Giulio Pavesi
is a PhD student in Computer Science at the University of Milan-Bicocca. His research interests include bioinformatics and discrete models of complex systems.

Giancarlo Mauri
is full professor of Computer Science at the University of Milan-Bicocca. His research interests are mainly in the area of theoretical computer science, and include: formal languages and automata, computational complexity, computational learning theory, neural networks, cellular automata and models of concurrent/parallel computing, bioinformatics.

Graziano Pesole
is Associate Professor of Molecular Biology at the University of Milan, Italy. His research interests include bioinformatics and molecular evolution.

***Keywords:** pattern discovery, multiple sequence alignment, regulatory elements, non-coding DNA, software*

Graziano Pesole, Department of General Physiology and Biochemistry,
University of Milan,
Via Celoria 26,
20133 Milan,
Italy

Tel: +39 02 5835 4915
Fax: +39 02 5835 4912
E-mail: Graziano.Pesole@unimi.it

Methods for pattern discovery in unaligned biological sequences

Giulio Pavesi, Giancarlo Mauri and Graziano Pesole

Date received (in revised form): 20th August, 2001

Abstract

Pattern discovery in biological sequences is the problem of finding patterns that are over-represented in a set of unaligned DNA or protein sequences of related biological function. Such patterns could correspond to regions of the sequences responsible for the function itself, and could be used later for the functional annotation of newly determined sequences. Despite many studies, this problem can be considered far from being solved. The main difficulty lies in the fact that significant patterns can appear within each sequence with mutations, insertions or deletions of nucleotides or amino acids, without losing their biological function. This paper provides a survey of a number of existing pattern discovery algorithms, focusing both on the methods underlying them and their availability for the scientific community.

INTRODUCTION

In the last few years, the amount of biological data generated by the scientific community has grown exponentially, and a huge number of sequences are now available in several databases. Thus, the focus of many research projects has shifted from the generation of the data to their analysis, that is, the extraction of any kind of biological meaning from the sequences.

Given a set of functionally related sequences, the main aim of pattern-discovery algorithms is to find new and *a priori* unknown patterns that appear in every sequence, or at least in a significant number of sequences, of the set. Such patterns could correspond to the regions of the sequences responsible for their function, and could be also used later for the functional annotation of newly determined sequences.

From a computational point of view, the main difficulty lies in the fact that the same pattern can appear within each sequence in a different and approximate form (ie with mutations, insertions or deletions of nucleotides or amino acids), keeping intact its biological function. The longer the patterns and the more degraded their occurrences, the harder (and slower) it is to find them for pattern-discovery algorithms. A large number of different methods has been so far introduced, but we do not have yet good models or reliable algorithms that guarantee to find all (or most of) the biologically meaningful patterns.

This paper, without the claim of being exhaustive, provides a survey of a number of different approaches to the problem, presenting the main ideas underlying the methods and mentioning, when available, the software tools based on the different algorithms.

Definitions

We first give some definitions, showing how the problem looks from a computational point of view. Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be an alphabet. We denote with Σ^* the set of all the strings (sequences) on Σ of arbitrary length, including the empty string, and with $|\Sigma|$ the cardinality of the alphabet. Given a string $s = s_1 \ldots s_n$, all strings $s_i \ldots s_j$, $1 \leqslant i < j \leqslant n$ are *substrings* of s. Protein sequences are strings over a 20-letter alphabet $\Sigma_P = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$, while nucleotide sequences are strings over 4-letter alphabets $\Sigma_{\mathrm{DNA}} = \{A, C, G, T\}$ and $\Sigma_{\mathrm{RNA}} = \{A, C, G, U\}$. A *pattern* is simply a short sequence over one of these alphabets. Given an alphabet Σ, a sequence $s = s_1 \ldots s_n$ and a pattern $p = p_1 \ldots p_m$ in Σ^*, we say that p occurs *exactly* in s if there exists an index i such that $s_{i+l} = p_l$, for all $l \in \{1, \ldots, m\}$, that is, s contains p as a substring. Then, given a *distance measure* on strings d: $\Sigma^* \times \Sigma^* \rightarrow \mathscr{R}$, we denote with $D(p, s)$ the distance of a pattern p from a sequence s, and define it as:

$$D(p, s) = \min_{i,j} d(p, s_i \ldots s_j)$$

That is, the distance of pattern p the sequence s is the distance from p of the substring $s_i \ldots s_j$ of s *closest* (according to d) to p. For DNA sequences, the most widely used measures are the *Hamming distance* that, given two strings of equal length, counts the number of positions in which they differ, and the *Levinstein distance* that given two strings of arbitrary length is defined as the minimum number of *edit operations* (that is, mutation, insertion or deletion of letters) needed to transform one string into the other. For proteins, the main difference is that different weights can be assigned to mismatches, taking into account similarities between amino acids, or analogously two or more amino acids can be assigned to the same class and considered as 'matching', for example according to PAM[1] or BLOSUM[2] matrices. Then, given a threshold value τ for D, we say that p occurs *approximately* in s if $D(s, p) \leqslant \tau$.

In its simplest version, the pattern-discovery problem can thus be formulated as follows:

> **Pattern-discovery problem**: Given a set of k unaligned biological sequences, a distance measure d, and a threshold value τ for d, find all patterns that occur in at least q sequences out of k within distance τ from the sequence.

From now on, we will refer to patterns satisfying the constraints of the problem with the terms *conserved patterns* or *motifs*. Of course, the problem could be defined in different terms: for example, one might be interested in finding all patterns that appear (approximately) in a 'surprisingly' high number of sequences (without explicitly specifying a minimum number q), with respect to their expected number of occurrences, or in just finding the pattern whose sum of distances from the sequences is the minimum among all possible patterns of a given length.

The methods so far proposed can be roughly grouped, as in Brazma *et al.*[3], into two categories: *pattern-driven* and *sequence-driven* methods. In the former, also known as *enumerative* or *exhaustive* methods, a set of candidate patterns, possibly comprising every pattern on Σ up to a given length, is generated, and the occurrences of each pattern are found in the sequences. Conversely, in the sequence-driven approach (also called *alignment-based*), the algorithm tries to determine the conserved pattern(s) by comparing the sequences and looking for similar regions. In the following sections, we describe some of the most widely known algorithms for each category, indicating, when available, the software packages implementing the algorithms.

PATTERN-DRIVEN METHODS

The simplest solution to the problem is to generate all possible patterns up to a maximum length l on Σ (or of exact length l, if it is known in advance), search separately for the approximate occurrences of each one in the set of sequences, score each pattern according to a given significance measure, and report the highest-scoring patterns. Clearly, this approach guarantees to find all patterns that satisfy the input constraints, that is, that occur in at least q sequences of the set. Moreover, the sequences can be organised in suitable indexing structures such that the time needed by the algorithm to search for a single pattern is linear in the overall length of the sequences. Alas, this method has an evident downside: the number of candidate patterns, and therefore the time required by the algorithm, grows exponentially in their length. In fact, if no assumptions are made on how patterns must appear in the sequences, there are $|\Sigma|^l$ candidate patterns for each length l. Moreover, computing some statistical significance score for each pattern further increases the time required by the algorithm. Methods of this kind were introduced in the 1980s,[4–6] and worked well in practice for short patterns and especially on DNA sequences, since in the latter case the alphabet is smaller. This is the approach followed recently by Tompa,[7] where it is applied to the identification of ribosome binding sites in mRNA. The algorithm enumerates all patterns of length seven, allowing at most one mutation in each occurrence. Patterns are then ranked according to their z-score, which compares the actual number of occurrences of each pattern with its expected value and its standard deviation.

The same method is extended in Sinha and Tompa[8] to transcription factor binding sites that include *gapped* motifs, that is, motifs composed of two conserved parts separated by a random region of variable length. The Coresearch algorithm[9] also starts by enumerating all DNA patterns of length seven, and then tries to expand them by analysing the sequence regions around the positions where they occur.

If no errors are allowed, the search can be limited only to patterns that occur at least once in the sequences, that are $n - l + 1$ for each length l. Moreover, in this case extremely efficient algorithms can be based on *suffix trees*,[10] as in Hui,[11] where the longest substring common to at least q of the input sequences is found in linear time, or in Apostolic *et al.*,[12] where the most significant patterns, under different statistical measures of significance, can be found again in time linear in the input size. The latter algorithm has been implemented in the Verbumculus software package. A detailed introduction to suffix trees and their applications in computational biology can be found in Gusfield.[13] Likewise, only exact occurrences of patterns are considered in van Helden *et al.*[14] for the analysis of hexanucleotide frequencies in the upstream regions of yeast genes. Each pattern is assigned a significance score based on the actual number of occurrences in upstream regions compared to an expected value based on the nucleotide frequencies in all the non-coding regions of the yeast genome. This algorithm, as well as other sequence analysis tools, has been implemented in the Yeast Tools page accessible via WWW interface. Limiting the set of patterns to those that occur at least once exactly ($O(n)$) but also allowing mismatches is the approach followed in Li *et al.*[15] and Gelgand *et al.*[16]. In the latter, the method is applied to the prediction of transcription regulatory sites in Archaea.

Suffix trees seem to be a good choice for pattern-driven methods also when one is working with approximate occurrences, as shown by Sagot[17] for DNA sequences. Values for maximum pattern length and maximum number of mutations allowed have to be given as input to the algorithm. On the other hand, patterns are not required to occur exactly in the sequences. The time complexity of the algorithm is reduced to be exponential in the number of mutations instead of the pattern length. This approach is also extended to gapped motifs by Marsan and Sagot.[18]

Exhaustive enumeration, when working only with mutations, can be also made more efficient by imposing restrictions on the location of mismatches. In other words, some positions of the pattern believed to be directly responsible for its biological function have to be perfectly conserved, while other positions can contain any other amino acid or nucleotide. This is the approach followed by Neuwald and Green,[19], Rigoutsos and Floratos[20] and Califano.[21] In these algorithms, the sequence alphabet Σ can be either Σ_P or Σ_{DNA}, and patterns are defined over the alphabet $\Sigma \cup \{*\}$, where $*$ is a 'don't care' or 'wild-card' symbol. That is, occurrences of patterns must match positions of the pattern containing a letter from Σ, and contain any other symbol in positions corresponding to the wild-card symbol $*$. Matches between amino acids can be also defined according to PAM or BLOSUM matrices: all amino acids that are similar enough according to the matrices can be defined by the user as belonging to the same class, and therefore as 'matching'. Also, occurrences of a pattern must match the first letter, and 'don't care' symbols must be spread along the pattern according to given density constraints. No pattern length has to be given as input beforehand. The main advantage of this approach is that exhaustive enumeration of the patterns can be avoided, that is, long patterns are built by combining shorter ones. Patterns found by the algorithm are finally output ranked according to their statistical significance. Variants of this method have been implemented in the ASSET, Teiresias and SPLASH software packages. A similar approach (patterns containing wild-cards) has been applied in Brazma *et al.*[22] to the analysis of upstream regions of the putative yeast genes. A pattern-driven approach, combined with a refinement based on hidden Markov models,[23] is adopted in the Yebis program.[24] First, all patterns up to a given length are generated and located in the sequences using the WordUP algorithm.[25] With each pattern is associated a score based on the comparison between the actual number of sequences carrying that pattern and its expected number. Then, the most significant ones are divided into groups of similar patterns, and a multiple alignment of the patterns is produced for each group, defining a significance value for each position of the motif. Finally, a hidden Markov model is generated for each group, according to the corresponding alignment. The latter serves as a model for the source generating the motif representing the group. The last two steps are iterated by the algorithm. That is, the set of sequences is scanned again by each constructed model, looking for segments that fit the model parameters, trying to expand the motif regions. The substrings selected are again aligned and a new hidden Markov model is generated. The algorithm stops when no improvements to the models can be made.

The search space of the candidate patterns is instead reduced a priori in the Pratt algorithm.[26] Given k input sequences, the algorithm looks for patterns that occur in at least q sequences, where the latter parameter is set by the user. The idea is that, if a pattern has to occur in at least q sequences, then any subset of $k - q + 1$ sequences will contain one of its occurrences. Thus, the search is executed only for those patterns that occur at least once in the $k - q + 1$ shortest sequences, up to a maximum length set by the user, that are represented with a special data structure called *pattern graph*. The search is executed by a depth-first traversal of the space of the patterns, combined with a branch-and-bound technique to reduce the size of the space further. Pratt can find patterns containing wild-card regions (ie stretches of wildcard symbols) whose length can vary in the different occurrences.

SEQUENCE-DRIVEN METHODS

Most of the general-purpose software packages available for sequence analysis are instead based on sequence-driven algorithms. Unlike pattern-driven methods, it is impossible to guarantee optimal results with methods of this kind. The reason is that the problems to which comparison of multiple sequences can be reduced are NP-hard,[27] that is, can be solved only with algorithms that take time exponential in the input size. Therefore, all sequence-driven methods are based on different heuristics, with the aim of obtaining optimal results while keeping an efficient running time. The advantage is that no upper limit has to be imposed on the length of the patterns. Moreover, these algorithms usually do not need as input a maximum threshold value for the pattern distance from the sequences. The main idea underlying the three algorithms that we describe in this section is to represent conserved patterns with a model, built by combining substrings of the input sequences. The hypothesis is that the set of biological sequences containing a pattern can be seen as generated by two separate sources: one emitting 'background noise' with probability equalling the letter frequencies in the sequences, and another one generating the instances of the real motif, according to two separate probability distributions. The goal of the algorithm is to build a model for the source generating the pattern. Then, for each input sequence, the substring that best fits the model is considered the occurrence of the corresponding pattern. Further details on how models are represented and evaluated will be shown in the section on 'Measures of significance'. However, since considering all the possible models would not be computationally feasible, each algorithm uses a different heuristic.

A *greedy* approach to the problem was introduced in Hertz *et al.*,[28] and implemented in the software package Consensus. Given as input a set of sequences $S_1 \ldots S_k$ the basic version of the algorithm requires as input the length w of the motif to be found, and assumes that the latter occurs once in each sequence. The steps performed by the algorithm can be summed up as follows:

1. All the substrings of S_1 of length w are compared to the substrings of length w of S_2. Each comparison produces a $20 \times w$ (or $4 \times w$) alignment matrix F. Basically, entry $F_{i,j}$ in the matrix is the frequency with which residue (or nucleotide) i occurs in the jth position of the substrings selected. Also, each matrix is scored according to its information content, that is, the more similar two substrings are and the more their letters differ from the background distribution, the higher is the score (see the section on 'Measures of significance' for details on how the information content is computed); the highest scoring matrices, and the corresponding substrings are saved.

2. Each substring of length w of sequence S_3 is paired with the matrices saved at step 1, generating a new set of three-sequence alignment matrices; each one is scored as in the previous step, and the highest scoring ones are saved.

3. Step 2 is repeated for each sequence of the set, until the alignment matrices contain one substring for each sequence.

The algorithm is greedy in that the best partial alignments are saved at each step in the hope that they will eventually lead to the optimal one corresponding to the actual occurrences of the most conserved pattern. Obviously, the more conserved the latter is, the more likely is the algorithm to find it. Otherwise, the risk is to store in the first steps matrices corresponding to random (but similar enough) patterns, and to discard the one corresponding to occurrences of the real motif. Further improvements to the algorithm include the possibility of finding patterns that do not occur or appear more than once in each sequence, and avoid explicitly requiring the length parameter to the user (WConsensus).[29] However, the user still needs to vary a bias value that directly influences the length of the pattern.

Perhaps the most widely used and known pattern discovery method is the Gibbs sampling approach introduced by Lawrence *et al.*[30] As in the previous example, the basic version of the algorithm assumes that a pattern appears with mutations in each sequence of a set of k protein or DNA sequences, and needs as input the length w of the pattern. The algorithm iterates many times the following steps:

1. A substring of length w is chosen in

each of the k sequences (at the beginning, with uniform probability).

2. One of the k sequences is chosen at random: let S be this sequence.

3. A $20 \times w$ (or $4 \times w$) frequency matrix F is created from the substrings taken from the other $k - 1$ sequences. As for Consensus, entry $F_{i,j}$ in the matrix is the frequency with which residue i occurs in the jth position in all the $k - 1$ substrings.

4. For each position i in S, let P_i the substring of length w of S starting at i. For each P_i a likelihood value p_i is computed, representing how well substring P_i fits the model induced by the matrix F with respect to the background amino acid or nucleotide distribution.

5. A new probability value, proportional to p_i, is assigned to each position i of S. Thus, starting positions of patterns that fit well in the model determined by F are more likely to be chosen at the next cycle.

6. Go to 1: now the probability values for the substrings of sequence S to be picked are those computed at the previous step.

This algorithm is also known as the *site sampler*. The higher is the number of occurrences of a pattern with respect to the background amino acid or nucleotide distribution, the more likely is the algorithm to find it. Thus, the set of input sequences needs to be quite large (at least 15 or more sequences) for weakly conserved patterns to reach statistical significance. The algorithm is fast, usually taking $O(k)$ time, where k is the number of input sequences. However, given its probabilistic nature, it often has to be run different times, and the final results can be obtained by comparing the outputs of each run. Additions to the basic algorithm have been made,[31] where a method to determine automatically the length of the conserved pattern was introduced, allowing at the same time gaps (wildcard positions) within the pattern itself. Also, multiple occurrences of the pattern within the same sequence were allowed, or, conversely, the pattern did not have to occur in every sequence (algorithm known as *motif sampler*). Variants of the Gibbs sampling technique applied to DNA sequences are described in Rocke and Tompa,[32] Huges *et al.*[33] and Workman and Stormo.[34] In the first, the algorithm is extended to deal with gapped motifs. AlignACE[33] is a program where the basic Gibbs sampling algorithm is fine tuned in order to work on DNA sequences, including for example both strands of each input sequence and introducing a different sampling technique. In the ANN-Spec algorithm,[34] a Gibbs sampling method is combined with an artificial neural network that replaces the frequency matrix.

The MEME (Multiple Expectation maximisation for Motif Elicitation) algorithm[35] builds a different model starting from each pattern of length w occurring in the sequences, and tries to improve it using an expectation maximisation algorithm (EM), that is, it looks for the parameter's values that maximise the likelihood of the data to be generated by the sources (background and motif) by using a local search technique. On the other hand, Gibbs Sampler can be seen as a stochastic variant of this method, thus less likely to end on local optima. After the occurrences of some motif have been determined by the algorithm, they are removed from the sequences and the algorithm is restarted. Thus, MEME can detect multiple motifs within the same set of sequences within a single run.

APPLICATION TRIAL

Usually, all pattern-discovery algorithms are presented together with a real case study, where some known DNA or protein motif is successfully identified by the program in a set of sequences. In some cases, the merits of a new algorithm are shown by comparing it to other similar methods, and also the time needed to perform the task is reported. Moreover, pattern-driven algorithms are usually fine-tuned according to the type of motif or signals the authors are looking for.

A very interesting experiment was performed by Pevzner and Sze,[36] where the three methods described in the previous section were tested on the same problem (also called *Planted (l,d) motif problem* by Buhler and Tompa[37]). Rather than using a real biological benchmark, the authors generated a set of 20 random DNA sequences with uniform nucleotide distribution, and the same pattern (l nucleotides long) was implanted in each sequence with a fixed number (d) of mutations occurring at random positions. The performance of the algorithms was measured according to the length of the sequences, of the pattern, and the number of mutations. The results (some are shown in Tables 1 and 2) were similar for all the three methods, with a slightly better performance obtained by the Gibbs Sampler, which seems to suffer mostly from the reduced number of sequences. In particular, the main factor affecting the performance of the programs seems to be how much the conserved pattern 'stands out' against background noise. For example, as shown in Table 2, a motif of length 15 with four mutations is found on short sequences, but lost in sequences longer than 300–400 nucleotides.

More recently, the same benchmark problem has been addressed also by the Winnower,[36] SP-Star,[36] Projection[37] and Weeder[38] algorithms. All tests reported in the respective papers show how these programs basically outperform the three sequence-driven algorithms described in the previous section on the planted motif problem. In particular, the last two algorithms allow the user to estimate *a priori* the probability of finding a given motif according to its length and the maximum number of mutations allowed for its occurrences: the higher is the probability, the slower is the program. Thus, the user can choose an optimal trade-off between time and accuracy. Both methods can guarantee to find (in acceptable time) with high probability patterns of arbitrary length, as long as their number of occurrences is higher than random patterns.

In any case, deciding *a priori* which program or tool can be considered the most suitable for a given problem is sometimes as hard as solving the problem itself. The evaluation is first of all strongly dependent on the type of sequences, since some algorithms have been explicitly designed to work on nucleotide sequences rather than proteins, or vice versa. For example, pattern-driven methods can detect longer motifs in DNA sequences, since the alphabet size is smaller. Then, one should consider what kind of pattern one is looking for. Short (or well-conserved) patterns can be detected efficiently by virtually all methods, either sequence- or pattern-driven. Pattern-driven methods are perhaps best suited to perform a once-for-all analysis of the data, and to build a list of the most significant motifs, even on genome-scale datasets. For longer or less conserved motifs, even if searched for in

Table 1: Maximum number of mismatches tolerated (average performance coefficient greater than 50 per cent) by Consensus, Gibbs Sampler and MEME, according to the length of the conserved pattern, in a sample of 20 random DNA sequences (with uniform nucleotide distribution) of length 600. Consensus fails for (13,3) signals, MEME fails to find (11,2), (13,3), and (20,6) signals

Pattern length	8	9	10	11	12	13	14	15	16	17	18	19	20
Mismatches	1	1	1	2	2	3	3	3	4	4	5	5	6

Source: Pevzner[36]

Table 2: Comparison of the performance of the programs on 20 random DNA sequences, with uniform nucleotide composition, with implanted patterns of length 15 with 4 random mutations in each occurrence, according to the sequence length

Sequence Length	100	200	300	400	500	600	700	800	900
Consensus	0.92	0.94	0.53	0.31	0.29	0.07	0.15	0.09	0.01
Gibbs Sampler	0.93	0.96	0.51	0.46	0.29	0.12	0.09	0.34	0.00
MEME	0.91	0.78	0.59	0.37	0.17	0.10	0.02	0.03	0.00

Source: Pevzner[36]

shorter sequences, additional considerations should be taken into account. The planted motif test made no assumption on the position of mutations in the occurrences of patterns. But, in most of the cases, biologically significant patterns tend to have some well-conserved positions (ie where mutations do not occur), while other are much less conserved or even completely random. This is, for example, the assumption made by the authors, among others, of SPLASH and Teiresias. Thus, the real biological problem is somehow different from the pure 'algorithmic' one. This fact is the reason why sequence-driven methods have provided successful results on biological instances that, if formalised as a planted motif problem, are perhaps harder than the ones presented by Pevzner,[36] such as the CRP binding sites (one of the most variable of all known DNA binding sites) used by the authors of Consensus[28] and MEME[35] as a benchmark example. Thus, sequence-driven methods might be the best choice when little is known about the hidden motifs (ie no assumption can be made on the maximum number of errors in each occurrence). Even if optimal results cannot be guaranteed, they output some motif (by looking for similar regions), and a conserved pattern of biological interest could correspond to sub-optimal results (that is, not to the highest-scoring matrix).

MEASURES OF SIGNIFICANCE

In both pattern- and sequence-driven methods, an algorithm might output more than one conserved motif. Or, as in the case of probabilistic methods, the algorithm might give different results at each run. As we briefly mentioned before, nearly all the programs and algorithms available associated with the motifs found a *score* based on some statistical measure of significance, expressing how 'surprising' it is to find them with respect to the frequency of each residue in the sequences, hoping at the same time to reflect as much as possible their biological relevance.

When only exact occurrences are considered, a thorough analysis of the statistical properties of biological sequences and of different probability distributions for patterns occurring in them can be found in Reinert *et al.*[39]. In this case, as shown by Apostolico *et al.*[12] and implemented in the Verbumculus program, finding the most significant patterns, under different measures, takes linear time in the length of the sequences. The scores used basically compare the number of occurrences of a pattern with an expected value, like the chi-square score or the z-score used also for motifs containing mismatches.[7] Expected values are in turn computed with different techniques, as the Poisson approximation[12] and the binomial/normal approximation.[7] The z-score is also used in SPLASH[21] for patterns containing wild-card symbols. In this case, the number of occurrences of a pattern is compared with the expected value and standard deviation of the number of patterns with the same features (ie length, number of conserved positions) in a random set of sequences with the same letter frequencies.

In the sequence-driven methods we mentioned, instead, the approach followed is slightly different. A motif is described by a $|\Sigma| \times m$ matrix F, where m is the length of the motif and Σ is the sequence alphabet (either Σ_P or Σ_{DNA}). Entry $f_{i,j}$ in the matrix is the frequency with which letter Σ_i of the alphabet is found in position j in the occurrences of the motif, and $\Sigma_i f_{i,j} = 1$. All the methods described in the previous section basically try to build the most significant frequency matrix. The assumption is that the most interesting motifs correspond to matrices whose letter frequencies most differ from the *a priori* probability of the letters in the sequences ($\Pr(\sigma_i)$), that is, the frequency of the letters in the 'background noise'. The more the matrix differs, the highest is its score. One way to compute this score is to use the *information content* (or *relative entropy*) of the matrix measured in *bits*:

$$I(F) = \sum_{j=1}^{m} \sum_{i=1}^{|\Sigma|} f_{i,j} \log \frac{f_{i,j}}{\Pr(\sigma_i)} \qquad (1)$$

That is, the information content is given by the sum of the information content of each position of the motif (inner sum). When $f_{i,j} = \Pr(\sigma_i)$, that is, the frequency in the motif equals the background distribution for all letters, the value is at a minimum and equals zero. The information content is maximised when the least expected letter occurs exclusively in the motif, ie $f_{l,j} = 1$ and $\Pr(\sigma_l) \leqslant \Pr(\sigma_i)$ for all i. Schneider *et al.*[40] show that the frequency of binding sites for a DNA binding protein approximately equals $e^{-I(p)}$. When the frequency of a letter is zero, the value $0 \log 0$ can be either evaluated as zero, or avoided by introducing a pseudo-count term:[30]

$$f_{i,j} = \frac{c_{i,j} + b_i}{N - 1 + \Sigma_k b_k} \qquad (2)$$

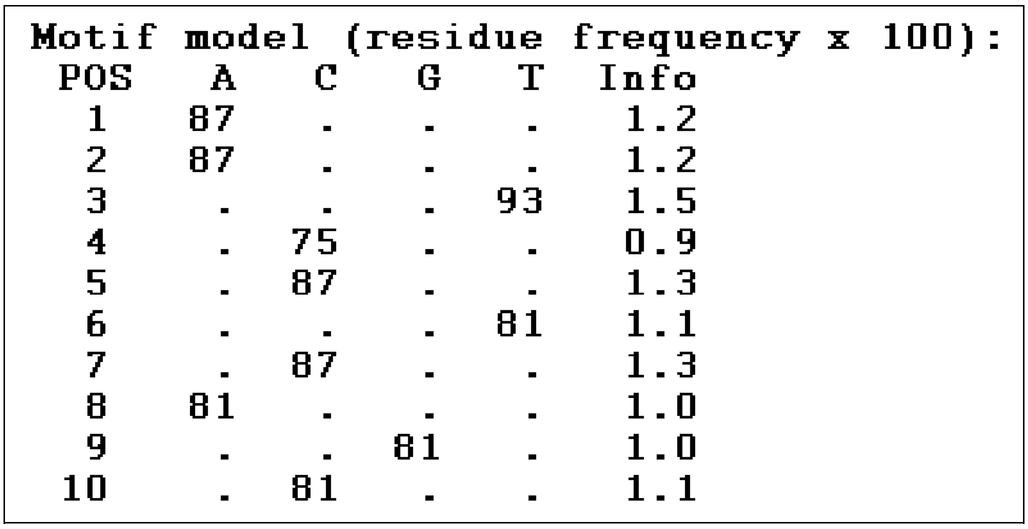
where $c_{i,j}$ is the number of occurrences of letter σ_i in position j of the motif, b_i is the pseudo-count value associated with σ_i, usuallydefinedas$a{\cdot}\Pr(\sigma_i)$with$a > 0$, N is the number of input sequences (assuming that the pattern appears once in every sequence) and $\Sigma_k b_k$ is the sum of the pseudo-counts associated with all the letters of Σ.

In order to include in the measure the number of sequences the pattern appears in (let k be this number), the information content can be multiplied by $-k$, yielding the *log-likelihood ratio* or *maximum a posteriori (MAP)* score:

$$L(F) = \sum_{j=1}^{m} \sum_{i=1}^{|\Sigma|} n_{i,j} \log \frac{\Pr(\sigma_i)}{f_{i,j}} \qquad (3)$$

Notice that the frequency of occurrence of each letter is replaced by $n_{i,j}$, that is, the actual number of times letter σ_l appears in position j in the occurrences of the pattern. A more detailed discussion on the different ways to describe a motif with a weight matrix (and related sequence driven methods) can be found in Stormo.[41]

Another way of defining the significance of a given motif, starting from its information content, is to compute its

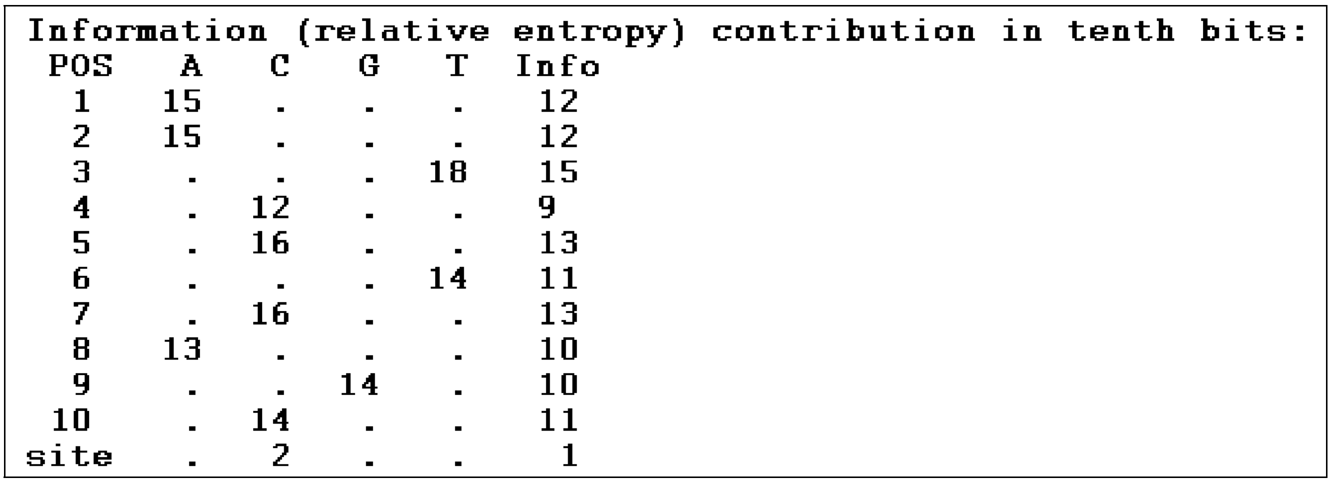


```
Motif model (residue frequency x 100):
 POS   A   C   G   T  Info
  1   87   .   .   .   1.2
  2   87   .   .   .   1.2
  3    .   .   .  93   1.5
  4    .  75   .   .   0.9
  5    .  87   .   .   1.3
  6    .   .   .  81   1.1
  7    .  87   .   .   1.3
  8   81   .   .   .   1.0
  9    .   .  81   .   1.0
 10    .  81   .   .   1.1
```

Figure 1: Residue frequency matrix (adjusted with the pseudo-count term) output by Gibbs Sampler for motif AATCCTCAGC occurring in 15 random (with uniform nucleotide distribution) DNA sequences with at most one mutation. For each position, it reports the frequency (×100) of occurrence of each nucleotide. Statistically negligible values are not reported. The rightmost column reports the information content of each position, computed according to Equation 1

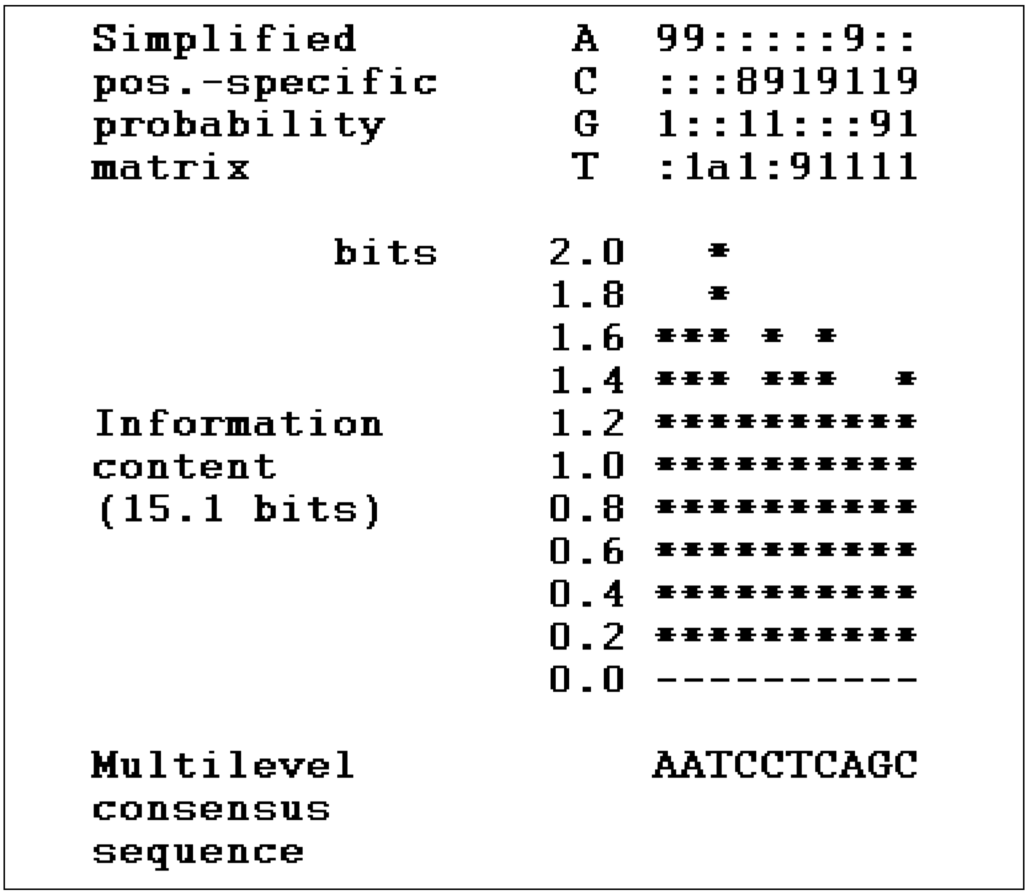
p-value, intuitively; the probability of observing another motif with equal or greater information content, given the length of the motif itself and the number of input sequences, as in Consensus and MEME. A method used to estimate the *p*-value of a motif is described, for example, in Hertz and Stormo.[29]

Some examples of what the output of Gibbs Sampler, MEME and Consensus looks like and can be interpreted are shown in Figures 1–5. All three programs



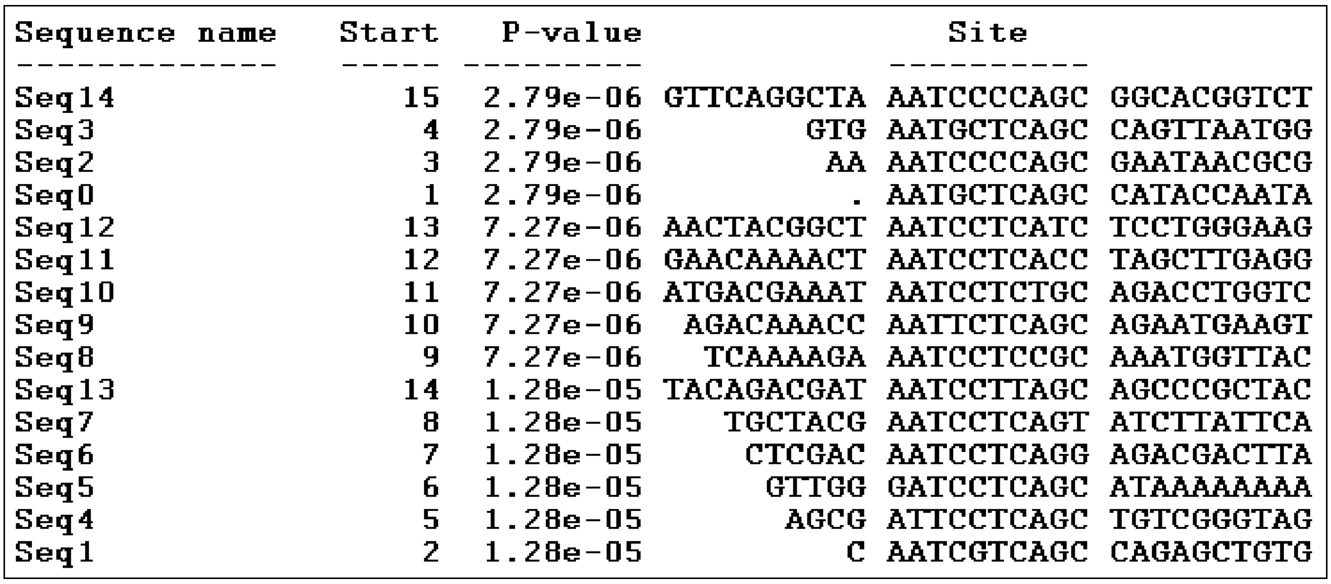
```
Information (relative entropy) contribution in tenth bits:
 POS    A    C    G    T   Info
   1   15    .    .    .   12
   2   15    .    .    .   12
   3    .    .    .   18   15
   4    .   12    .    .   9
   5    .   16    .    .   13
   6    .    .    .   14   11
   7    .   16    .    .   13
   8   13    .    .    .   10
   9    .    .   14    .   10
  10    .   14    .    .   11
site    .    2    .    .    1
```

Figure 2: Information content (Equation 1) of motif AATCCTCAGC as reported by Gibbs Sampler. The log-likelihood ratio is computed according to Equation 3. The bottom row of the matrix reports the information content of each column (that is, the sum of the content corresponding to each nucleotide instead of each position)



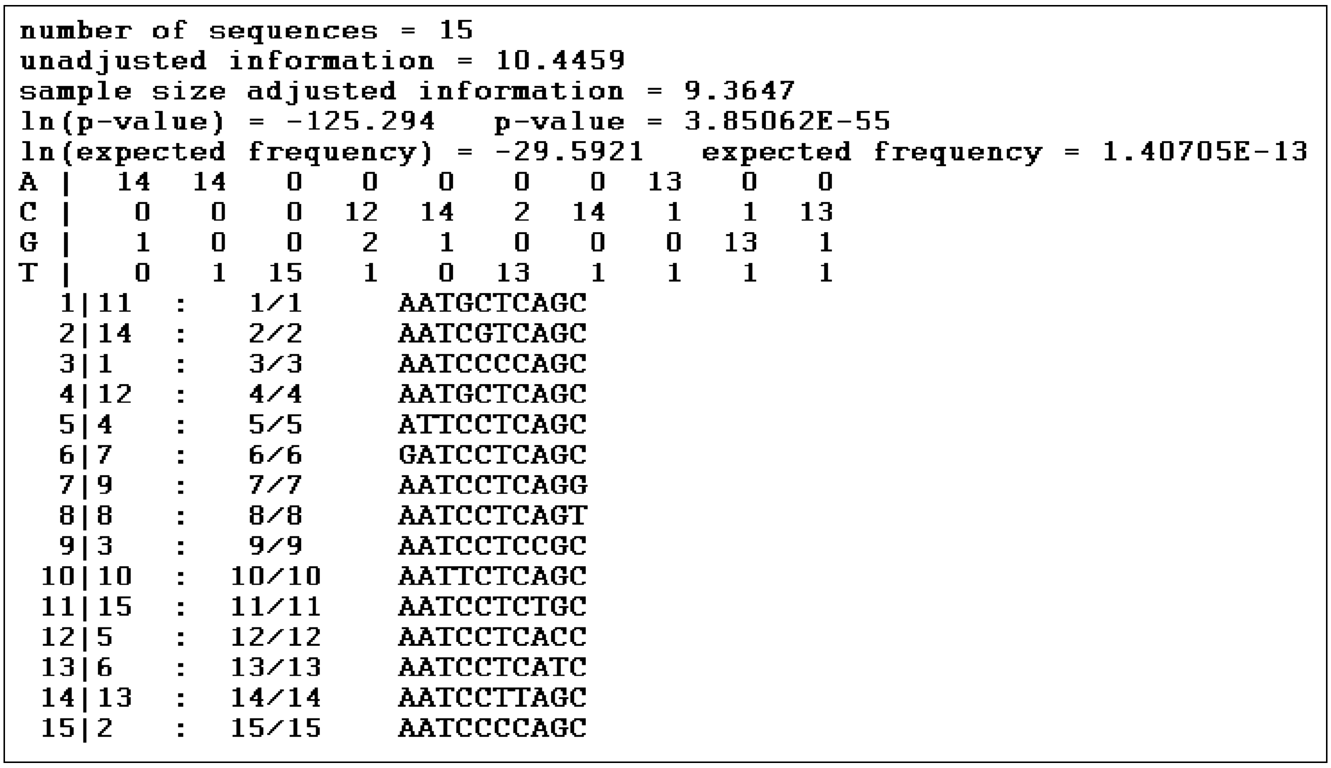
```
Simplified          A  99::::9::
pos.-specific       C  :::8919119
probability         G  1::11:::91
matrix              T  :1a1:91111

          bits    2.0    *
                  1.8    *
                  1.6 *** * *
                  1.4 *** ***  *
Information       1.2 **********
content           1.0 **********
(15.1 bits)       0.8 **********
                  0.6 **********
                  0.4 **********
                  0.2 **********
                  0.0 ----------

Multilevel            AATCCTCAGC
consensus
sequence
```

Figure 3: Frequency matrix (without pseudo-count term) and information content as output by MEME



```
Sequence name    Start   P-value                Site
-------------    ----- ---------             ----------
Seq14               15  2.79e-06 GTTCAGGCTA AATCCCCAGC GGCACGGTCT
Seq3                 4  2.79e-06        GTG AATGCTCAGC CAGTTAATGG
Seq2                 3  2.79e-06         AA AATCCCCAGC GAATAACGCG
Seq0                 1  2.79e-06          . AATGCTCAGC CATACCAATA
Seq12               13  7.27e-06 AACTACGGCT AATCCTCATC TCCTGGGAAG
Seq11               12  7.27e-06 GAACAAAACT AATCCTCACC TAGCTTGAGG
Seq10               11  7.27e-06 ATGACGAAAT AATCCTCTGC AGACCTGGTC
Seq9                10  7.27e-06  AGACAAACC AATTCTCAGC AGAATGAAGT
Seq8                 9  7.27e-06   TCAAAAGA AATCCTCCGC AAATGGTTAC
Seq13               14  1.28e-05 TACAGACGAT AATCCTTAGC AGCCCGCTAC
Seq7                 8  1.28e-05    TGCTACG AATCCTCAGT ATCTTATTCA
Seq6                 7  1.28e-05     CTCGAC AATCCTCAGG AGACGACTTA
Seq5                 6  1.28e-05      GTTGG GATCCTCAGC ATAAAAAAAA
Seq4                 5  1.28e-05       AGCG ATTCCTCAGC TGTCGGGTAG
Seq1                 2  1.28e-05          C AATCGTCAGC CAGAGCTGTG
```

Figure 4: Alignment output by MEME, indicating the starting position of the motif in each sequence. The *p*-value of a site is computed from the match score of the site with the position specific scoring matrix for the motif. The *p*-value gives the probability of a random string (generated with the background letter frequencies) having the same match score or higher



```
number of sequences = 15
unadjusted information = 10.4459
sample size adjusted information = 9.3647
ln(p-value) = -125.294   p-value = 3.85062E-55
ln(expected frequency) = -29.5921   expected frequency = 1.40705E-13
A |  14  14   0   0   0   0   0  13   0   0
C |   0   0   0  12  14   2  14   1   1  13
G |   1   0   0   2   1   0   0   0  13   1
T |   0   1  15   1   0  13   1   1   1   1
   1|11   :   1/1     AATGCTCAGC
   2|14   :   2/2     AATCGTCAGC
   3|1    :   3/3     AATCCCCAGC
   4|12   :   4/4     AATGCTCAGC
   5|4    :   5/5     ATTCCTCAGC
   6|7    :   6/6     GATCCTCAGC
   7|9    :   7/7     AATCCTCAGG
   8|8    :   8/8     AATCCTCAGT
   9|3    :   9/9     AATCCTCCGC
  10|10   :  10/10    AATTCTCAGC
  11|15   :  11/11    AATCCTCTGC
  12|5    :  12/12    AATCCTCACC
  13|6    :  13/13    AATCCTCATC
  14|13   :  14/14    AATCCTTAGC
  15|2    :  15/15    AATCCCCAGC
```

Figure 5: Output of Consensus, with the frequency matrix (number of occurrences of each nucleotide in each position), *p*-value, and position of the motif within the sequences

output the frequency matrix (even if in different ways and styles), and the score associated with it, either based on information content (Gibbs Sampler), or *p*-value, and an alignment of the sequences centred on the motifs. In the example shown in the figures, we generated a sample of 15 DNA sequences with uniform nucleotide probability, and planted in each sequence motif AATCCTCAGC with at most one mutation in each occurrence. The motif was found by all the programs.

CONCLUSIONS

In the last few years, the growing amount of biological data needing to be analysed and the excitement deriving from the completion of whole genome projects has brought under the spotlight the problem of developing powerful and efficient tools for the analysis and the comparison of biological sequences.

This paper has provided, without the claim of being exhaustive, a survey of different algorithms and methods for the automatic discovery of patterns in biological sequences. For each one, we described the underlying approach, and tried to show which kind of patterns it can actually detect. When available, pointers to the software packages based on the algorithms are given (see Table 3). In any case, the ultimate tool, able to detect all the biologically interesting and meaningful patterns has yet to be developed. We believe that in the next few years the growing interest for this problem, both in biologists and computer scientists, and most of all their cooperation will yield significant results and powerful tools able to provide substantial help to researchers in this field.

Table 3: Pattern-discovery programs downloadable (DL) or accessible via World Wide Web interface free of charge. The availability of downloadable programs may depend on hardware platform or operating system

Program name	Web site	DL	WWW
AlignACE[33]	http://atlas.med.harvard.edu/	+	+
(W)Consensus[28]	ftp://beagle.colorado.edu/pub/consensus	+	−
	http://bioweb.pasteur.fr/seqanal/interfaces/consensus-simple.html	−	+
CORE SEARCH[9]	http://www.gsf.de/biodv/coresearch.html	+	−
Gibbs Sampler[31]	ftp://ncbi.nlm.nih.gov/pub/neuwald	+	−
	http://bayesweb.wadsworth.org/gibbs/gibbs.html	−	+
MEME[35]	http://meme.sdsc.edu/meme/website	+	+
Pratt[26]	http://www.ii.uib.no/~inge/Pratt.html	+	+
SPLASH[21]	http://www.research.ibm.com/splash	+	−
Teiresias[20]	http://www.research.ibm.com/bioinformatics	+	+
Verbumculus[12]	http://www.cs.purdue.edu/~stelo/Verbumculus	+	+
WordUP[25]	http://bighost.area.ba.cnr.it/BIG/WordUP	−	+
Yeast Tools[14]	http://embnet.cifn.unam.mx/rsa-tools/	−	+
Yebis[24]	http://www-scc.jst.go.jp/YEBIS	−	+

ACKNOWLEDGEMENTS
This work has been supported by the Italian Ministero dell'Università e Ricerca Scientifica e Tecnologica, projects 'Bioinformatics and Genomics' (PRIN99), 'Studio di geni di interesse biomedico e agroalimentare' (no. 488/92), and 'Biotecnologie' (no. 95/95-MURST 95%).

References

1. Dayhoff, M. O., Schwartz, R. M. and Orcutt, B. C. (1978), 'A model of evolutionary change in proteins', *Atlas Protein Seq. Struct.*, Vol. 5, pp. 345–352.
2. Heinkoff, S. and Heinkoff, J. G. (1992), 'Amino acid substitution matrices from protein blocks', *Proc. Natl Acad. Sci. USA*, Vol. 89, pp. 10915–10919.
3. Brazma, A., Jonassen, I., Eidhammer, I. and Gilbert, D. (1998), 'Approaches to the automatic discovery of patterns in biosequences', *J. Comput. Biol.*, Vol. 5, pp. 279–305.
4. Queen, C., Wegman, M. N. and Korn, L. J. (1982), 'Improvements to a program for DNA analysis: a procedure to find homologies among many sequences', *Nucleic Acids Res.*, Vol. 10, pp. 449–456.
5. Staden, R. (1989), 'Methods for discovering novel motifs in nucleic acid sequences', *Computer Appl. Biosci.*, Vol. 5(4), pp. 293–298.
6. Wateman, M. S., Arratia, R. and Galas, M. (1982), 'Pattern recognition in several sequences: Consensus and alignment', *Bull. Math. Biol.*, Vol. 46(4), pp. 515–527.
7. Tompa, M. (1999), 'An exact method for finding short motifs in sequences, with application to the ribosome binding site problem', in Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB 1999), AAAI Press, Meulo Park, CA, pp. 262–271.
8. Sinha, S. and Tompa, M. (2000), 'A statistical method for finding transcription factor binding sites', in Procedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), AAAI Press, Meulo Park, CA, pp. 344–354.
9. Wolferstetter, F., Frech, K., Herrmann, G. and Werner, T. (1996), 'Identification of functional elements in unaligned nucleic acids sequences by a novel tuple search algorithm', *Computer Appl. Biosci.*, Vol. 12(1), pp. 71–80.
10. McCreight, E. M. (1976), 'A space-

economical suffix tree construction algorithm', *J. ACM*, Vol. 23, pp. 262–272.

11. Hui, L. C. K. (1992), 'Color set size problem with application to string matching', in Proceedings of the 3rd Symposium on Combinatorial Pattern Matching, Springer Verlag, LNCS 644, pp. 227–240.
12. Apostolico, A., Bock, M. E., Lonardi, S. and Xu, X. (2000), 'Efficient detection of unusual words', *J. Comput. Biol*, Vol. 7, pp. 71–94.
13. Gusfield, D. (1997), 'Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology', Cambridge University Press, New York.
14 . van Helden, J., Andrè, B. and Collado-Vides, J. (1998), 'Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies', *J. Molec. Biol.*, Vol. 281, pp. 827–842.
15. Li, M., Ma, B. and Wang, L. (1999), 'Finding similar regions in many strings'. In 'Proceedings of the 31st Annual ACM Symposium on Theory of Computing', pp. 473–482.
16. Gelfand, M., Koonin, E. and Mironov, A. (2000), 'Prediction of transcription regulatory sites in archaea by a comparative genomic approach', *Nucleic Acids Res.*, Vol. 28(3), pp. 695–705.
17. Sagot, M. F. (1998), 'Spelling approximate repeated or common motifs using a suffix tree, in Proceedings of Latin 98, Springer Verlag LNCS 1380, pp. 111–127.
18. Marsan, M. and Sagot, M. F. (2000), 'Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory site consensus identification', *J. Comput. Biol.*, Vol. 7, pp. 345–360.
19. Neuwald, A. F. and Green, P. (1994), 'Detecting patterns in protein sequences', *J. Mol. Biol.*, Vol. 239, pp. 698–712.
20. Rigoutsos, I. and Floratos, A. (1998), 'Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm', *Bioinformatics*, Vol. 14, pp. 55–67.
21. Califano, A. (2000), 'SPLASH: structural pattern localization analysis by sequential histograms', *Bioinformatics*, Vol. 16, pp. 341–357.
22. Brazma, A., Jonassen, I., Vilo, J. and Ukkonen, E. (1998), 'Predicting gene regulatory elements in silico on a genomic scale', *Genome Res.*, Vol. 8, pp. 1202–1215.
23. Levinson, S. E., Rabiner, L. R. and Sondhi, M. M. (1983), 'An introduction to the application of the theory of probabilistic function of a Markov process to automatic speech recognition', *Bell Syst. Tech. J.*, Vol. 62, pp. 1035–1074.
24. Yada, T., Totoki, Y., Ishikawa, M. *et al.* (1998), 'Automatic extraction of motifs represented in the hidden Markov model from a number of DNA sequences', *Bioinformatics*, Vol. 14, pp. 317–325.
25. Pesole, G., Prunella, N., Liuni, S. *et al.* (1992), 'WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences', *Nucleic Acids Res.*, Vol. 20(11), pp. 2871–2875.
26. Jonassen, I. (1997), 'Efficient discovery of conserved patterns using a pattern graph', *Computer Appl. Biosci.*, Vol. 13, pp. 509–522.
27. Garey, M. and Johnson, D. (1979), 'Computers and Intractability: a Guide to NP Completeness'. W.H. Freeman, New York.
28. Hertz, G., Hartzell, G. and Stormo, G. (1990), 'Identification of consensus patterns in unaligned DNA sequences known to be functionally related', *Computer Appl. Biosci.*, Vol. 6, pp. 81–92.
29. Hertz, G. and Stormo, G. (1999), 'Identifying DNA and protein patterns with statistically significant alignment of multiple sequences', *Bioinformatics*, Vol. 15, pp. 563–577.
30. Lawrence, C., Altschul, S., Boguski, M. *et al.* (1993), 'Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment', *Science*, Vol. 262, pp. 208–214.
31. Neuwald, A., Liu, J. and Lawrence, C. (1995), 'Gibbs motif sampling: detection of outer membrane repeats', *Protein Sci.*, Vol. 4, pp. 1618–1632.
32. Rocke, E. and Tompa, M. (1998), 'An algorithm for finding novel gapped motifs in DNA sequences', in Proceedings of the 2nd Annual International Conference on Computational Molecular Biology (RECOMB 98), pp. 228–233.
33. Huges, J. D., Estep, P., Tavazoie, S. and Church, G. M. (2000), 'Computational identification of Cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*', *J. Molec Biol.*, Vol. 296, pp. 1205–1214.
34. Workman, C. T. and Stormo, G. D. (2000), 'Ann-Spec: a method for discovering transcription factor binding sites with improved specificity', *Pacific Symposium on Biocomputing (PSB) 2000*, pp. 464–475.
35. Bailey, T. and Elkan, C. (1995), 'Unsupervised learning of multiple motifs in biopolymers using expectation maximisation', *Machine Learning*, Vol. 21, pp. 51–80.
36. Pevzner, P. A. and Sze, S. (2000), 'Combinatorial approaches to finding subtle signals in DNA sequences' in Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), AAAI Press, Meulo Park, CA, pp. 269–278.

37. Buhler, J. and Tompa, M. (2001), 'Finding motifs using random projections', 'Proc. of the 5th Annual International Conference on Computational Molecular Biology (RECOMB 2001)', pp. 69–76.

38. Pavesi, G., Mauri, G. and Pesole, G. (2001), 'An algorithm for finding signals of unknown length in DNA sequences', in Proceedings of the 9th International Conference on Intelligent Systems for Molecular Biology (ISMB 2001), *Bioinformatics* , 17 suppl. 1, AAAI Press, Meulo Park, CA, pp. S207–S214.

39. Reinert, G., Scabath, S. and Waterman, M. S. (2000), 'Probablistic and statistical properties of words', *J. Comp. Biol.*, Vol. 7, pp. 1–48.

40. Schneider, T. D., Stormo, G. D., Gold, L. and Ehrenfeucht, A. (1986), 'Information content of binding sites on nucleotide sequences', *J. Mol. Biol.*, Vol. 188, pp. 415–431.

41. Stormo, G. (2000), 'DNA binding sites: representation and discovery', *Bioinformatics*, Vol. 16(1), pp. 16–23.