paths from the root to those nodes go through nodes at heights $i_1, \ldots, i_{L-1}$ whose incoming edges are labelled by $a_1, \ldots, a_{L-1}$, respectively, divided by the total number of represented words.*

Conditional probabilities can be computed from Proposition 2 and the relation $P(A|B) = P(A,B)/P(B)$. For example, according to the trie of Figure 1, the probability of having a `g` in the third position, given that the base at the first position is a `c` is given by $\Pr[x_3 = \texttt{g} | x_1 = \texttt{c}] = \#[\mathbf{x}_{1,3} = \texttt{cg}]/\#[x_1 = \texttt{c}] = 5/9 \approx 0.56$.

Finally, we note that, for a K-order Markov motif model $\mathbf{m}$ built from a set of motif occurrences $\mathcal{X}$, we need to compute transition probabilities of the kind $P_{\mathbf{m}}(x_j | x_{j-K} \cdots x_{j-1})$ and thus we need to count the occurrences of contiguous k-mers $x_{j-K} \cdots x_j$ on $\mathcal{X}$. This can be easily done as follows. Starting from the nodes at height $j-K-1$ in the trie, we follow the edges labelled by each position of the k-mer, $x_{j-K}, x_{j-K+1}, \ldots, x_j$, in sequence, and we sum up the leaf counts of the final nodes whenever we can reach the end of the k-mer. Hence we need to visit at most

$$1+2+\cdots+K+\sum_{j=K+1}^{W} |\mathcal{A}|^{j-K-1} \cdot K = K\left(\frac{1+K}{2} + \frac{|\mathcal{A}|^{W-K}-1}{|\mathcal{A}|-1}\right)$$

vertices for computing $P_{\mathbf{m}}(x_1 \cdots x_W)$. However, this worst case $O(K|\mathcal{A}|^{W-K})$ behaviour is rare in practice since it happens only when the motif trie is complete. Most often the trie is very sparse for it contains only a few occurrences of the motif. In addition, once $P_{\mathbf{m}}(x_j | x_{j-K} \cdots x_{j-1})$ is computed, it can be stored for subsequent use.

2.3 Computing *P*-values for Markov motif models

We consider now the problem of computing motif occurrence P-values for K-order Markov models.

Definition 2. *Let $\mathbf{m}$ be a K-order motif model of length W. We define the log-likelihood score of a word $\mathbf{x}$ relative to $\mathbf{m}$ as*

$$\ell(\mathbf{x}; \mathbf{m}) \overset{def}{=} \log P_{\mathbf{m}}(\mathbf{x}).$$

Given also a background sequence model $\mathbf{m}_0$, we define the P*-value of a score l as*

$$P\text{-}value(l; \mathbf{m}) \overset{def}{=} P_{\mathbf{m}_0}(\{\mathbf{x} \in \mathcal{A}^W \mid \ell(\mathbf{x}; \mathbf{m}) \geq l\}).$$

In words, it represents the probability, under the background (null) model, for a random sequence to have a score at least as good as l relative to the motif model $\mathbf{m}$.

For the PWM model, the problem of computing P-values is well studied in the literature and has been recently demonstrated to be NP-hard (Touzet and Varre, 2007; Zhang *et al.*, 2007). The naive solution consists in exhaustively enumerating each word of the desired size W, computing its log-likelihood under the motif model $\mathbf{m}$ and comparing it to the target score l. Approximate methods exist which avoid exhaustive enumeration by sampling words from the background distribution and estimating the P-value through the empirical expected value of the binary indicator function $\mathbf{1}(\ell(\mathbf{y},\mathbf{m})\geq l)$. Here, however, we consider extensions of recent exact PWM-based techniques for our case of K-order Markov models.

2.3.1 Branch-and-bound P-value computation The algorithm presented by Bejerano (2003) is based on a recursive enumeration of the words of length W in such a way that, at each point of the enumeration tree, we have a prefix of length $j\leq W$ and we can compute bounds for the log-likelihood score of the words having this prefix. At this point, we compare these bounds with the target score l to decide whether or not to pursue the enumeration further down the tree. Procedures employing this kind of heuristic fall into a broad class of algorithms commonly referred to as *branch and bound* algorithms (e.g. Michalewicz and Fogel (2004), Sec. 4.4). A skeleton of this particular procedure is shown in Figure 2.

The algorithm of Figure 2 can also be used for K-order Markov models, its efficacy depending on the estimation of the bounds performed in line 7. In order to explain our solution for this problem, we need some notation. First, we notice that the log-likelihood score of a word $\mathbf{x}=x_1\cdots x_W$ w.r.t. a K-order Markov model $\mathbf{m}$ of length W decomposes additively into $\ell(\mathbf{x};\mathbf{m})=\sum_{j=1}^{W}\log P(x_j|x_{j-K}\cdots x_{j-1})$. We denote by $\ell_j(\mathbf{x};\mathbf{m})=P(x_j|x_{j-K}\cdots x_{j-1})$ the term corresponding to position j, which we call the *j-position score*. The j-position score depends only on the character a at that position and the k-mer $\mathbf{k}\in\mathcal{A}^{\min\{j-1,K\}}$ that precedes it. Thus we write $\ell_j(\mathbf{k}\cdot a;\mathbf{m})$ to denote the j-position score of a word $\mathbf{x}$ with $x_j=a$ and $x_{j-|\mathbf{k}|\cdots j-1}=\mathbf{k}$. We denote by $\ell_{u\cdots v}(\mathbf{x};\mathbf{m})=\sum_{j=u}^{v}\ell_j(\mathbf{x};\mathbf{m})$ the partial sum of the position scores for positions from u to v, or the *subword score* from u to v. If $u=1$, then we call it the *v-prefix score* of $\mathbf{x}$. If $v=W$, it is called the *u-suffix score* of $\mathbf{x}$.[1] Notice that, as for the position score, the subword score $\ell_{u\cdots v}(\mathbf{x};\mathbf{m})$ depends not only on the subword $x_u\cdots x_v$ but also on the k-mer $\mathbf{k}$ that precedes it. We thus write $\ell_{u\cdots v}(\mathbf{k}\cdot\mathbf{z};\mathbf{m})$ to denote the subword score from u to v of a word $\mathbf{x}$ s.t. $\mathbf{x}_{u-|\mathbf{k}|\cdots u-1}=\mathbf{k}$ and $\mathbf{x}_{u\cdots v}=\mathbf{z}$. For the v-prefix score of a word starting with $\mathbf{v}\in\mathcal{A}^{v}$ we write $\ell_{\cdots v}(\mathbf{v};\mathbf{m})$, and for the u-suffix score of a word ending with the suffix $\mathbf{u}\in\mathcal{A}^{W-u+1}$ preceded by the k-mer $\mathbf{k}$ we write $\ell_{u\cdots}(\mathbf{k}\cdot\mathbf{u};\mathbf{m})$. Finally, we denote by $\ell^{\max}_{u\cdots v}(\mathbf{k}\cdots\mathbf{m})=\max_{\mathbf{z}\in\mathcal{A}^{v-u+1}}\ell_{u\cdots v}(\mathbf{k}\cdot\mathbf{z};\mathbf{m})$ the maximum subword score from u to v of a word with the 'seed' k-mer $\mathbf{x}_{u-|\mathbf{k}|\cdots u-1}=\mathbf{k}$ preceding the subword. By the same token, we denote by $\ell^{\max}_{\cdots v}(\cdot;\mathbf{m})=\max_{\mathbf{v}\in\mathcal{A}^{v}}\ell_{\cdots v}(\mathbf{v};\mathbf{m})$ the maximum v-prefix score and by $\ell^{\max}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})=\max_{\mathbf{u}\in\mathcal{A}^{W-u+1}}\ell_{u\cdots}(\mathbf{k}\cdot\mathbf{u};\mathbf{m})$ the maximum u-suffix score for of a word with the 'seed' k-mer $\mathbf{x}_{u-|\mathbf{k}|\cdots u-1}=\mathbf{k}$. The minimum subword, prefix and suffix scores, $\ell^{\min}_{u\cdots v}(\mathbf{k}\cdot\ ;\mathbf{m})$, $\ell^{\min}_{\cdots v}(\cdot;\mathbf{m})$ and $\ell^{\min}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})$ are defined accordingly.

Back to the branch and bound problem, suppose that we are at a point in the enumeration tree where we have spelled a $(j-1)$-prefix $\mathbf{y}$ ending with the k-mer $\mathbf{k}$, that is, the prefix is of the form $\mathbf{y}=*\mathbf{k}$ for some $\mathbf{k}\in\mathcal{A}^{\min\{K,j-2\}}$. Then we propose to use the bounds $L^{\min}(\mathbf{y})=\ell^{\min}_{j\cdots}(\mathbf{k}\cdot;\mathbf{m})$ and $L^{\max}(\mathbf{y})=\ell^{\max}_{j\cdots}(\mathbf{k}\cdot;\mathbf{m})$ in line 7 of the algorithm of Figure 2. Notice that these are the most strict bounds we can use knowing only the prefix $\mathbf{y}$. The computation of these bounds is done by a dynamic programming procedure based on the following proposition.

LEMMA 1. *Let $\mathbf{m}$ be a K-order Markov motif model of length W, $0\leq u\leq v\leq W$, and $\mathbf{k}\in\mathcal{A}^{\min\{K,u-1\}}$. In addition, let $\ell^{\max}_{u\cdots v}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})$ denote the maximum subword score from u to v of a word $\mathbf{x}$ s.t. $\mathbf{x}_{u-|\mathbf{k}|\cdots u-1}=\mathbf{k}$ and $\mathbf{x}_{u\cdots v}=*\mathbf{z}$, that is, we require that the subword $\mathbf{x}_{u\cdots v}$ ends with $\mathbf{z}$. Then the following holds:*

(i) If $|\mathbf{z}|\geq\min\{v-u+1,K\}$, then, for all $a\in\mathcal{A}$,

$$\ell^{\max}_{u\cdots v+1}(\mathbf{k}\cdot*\mathbf{z}a;\mathbf{m})=\ell^{\max}_{u\cdots v}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})+\ell_{v+1}(X_{v+1}=a|\mathbf{X}_{u-K\cdots v}=\mathbf{k}*\mathbf{z};\mathbf{m})$$

(ii) $\ell^{\max}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})=\max_{\mathbf{z}\in\mathcal{A}^{z}}\ell^{\max}_{u\cdots W}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})$ for any $0\leq z\leq W-u+1$.

PROOF. To prove proposition (i) notice that, by definition, $\ell^{\max}_{u\cdots v+1}(\mathbf{k}\cdot*\mathbf{z}a;\mathbf{m})$ corresponds to the best subword score from u to $v+1$ of a word $\mathbf{x}$ s.t. $\mathbf{x}_{u-|\mathbf{k}|\cdots u-1}=\mathbf{k}$ and $\mathbf{x}_{u\cdots v+1}=*\mathbf{z}a$. The subword score from u to $v+1$ of one such a word is of the form

$$\ell_{u\cdots v+1}(\mathbf{x};\mathbf{m})=\ell_u(\mathbf{x};\mathbf{m})+\cdots+\ell_v(\mathbf{x};\mathbf{m})+\ell_{v+1}(\mathbf{x};\mathbf{m}).$$

The condition $|\mathbf{z}|\geq\min\{v-u+1,K\}$ guarantees that, if the size of the subword (i.e. $u-v+1$) is greater than K, then $\mathbf{z}$ has size at least K. Otherwise, $\mathbf{z}$ fills the whole space $u\cdots v$. This means that all the letters affecting the last term of the subword score $\ell_{u\cdots v+1}(\mathbf{x};\mathbf{m})$ are fixed. Hence $\ell_{v+1}(\mathbf{x};\mathbf{m})$ is constant and

$$\begin{aligned}\ell^{\max}_{u\cdots v+1}(\mathbf{k}\cdot*\mathbf{z}a;\mathbf{m})&=\max_{\mathbf{x}}\{\ell_u(\mathbf{x};\mathbf{m})+\cdots+\ell_v(\mathbf{x};\mathbf{m})+\ell_{v+1}(\mathbf{x};\mathbf{m})\}\\&=\max_{\mathbf{x}}\{\ell_u(\mathbf{x};\mathbf{m})+\cdots+\ell_v(\mathbf{x};\mathbf{m})\}+\ell_{v+1}(\mathbf{x};\mathbf{m})\\&=\ell^{\max}_{u\cdots v}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})+\ell_{v+1}(\mathbf{x};\mathbf{m}).\end{aligned}$$

Part (ii) follows directly from the definition of $\ell^{\max}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})$ and $\ell^{\max}_{u\cdots W}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})$. It is only saying that the best score for a suffix starting at position u preceded by $\mathbf{k}$ is the best score for a suffix starting at position u preceded by $\mathbf{k}$ and ending with a certain $\mathbf{z}$ of size z, among all possible choices of $\mathbf{z}$, no matter its length.

Lemma 1 still holds if we consider minimum scores instead of maximum scores, and the proof is analogous. Based on this result, we can compute $\ell^{\max}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})$ [and equally $\ell^{\min}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})$] as shown in Figure 3. For each position $j=u,\ldots,W$ we build a table of the best subword scores $\ell^{\max}_{u\cdots j}(\mathbf{k}\cdot*\mathbf{z};\mathbf{m})$

```
Algorithm BranchAndBoundPvalue
Input    m: the motif model of length $W$
         $\mathbf{m}_0$: the background model
         $l$: the log-likelihood threshold
         $\mathbf{y}$: a prefix of length $j \leq W$
Output $P_{\mathbf{m}_0}(\{\mathbf{z}\in\mathcal{A}^W \mid \mathbf{z}_{1\ldots j}=\mathbf{y} \wedge \ell(\mathbf{z};\mathbf{m})\geq l\})$
 1  if $|\mathbf{y}| = W$ then
 2      if $\ell(\mathbf{y};\mathbf{m}) \geq l$ then
 3          return $P_{\mathbf{m}_0}(\mathbf{y})$
 4      else
 5          return 0
 6  else
 7      Estimate bounds $L^{\min}(\mathbf{y})$ and $L^{\max}(\mathbf{y})$ s.t.
        $\forall\mathbf{z}\in\mathcal{A}^W$ with $\mathbf{z}_{1\ldots j}=\mathbf{y}$, we have $L^{\min}(\mathbf{y})\leq\ell(\mathbf{z};\mathbf{m})\leq$
        $L^{\max}(\mathbf{y})$
 8      if $L^{\min}(\mathbf{y}) \geq l$ then
 9          return $P_{\mathbf{m}_0}(\{\mathbf{z}\in\mathcal{A}^W \mid \mathbf{z}_{1\ldots j}=\mathbf{y}\})$
10      else if $L^{\max}(\mathbf{y}) < l$ then
11          return 0
12      else
13          $s \leftarrow 0$
14          for each $a \in \mathcal{A}$ do
15              $s \leftarrow s +$ BranchAndBoundPvalue $(\mathbf{m},\mathbf{m}_0,l,\mathbf{y}a)$
16          return $s$
```

Fig. 2. Branch and bound motif P-value computation. This algorithm must be initially called with *BranchAndBoundPvalue*$(\mathbf{m},\mathbf{m}_0,l,\varepsilon)$.

[1]In the pattern-matching literature, the term u-suffix usually refers to the suffix on length u whereas here it means the suffix starting at position u.

```
Algorithm Max suffix log-score
Input   m: a Markov motif model of order $K$ and length $W$
        $u$: the initial position of the suffix ($1 \le u \le W$)
        $\mathbf{k} \in \mathcal{A}^{\min\{K,u-1\}}$: a 'seed' k-mer that precedes the suffix
Output  $\ell^{\max}_{u\cdots}(\mathbf{k}\cdot;\mathbf{m})$
1  Initialise $\mathbf{S}[u-1,\mathbf{k}] \leftarrow 0$
2  for $j = u,\ldots,W$ do
3      for each $\mathbf{y}$ s.t. $\mathbf{S}[j-1,\mathbf{y}]$ is defined do
4          for each $a \in \mathcal{A}$ do
5              $\widetilde{\mathbf{S}}[j,\mathbf{y}a] \leftarrow \mathbf{S}[j-1,\mathbf{y}] + \ell_j(\mathbf{y}\cdot a;\mathbf{m})$
6      if $j \le K$ then
7          for each $\mathbf{z} \in \mathcal{A}^j$ do
8              $\mathbf{S}[j,\mathbf{z}] \leftarrow \widetilde{\mathbf{S}}[j,\mathbf{z}]$
9      else
10         for each $\mathbf{z} \in \mathcal{A}^K$ do
11             $\mathbf{S}[j,\mathbf{z}] \leftarrow \max_{a\in\mathcal{A}}\{\widetilde{\mathbf{S}}[j,a\mathbf{z}]\}$
12 return $\max_{\mathbf{z}}\{\mathbf{S}[W,\mathbf{z}]\}$
```

Fig. 3. Dynamic programming computation of the maximum suffix log-score. The minimum suffix log-score can be computed similarly, only by replacing 'max' by 'min' in lines 11 and 12.

for all possible $\mathbf{z}$ of size $z=\min\{K,j-u+1\}$. We start with the value $\ell^{\max}_{u\cdots u-1}(\mathbf{k};\mathbf{m})=0$ and use the entries of the table corresponding to position j to build the entries of position $j+1$ with the aid of item (i) of Lemma 1. When the size of the subword $u\ldots j$ is bigger than K, the best subword scores from u to $j+1$, obtained after applying relation (i), would refer to subwords ending with all possible suffixes of length $K+1$. However we do not need to keep all this information since, for the next iteration, only the last K letters of the subword will matter. We hence shrink the table by considering only the best scores based on the last K positions of the subword. At the end, we use item (ii) of the lemma to return the desired result.

As a final note, we remark that the algorithm of Figure 3 takes time $O((W-u)\cdot|\mathcal{A}|^{K+1})$. Indeed, for each position $j=u,\ldots,W$, we need to build an extended table based on the $|\mathcal{A}|^K$ entries of the previous iteration and the possible $|\mathcal{A}|$ new letters, hence $|\mathcal{A}|^{K+1}$ entries. Then we eventually need to shrink this table, which demands visiting again its $|\mathcal{A}|^{K+1}$ entries. At the end of the algorithm, we have to scan the final $|\mathcal{A}|^K$ entries corresponding to the last position to obtain the final result. We also remark that, for each prefix in the recursive enumeration, the computation of the maximum and minimum suffix score bounds can be done simultaneously. Moreover, all prefixes of the same length ending with the same k-mer will share the same bounds, therefore this computation needs to be carried out only once per k-mer and per prefix length. Thus the worst case total cost of computing score bounds along the enumeration procedure is $\sum_{u=1}^{W}|\mathcal{A}|^K\cdot(W-u)\cdot|\mathcal{A}|^{K+1}=|\mathcal{A}|^{2K+1}\cdot W\cdot(W-1)/2=O(W^2|\mathcal{A}|^K)$.

2.3.2 *Computing P-values by iterative model refining*

We discuss now our extension of the method proposed by Touzet and Varre (2007). We begin with the very general observation that an essential factor impacting the efficiency of score P-value algorithms is the number of possible scores. We say that a score s is *possible* for a model $\mathbf{m}$ whenever there is a word $\mathbf{x}$ s.t. the score of $\mathbf{x}$ under $\mathbf{m}$ equals s. Indeed, in a principled enumeration schema like the one discussed in the previous section, if there are many possible scores then, at any given point, it is more likely that some of them will fail to observe our pruning conditions forcing the recursion to go down. On the other hand, if the number of possible scores is small, the number of possible sequences being the same, the score distribution is more 'concentrated', which results in more sequences being counted or discarded by groups, hence abbreviating the enumeration. Therefore, it would be interesting to reduce the number of possible scores of a model, for example, by truncating the values of its parameters at an arbitrary precision. Of course, we cannot do this in general for any score threshold, since this could result in some sequences whose scores are normally above the threshold not being counted due to the reduction in their scores caused by the precision loss. However, for sufficiently extreme P-values and for an adequate precision, we may miss no sequence while speeding up the P-value computation.

To formalize these ideas, let us re-phrase some key definitions found in Touzet and Varre (2007). First, given a number $x\in\mathbb{R}$ and a *precision* $\epsilon\ge 1$, we define the *round value* of x at precision ϵ as $[x]_\epsilon=\frac{1}{\epsilon}\lfloor\epsilon\cdot x\rfloor$, where $\lfloor\cdot\rfloor$ denotes the 'integer part' function. The precision ϵ will be normally taken to be a non-negative power of 10, i.e. $\epsilon=10^k$ for $k\ge 0$ and, in this case, the rounding operation will correspond to truncating the number at its k-th decimal place. Next, given a K-order Markov motif model of length W, $\mathbf{m}$, we call the derived ϵ*-round model* $\mathbf{m}_\epsilon$ the model obtaining by rounding each parameter of $\mathbf{m}$ at precision ϵ, that is

$$\ell(\mathbf{x};\mathbf{m}_\epsilon)=\sum_{j=1}^{W}\log[P_{\mathbf{m}}(x_j|x_{j-K}\cdots x_{j-1})]_\epsilon.$$

Since, for any $x\ge 0$, $[x]_\epsilon\le x$, and since log is a monotonically increasing function, it follows that $\ell(\mathbf{x};\mathbf{m}_\epsilon)\le\ell(\mathbf{x};\mathbf{m})$ for all $\epsilon\ge 1$ and for all $\mathbf{x}\in\mathcal{A}^W$. We can thus define the *maximal error* associated to the precision ϵ as

$$\Delta_{\mathbf{m},\epsilon}\overset{\text{def}}{=}\max_{\mathbf{x}\in\mathcal{A}^W}\{\ell(\mathbf{x};\mathbf{m})-\ell(\mathbf{x};\mathbf{m}_\epsilon)\}.$$

By this very definition of the maximum error, we have that, for any $\epsilon\ge 1$ and $\mathbf{x}\in\mathcal{A}^W$

$$\ell(\mathbf{x};\mathbf{m}_\epsilon)\le\ell(\mathbf{x};\mathbf{m})\le\ell(\mathbf{x};\mathbf{m}_\epsilon)+\Delta_{\mathbf{m},\epsilon}.$$

Finally, given a null model $\mathbf{m}_0$, we define the log-score distribution of $\mathbf{m}$ (under $\mathbf{m}_0$) as the function

$$Q_{\mathbf{m_0}}(s;\mathbf{m})\overset{\text{def}}{=}P_{\mathbf{m}_0}(\{\mathbf{x}\in\mathcal{A}^W\mid\ell(\mathbf{x};\mathbf{m})=s\}),$$

that is, for $s\in\mathbb{R}$, $Q_{\mathbf{m_0}}(s;\mathbf{m})$ denotes the probability under $\mathbf{m}_0$ for a sequence to have log-score s in $\mathbf{m}$.

The algorithm we are about to describe is based on the following proposition.

Lemma 2. *Let* $\mathbf{m}$ *be a K-order Markov model,* $\epsilon'\ge\epsilon$ *be two precisions and* $t'\le t$ *be two real numbers s.t. P-value*$(t;\mathbf{m}_\epsilon)=$*P-value*$(t-\Delta_{\mathbf{m},\epsilon};\mathbf{m}_\epsilon)$ *and P-value*$(t';\mathbf{m}_{\epsilon'})=$*P-value*$(t'-\Delta_{\mathbf{m},\epsilon'};\mathbf{m}_{\epsilon'})$*. Then*

$$\sum_{t'\le s<t}Q_{\mathbf{m_0}}(s;\mathbf{m})=\sum_{t'\le s<t}Q_{\mathbf{m_0}}(s;\mathbf{m}_{\epsilon'}).$$

Lemma 2 is essentially the same as Lemma 8 in Touzet and Varre (2007). We refer thus to that paper for a proof. This result says that we can approximate the log-score distribution of the original model $\mathbf{m}$ with no error by the score distribution of the round model $\mathbf{m}'_\epsilon$ inside the interval $[t',t)$. If, for a still higher precision ϵ'', we could find a value $t''\le t'$ s.t. P-value$(t'';\mathbf{m}_{\epsilon''})=P$-value$(t''-\Delta_{\mathbf{m},\epsilon''};\mathbf{m}_{\epsilon''})$ then, according to the lemma, we would be able to approximate the original score distribution inside the adjacent interval $[t'',t')$ with the refined model $\mathbf{m}_{\epsilon''}$. The net effect is that we would have the original score distribution in the interval $[t'',t)$ exactly estimated from two round distributions. If we continue increasing the precision, we get an increasingly larger interval.

This idea is used to compute P-value$(l,\mathbf{m})$ by approximating the log-score distribution on a series of adjacent intervals covering the entire range $[l,+\infty)$ using round models of increasing precision. The procedure is shown in Figure 4. The two main problems of this algorithm when applied to K-order Markov models correspond to lines 5 and 6. The former refers to the computation of the maximum error, which is a trivial problem when PWMs are used, but which is more complicated in our case. The latter corresponds to the computation of the score distribution, which is the core of the algorithm with exponential behaviour. The strategy to alleviate its deleterious impact consists in starting with models with low precision and augmenting it progressively. The increase in the precision up to a reasonable level may still be necessary, but it is also partially compensated by the fact

```
Algorithm Iterative P-value refinement
Input   $\mathbf{m}$: the ($K$-order Markov) motif model of length $W$
        $\mathbf{m}_0$: the background model
        $l$: The log-likelihood score threshold
        $\epsilon_0 > 0$: the initial precison
        $\kappa \geq 1$: the precision update factor
Output  $P$-value$(\mathbf{x};\mathbf{m})$
 1  $t \leftarrow$ any value $> 0$
 2  $\epsilon \leftarrow \epsilon_0$
 3  $p \leftarrow 0$
 4  repeat
 5      Compute the maximum error $\Delta_{\mathbf{m},\epsilon}$
 6      Compute $Q_{\mathbf{m}_0}(s;\mathbf{m}_\epsilon)$ for all accessible score $s \in [l-\Delta_{\mathbf{m},\epsilon}, t)$
 7      Search for the smallest $t'$ s.t.
        $P$-value$(t';\mathbf{m}_\epsilon) = P$-value$(t'-\Delta_{\mathbf{m},\epsilon};\mathbf{m}_\epsilon)$
 8      if such a $t'$ exists then
 9          $p \leftarrow p + \sum_{s\in[t',t)} Q_{\mathbf{m}_0}(s;\mathbf{m}_\epsilon)$
10          $t \leftarrow t'$
11      $\epsilon \leftarrow \kappa \cdot \epsilon$
12  until $l = t$
13  return $p$
```

Fig. 4. *P*-value iterative refinement.

that the interval over which the distribution is computed diminishes as the precision grows.

For the purpose of computing the maximum error associated to the round model $\mathbf{m}_\epsilon$, we have elaborated a dynamic programming algorithm which is based on the following proposition.

LEMMA 3. *Let* $\Delta_{\mathbf{m},\epsilon}[j,\mathbf{k}] = \max_{\mathbf{y}=*\mathbf{k}\in\mathcal{A}^j}\{\ell_{\dots j}(\mathbf{y};\mathbf{m}) - \ell_{\dots j}(\mathbf{y};\mathbf{m}_\epsilon)\}$, *i.e.* $\Delta_{\mathbf{m},\epsilon}[j,\mathbf{k}]$ *denotes the maximum rounding error for a prefix of length j ending with* $\mathbf{k}$ *(by convention,* $\Delta_{\mathbf{m},\epsilon}[0,\varepsilon]=0$*). Then, the following holds:*

(i) *If* $|\mathbf{k}| \geq \min\{K,j\}$ *then,* $\forall a \in \mathcal{A}$,

$$\Delta_{\mathbf{m},\epsilon}[j+1,\mathbf{k}a] = \Delta_{\mathbf{m},\epsilon}[j,\mathbf{k}] + [\ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}) - \ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}_\epsilon)].$$

(ii) $\Delta_{\mathbf{m},\epsilon} = \max_{\mathbf{k}\in\mathcal{A}^k}\Delta_{\mathbf{m},\epsilon}[W,\mathbf{k}]$, *for any* $0 < k \leq W$.

PROOF. For part (i), let $\mathbf{y} = *\mathbf{k}a \in \mathcal{A}^{j+1}$, that is, $\mathbf{y}$ is a prefix of length $j+1$ ending with $\mathbf{k}a$. Then

$$\begin{aligned}
&\ell_{\dots j+1}(\mathbf{y};\mathbf{m}) - \ell_{\dots j+1}(\mathbf{y};\mathbf{m}_\epsilon)\\
&= \sum_{i=1}^{j+1}\log P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1}) - \sum_{i=1}^{j+1}\log[P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1})]_\epsilon\\
&= \left\{\sum_{i=1}^{j}\log P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1}) - \sum_{i=1}^{j}\log[P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1})]_\epsilon\right\}\\
&\quad + \left\{\log P_{\mathbf{m}}(y_{j+1}|y_{j-|\mathbf{k}|+1\dots j}) - \log[P_{\mathbf{m}}[(y_{j+1}|y_{j-|\mathbf{k}|+1\dots j})]_\epsilon\right\}.
\end{aligned}$$

Because of the restriction on the size of $\mathbf{k}$, the last term of the sum above is fixed, that is,

$$\begin{aligned}
&\ell_{\dots j+1}(\mathbf{y};\mathbf{m}) - \ell_{\dots j+1}(\mathbf{y};\mathbf{m}_\epsilon)\\
&= \left\{\sum_{i=1}^{j}\log P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1}) - \sum_{i=1}^{j}\log[P_{\mathbf{m}}(y_i|y_{i-K}\cdots y_{i-1})]_\epsilon\right\}\\
&\quad + [\ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}) - \ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}_\epsilon)].
\end{aligned}$$

```
Algorithm Max rounding error
Input   $\mathbf{m}$: the motif model of length $W$
        $\epsilon$: the rounding precision
Output  $\Delta_{\mathbf{m},\epsilon} = \max_{\mathbf{x}\in\mathcal{A}^W}\{\ell(\mathbf{x};\mathbf{m}) - \ell(\mathbf{x};\mathbf{m}_\epsilon)\}$
 1  Initialise $\mathbf{D}[0,\varepsilon] \leftarrow 0$
 2  for $j = 1,\dots,W$ do
 3      for each $\mathbf{k}$ s.t. $\mathbf{D}[j-1,\mathbf{k}]$ is defined do
 4          for each $a \in \mathcal{A}$ do
 5              $\widetilde{\mathbf{D}}[j,\mathbf{k}a] \leftarrow \mathbf{D}[j-1,\mathbf{k}]$
                    $+ \log P_{\mathbf{m}}(Y_{j+1} = a|Y_{j-|\mathbf{k}|+1\cdots j} = \mathbf{k})$
                    $- \log[P_{\mathbf{m}}(Y_{j+1} = a|Y_{j-|\mathbf{k}|+1\cdots j} = \mathbf{k})]_\epsilon$
 6      if $j \leq K$ then
 7          for each $\mathbf{k} \in \mathcal{A}^j$ do
 8              $\mathbf{D}[j,\mathbf{k}] \leftarrow \widetilde{\mathbf{D}}[j,\mathbf{k}]$
 9      else
10          for each $\mathbf{k} \in \mathcal{A}^K$ do
11              $\mathbf{D}[j,\mathbf{z}] \leftarrow \max_{a\in\mathcal{A}}\{\widetilde{\mathbf{D}}[j,a\mathbf{z}]\}$
12  return $\max_{\mathbf{k}}\{\mathbf{D}[W,\mathbf{k}]\}$
```

Fig. 5. Maximum rounding error for *K*-order Markov models.

Hence

$$\begin{aligned}
\Delta_{\mathbf{m},\epsilon}[j+1,\mathbf{k}a] &= \max_{\mathbf{y}=*\mathbf{k}a\in\mathcal{A}^{j+1}}\left\{\ell_{\dots j+1}(\mathbf{y};\mathbf{m}) - \ell_{\dots j+1}(\mathbf{y};\mathbf{m}_\epsilon)\right\}\\
&= \max_{\mathbf{y}'=*\mathbf{k}\in\mathcal{A}^j}\left\{\ell_{\dots j}(\mathbf{y}';\mathbf{m}) - \ell_{\dots j}(\mathbf{y}';\mathbf{m}_\epsilon)\right\}\\
&\quad + [\ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}) - \ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}_\epsilon)]\\
&= \Delta_{\mathbf{m},\epsilon}[j,\mathbf{k}] + [\ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}) - \ell_{j+1}(\mathbf{k}\cdot a;\mathbf{m}_\epsilon)].
\end{aligned}$$

Part (ii) is a direct consequence of the definition of $\Delta_{\mathbf{m},\epsilon}[W,\mathbf{k}]$. It is only saying that the maximum error for a W-word is the maximum error for a W-word (=W-prefix) ending with a certain $\mathbf{k}$ among all possible choices for $\mathbf{k}$.

The algorithm of Figure 5 computes $\Delta_{\mathbf{m},\epsilon}$ by using Lemma 3 to build tables $\mathbf{D}[j,\mathbf{k}]$ containing the values of $\Delta_{\mathbf{m},\epsilon}[j,\mathbf{k}]$ for $j=1,\dots,W$ and $\mathbf{k}\in\mathcal{A}^{\min\{j,K\}}$. It is easy to see that, for each $j=1,\dots,W$, the algorithm takes at most $|\mathcal{A}|^{K+1}$ steps to build the table $\widetilde{\mathbf{D}}[j,\mathbf{k}a]$, and then $|\mathcal{A}|^{K+1}$ additional steps to read all its elements and 'compress' it into the final table $\mathbf{D}[j,\mathbf{k}]$. Hence the overall time complexity of the algorithm is $O(W\cdot|\mathcal{A}|^{K+1})$.

Finally, we need to discuss how to compute the score distribution $Q_{\mathbf{m}_0}(s;\mathbf{m})$. More specifically, our objective is to compute $Q_{\mathbf{m}_0}(s;\mathbf{m})$ for every possible score s within a given interval $[c,d]$. The idea is to perform the enumeration of the words in $\mathcal{A}^W$ and select the words $\mathbf{x}$ satisfying $c \leq \ell(\mathbf{x};\mathbf{m}) \leq d$, grouping them by their scores. As in the branch and bound P-value computation, we use bounds for the scores of a word starting with a given prefix to prune the enumeration. However, we now perform a breadth-first enumeration whereas in the branch and bound algorithm we performed a depth-first enumeration. This is necessary because we keep track of the possible prefix scores for all prefixes of length j in order to compute the possible scores for prefixes of length $j+1$. As a matter of fact, we do not need to keep track of all possible prefixes of length j and their scores for the computation of the possible prefix scores of length $j+1$ since only the last $k=\min\{K,j\}$ letters of the j-prefix affect the $(j+1)$-position score. We hence need only to keep a table with the possible combinations of $(j,\mathbf{k},s)$, where j is the size of the prefix, $\mathbf{k}\in\mathcal{A}^{\min\{K,j\}}$ is such that the prefix is of the form $*\mathbf{k}$ and s is a possible score for one or more prefixes of this form. The value of the entry in the table indexed by $(j,\mathbf{k},s)$ corresponds to $P_{\mathbf{m}_0}(\{\mathbf{y}\in\mathcal{A}^j \mid \mathbf{y}=*\mathbf{k}\wedge\ell_{\dots j}(\mathbf{y};\mathbf{m})=s\})$. In the end, the entries of the form $(j=W,\mathbf{k},s)$ summarize all possible scores in $[c,d]$ and the corresponding words of length W, grouped by each possible suffix $\mathbf{k}$, with

```
Algorithm Log-likelihood score distribution
Input   $\mathbf{m}$: the motif model of length $W$
        $\mathbf{m}_0$: the background model
        $c$: the lower limit of the score interval
        $d$: the upper limit of the score interval
Output  $\mathbf{Q}[s] = Q_{\mathbf{m}_0}(s;\mathbf{m})$ for all possible score $s \in [c,d]$
1  Initialise $\mathbf{S}[0,\varepsilon,0] \leftarrow 1$
2  for $j = 1,\dots,W$ do
3      for each $(\mathbf{k},s)$ s.t. $\mathbf{S}[j-1,\mathbf{k},s]$ is defined do
4          for each $a \in \mathcal{A}$ do
5              $s' \leftarrow s + \ell_j(\mathbf{k}\cdot a;\mathbf{m})$
6              $p' \leftarrow \mathbf{S}[j-1,\mathbf{k},s] \times P_{\mathbf{m}_0}(X_j = a|X_{j-|\mathbf{k}|\cdots j-1} = \mathbf{k})$
7              Let $\mathbf{k}'$ be the suffix of $\mathbf{k}a$ of size $k = \min\{j,K\}$
8              Compute $\ell^{\min}_{j+1\dots}(\mathbf{k}'\cdot;\mathbf{m})$, $\ell^{\max}_{j+1\dots}(\mathbf{k}'\cdot;\mathbf{m})$ if necessary
                 and store the result for subsequent use
9              if $c - \ell^{\max}_{j+1\dots}(\mathbf{k}'\cdot;\mathbf{m}) \leq s' \leq d - \ell^{\min}_{j+1\dots}(\mathbf{k}'\cdot;\mathbf{m})$ then
10                 if $\mathbf{S}[j,\mathbf{k}',s']$ is not defined then
11                     Define $\mathbf{S}[j,\mathbf{k}',s'] = p'$
12                 else
13                     $\mathbf{S}[j,\mathbf{k}',s'] \leftarrow \mathbf{S}[j,\mathbf{k}',s'] + p'$
14 for each $(\mathbf{k},s)$ s.t. $\mathbf{S}[W,\mathbf{k},s]$ is defined do
15     if $\mathbf{Q}[s]$ is not defined then
16         Define $\mathbf{Q}[s] = \mathbf{S}[W,\mathbf{k},s]$
17     else
18         $\mathbf{Q}[s] \leftarrow \mathbf{Q}[s] + \mathbf{S}[W,\mathbf{k},s]$
19 return $\mathbf{Q}$
```

Fig. 6. Log-likelihood score distribution.

their joint probabilities under the background model. The procedure is shown in Figure 6.

The core of the algorithm consists in computing the score distribution tables $\mathbf{S}[j,\mathbf{k},s]$ by taking the entries of $\mathbf{S}[j-1,\mathbf{k},s]$ and considering their extension to position j upon the addition of each character $a \in \mathcal{A}$, thus $|\mathbf{S}[j-1,\mathbf{k},s]|\cdot|\mathcal{A}|$ possibilities. For each possible extension, the algorithm makes use of suffix score bounds, as in the branch and bound procedure, and so the maximum time cost for computing these bounds is the same as in that algorithm. The maximum time of the table construction phase is thus given by $\sum_{j=1}^{W}|\mathbf{S}[j-1,\mathbf{k},s]|\cdot|\mathcal{A}|+\sum_{j=1}^{W}(W-j)|\mathcal{A}|^{2K+1}$. The final phase of the algorithm consists in reading the entries of $\mathbf{S}[W,\mathbf{k},s]$ to produce the final table $\mathbf{Q}[s]$. Overall, the worst case occurs when each prefix of length $j=1,\dots,W$ originates a distinct prefix score, all of them passing the pruning criteria. In this case, $|\mathbf{S}[j,\mathbf{k},s]|=|\mathcal{A}|^j$ and thus the the total cost of the algorithm (assuming $W>K$) is given by $\sum_{j=1}^{W}|\mathcal{A}|^j+\sum_{j=1}^{W}(W-j)|\mathcal{A}|^{2K+1} = \frac{|\mathcal{A}|^{W+1}-|\mathcal{A}|}{|\mathcal{A}|-1}+\frac{|\mathcal{A}|^{2K+1}(W^2-W)}{2} = O(|\mathcal{A}|^W)$, as expected. However, in practice, the iterative refinement algorithm considerably reduces the size of the score distribution tables (and therefore the running time of the algorithm) by, on the one hand, working with models of the lowest possible precision, which reduces the number of possible scores and, on the other hand, by decreasing the score interval as the model precision increases, which makes the pruning filter more strict.

2.3.3 Computing the log-likelihood threshold for a given P-value Despite the speed-up brought by the P-value computation algorithms proposed here, this problem remains computationally costly. This is why, instead of computing the P-value of a putative occurrence in order to determine if it is significant or not, it is preferable to have a likelihood threshold and test whether the likelihood of this occurrence is greater than this threshold. The algorithm proposed by Touzet and Varre (2007) for computing the maximum possible score whose P-value is greater than a given value p can also be used for K-order Markov models once we have extended the algorithms for computing the log-likelihood distribution (Fig. 6) and the maximum error (Fig. 5), and by using our adapted branch and bound algorithm as the counterpart of the algorithm 'FastPvalue' of that paper.

3 DISCUSSION

To illustrate our discussion, we report some results of an experiment performed to study the behaviour of our P-value algorithms. We have selected seven motifs with lengths ranging from 6 to 12 from the dataset used in Barash *et al.* (2003), which contains TBFS alignments extracted from the TRANSFAC database (Wingender *et al.*, 2000). We chose, for each length, the motif with the greatest number of complete aligned sites (i.e. gaps were not allowed). Next, we used each of these motifs to build a K-order Markov model for $K=0,1,2$ using the motif trie representation. Then, for each of these models, we took the same set of 20 words sampled from the uniform background model, and computed the P-value of their log-scores using the brute-force (BF), branch and bound (BB) and iterative refinement (IR) methods. The mean computa-tion times are summarized in Table 1. By analysing this table we notice that:

- The implemented heuristics significantly improve over the naive method in practice, although for small motif lengths, the gain is not substantial due to the overhead brought by the calculation of score bounds and score distributions.
- As expected, the mean computation time per P-value of the naive method remains more or less unaltered for a fixed motif length, no matter what the motif order is (SD is also low). The BB and IR methods, on the contrary, are more sensitive to the variation of the model order, since the number of possible scores varies exponentially with this number.
- For a fixed motif order, all algorithms are negatively affected by an increase of the motif length. However, the actual computation times depend on each particular model and on the distribution of the scores. It may happen that a model with a bigger length gives lower computation times if the test sequences give more extreme P-values. Hence it is difficult to compare different values from the same column.
- Comparing the values of each line, we can see that, in general, the IR method performs better than the BB method. The difference is accentuated as both the motif size and order grows since this increases exponentially the number of possible scores, a problem which is addressed more directly in the IR method by using models with lower precision.

4 CONCLUSION

In this article, we have considered two practical issues regarding the use of high-order Markov models for the representation of biological sequence motifs. First, we proposed a compact representation for a set of example motif occurrences from which position-specific conditional base probabilities can be efficiently computed. This representation can be useful in situations where the model is constantly updated, for instance, during the course of a motif learning procedure. As an example, we have included, in the software library mentioned in the abstract, an implementation of the motif site sampler (Lawrence *et al.*, 1993) which uses the trie representation to infer K-order Markov motifs in a set of unaligned sequences. In the original Gibbs sampling procedure, the model parameters are updated as soon as a new site is sampled. With the proposed representation, we only include the newly sampled

Table 1. *P*-value computation times for different values of K and W using three methods: BF, BB and IR.

W	BF	BB	IR
		$K=0$	
6	19.4 (5.11)	**10.3** (4.1)	13.6 (6.34)
7	99.85 (2.43)	**13.45** (6.36)	40.05 (14.0)
8	393.5 (1.6)	42.8 (23.3)	**31.6** (6.26)
9	2608.65 (11.77)	**274.4** (89.14)	551.6 (179.65)
10	18334.35 (26.38)	812.75 (280.43)	**487.75** (102.03)
11	?	3155.9 (1331.48)	**71.6** (9.18)
12	?	42891.95 (16387.32)	**26459.4** (10752.97)
		$K=1$	
6	15.95 (1.09)	**5.6** (2.11)	20.25 (2.8)
7	89.8 (1.23)	**36.85** (18.8)	58.3 (14.84)
8	345.65 (1.38)	164.0 (73.02)	**76.3** (18.22)
9	2417.15 (3.13)	**702.75** (306.62)	1023.75 (344.67)
10	19626.5 (19.36)	3780.5 (2135.76)	**1124.95** (376.28)
11	?	3010.15 (1146.59)	**363.25** (90.32)
12	?	**53487.65** (14953.72)	77476.75 (30706.51)
		$K=2$	
6	15.45 (0.68)	**5.65** (9.05)	53.4 (3.03)
7	69.1 (0.44)	**30.4** (20.75)	76.85 (7.76)
8	288.45 (3.26)	144.1 (58.07)	**136.35** (13.78)
9	2024.3 (2.22)	759.85 (252.66)	**602.65** (118.77)
10	17432.05 (28.64)	6668.05 (2722.57)	**1052.1** (264.83)
11	?	1779.45 (810.61)	**364.75** (77.89)
12	?	39105.9 (18679.04)	**14084.05** (4488.29)

Shown are the mean time per *P*-value computation and the SD (inside parenthesis) for each motif length and for each of the computation methods. The best mean performance for each pair (K,W) is shown in boldface. The naive enumeration method was not tested for $W > 10$.

occurrence in the trie, which can be done in linear time. Then we considered the problem of assessing the statistical significance of putative occurrences through the computation of exact *P*-values. We proposed extensions of PWM-based techniques to deal with K-order Markov models. These PWM algorithms figure among the most efficient algorithms in their category, and these extensions enrich the algorithmic toolkit of Markov dependency models contributing to the dissemination of their use. It is known, however, that standard K-order Markov models can suffer from the scarcity of training data, and therefore it will be interesting to study the adaptation of the proposed representation and algorithms to more general Markov-type dependency models such as the ones proposed by Zhao *et al.* (2005) or Huang *et al.* (2006), for instance.

ACKNOWLEDGEMENTS

The authors thank Augusto Vellozo for the helpful discussions.

Funding: P.F. was funded by a doctoral scholarship from the CAPES Foundation (Brazilian ministry of education) under co-supervision of K.G. and M.F.S. This project was also funded by an ANR research grant (Project REGLIS no. 05-NT05-3_45205) and counted on the computational infrastructure of the PRABI (Rhône-Alpes Pole of Bioinformatics).

Conflict of Interest: none declared.

REFERENCES

Bailey,T.L. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **2**, 28–36.

Barash,Y. *et al.* (2003) Modeling dependencies in protein-DNA binding sites. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB'03).* ACM, New York, NY, USA, pp. 28–37.

Beckstette,M. *et al.* (2006) Fast index based algorithms and software for matching position specific scoring matrices. *BMC Bioinformatics*, **7**, 389.

Bejerano,G. (2003) Efficient exact *P*-value computation and applications to biosequence analysis. In *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB'03).* ACM, New York, NY, USA, pp. 38–47.

Durbin,R. *et al.* (1999) *Biological sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK.

Ellrott,K. *et al.* (2002) Identifying transcription factor binding sites through markov chain optimization. *Bioinformatics*, **18**(Suppl 2), S100–S109.

Fredkin,E. (1960) Trie memory. *Comm. ACM*, **3**, 490–499.

GuhaThakurta,D. (2006) Computational identification of transcriptional regulatory elements in DNA sequence. *Nucleic Acids Res.*, **34**, 3585–3598.

Huang,W. *et al.* (2006) Optimized mixed markov models for motif identification. *BMC Bioinformatics*, **7**, 279.

Knuth,D.E. (1998) Sorting and searching. In *The Art of Computer Programming*, vol. 3 of *The Art of Computer Programming*. 2nd edn. Addison, Reading, MA, USA.

Lawrence,C.E. *et al.* (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Michalewicz,Z. and Fogel,D. (2004) *How to Solve it: Modern Heuristics*. Springer, Berlin, Germany.

Pizzi,C. *et al.* (2007) Fast search algorithms for position specific scoring matrices. In *Proceedings of the Bioinfomatics Research and Development BIRD 2007*, vol. 4414 of *Lecture Notes in Bioinformatics*. Springer, Berlin, Germany, pp. 239–250.

Touzet,H. and Varre,J.-S. (2007) Efficient and accurate *P*-value computation for position weight matrices. *Algorithms Mol. Biol.*, **2**, 15.

Wingender,E. *et al* (2000) Transfac: an integrated system for gene expression regulation. *Nucleic Acids Res.*, **28**, 316–319.

Zhang,J. *et al.* (2007) Computing exact *P*-values for DNA motifs. *Bioinformatics*, **23**, 531–537.

Zhao,X. *et al.* (2005) Finding short DNA motifs using permuted markov models. *J. Comput. Biol.*, **12**, 894–906.