A NOVEL APPROACH FOR STRUCTURED CONSENSUS MOTIF INFERENCE UNDER SPECIFICITY AND QUORUM CONSTRAINTS

CHRISTINE SINOQUET

LINA, Université de Nantes,
2 rue de la Houssinière, BP 92208, 44322 Nantes Cedex, France,
E-mail: christine.sinoquet@univ-nantes.fr

We address the issue of structured motif *inference*. This problem is stated as follows: given a set of n DNA sequences and a quorum q (%), find the optimal structured consensus motif described as gaps alternating with specific regions and shared by at least $q \times n$ sequences. Our proposal is in the domain of metaheuristics: it runs solutions to convergence through a cooperation between a sampling strategy of the search space and a quick detection of local similarities in small sequence samples. The contributions of this paper are: (1) the design of a stochastic method whose genuine novelty rests on driving the search with a threshold frequency f discrimining between specific regions and gaps; (2) the original way for justifying the operations especially designed; (3) the implementation of a mining tool well adapted to biologists'exigencies: few input parameters are required (quorum q, minimal threshold frequency f, maximal gap length g). Our approach proves efficient on simulated data, promoter sites in Dicot plants and transcription factor binding sites in *E. coli* genome. Our algorithm, Kaos, compares favorably with MEME and STARS in terms of accuracy.

1. Introduction

In the last fifteen years a lot of work has appeared in the literature which addresses the general topic of *consensus motif inference* (CMI) in a set of biological sequences (Gibbs sampling,[10,16] Pratt,[9] CONSENSUS,[8] Teiresias,[14] MEME,[2] PROJECTIONS,[4] Smile,[12,6] Winnower,[13] Splash,[5] STARS,[11] MoDEL,[7] ...). The reader is directed to some recent surveys[3,11,7]. Though the problem is incontestably not new, it remains an important and difficult one in sequence analysis. Exact algorithms are not convenient for large datasets or long sequences. Therefore, approximate algorithms have to be designed. There was room for an approach dedicated to the specific problem of *structured* motif inference. Let us first remind of a peculiar instance of the CMI problem, called the local multiple alignment problem (LMA). In its general form, this problem may be stated as follows: identify a set of sub-words $\{o_{i_1}, o_{i_2}, \cdots, o_{i_{r_i}}\}$, called occurrences, under the four following constraints: (1) any o_{i_j} verifies a similarity constraint with any other o_{i_p}; (2) the occurrences have the same length; (3) each sequence s_i contains 0 or 1 such occurrence o_{i_k}; (4) there are at least $q \times n$ sequences s_i contributing to the set of occurrences (quorum constraint). Generally, LMA algorithms output the set of occurrences, a position-specific scoring matrix ($pssm$) M and the consensus motif. For any character c of the DNA sequence alphabet $\mathscr{A}$, $M[c, j]$ yields the frequency of character c at position j over the set of occurrences. The consensus motif is computed as the word with the most frequent characters in the $pssm$ M. But a

central issue in post-genomics is the automatic discovery of functional biological motifs with the aim of identifying a structure common to a set of sequences. The structure may be described as specific regions alternating with gaps. Gaps are regions which contain no intrinsic information. Thus we are interested in structured local multiple alignments (SLMA$_s$). A structured motif is a word built on alphabet $\mathscr{A} \cup \{\mathtt{x}\}$, where x denotes the wild-card character. A wild-card character in position j of the motif indicates that the frequencies of all characters in $\mathscr{A}$ are below a given threshold frequency, say, f. A gap is made up of contiguous wild-card characters. Here our concern is the retrieval of structured motifs such as ACTGxxxxCTTxxGGxxxAAGA, for example.

Naively deriving a structured consensus motif from the $pssm$ built by a classical LMA method does not yield the optimal solution for large datasets or long sequences. Nevertheless, it is wise to benefit from an existing LMA algorithm for local similarity search. On the other hand, for systematic data mining purpose, one can not waste time with successive guesses at the putative structure. The number of input parameters for a search must be reduced: quorum (q); threshold frequency (f); minimal motif length ($minMotifLength$), maximal gap length (g). Thirdly, we are aware of the robustness of stochastic methods in the domain of CMI (Gibbs sampler,[10] MEME,[2] PROJECTIONS,[4] STARS,[11]). Finally, we wish to design an algorithm with a low memory cost. These motivations lead to our investigating a stochastic sampling strategy cooperating with an LMA algorithm.

The remainder of the paper is organized as follows. Section 2 introduces specific terminology and notions we use subsequently. Section 3 is dedicated to the presentation of our algorithm, Kaos. The discussion of the results obtained on simulated and biological data may be found in Section 4.

2. Definitions

For DNA sequences, the alphabet $\mathscr{A}$ is $\{A,\ C,\ T,\ G\}$.

Definition 2.1. ***(LMA)*** Given n sequences $s_1,\ s_2 \cdots s_n$ built on alphabet $\mathscr{A}$, an integer k and a similarity criterion, an LMA is a set of substrings $o_1,\ o_2 \cdots o_n$ of $s_1,\ s_2 \cdots s_n$ such that $o_1,\ o_2 \cdots o_n$ have common length k and maximize the similarity criterion. Note that an LMA is easily represented with a matrix. In the following, we will also call LMA any process yielding such a set of substrings.

Definition 2.2. **(pssm, support, hits)** Let $\mathcal{S} = s_1,\ s_2 \cdots s_n$ be n sequences built on alphabet $\mathscr{A}$, let $\mathcal{O} = \{o_1,\ o_2 \cdots o_n\}$ be an LMA of length k built from these sequences considering a given similarity criterion and let $minSeq$ and $minMotifLength$ be two integers. A *pssm* M associated with this LMA is any matrix of reals of size $|\mathscr{A}| \times l$ ($l \geq minMotifLength$) induced from a sub-matrix L of the LMA matrix as follows: L has at least $minSeq$ lines and $minMotifLength$ columns; M compiles the frequencies of the characters of $\mathscr{A}$, for each column of L. The **hits** of the $pssm$ M are the locations in the sequences of the sub-words aligned in L. These sub-words are the **support** of the *pssm*.

Definition 2.3. **(specific character)** Let f be a given threshold frequency, $0 \leq f \leq 100$. If the highest frequency at column j, $M^+[j]$, is over f and is obtained for character c, then

c is a *specific* character and is refered to as $char(M^+[j])$.

Definition 2.4. (wild-card character, mask) Let us choose x ($\notin \mathscr{A}$) as the *wild-card character*. We denote $mask(M)$ the string $\in \mathscr{A}^l \cup \{\mathsf{x}\}$ verifying:
forall $1 \leq j \leq l$, $mask(M)[j] = \begin{cases} \mathsf{x} \ if \ M^+[j] < f \\ char(M^+[j]) \ otherwise. \end{cases}$

Definition 2.5. (gap) A gap is any word built on alphabet $\{\mathsf{x}\}$ which is one of the longest substrings of contiguous wild-card characters in $mask(M)$.

The only constraint on the motif structure is g, the maximal gap length allowed.

Definition 2.6. (property $\mathcal{G}(f,g)$) Property $\mathcal{G}(f,g,M)$ holds if and only if the length of the longest gap in M's mask is less than or equal to g for the threshold frequency f and this mask has no gap at either extremity.

3. Cooperation between a sampling strategy and an LMA

3.1. *Moving in the $pssm$ search space*

To solve a combinatorial optimization problem, all metaheuristics find a balance between search intensification (identifying solutions of better quality from known solutions) and search diversification (escaping from local optima). Our approach successively infers solutions through iterations considering small samples of m sequences chosen at random among the n initial sequences. If at least $q \times n$ sequences share a similarity, it is likely that $q \times m$ sequences in the samples share this similarity on average. The searched space $\mathcal{E}$ consists of *pssm*$_s$. But we would emphasize that the definite originality of our method lies in the following points: (1) where Gibbs sampling, PROJECTIONS, MEME and CONSENSUS (at each of its iterations) consider $pssm_s$ for a *given* motif length and $pssm_s$ supported by the *same* number of sequences, on the contrary, our method considers the space of *pssm*$_s$ with minimal second dimension $minMotifLength$ and minimal support size $minSeq$; (2) moreover, for structured motif inference purpose, we impose straightaway that we only move in the $pssm$ sub-space where all elements verify property $\mathcal{G}(f,g)$. Each *pssm* represents a similarity shared by, say, r sequences belonging to the initial set. The higher r is, the more likely is the *pssm* the optimal solution. The objectives for the search are increasing the support size (quorum constraint) and optimizing a criterion $\mathscr{C}$ designed to evaluate the specificity of solutions.

3.2. *Sketch of the algorithm*

3.2.1. *Sequence sampling and first intensification level*

The sketch of the search is given in algorithm 1. Any iteration begins with the generation at random of two sequence samples S_1 and S_2 (line 3). The plug-in LMA software tuned with length w yields two LMA matrices of size $m \times w$ (line 4). An exact procedure is implemented to scan each of these matrices and identify the largest sub-matrices (with

respect to the numbers of lines and columns) whose $pssm_s$ verify the constraint $\mathcal{G}(f,g)$. This procedure will be described in an extended version of the paper.

At line 5, operator $\otimes$ processes each pair $(s_{1_1},\ s_{1_2})$ of $\mathscr{S}_{1_1} \times \mathscr{S}_{1_2}$ shifting one matrix with regard to the other one to identify a local similarity. As a result, the elementary operation $s_{1_1} \times s_{1_2}$ outputs the **unique** $pssm\ s_2$ (if it exists) which satisfies property $\mathcal{G}(f,g)$ and optimizes criterion $\mathscr{C}$. s_2 is computed according to the formula: $s_2[c,j] = \frac{n_{1_1} s_{1_1}[c,j_1] + n_{1_2} s_{1_2}[c,j_2]}{n_{1_1}+n_{1_2}}$. j_1 and j_2 are the columns of matrices s_{1_1} and s_{1_2} which contribute to the column j of matrix s_2. n_{1_1} and n_{1_2} are the support sizes of s_{1_1} and s_{1_2}. Indeed, such computations need only be done for some shifts. The other shifts are efficiently rejected through straightforward focuses on the gaps in s_{1_1} (or s_{1_2}) (starting with the longest ones), and the corresponding regions in s_{1_2} (or s_{1_1}). Thus it is likely that the longest gaps in the potential solution corresponding to the current shift will be pointed out. For a given shift, optimization is performed through scanning $\mathscr{L}$, the list of pairs $(begin,\ end)$ for all gaps in s_{1_1} and s_{1_2}. This list is sorted in decreasing order with respect to the gap lengths. The principle of the optimization will be detailed in the extended version of the paper.

3.2.2. *Starting points and second intensification level*

If they are of sufficient quality with respect to criterion $\mathscr{C}$, some elements of $\mathscr{S}_2$ will be chosen as "starting points" for moving in the search space $\mathcal{E}$ (line 6). A starting point from $\mathscr{S}_2$ will replace a "bad" current solution in $\mathscr{S}_3$. Moving in $\mathcal{E}$ is performed with the objective of maximizing the support sizes (see definition 2.2) of the current solutions in $\mathscr{S}_3$. Here, intensification is obtained with a second operator $\oplus$ (line 6) which compares the elements of all pairs in $\mathscr{S}_3 \times \mathscr{S}_2$: if a solution s_2 in $\mathscr{S}_2$ has the same *mask* as a solution s_3 belonging to $\mathscr{S}_3$, then the support and frequencies of s_3 are updated. $\oplus$ is a specialized version of $\otimes$. Furthermore, if updating the frequencies for a given solution s^* with support size above $q \times n$ only entails variations within a given percentage, say, 2%, then the convergence criterion is satisfied (line 7), the search stops and it yields the pair $\{s^*,\ \mathscr{S}_3\}$ which will be processed afterwards. Now the reader can understand that a "bad" solution s_3 in $\mathscr{S}_3$ replaced with a starting point s_2 is characterized as follows: (1) it was not much reinforced by operator $\oplus$ through successive iterations and therefore it has a low support size; (2) s_2 scores over s_3 with regard to criterion $\mathscr{C}$. To sump up, diversification by generation of starting points and intensification in $\mathcal{E}$ are simultaneous processes iterated until a convergence criterion is satisfied for a solution s^*.

3.3. *Justification of the operators implemented and the criterion optimized*

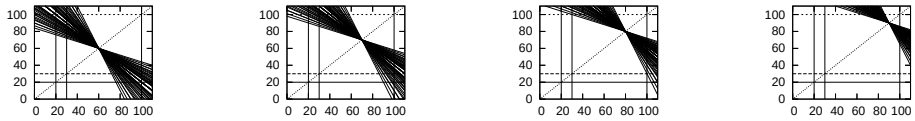
To evaluate the pertinency of operator $\otimes$, we must check that it is unlikely that false positive solutions corresponding to local similarities might be retained in $\mathscr{S}_2$. If we call f_1 and f_2 the two frequencies involved in the formula $s_2[c,j] = \frac{n_{1_1} s_{1_1}[c,j_1] + n_{1_2} s_{1_2}[c,j_2]}{n_{1_1}+n_{1_2}}$, Figure 1 shows variations of f_2 *versus* f_1 for 4 values of f and different values of ratio $\frac{n_{1_1}}{n_{1_2}}$ under the constraint $\frac{n_{1_1}\ f_1 + n_{1_2}\ f_2}{n_{1_1}+n_{1_2}} \geq f$. The ratio values considered here are the 49 our

Algorithm 1 $Search(g, f, q)$

Input: a set S of n sequences; maximal gap length g; minimal frequency f; quorum q; sample size m.
Output: answer 'Yes'/'No'; if 'Yes', a mask $\mathcal{M}ask$, the corresponding *pssm* M, hits $hits$ for at least $q \times n$ sequences.
Search space: $\mathcal{E}$, the set of *pssm*$_s$ verifying constraint $\mathcal{G}(f, g)$
Solution sets: $\mathscr{S}_{1_1}$, $\mathscr{S}_{1_2}$, $\mathscr{S}_2$ *and* $\mathscr{S}_3$
Operators: $\otimes$ and $\oplus$: $\mathscr{P}(\mathcal{E}) \times \mathscr{P}(\mathcal{E}) \rightarrow \mathscr{P}(\mathcal{E})$ ($\mathscr{P}(E)$: set of all subsets of E)

1: $\mathscr{S}_3 \leftarrow \emptyset$
2: **repeat**
3: Step 1: generate at random from S two samples of m sequences respectively, S_1 and S_2.
4: identify at most n_1 best solutions $\mathscr{S}_{1_1}$ and at most n_1 best solutions $\mathscr{S}_{1_2}$ running plug-in algorithm LMA respectively on samples S_1 and S_2, under constraint $\mathcal{G}(f, g)$ and using criterion $\mathscr{C}$.
5: Step 2: identify at most n_2 best solutions $\mathscr{S}_2$ from $\mathscr{S}_{1_1} \otimes \mathscr{S}_{1_2}$, under constraint $\mathcal{G}(f, g)$ and with respect to criterion $\mathscr{C}$.
6: Step 3: update $\mathscr{S}_3$ with $\mathscr{S}_3 \oplus \mathscr{S}_2$ and possibly new starting points from $\mathscr{S}_2$, under constraint $\mathcal{G}(f, g)$ and using criterion $\mathscr{C}$.
7: **until** (convergence criterion is satisfied for a solution s^* in $\mathscr{S}_3$ verifying quorum constraint) or timeout is reached
8: **if** such a solution s^* exists **then** motifAssembling(s^*, $\mathscr{S}_3$); **return** 'Yes', $\mathcal{M}ask$, M, $hits$
9: **return** 'No'

implementation works with ($n_{1_1}, n_{1_2} \in [4, 10]$; $m = 10; minSeq = 4$, see definition 2.2). As intuitively expected, the probability that the frequency $f_1 = 25\%$ of an aleatory character might be compensated by a sufficiently high value of $f_{2min} = h(f_1) = (f - f_1)\ \frac{n_{1_1}}{n_{1_2}} + f$, to yield a false specific character, drastically decreases as f increases. We draw vertical lines $x = 25 \pm 5\%$ and horizontal lines $y = 25 \pm 5\%$ to delimit areas for frequencies near to 25%. Through $f = 60\%$ to $f = 90\%$, the two areas delimited intersect a decreasing number of lines $y = h(x)$.



f = 60% f = 70% f = 80% f = 90%

Figure 1: Probability for reinforcement of a specific character c through operation $\otimes$, under various values for the threshold frequency f ($f \geq 60\%$). We draw the lines corresponding to $f_{2min} = h(f_1) = (f - f_1)\ \frac{n_{1_1}}{n_{1_2}} + f$ for 49 values of the ratio $\frac{n_{1_1}}{n_{1_2}}$; $n_{1_1}, n_{1_2} \in [4, 10]$.

Conclusion 3.1. For DNA sequences, if the threshold frequency f is greater than or equal to 70% and if all 49 possibilities for $\frac{n_{1_1}}{n_{1_2}}$ with $n_{1_1},\ n_{1_2} \in [4, 10]$ are equiprobable, then the probability that an aleatory character in s_{1_1} compensates the same character in s_{1_2}, to yield a specific character in s_2, decreases from 0.45 to 0 (0.29 for $f = 72\%$; 0.12 for $f = 75\%$).

With this knowledge, we study the behaviour of operator $\otimes$ in the three cases: TP $\times$ TP, FP $\times$ TP, FP $\times$ FP, where FP denotes a false positive solution (a local similarity) and TP corresponds to a sub-optimal solution. The optimal solution is a similarity shared by $n \times q$ sequences at least and a TP only differs from the optimal solution by some false wild-card or specific characters. Table 3 in Appendix details our reasoning, which is based on conclusion 3.1. We draw the following conclusion from the comparison of columns 3, 3', 5 and 5' of Table 3 (see lines 3, 6 and 9):

Conclusion 3.2. Table 3 shows that crossing operands one of which at least is a false positive solution is likely to entail the generation of wild-card characters whereas specific characters are generated instead if both operands are true positive solutions.

Operator $\otimes$ implements shifts of a $pssm$ with respect to another $pssm$. If one of the operands at least is a FP, it is unlikely that many specific characters of both $pssm_s$ correspond. Were it the case for a pair of specific characters of both operands for a given shift, it is unlikely that these specific characters would be identical. So the cases mentioned at (6,2) and (9,2) in Table 3 are highly improbable. As a consequence of this and conclusion 3.2, conclusion 3.3 is stated as follows:

Conclusion 3.3. Crossing operands ($s_{1_1} \times s_{1_2}$) one of which at least is a false positive solution is likely to yield a $pssm$ whose mask has wild-card characters in the majority. The resulting $pssm$ will be rejected because of the presence of gaps at extremities or because they do not verify the maximal gap length constraint.

The criterion $\mathscr{C}$ can not be the usual log-likelihood ratio[1] because contrary to other LMA methods, our $pssm_s$ are **not** supported by the same numbers of sequences. Conclusion 3.3 gives a strong motivation to reward solutions with the following criterion: $\sum_{j=1,\ M^+[j] \geq f}^{l} M^+[j]$.

3.4. *Final processing*

3.4.1. *Motif assembling*

At line 8, among the solutions in $\mathscr{S}_3$, some may be false positive solutions while others may correspond to sub-regions of the final consensus motif, either strictly included in the sub-region corresponding to s^*, overlapping it or totally disjointed from it. Procedure *motifAssembling* chooses s^* as a first "reference" and refines it. Then it finds the solution in $\mathscr{S}_3$ having the greatest number of co-occurring hits with the reference and satisfying quorum constraint. This solution is refined in its turn and is used as the new reference. This process is repeated until no such reference can be identified. The procedure will be detailed in the extended version of this paper.

3.4.2. *Estimation of the motif statistical significance*

It is not a rule that all functional biological motifs should necessarily be less represented than other words in a dataset. Anyway, we must estimate the statistical significance of the motif predicted. We consider a motif of length l with ns specific characters. The probability that such a motif occurs by chance at least one time in each of n sequences of maximal size t, with at most d mismatches relative to the specific characters, is approximated by $(1 - (1 - \sum_{m=0}^{d} \binom{ns}{m} (\frac{|\mathscr{A}|-1}{|\mathscr{A}|})^m (\frac{1}{|\mathscr{A}|})^{ns-m})^{t-l+1})^n$.

3.5. *Complexity*

We chose MoDEL[7] as the plug-in LMA software for its rapidity. The complexity of an LMA performed with MoDEL is $O(m\,t\,w\,b)$ where w is the **chosen length** for the local alignment,

b is the intrinsic number of iterations for this method (default value is 45), m is the size of the sequence samples and t is the maximal length for the n sequences. We suppose that the convergence for procedure $Search(g,\ f,\ q)$ is obtained at u^{th} iteration. The complexity is then $O(u\ m\ t\ w\ b + \lambda^2\ {n_3}^2\ n)$ where λ is the maximal number of hits per sequence for any of the n_3 solutions in $\mathscr{S}_3$. The memory cost is $O(2\ n_1 + n_2 + n_3)\ (n\ \lambda + w)$ where n_1 and n_2 are the sizes of $\mathscr{S}_{1_1}$ and $\mathscr{S}_2$. Due to space limitation the corresponding proofs will be published in the extended version of the paper.

4. Experimental results

4.1. *Simulations*

Generating a consensus motif at random, we also generated $n \times q$ occurrences, blurred them with mismatches (in the specific regions only) and inserted them in n aleatory generated sequences. We compared the retrieved consensus motif with the real one and computed statistics: $cover$ (Is the whole motif retrieved?) and $exactness$ (Are there errors?) (not detailed here). Table 1 includes more details on the false wild-card or the false specific characters predicted. For some tests ($b,\ c,\ d$ and e), we adapted to our concerns the so-called challenge motif problem stated by Pevzner and Sze[13]. The first conclusion to draw from Table 1 is that our method is accurate. The lowest average exactness is 0.92, obtained for motif a and c under rather hard conditions: $q = 70\%;\ f = 80\%$. The cover is quite satisfying too: it never drops below 0.90. On the other hand, we never encounter more than one false wild-card character or one false specific character on average, except for b_2 and c_2 (respectively 1.03 and 1.02 false wild-card characters).

A second benchmark was designed to study whether tuning the parameter g (maximal gap length) with different values alters the results. The benchmark consists of 50 sequences generated at random (length in $[50,\ 300]$) and containing CTACTxxxATCCTTGGGxxxxxxxxTCGTxxxAAACTTGCTAGATTCAGGGAxxxGACGGAGGGTA whose longest gap has length 8. Successively tuning g to 8, 15 and 25 does not alter the quality of the ouputs obtained for 20 runs.

4.2. *Biological benchmarks*

Finally we ran Kaos on the biological datasets collected for STARS evaluation[17,18]. The motifs considered consist of (1) the Tata box TATAxATA in 131 sequences from various Dicot plants (32880 nucleotides, sequence length 251), (2) the binding site TGTAAxxxxxxxTTxAC for TyrR protein in *E. coli* genome (5 sequences, 3585 nucleotides, sequence length in [251-2021]) and (3) the binding site CTGTAxAxxxAxxCAG for LexA protein in *E. coli* genome (16 sequences, 28941 nucleotides, sequence length in [100-3842]). We compared MEME, STARS and 20 runs of Kaos under the constraints $q = 100\%$ and $f = 80\%$. To test our method with quorum $q = 70\%$ (and $f = 80\%$), we replaced 30% of the sequences in the initial datasets (1) and (3) with as many sequences of the same lengths chosen at random in the adequate genomes. Then we compared 20 runs of Kaos with MEME, which allows the retrieval of zero or one occurrence of the motif per sequence. We had to convert

MEME's outputs into structured motifs for comparison purpose. We compared the lengths of the motifs and the locations of false wild-card/specific characters for the motifs predicted by Kaos and any other algorithm. We conclude that in both cases ($q = 100\%$ and $q = 70\%$) Kaos is as efficient as MEME and STARS (see extended version).

Then we tested the behaviour of Kaos when the quorum decreases. We chose a difficult case: the binding sites for PurR protein (18 sequences, 44592 nucleotides) (see Table 2). From 90% to 70% all solutions found by Kaos are consistent with the real motif, with each other and with MEME's results. 60% is the quorum value below which MEME and Kaos retrieve a consensus depending on the sequences generated at random.

Table 1: Performances of Kaos with 2 quorum values and 2 frequency values for various motifs.

	(a1)	(a2)	(a3)	(a4)	(b1)	(b2)	(c1)	(c2)	(d1)	(d2)	(e1)	(e2)
q	100	70	100	70	100	70	100	70	100	70	100	70
f	100	100	80	80	80	80	80	80	80	80	80	80
co	0.97	0.97	0.91	0.90	1.0	1.0	1.0	0.97	0.92	0.93	0.91	0.95
ex	1.0	1.0	0.98	0.92	0.98	0.96	0.95	0.92	0.96	0.95	1.0	1.0
fw	0.49	0.5	0.58	0.57	0.5	1.03	0.2	1.02	0.37	0.47	0.11	0.3
fs	0.38	0.37	0.13	0.9	0	0.44	0.9	0.3	0.22	0.21	0.2	0.95

	n	t	l	ns	$q = 100\%$			$q = 70\%$		
					$d = 2$	$d = 4$	$d = 7$	$d = 2$	$d = 4$	$d = 7$
(a)	100	50	14	10	2.4 E-182	5.4 E-29	0.999	1.4 E-96	2.2 E-4	0.999
(a)	100	300	14	10	1.3 E-95	0.720	1.0	3.3 E-43	1.0	1.0
(b)	20	600	16	12	$d = 3$: 1.7 E-14			$d = 3$: 2.5 E-6		
(c)	20	600	18	14	$d = 4$: 1.4 E-15			$d = 4$: 5.1 E-7		
(d)	20	600	20	16	$d = 5$: 4.8 E-17			$d = 5$: 5.8 E-8		
(e)	20	600	22	18	$d = 6$: 9.1 E-19			$d = 6$: 4.3 E-9		

(a) GCGxxAAGCAxxCC (b) TGATxxTGAxxACGCC (c) TTTxCTCxCGxCCGxgag
(d) AATTTxxTCCTAGxTxTACG (e) CTTGGACxCGAxCCTCxxCGCC

Note: The maximal gap length g is set to 5. (a), (b), (c), (d) and (e) refer to various motifs. In each subcase (a_i), 100 sequences of lengths ranging from 50 to 300 have been generated under quorum $(q\%)$ and minimal frequency $(f\%)$ constraints. In cases (b), (c), (d) and (e), 20 sequences of length 600 have been generated. Average values for cover (co), exactness (ex) and number of false wild-card/specific characters (fw/fs) predicted have been computed for 100 runs. Bottom section: statistical significance. n is the number of sequences of size t, l is the motif length, ns is the number of specific characters, d is the maximal number of mismatches observed per occurrence. For motif (a) different values of d were encountered in the worst cases ($f = 70\%$).

Table 2: Robustness of Kaos with respect to quorum q; comparison with other methods.

q	100%	90%	80%	70%
MEME	ACGCAAACGTTTGCGTT	see 100%	ACGCAAACGTTTGCGT	ACGCAAACGTTTACGTT
MEME(∗)	AxGCAAACGxTTxCxT	see 100%	AxGCAAACGTTTxCxT	AxGCAAACGTTTACTT (8) AxGCAAACGTTTxCGT (3) AxGCAAACGTTTCxT (9)
STARS	GCxAxCGTTTTC			
Kaos	GxAAxCGxTTxC (7) AxGxAAACGTTTxCxT (13)†	AxGxAAxCGxTTxC (10) AxGxAAACGxTTxC (1) CGCAAxCGxTTxC (9)	GxAAxCGxTTxCxT (12) CGxAAACGTTTxCxT (8)	AxGxAAACGxxTxC (6) ‡ CAAAxGTxTCxGT (7) CxAACGTxTTxGT (7)

Note: $f = 80\%$. The maximal gap length is 5. The dataset contains sequences with PurR protein binding sites and random sequences. The known motif is AxGxAAxCGxTTxCxT. Sequences have lengths in range [299-5864], with average 2477. (*) For an easier comparison, the ouputs of MEME have been converted into structured motifs. The numbers in brackets indicate how many runs over 20 output the corresponding result. Statistical significance: † 1.2 E-69; $d = 0$ ‡ 0.999; $d = 4$. d is the maximal number of mismatches observed per occurrence.

5. Conclusion

We presented a novel method for SLMA under minimal specific character frequency, maximal gap length and quorum constraints. *pssm* convergence is obtained in an original way: strengthening (literally *merging*) the best candidates satisfying frequency and gap constraints, which definitely distinguishes our algorithm from all known methods. Kaos is robust with regard to the following criteria: maximal gap length specified with too high a value, parameters q and f. Besides, our method has a low memory cost. Finally, our software is easy to use since only three input parameters are required and the maximal gap length may be overestimated. Our results show that it is worth doing future work. The next step will consist in examining which parts of the algorithm may be speeded up. A more thorough examination of the choice for scoring functions is also one of our future tasks.

Acknowledgements

We wish to thank A. Mancheron for kindly putting at our disposal the datasets he collected. Thanks are also due to D. Hernandez for providing the beta version of the MoDEL software.

Appendix

Table 3: Study of the behaviour of operator $\otimes$ *w.r.t.* the type of operands (FP or TP, see text, subsection 3.3).

			1	2	3	3'	4	4'	5	5'	6	6'
		s_{opt}	x	c	c_1	c_1	x	x	c	c	x	x
		s_{1_1}	x	c	c_1	c_2	c_1	c_2	c	x	c	x
		s_{1_2}	x	c	c_2	c_1	c_2	c_1	x	c	x	c
1	TP×TP	s_{1_1}			$1c_1$ $2c_2$	$1c_2$ $2c_1$	c_1 c_2?	c_2 c_1?	c	$c+$	c	c eq
2		s_{1_2}			$1c_2$	$1c_1$	c_2	c_1	$c+$	c	c eq	c
3		s_2	x	c	c_1	c_1	x	x	c	c	x	x
4	TP×FP	s_{1_1}			$1c_1$ $2c_2$?	$1c_2$ $2c_1$?	$1c_1$ $2c_2$?	see 4	c	$c+$	c	c eq
5		s_{1_2}			$1c_2$	$1c_1$	$1c_2$	see 4	c?	c	c?	c
6		s_2	x	c	x	x	x	x	x	c	x	x
7	FP×FP	s_{1_1}			c_1 c_2?	see 3	see 3	see 3	c	x	see 5	see 5
8		s_{1_2}			c_2				x	c		
9		s_2	x	c	x				x	x		

Note: This table relies on conclusion 3.1. The operands are s_{1_1} and s_{1_2}. The elementary operator for $\otimes$ is denoted $\times$. We give the most probable result for $s_2 = s_{1_1} \times s_{1_2}$, considering the 2 contributing columns of s_{1_1} and s_{1_2} and the resulting column for s_2. x denotes a wild-card character in the masks of the $pssm_s$ considered. c, c_1 and c_2 denote specific characters. s_{opt} is the optimal solution. Indicating the character of s_{opt} is obviously only of concern for operations involving TP$_s$. Notations: $'1\ c'_1$ recalls that c_1 **has** the highest frequency rank in the column of the $pssm$ considered; $'2\ c'_2$ means that c_2 is **likely** to have the second frequency rank; $'c_1+'$ denotes the character c_1 is likely to have a high frequency rank; $'c_1?'$ means that there is no possible guess as to what the rank for c_1 can be; 'c_1 eq' means that c_1 has a frequency averaging $\frac{1}{|\mathscr{A}|}$ as all equiprobable characters. We comment line TP×TP and columns 3 and 3' to show how to read this table: if one of the TP$_s$ has c_1 as a specific character, corresponding to the specific character of s_{opt}, and the other TP has c_2 as a specific character, since both are TP$_s$, it is likely that c_1 appears with the second frequency rank for the second TP. The cases at lines and columns (4-6,5) and (4-6,5') are not symmetrical. c may be encountered in a FP with either a low or high probability (though < f since the corresponding character of the mask is a wild-card character). Thus the notation $'c?'$ is adequate. On the contrary, there is a high probability that the character c is high-ranked if s_{1_1} has a false wild-card character and s_{opt} has character c in its mask. In this case, the adequate notation is $c+$.

Bibliography

1. T.L. Bailey, Likelihood vs. information in aligning biopolymer sequences, USCD technical report cs93-318. University of California, San Diego, 1993.
2. T.L. Bailey, C. Elkan, Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning*, 21, 51–80, 1995.
3. B. Brejová, C. DiMarco, T. Vinar, S.R. Hidalgo, G. Hoguin, C. Patten, Finding patterns in biological sequences. Tech. Rep. CS798g, University of Waterloo, 2000.
4. J. Buhler, M. Tompa, Finding motifs using random projections. In *Proceedings of the Fifth International Conference on Computational Molecular Biology (RECOMB)*, 69–76, Montréal, Canada, ACM Press, apr, 2001.
5. A. Califano, SPLASH: Structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16(4), 341–357, 2000.
6. A.M. Carvalho, A.T. Freitas, A.L. Oliveira, M.-F. Sagot, A highly scalable algorithm for the extraction of cis-regulatory regions. In Yi-Ping Phoebe Chen and Limsoon Wong, ed., *Proceedings of the 3rd Asia Pacifi c Bioinformatics Conference (APBC)*, volume 1 of *Advances in Bioinformatics and Computational Biology*, Imperial College Press, 273–282, 2005.
7. D. Hernandez, R. Gras, R. Appel, MODEL: an efficient strategy for ungapped local multiple alignment. *Computational Biology and Chemistry*, 28, 2, 119-128, apr, 2004.
8. G. Hertz, G. Stormo, Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15, 563–577, 1999.
9. I. Jonassen, Efficient discovery of conserved patterns using a pattern graph. *Computer Applications in the Biosciences*, 13, 509-522, 1997.
10. C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, J.C. Wootton, Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262, 208–214, oct, 1993.
11. A. Mancheron, I. Rusu, Pattern discovery allowing gaps, substitution matrices and multiple score functions. In *Proceedings of the Third Workshop of Algorithms in Bioinformatics (WABI)*, 2812, 129–145, Budapest, Hungary, Springer-Verlag, LNBI, sep, 2003.
12. L. Marsan, M.-F. Sagot, Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory site consensus identification. *Journal of Computational Biology*, 7, 345–360, 2000.
13. P.A. Pevzner, S.-H. Sze, Combinatorial algorithm for finding subtle signals in DNA sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 269–278, San Diego, California, aug, 2000.
14. I. Rigoutsos, A. Floratos, Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14(1), 55–67, 1998.
15. T.D. Schneider, G.D. Stormo, L. Gold, and A. Ehrenfeucht, Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188, 415-431, 1986.
16. W. Thompson, E.C. Rouchka, C.E. Lawrence, Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Research*, 31, 13, 3580–3585, 2003.
17. http://www.softberry.com/berry.phtml
18. http://arep.med.harvard.edu/ecoli_matrices/