MoDEL: an efficient strategy for ungapped local multiple alignment

David Hernandez*, Robin Gras, Ron Appel

Proteome Informatics Group, Swiss Institute of Bioinformatics, CMU, 1 rue Michel-Servet, CH 1211 Geneva 4, Switzerland

Received 20 December 2003; received in revised form 26 December 2003; accepted 6 January 2004

Abstract

We introduce a method for ungapped local multiple alignment (ULMA) in a given set of amino acid or nucleotide sequences. This method explores two search spaces using a linked optimization strategy. The first search space $M$ consists of all possible words of a given length $W$, defined on the residue alphabet. An evolutionary algorithm searches this space globally. The second search space $P$ consists of all possible ULMAs in the sequence set, each ULMA being represented by a position vector defining exactly one subsequence of length $W$ per sequence. This search space is sampled with hill-climbing processes. The search of both spaces are coupled by projecting high scoring results from the global evolutionary search of $M$ onto $P$. The hill-climbing processes then refine the optimization by local search, using the relative entropy between the ULMA and background residue frequencies as an objective function. We demonstrate some advantages of our strategy by analyzing difficult natural amino acid sequences and artificial datasets. A web interface is available at http://idefix.univ-rennes1.fr:8080/PatternDiscovery/

Keywords: Local multiple alignment; Pattern discovery; Genetic algorithm

1. Introduction

Localization of conserved motifs or domains in nucleotide or amino acid sequences is an essential goal of much biological research. Alignments of such motifs can form the basis of pattern signatures or probabilistic models that can be used to identify similar motifs or domains in new sequences (Bucher, 1990; Sigrist et al., 2002; Bateman et al., 2003, Attwood et al., 2003), and can provide a starting point for evolutionary studies. In most cases, the localization is performed with multiple alignment methods like CLUSTALW (Thompson et al., 1994), T-COFFEE (Notredame et al., 2000), or DIALIGN2 (Morgenstern, 1999), and can provide a starting point for evolutionary studies. In most cases, the localization is performed with multiple alignment methods like CLUSTALW (Thompson et al., 1994), T-COFFEE (Notredame et al., 2000), or DIALIGN2 (Morgenstern, 1999), which are based on pairwise alignments. However, these pairwise methods could miss a subtle similarity that makes sense only when all occurrences are considered simultaneously (Notredame, 2000). Accordingly, other methods investigate the multiple sequences together, or infer consensus words or patterns that match all or a given number of sequences in the dataset. Generally, these methods allow the detection of weaker signals in the sequences, but on the other hand, they need more constraints to reduce the search space. They are thus more specific than classical multiple alignments. A large variety of such methods have been developed, with different specific goals. Here, we will use the term occurrence to speak about any contiguous set of symbols present in the sequence set. A class of methods infers gapped-words (words with fixed or variable length wild-cards), having exact occurrences in a given number of sequences. The main ones are Pratt (Jonassen et al., 1995), Splash (Califano, 2000) and TERE-SIAS (Rigoutsos and Floratos, 1998), which are exact algorithms (resulting solutions are proven to be optimal given the selected search criteria). Another class brings together methods that look for a set of occurrences, each one being at a Hamming distance of $k$ or at most $k$ of an external (a priori unknown) consensus word, which could not appear in the sequence set (a hypothetical common ancestor). The main methods are PatternBranching (Price et al., 2003), Winnower (Pevzner and Sze, 2000), Multiprofiler (Keich and Pevzner, 2002), and Smile (Marsan and Sagot, 2000), which is an exact algorithm that allows in addition the detection of several consensus words linked by variable spacers. Other methods like OligoAnalysis (van Helden et al., 1998) and YMF (Sinha and Tompa, 2000), are based on an exhaustive enumeration in the search space, to find over-represented words. Due to the computational expenses, they are limited to small words in DNA only. The class of methods that this paper focuses on, looks...
for a set of conserved occurrences, from the point of view of information content. Considering that this problem has been shown to be NP-hard (Akuwiti et al., 2000), all these methods use heuristics. Apart from CONSENSUS (Hertz et al., 1990; Hertz and Stormo, 1999), which builds a solution using greedy choices, other programs sample the search space, optimizing an objective function. The well-known methods are the Gibbs Sampler (Lawrence et al., 1993; Neuwald et al., 1995), AlignACE (Hughes et al., 2000) and MEME (Bailey, 1993), and both objective functions have already been shown, our paper focuses only to the background frequencies. A frequency matrix P is first derived from the ULMA as follows: for \( k = 1, \ldots, K \) and \( j = 0, \ldots, W-1 \):

\[
F_{k,j} = \frac{1}{N} \sum_{i=1}^{N} J(k, S_i, p_i) \quad \text{with} \quad J(k, a) = \begin{cases} 
1, & a = \sigma_k \\
0, & \text{else}
\end{cases}
\]

The fitness value \( I(P) \) of vector \( P \) (a ULMA) is given by the relative entropy of \( F \):

\[
I(P) = \sum_{i=1}^{K} \sum_{j=1}^{W} F_{i,j} \log \left( \frac{F_{i,j}}{F^0} \right)
\]

with \( F^0 \) corresponding to the background distribution estimated from the whole set. \( F^0 \) is estimated at the beginning and is kept unchanged during the rest of the process. The relative entropy is directly related to the log-likelihood ratio (Bailey, 1993), and both objective functions have already been used to describe the level of conservation inside a ULMA (Lawrence et al., 1993; Hertz et al., 1990; Lawrence and Reilly, 1990). Since the biological meaning of this function has already been shown, our paper focuses only on its optimization aspects.

2.3. Projection operators

The exploration process takes advantage of both word and ULMA representations, mapping them together by two projection operators. Words are used as hypothetical common ancestors to build ULMAs. In a similar way, consensus words are deduced from ULMAs. The first operator projects points from \( M \) onto \( \mathcal{P} \) (MtoP) (Fig. 1). The MtoP operator requires an alignment of the word \( M \) with all the sequences of \( S \). For \( i = 1, \ldots, N \) and \( j = 1, \ldots, T-W+1 \):

\[
p_i = \arg\max_{j} \sum_{k=1}^{W} J(S_i, j+k-1, m_k)
\]

Then, \( p_i \) corresponds to the position on \( S_i \) where the match count is maximal (minimal Hamming distance). This procedure often leads to a position vector with several ambiguous
It is likely that several positions on the same sequence will maximize the match count, leading to a position vector having some ambiguous dimensions. A greedy algorithm clears these ambiguities up. Non-ambiguous positions present a much greater similarity that can be reached from $M$ by a projection. In this case, we use a greedy algorithm to clear up ambiguities. Non-ambiguous dimensions are first selected to build a core. Remaining dimensions are then added one by one, each time choosing the one that maximizes the information content of the growing ULMA. This is carried out by choosing the most likely word according to the frequency matrix $F$ (Eq. (1)). More formally, for $i = 1, \ldots, K$ and $j = 1, \ldots, W$:

$$m_j = \sigma_k \text{ with } k = \arg\max_j (F_{i,k}) \quad (4)$$

If several $k$ are possible for a given $j$, a single one is chosen at random.

Note that the correspondence established between the two search spaces is not symmetrical. A word projected to $P$ and projected back to $M$ will not necessarily be the same as the initial one. This is also true for a ULMA.

### 3. Sampling

MoDEL combines two exploration strategies. The first one performs a global search in $P$ by sampling the $Q$ search space with an evolutionary algorithm. It is described in Sections 3.1 and 3.2. The second one is dedicated to a local search, optimizing the resulting $Q$ points with two hill-climbing processes, and is described in Section 3.3.

#### 3.1. The evolutionary algorithm

Evolutionary algorithms (Goldberg, 1989, 2002) are general optimization methods. Their principle roughly consists of simulating the evolution of a population made up of potential solutions (initially randomly chosen) with rules inspired from the theory of evolution. Several points of the search space are simultaneously sampled in order to (1) create a competition between them and (2) take advantage of the building block concept (Goldberg, 1989, 2002) to build new solutions, by assembling relevant sub-parts of selected solutions. A set of search space points that are simultaneously sampled is called a population of solutions (we used a population size of 200). The population of solutions is iteratively modified by stochastic operators. We can distinguish two main steps during an iteration. (1) The evaluation step: a fitness value is calculated for each potential solution of the current population (the parents). This value corresponds to the chosen objective function. (2) The breeding: in order to move in the search space, new solutions (the children) are created from the parents through the genetic operators (presented in Section 3.2). The convergence is achieved by a selection function, which ensures that the best solutions
are more likely to produce descendants. This selection is stochastic and is a function of the fitness value. The higher the fitness value, the more the corresponding parent is likely to be chosen. We used the tournament selection (Blickle and Thiele, 1995), which works by choosing the best parent out of \( n \) randomly chosen ones in the population (\( n = 2 \) in our case). We chose this selection scheme because solutions are only selected according to their rank and thus diversity in the population is maintained. Selection is done with replacement, which means that some solutions could be selected more than once, while others could be eliminated. See Algorithm 2 for a more formal view of the whole process.

**Algorithm 2** (Evolutionary algorithm).

initialize population(0)

\[ i = 0 \]

while \( i \leq \text{max}_\text{iteration} \) {
    evaluate population(\( i \))
    until population(\( i + 1 \)) is full
    \[ A = \text{select a solution from population(} i \text{)} \]
    \[ B = \text{select a solution from population(} i \text{)} \]
    apply genetic operators on \( A \) and \( B \) include \( A \) and \( B \) in population(\( i + 1 \))
    end until \( i = i + 1 \)
end while

3.2. Genetic operators

Genetic operators are used to build new solutions from parents and therefore to move in the search space in order to localize better solutions. They are functions of search space points (words in our case). Genetic operators are stochastically applied to the selected parents. Each operator is associated with an occurrence probability which is empirically determined. The global setting of these probabilities must balance the exploration (large modifications of the solutions) and the exploitation (small modifications, ascending progression). If the convergence is too strong, the algorithm is doomed to become stuck in a local maximum. On the other hand, a too strong exploration would maximize a similarity score. This score is calculated as follows: given \( M^A = m_1^A, \ldots, m_k^A \) and \( M^B = m_1^B, \ldots, m_k^B \) two words of length \( W \), there will be \( 2^W - 1 \) possible layouts (in case of a classical crossover, \( m_1^B \) is set in front of \( m_1^A \)). The score of a particular layout will be given by the match count between the two overlapping sub-words, divided by the total length of the alignment. Exchanges are then performed on the overlapping sub-words of the maximum layout, in the same way as for the point-to-point crossover.

Unary operators are the mutation and the slide. The mutation operator (MO) simply reinitializes some positions in the word. The slide operator shifts the symbols to the left or to the right and inserts a random symbol in place of the ejected one. Here, the symbols are shifted to the left and a ‘G’ is inserted.

Fig. 2. The crossovers and the slide operators. (A) One point crossover simply exchanges two suffixes determined by a randomly chosen position in the words. (B) Point-to-point crossover exchanges independently each position with a given probability. (C) Slide crossover tries to maximize the likelihood that exchanged positions correspond to homologous information. The two words are aligned in order to find the layout that maximizes a score. Symbols are then exchanged independently with some probability. (D) The slide operator shifts the symbols to the left or to the right and inserts a random symbol in place of the ejected one. Here, the symbols are shifted to the left and a ‘G’ is inserted.

symbols. We use three different crossover schemes. (1) One point crossover (OPC): the two parent-words are aligned face to face. A position in the alignment is randomly chosen, and all symbols that come after this position are swapped. (2) Point-to-point crossover (PPC): words are aligned face to face, but each position is independently swapped with some probability. (3) Slide crossover (SC): this problem-specific crossover has been designed to produce more structured moves in the search space than OPC and PPC. Two words could match the same occurrences in the sequence set, except for a shift of one or several symbols. In such a case, performing a PPC or a PPC will not be constructive because by aligning each word exactly face to face, some “non-homologous” information will be exchanged. To minimize this effect, the two words are not necessarily placed face to face before the exchange. Shifts are performed in order to find a layout that maximizes a similarity score. This score is calculated as follows: given \( M^A = m_1^A, \ldots, m_k^A \) and \( M^B = m_1^B, \ldots, m_k^B \), two words of length \( W \), there will be \( 2^W - 1 \) possible layouts (in case of a classical crossover, \( m_1^B \) is set in front of \( m_1^A \)). The score of a particular layout will be given by the match count between the two overlapping sub-words, divided by the total length of the alignment. Exchanges are then performed on the overlapping sub-words of the maximum layout, in the same way as for the point-to-point crossover.

Unary operators are the mutation and the slide. The mutation operator (MO) simply reinitializes some positions in the word. The slide operator (SO) shifts the symbols one position to the left or to the right. The ejected symbol is lost and a new random one is inserted.

3.3. Local search in \( P \)

Exploring solely with the evolutionary algorithm is not sufficient: the global maximum might be located in an area of \( P \) that is not covered by \( Q \). A local search in \( P \) is neces-
sary to locate possible higher scoring ULMAs. After each iteration of the evolutionary algorithm, about 5% of the newly created solutions are chosen by tournament selection to be optimized with a local search in $P$. This is performed by coupling together two hill-climbing optimizations: a ‘sequence-neighborhood’ optimization and the phase shift correction (Lawrence et al., 1993). Sequence-neighborhood optimization works as follows: given a position vector $P = p_1, p_2, \ldots, p_N$, only one dimension $p_i$ is modified at a time, keeping the others fixed. This dimension is sampled on all its possible values (corresponding to the $T - W + 1$ sub-words of $S_i$), and is updated to the one that maximizes the overall information content $I(P)$. All dimensions are modified one after the other, either in a predefined order, or by random selection without replacement. The whole process is iterated until complete convergence, which usually takes less than ten iterations. Informed readers certainly noticed that moves performed by the sequence-neighborhood operator are very similar to those performed by the Gibbs Site Sampler (Lawrence et al., 1993). The difference is that the Gibbs Site Sampler uses a stochastic hill-climbing process on random points of $P$. This is performed by associating a probability to each possible move in the neighborhood to carry out a biased choice. The reason for the use of a stochastic operator is to avoid local optima. In the case of MoDEL, moves are deterministic in the sense that the best ones are always chosen. Search diversity is ensured by the evolutionary algorithm that produces relevant points of $Q$, and the hill-climbing operator is only used to search in a limited area around a given point of $Q$. The sequence-neighborhood operator is coupled together with the phase shift correction, that simply consists of shifting all $P_i$ together, several positions to the left and to the right to ensure that the ULMA is not in a shifted sub-optimal position.

These hill-climbing optimizations are not done on the whole population of solutions, but on a small fraction of selected solutions (about 5%), for two reasons: the first one is that the hill climbing process is time consuming (optimizing 5% of the solution produced by the evolutionary algorithm, requires about 50% of the total CPU-time). The second reason is that experience shows that different seeds may often converge toward the same solution (belonging to the same basin of attraction (Jones 1995)), thus using this optimization on the whole population does not significantly increase the performance.

The two remaining genetic operators cited in Section 3.2 follow a different concept than those described previously. As others work from $M$ to $M$ (produce a word from a word), they work from $M$ to $M$ via $P$. They use the hill-climbing optimizations presented above to produce a new word. The first one (SNtoM) uses the sequence-neighborhood process. A word is projected to $P$, then the resulting ULMA is optimized through the hill climbing process and projected back to $M$. The second one (PSnoM) follows the same principle, but with the phase shift correction. Both are deterministic and produce a very guided move in $M$.

The two remaining genetic operators cited in Section 3.2 follow a different concept than those described previously. As others work from $M$ to $M$ (produce a word from a word), they work from $M$ to $M$ via $P$. They use the hill-climbing optimizations presented above to produce a new word. The first one (SNtoM) uses the sequence-neighborhood process. A word is projected to $P$, then the resulting ULMA is optimized through the hill climbing process and projected back to $M$. The second one (PSnoM) follows the same principle, but with the phase shift correction. Both are deterministic and produce a very guided move in $M$.
4.1. HTH motif discovery

We performed comparisons of different exploration strategies, using a dataset of HTH domain-containing proteins. We would like to emphasize that these comparisons are not designed to show the accuracy of the relative entropy to recognize a biological motif, but rather to compare how different optimization techniques are able to localize the highest point in the search space, when the same CPU-time is allowed. Thus, these comparisons only involve programs that use the same constraints and equivalent objective functions.

4.1.1. The HTH dataset

The HTH structural motif is found in a large variety of DNA-binding proteins spread over all phyla. Approximately 23 major evolutionary lineages (clades) are known (Rosinski and Aitchley, 1999). Inside each clade, proteins are highly similar. However, there is near-maximal divergence across clades. This structural motif consists of two α-helices separated by a short turn. Its conformation allows proteins to specifically bind DNA. The second helix recognizes and binds to a specific sequence along the major groove of DNA, while the first helix stabilizes the complex. DNA binding specificity is rarely, if ever, conserved.

HTH occurrences are thus weakly similar, and quite difficult to recover. We would like to emphasize that these comparisons are not concerned with the histone H4 protein (H4), which has an average length of 217 amino acids (minimum = 61, maximum = 613). The highest-scoring ULMA we found (Fig. 4) has a relative entropy of 32.71 (we fixed a width equal to the highest-scoring ULMA we found (H4), which has a relative entropy of 32.71 (we fixed a width of 20 amino acids). We will denote this ULMA as the reference. According to the Swiss-Prot database annotations (Bouckaert et al., 2003) and the evolutionary study (Rosinski and Aitchley, 1999), all but one HTH occurrences are included in the reference ULMA. The only exception concerns the histone H4 protein (H4), for which no annotation is available in the Swiss-Prot database, and for which another occurrence is provided by the evolutionary study. We can however observe that the occurrence provided by the evolutionary study has a much lower score according to the relative entropy. Moreover, a comparison with the 3D structure of homologous proteins in the PDB database (Bernstein et al., 2000) shows that the occurrence included in the reference ULMA indeed corresponds to a HTH structure. However, our aim was not to perform an evolutionary analysis of this exception and it remains an open question.

It is worth mentioning that these HTH occurrences are not conserved enough to be recovered by methods like SMILE or Winnower, which look for a set of occurrences, each one being at a Hamming distance of k (or at most k) of an external, a priori unknown word. Classical multiple alignment methods like CLUSTALW or T-COFFEE (based on pairwise comparisons), also fail to align most of the HTH domains, because this ULMA makes sense only when all (or almost all) occurrences are considered simultaneously.

4.1.2. Recovering the reference ULMA

We first compared the ability of several algorithms to recover the reference ULMA. Our comparison involved, in addition to MoDEL, three widely known programs: the Gibbs Site Sampler1 (Lawrence et al., 1993), MEME Bailey and Elkan (1994) and CONSENSUS (Hertz and Stormo, 1999). All these programs optimize the same objective function or equivalent,2 and can be parameterized with the same constraints (fixed width ULMA, exactly one occurrence per sequence). The results showed that both CONSENSUS and MEME failed to recover the reference ULMA (despite its higher value), becoming stuck on a local optimum. Table 1 summarizes the scores reached by the four programs. Out of a concern to increase the performances of these programs, we carefully tested a number of exploration parameters, keeping the one that performed best. For the Gibbs Site Sampler, changing its convergence limit with the (~L100) option (optimize another random seed, if nothing better has been found since the last 100 iterations), significantly increased its accuracy on this dataset. The CONSENSUS program, in addition to the default settings, was run with the (~q 30000) option. This program builds the solution from partial solutions using greedy choice and this option


2 The Gibbs Site Sampler optimizes a slightly different objective function, using non-fixed background frequencies, but we ensured that it has no consequence in recovering the HTH domains.
Table 1

<table>
<thead>
<tr>
<th>RE</th>
<th>Sim</th>
<th>tCalc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MzDEL</td>
<td>32.71</td>
<td>1.00</td>
</tr>
<tr>
<td>Site Sampler*</td>
<td>32.71</td>
<td>1.00</td>
</tr>
<tr>
<td>CONSENSUS</td>
<td>31.96</td>
<td>0.01</td>
</tr>
<tr>
<td>CONSENSUS*</td>
<td>32.42</td>
<td>0.65</td>
</tr>
<tr>
<td>MEME</td>
<td>28.38</td>
<td>0.41</td>
</tr>
</tbody>
</table>

The asterisk (*) denotes programs that were run with different parameters than the default ones. The relative entropy score is indicated in the RE column. The similarity rate between the resulting ULMA and the reference one is indicated in the Sim column. tCalc is the CPU-time in seconds (Athlon, 1.6GHz). The sole methods that were able to reach the highest-scoring ULMA (32.71), are MzDEL and the Gibbs Site Sampler. Both CONSENSUS and MEME fail and always converge to a sub-optimal ULMA (due to their deterministic nature).

increases the number of partial solutions that are considered for the greedy choice (note that we tried higher value until running out of memory (~2Go), without getting a higher score). The MEME program was run with default parameters only because it was not improved by changing them. A similarity value between the resulting ULMA and the reference one is also indicated. It is calculated as the percentage of identical aligned symbols-pairs, between the resulting and the reference ULMA. Note that the ULMA recovered by CONSENSUS with default parameters is very different from the reference one, despite a high relative entropy. This very low similarity value cannot be interpreted as a poor performance, but rather as the presence of a lured local optimum. Because of their stochastic nature, the CPU-time of MzDEL and the Site Sampler correspond to the minimum search time required to find the reference ULMA, with a probability of at least 0.95 (see Section 4.1.3).

4.1.3. The MzDEL optimization over simpler ones

One may ask if using a complicated exploration strategy really gives an advantage over simple repeated optimizations (multi-start hill-climbing) on different random seeds, when the same CPU-time is allowed. To illustrate the relevance of the global search (evolutionary algorithm) as well as the MtoP projection operator, we compared the exploration abilities of MzDEL versus three simpler strategies. The first one consists simply of local hill-climbing optimizations of random points of $P$. The second one works in the same way but with random points of $Q$ (by converting random words to ULMAs using the MtoP operator). The last one is the Site Sampler, which can be roughly seen as a stochastic hill-climbing from random points of $P$. Fig. 5 shows the success rate (y-axis) plotted against the CPU-time (x-axis). The success rate is estimated over 100 independent runs. A success means that the reference ULMA has been recovered. Results show that the evolutionary algorithm significantly improves the exploration process, compared to simple hill-climbing optimizations from random seeds. The Site Sampler was run with the same convergence parameter as in the previous section (~L100). Keeping its default parameter (~L10) led to a lower performance than the exploration of random $P$ points. We therefore show that, even if the
evolutionary algorithm spends about 50% of the CPU-time of MoDEL, it gives a significant advantage in the exploration process, by choosing relevant points of Q, to be locally optimized.

4.2. Artificial dataset

This section gives another comparison between MoDEL and the Gibbs Site Sampler. Previous results suggested that under the one occurrence per sequence constraint, the Gibbs Site Sampler is superior to MEME and CONSENSUS, which has been also confirmed in Akutsu et al. (2000).

Using artificial datasets is not a trivial task, because to our knowledge, there is no way to generate a sequence set that contains a ULMA of known information content which is just above the noise level (required for the problem to be difficult). We chose to do these comparisons on sets of random sequences of cardinal k = 4 (DNA), each symbol having the same probability. It is obvious that on such sequences, any possible ULMA will not be statistically significant, but our aim is only to compare the exploration strategies of the two methods and their abilities to localize a point of the search space that maximizes a given objective function (information content in our case). Moreover, the ability to optimize a ULMA on random sequences could be very useful to estimate the null distribution of scores, with the goal of computing the p-value of a particular biological ULMA (Liu et al., 2001). The underlying idea in using random sequences as datasets is that sequences containing a strong signal will be easy for most methods: lower the signal and the problem becomes more difficult. Lowering the signal such that it reaches the noise level (becomes imperceptible) would be equivalent to using random sequences with no signal. The main disadvantage in using random sequences is that the global maximum is unknown, but comparison of exploration strategies could be performed in a relative way. One may ask if these search spaces are suitable for a neighborhood exploration process, that is, if there is enough local monotone properties that can be exploited to guide the search. We notice that both methods give much higher results than a random exploration. We can conclude that such search spaces have locally monotone properties that make their exploration by local heuristics pertinent.

We generated three series of data (A, B and C). Each series is made up of 500 sets of sequences of length 1000. Sets of the A series are made up of 20 sequences, those of the B series of 25 sequences, and those of the C series of 30 sequences. In order to obtain comparable results, both methods must optimize the same objective function. Both are information content related, but due to a little difference (non-fixed background), we implemented Gibbs Site Sampler’s one (maximum likelihood) to MoDEL. In this way, the results are directly comparable. Exploration settings were adjusted in a way that each method has the same time available and identical constraints (width of the ULMA and a distribution of one occurrence per sequence). MoDEL did 900 iterations on each sequence set. Gibbs Site Sampler settings were adjusted as follows: (−d) fixed width ULMA, (−L∞1000) optimize another random seed, if nothing better has been found since the last 1000 iterations, (−m8000) at most 8000 iterations (as in Section 4.1.2, these settings were adjusted in order to optimize its results). Both methods were run on the A, B, and C sequence set series. Table 2 gives, for each series, the frequency to which each method performed better than the other. Even if the score on a given sequence set is not the same for both methods, the resulting ULMA may differ slightly. To measure differences between solutions found by both methods on the same sequence set, we used the performance coefficient (PC) (Pevzner and Sze, 2000), which is a similarity measure between two ULMAs. Let K1 be the set of positions in the first ULMA and K2 be the set of positions in the second ULMA. PC is given by |K1∩K2|/|K1∪K2|. On the A series, even if MoDEL found better results in most of the cases, the resulting ULMAs differ only slightly (∼ 70% of the detected occurrences are the same for the two methods). However, MoDEL shows a significant improvement on the B and C series. Increasing the number of sequences in the set exponentially increases the search space size and makes the problem more difficult.

4.3. Discussion and results of the planted motif challenge

We discuss here the suitability of the so-called challenge problem (Pevzner and Sze, 2000), to benchmark information content based optimization methods. This challenge consists in discovering a motif from artificial DNA sequences generated according to the FM model (fixed number of mutation). Let M be a fixed (unknown) word of length l. Create N new words M1, M2, . . . , Mk that correspond to corrupted images of M, with k substitutions at random positions. These images are then hidden in a set of N random sequences of length T, exactly one per sequence. Given the sequences, recover the planted motif M (which is equivalent to recover the ULMA formed by its occurrences M1, M2, . . . , Mk in the dataset). Parameterization of the challenge problem is N = 20, T = 600, l = 15, and k = 4 (a (20, 600, 15, 4) signal).

The performance of a given algorithm is evaluated with the performance coefficient described in Section 4.2 by mea-
suring the similarity between the planted ULMA and the discovered one. A number of specific heuristic algorithms perform well on this problem. Such methods are Pattern-Branching (Price et al., 2003), Winnower (Pevzner and Sze, 2000), Multiprofiler (Keich and Pevzner, 2002), Projection (Buhler and Tompa, 2002) and SMILE (Marsan and Sagot, 2000; Marsan, 2002), which is an exact algorithm based on a suffix-tree representation. However in Price et al. (2003), Pevzner and Sze (2000), Keich and Pevzner (2002), Buhler and Tompa (2002), poor performances are reported for non-specific algorithms such as the Gibbs Site Sampler, MEME and CONSENSUS, which are methods that look for high information content ULMAs. It is important to mention some aspects concerning non-specific methods when applied to this challenge problem. First, these specific and non-specific methods are not equal in terms of constraints (the challenge itself does not specify anything about allowed constraints). While both type of methods use the length of the signal and its distribution of one occurrence per sequence, specific methods use, in addition, the number of substitutions $k$ in the planted occurrences. Dis-considering this constraint makes the problem more difficult in the sense that the signal will be less perceptible (or imper-ceptible). Table 3 clearly shows that for MoDEL and the Gibbs Site Sampler (as well for MEME and CONSENSUS by extension), a (20, 600, 15, 4)-signal is weaker than other ULMAs that appeared by chance. The planted motif is at the noise level, and therefore cannot be recovered using an information content based objective function and discarding the constraint of the number of substitutions $k$. This means that non-specific methods will miss the planted occurrences because their objective function will guide convergence on a superior ULMA that appears by chance. In other words, such results are strongly biased in favor of specific methods and such comparisons should be interpreted with caution.

We can however observe that the PC score is about 0.45 for both methods, which means that about 45% of the planted occurrences belong to the high entropy resulting ULMA. Using an information-content based objective function should be helpful to recover a (20, 600, 15, 4)-signal. We used this observation to build a modified version of MoDEL that is able to efficiently recover the planted motif. Those modifications are simply to limit exploration to the $Q$ search space, and to use the constraint of the number of substitutions $k$ in the stop condition. All explored words are checked until one that can match all sequences with 4 errors is found, which means that the planted occurrences has been recovered. By limiting the maximum exploration time to 2 min, MoDEL is able to recover the planted motif in 99.9% of the cases (averaged on 1000 independent instances), with an average CPU-time of $17 \times$ per run (Pentium IV 1.5GHz).

### 5. Conclusion

We propose an optimization strategy for the ULMA problem using relative entropy as an objective function. The exploration strategy uses two different representations: external consensus words and ULMAs. Our approach is applicable on both DNA and amino acid sequences, but is currently limited to fixed-width ULMAs made up of one occurrence per sequence in the dataset. Results show that our method is more likely to localize higher points in the solution space than the Site Sampler, MEME or CONSENSUS, under the same conditions (same CPU-time and constraints).

### Acknowledgements

We would like to thank Frederique Lisacek, Andrea Auchincloss and Stephanie Kappus, for useful discussions as well as a carefully reading of the manuscript and Emmanuelle Morin who developed the web interface.

### References


Bailey, T.L., 1993: Likelihood vs. information in aligning biopolymer sequences. Useul technical report c93-318, University of California, University of California, San Diego, La Jolla, CA.


### Table 3

<table>
<thead>
<tr>
<th>Method</th>
<th>PC</th>
<th>tCalc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoDEL</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>Site Sampler</td>
<td>0.99</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Runs are done on 500 independent instances of the challenge problem ($N = 20$, $T = 600$, $l = 15$, $k = 4$). In almost all the cases, a ULMA with a better score than the planted one is discovered. It is thus clearly shown that the signal is at the noise level for both MoDEL and the Site Sampler (as well for MEME and CONSENSUS by extension). $P_{c} < \epsilon$, show the frequency for which the two programs found respectively a higher, equal or lower information content ULMA than the planted one. $PC$ is the average CPU time in seconds per run.


