Finding Patterns in Biological Sequences

Broňa Brejová, Chrysanne DiMarco, Tomáš Vinař
Department of Computer Science
University of Waterloo

Sandra Romero Hidalgo
Department of Statistics
University of Waterloo

Gina Holguin, Cheryl Patten
Department of Biology
University of Waterloo

Technical report CS-2000-22
University of Waterloo
December 2000

Abstract

In this report we provide an overview of known techniques for discovery of patterns of biological sequences (DNA and proteins). We also provide biological motivation, and methods of biological verification of such patterns. Finally we list publicly available tools and databases for pattern discovery. On-line supplement is available through http://monod.uwaterloo.ca/supplements/00motif.

Contents

1 Introduction

The goal of our project was to study known techniques for discovering patterns in biological sequences. Patterns we want to find usually correspond to functionally or structurally important elements in proteins or DNA sequences. There is an assumption that these important regions are better conserved in evolution and therefore they occur more frequently than expected. Pattern discovery is one of the fundamental problems in bioinformatics. It can be used in multiple sequence alignment, protein structure and function prediction, characterization of protein families, promoter signal detection, and other areas.

Introduction to the problem and biological motivation of motif discovery can be found in Section 2.

Section 3 gives overview of known algorithms. We have concentrated on the problem of discovering previously unknown patterns. From biological point of view it is equally important to have tools for finding known patterns in new sequences, however this is usually not so interesting from algorithmic point of view. Therefore we touch on this issue only lightly in cases when it is not obvious. We also do not try to compare individual methods based on their performance or their ability to find most relevant patterns. The reason is that individual approaches vary widely in the type of pattern they try to find, performance guarantees and so on. Even authors of experimental comparative studies such as [Hudak and Mcclure, 1999] have difficulties to determine which method performed better on a given dataset. It is impossible to do so based only on descriptions of algorithms. Still we have tried to choose such algorithms to our study that seem to contain most interesting ideas.

Patterns found by algorithms may or may not be the patterns which we really want to find. Sections 4, and 5 provide means how to assess quality of the found patterns. In particular, section 4 discusses measures of statistical significance of a pattern and section 5 shows how to verify patterns using biological experiments.

To illustrate importance of pattern finding methods we included a few examples where such software tools led to new biological discoveries (section 6).

Finally, section 7 closes our discussion, and appendices A and B include list of publicly available software tools for motif discovery and databases of motifs.

2 Biological Motivation for Pattern Discovery

Nucleotide and protein sequences contain patterns or motifs that have been preserved through evolution because they are important to the structure or function of the molecule. In proteins, these conserved sequences may be involved in the binding of the protein to its substrate or to another protein, may comprise the active site of an enzyme or may determine the three dimensional structure of the protein. Nucleotide sequences outside of coding regions in general tend to be less conserved among organisms, except where they are important for function, that is, where they are involved in the regulation of gene expression. Discovery of motifs in protein and nucleotide sequences can lead to determination of function and to elucidation of evolutionary relationships among sequences.

2.1 Pattern discovery in proteins

With the accumulation of nucleotide sequences for the entire genomes of many different organisms, comes the need to make sense out of all of the information. Attempts have been made to organize all of the proteins encoded in these genomic sequences into families based on the presence of common signature sequences [Linial et al., 1997, Rigoutsos et al., 1999]. Members of protein families are often characterized by more than one motif (on average each family has 3-4 conserved regions) which increases the certainty that a protein has been assigned to a correct family [Nevill-Manning et al., 1998]. Hierarchical trees of protein clusters often reveal functional and evolutionary relationships among proteins. Starting with a single "seed" sequence, protein families can be characterized in order to find ancient ancestor sequences [Neuwald et al., 1997]. First, proteins related to a query sequence are found by searching the databases for similar sequences. Sequences revealed from this initial screen are then used as query sequences to search for other family members and the process is repeated to exhaustion. All of the sequences are aligned in order to identify conserved regions which are used to generate models that represent ancient conserved regions. The rationale behind this approach is that if protein A is related to protein B, and B is related to C, then A is also related to C. Model refinement parallels divergent evolution in that each subsequent alignment reveals progressively more distant relatives.

By this method, proteins are assigned to a family based on sequence homology as determined primarily by alignment. If an alignment finds homology between a query protein and a particular family of proteins, a phylogenetic relationship between them is automatically assumed [Barker et al., 1996]. There are two problems with this assumption: 1) significant sequence similarities are not always indicative of close evolutionary relations, and 2) despite limited sequence homology, proteins can have structural and mechanistic similarities, and even common ancestry not apparent through alignment. Perhaps structural information should also be considered when attempting to classify proteins that are highly divergent in homology, yet functionally equivalent.

Such an approach has been used to identify motifs in proteins that may be related through convergent evolution. Leucine zipper sequences, involved in protein dimerization, appear in diverse families that lack a common ancestor and thus may be an example of a convergent motif. These regions lack sequence similarity, being comprised of a repeating pattern of a single conserved leucine residue separated by six highly variable amino acids. This pattern repeats on average about four times in a protein. Because of the high variability in the sequence of the motif, pattern discovery is combined with secondary structure prediction; leucine zippers form coiled coil structures that are involved in dimer formation [Bornberg-Bauer et al., 1998]. Further confounding pattern prediction, leucine is sometimes replaced with methionine, valine or isoleucine. This flexibility in motif sequence which reflects flexibility in biological function (proteins with leucine zipper domains can often form different combinations of hetero- and homodimers) must be considered in pattern prediction for proteins.

Identification of patterns that have been conserved through evolution can lead to the association of these sequences with protein function or structure. A first step to function prediction is to look for sequence features that are common to groups of proteins with a specified activity but are absent from proteins without the activity. Savoie et al. [Savoie et al., 1999] used such an approach to develop a recognition rule for sequences that determine whether a peptide will activate a T-cell response. These motifs are essentially antigenic determinants that elicit an immune response and can be used to develop vaccines. The premise of all attempts to assign function to unknown proteins by pattern recognition is that highly conserved sequences have been preserved through evolution because they are important to the function or structure of the protein. While intuitively this seems a valid assumption, it is possible that some conserved sequences may simply correspond to regions with a lower rate of mutation.

Although functional motifs may not be apparent in the protein primary sequence when they consist of single conserved amino acid residues separated by long, variable regions, these conserved residues may come together to form a functional group when the protein is folded into its three dimensional structure [Califano, 2000]. On the other hand, patterns of conserved sequences can often highlight elements that are responsible for structural similarity between proteins and can be used to predict the three dimensional structure of a protein.

Because some amino acids share similar characteristics such as size, charge or hydrophobicity, substitutions are often permitted in protein motifs even where residues are important to structure or function. Nevill-Manning et al. [Nevill-Manning et al., 1998] describe a method for discovering conserved motifs that characterize a protein family but are somewhat flexible in the amino acids allowed in particular positions within the motif. Groups of amino acids occurring at each position in a motif with significant frequency were identified and used to characterize subsets of motifs that are biologically relevant. While a motif should be sensitive enough to allow identification of new family members with a minimum of false negatives, there may be a tradeoff in terms of specificity; lower specificity leads to the identification of false positive sequences.

Once biological dictionaries of protein sequence patterns are constructed (see Appendix B for examples of protein motif databases), they can be used to predict the function of newly discovered or unknown proteins, or to screen genomic databases for other proteins with similar function. Chloroplasts are the photosynthetic organelles of plant cells that also perform many other functions such as hormone synthesis. More than 200 proteins are involved in chloroplast activity; some of these are encoded on the chloroplast genome while others are encoded on the nuclear genome of the plant cell. The latter proteins are synthesized in the cytosol and are then transported to the chloroplast. Thus, targeting of proteins to the chloroplast and localization of proteins within the chloroplast are complex and are specified by sequences within the protein. Peltier et al. [Peltier et al., 2000] systematically characterized chloroplast proteins by two dimensional gel

electrophoresis, mass spectrometry, and protein sequencing. Using a combination of de novo motif discovery and detection of known motifs, they identified motifs that specify the function and location of many of the chloroplast proteins.

Protein function can be determined by detection of characteristic motifs even in the absence of homology outside of the motif sequence. RNA genomes found in many viruses such as HIV and Ebola, replicate frequently and rapidly and therefore accumulate errors at a high rate. Thus, genomic sequences among these viruses tend to be highly divergent. Although biological and biochemical data indicate a common ancestry, there is often no statistically significant homology among sequences. To predict the function of a protein from an RNA virus, one could look for conserved motifs [Mcclure and Kowalski, 1999]. Viral genomes are typically small and encode only a few proteins necessary for viral replication. Conserved motifs are therefore likely to specify the function or structure of a protein because if the sequence deviated significantly from the consensus sequence, then the protein would likely not be functional and the virus would be unable to reproduce.

2.2 Pattern discovery in non-coding regions

Similar to patterns in proteins, motifs in non-coding sequences can be used to determine the function of nucleotide sequences on a global level, for example, to find all promoters in a genome, or to determine specific function such as regions involved in tissue-specific regulation of gene expression. Non-coding sequences are generally not well conserved, therefore, the presence of a conserved sequence in the region upstream of a gene usually implies that it is functionally important. Finding all promoters in large genomic sequences necessitates the identification of features that are common to all promoters but are not present in non-promoter sequences. This is a difficult problem, especially in eukaryotic organisms which do not have a single core promoter and are usually associated with multiple regulatory factors. Some approaches include the identification of global signals that interact with RNA polymerase and general transcription factors (e.g., TATA and CAAT boxes, CpG islands), the detection of upstream regions with a high density of transcription factor binding sites (although these are often not clustered), and the identification of sequence characteristics that influence DNA three dimensional structure (regions downstream of the TATA box tend to be highly bendable while regions upstream have low bendability) [Pedersen et al., 1999].

Often the goal is not only to locate promoters, but to understand how genes under the control of these promoters are regulated. This involves identifying specific regulatory sequences in promoter regions, for example, that bind to specific transcription factors in response to a biological signal. Known transcription factor binding motifs can be modeled to find additional binding sites in a genome and thereby identify genes that are regulated in the same manner. Geraghty et al. [Geraghty et al., 1999] were able to identify new genes that were coordinately regulated by the fatty acid oleate in the genome of Saccharomyces cerevisiae. These genes have a common upstream regulatory sequence known as the oleate response element (ORE). The S. cerevisiae genome database was screened for this element, constraining the search to within 500 bases upstream of ORFs greater than 100 codons. Because genes controlled by an ORE were expected to be targeted to the peroxisome (membrane bound organelles involved in, among other things, fatty acid metabolism), the coding sequences downstream of predicted OREs were screened for the presence of a peroxisome motif. The proteins encoded by these genes provided new insights into interesting metabolic mechanisms of the cell. Similarly, Roulet et al. [Roulet et al., 2000] found new sites that bind to members of the CTF/NFI family of eukaryotic transcription factors. These motifs are difficult to detect because they consist of two short motif sequences separated by a spacer of variable length; half sites are also recognized. By synthesizing a series of oligonucleotides with variations on the consensus sequence and determining their binding efficiency to a CTF/NFI transcription factor by gel shift assay, they were able to develop a model that was successfully used to find new binding sites.

Prediction of regulatory protein binding sites can help to infer the function of a gene when homologous genes of known function are not available. Yada et al. [Yada et al., 1997] outline the development of a recognition rule for all of the different sigma factor binding sites (a sigma factor is a component of RNA polymerase involved in promoter recognition) in the Bacillus subtilis genome. They applied this information to the prediction of sigma factors that would bind to the promoters of uncharacterized genes and initiate expression. The function of these unknown genes can then be hypothesized by analogy to the known function of other genes regulated by the same

sigma factor.

When the transcription factor binding motifs are unknown, they can be found by searching for common elements in the upstream regions of genes that are known to be coregulated (such genes are known as regulons) [Mironov et al., 1999, Hughes et al., 2000]. A comparative approach can be used to predict regulatory sequences in different genomes, however, the cognate regulatory factor must be known to be conserved [Gelfand et al., 2000]. Travasoie et al. [Tavazoie et al., 1999] examined life cycle-dependant patterns of gene expression in Saccharomyces cerevisiae by first collecting and analyzing mRNA using microarrays, and grouping the corresponding open reading frames according to the specific point in the yeast's life cycle in which they were expressed. Sequence patterns that are common and specific to each group were then identified in the upstream regions of these genes; these are likely to be involved in developmentally-specific regulation of gene expression. The motifs were used to detect additional sites in the genome with the goal of eventually understanding the regulatory networks within the cell. The advantage of this approach is that it is not influenced by any prior knowledge of genes that might be expressed or the organism that they are expressed in, and is therefore particularly useful to understand regulation in organisms for which very little biology is known. A similar approach could be used to analyze gene expression in different tissues, in response to a given stimulus (for example, an environmental signal), or as a cell transitions from a normal to an abnormal state.

It may be of interest to find motifs in RNA sequences; however, RNA molecules such as tRNA, rRNA and catalytic RNA are usually more conserved in structure than in sequence. The properties of an RNA molecule are often determined by its structure which can take various forms as a consequence of intramolecular basepairing within the single stranded molecule, for example, pseudoknots, hairpin loops, bulges, etc. Thus, aligning RNA solely on the basis of conserved sequences is often misleading. Rather, to look for motifs, an RNA sequence is searched for regions that could potentially basepair to form secondary structure; of course, distance constraints would have to be applied and a minimun number of base pairs would be required [Gorodkin et al., 1997b]. One mechanism of transcription termination in bacteria, the so-called rho independent termination, involves the formation of a secondary structure known as a stem-loop (or hairpin) in mRNA just upstream of the termination site. Intramolecular complementary base pairing results in formation of a stem which is capped by a loop of unpaired bases. Ermolaeva et al. [Ermolaeva et al., 2000] used secondary structure patterns to predict transcription terminators in twelve bacterial genomes. They searched genomic sequences for an mRNA motif characterized by a stable stem sequence (i.e., high in GC content, with only one basepair mismatch in a sequence that would generate a stem of 4-20 nucleotides), followed closely by a short U-rich region, and within a reasonable distance of an open reading frame.

The recognition of regulatory sequences in eukaryotic and prokaryotic DNA sequences has had relatively limited success. Regulation of gene expression is complex. It is common for transcription factors to bind to their DNA target sites in cooperation with many other factors that act synergistically to induce gene expression. In many cases transcription factors can form different combinations of hetero- and homodimers that bind with different specificities; these can often bind in their monomeric form. Other problems that hinder the development of good models for promoter prediction include the relatively low number of characterized promoters and poor understanding of the signals for start and stop of transcription and translation, especially in eukaryotes. In a review by Ficket and Hatzigeorgiou [Fickett and Hatzigeorgiou, 1997], currently available promoter prediction programs were tested and found at best about half of eukaryotic promoters.

For the detection of protein binding sequences in DNA, the spatial distribution of amino acids with respect to a DNA substrate should be examined in addition to the interaction of the protein with a specific DNA sequence. Kono and Sarai [Kono and Sarai, 1999] surmised that the distribution of amino acids around bases found in 130 protein-DNA complexes in the protein data bank could be used to derive empirical interaction potentials, and thus to predict DNA target sites for DNA-binding proteins. A strict sequence correspondence between amino acids and bases was not found, although preferences were evident. However, the interactions between amino acids with a strong base preference and their cognate DNA target could be formed with other amino acids.

Sometimes the initial assumptions used to find motifs are oversimplified. The objective of Kochetov et al. [Kochetov et al., 1999] was to predict properties of mRNA sequences that influence levels of

translation. They compared the mRNA sequences of highly expressed genes with the mRNA sequences of poorly expressed genes to detect sequence features essential for efficient expression. The selection of highly expressed mRNA was based on the assumption that an abundance of polypeptides is a consequence of efficient translation, and conversely, that inefficient expression results in a scarcity of polypeptides. This approach does not take into account other factors that influence polypeptide levels such as RNA stability, promoter activity, etc. It is possible to have a very efficient translation process and still have a short supply of polypeptides. For the cell, more does not necessarily mean better and it is doubtful that efficient translation processes in the cell have been selected through evolution because they result in greater production of polypetides.

2.3 Tandem Repeats

A particularly interesting problem in pattern-finding involves the detection of *tandem repeats*, which are two or more contiguous, approximate copies of a pattern of nucleotides. Tandem duplication occurs as a result of mutational events in which an original segment of DNA, the *pattern*, is converted into a sequence of individual copies. With the progression of time, the individual copies within a tandem repeat may undergo other "uncoordinated" mutations which render the once-identical copies in the original pattern now only *approximate* variations of each other.

The prevalence of tandem repeats is surprisingly high in genomic sequences. [Benson, 1999] notes that "Tandem repeats are presumed to occur frequently in genomic sequences, comprising perhaps 10% or more of the human genome, But, accurate characterization of the properties of tandem repeats has been limited by the inability to easily detect them". As Benson also goes on to say, the detection of tandem repeats has come to assume an increasing importance in genomic research, for both positive and negative reasons. On the negative side, the appearance of specific kinds of tandem repeats has been linked to a number of different diseases, including Huntington's disease, myotonic dystrophy, spinal and bulbar muscular atrophy, and Friedrich's atxia. In each of these cases, individuals with the disease have a huge increase in the number of copies of a trinucleotide pattern, into the hundreds or even thousands. On the positive side, however, it appears that tandem repeats may play a role in gene regulation (interacting with transcription factors, altering the structure of the chromatin, or acting as protein binding sites [Benson, 1999, p.573] and in the development of immune system cells.

3 Algorithms

3.1 Introduction

Algorithmic approaches to pattern discovery exhibit surprising variety. They can be classified according to different more or less orthogonal criteria. In our report we group algorithms together mainly based on the approaches they use. In this introduction we introduce other possible classifications of the algorithms. We concentrate on two issues: how is the biological task formulated in computer science terms, and what kind of patterns are used in the programs. We also introduce notation used throughout this section.

3.1.1 Computer science questions

Problem of pattern discovery appears in different areas of biology. Good examples are protein binding sites (including but not limited to discovery of elements regulating gene expression) and motifs conserved in members of protein families. More detailed discussion of biological aspects of motif finding see Section 2. Now we will discuss how to translate such biological questions to more formal computer science problems.

Classification problems. One class of problems are classification problems. These occur for example in protein families. One of the goals of finding common motifs in protein families is to use these motifs as a classifiers: given an unknown protein we can classify it as a member or non-member of a family, based on the fact whether it contains the motifs characteristic for the family. In this case we may formulate question as a machine learning problem: given a set of sequences belonging to the family (positive examples) and a set of sequences not belonging to the family (negative examples) one may wish to find a function f which for each protein decides whether it belongs to the family or not. In context of motif discovery we are mostly interested in such classes of functions f that involve matching some discovered patterns against the unknown sequence. Note, that negative examples are simply other known proteins taken from protein databases such as SWISS-PROT. Quite often people start only from positive examples and negative examples use

for evaluation of their classifiers. In this case they usually solve the problem of finding suitable significant patterns as described below. A detailed discussion of possible formalizations of pattern finding as classification problem can be found in a review [Brazma et al., 1998].

Finding significant patterns. Motif discovery is not always formulated as a classification problem. For example if we want to find a regulatory element, we might have a set of regions likely to contain this element. However it does not mean, that this element cannot occur in other places in genome or that all of these sequences must contain common regulatory element. Also in a context of protein family motifs we are interested in finding conserved regions that may indicate structurally or functionally important elements, regardless whether they have enough specificity to distinguish between this family and other families. In this context it is more complicated to formulate the question precisely. Usually people define class of patterns they want to find and they are interested in discovering the highest scoring pattern from this class that has enough support. Various approaches differ in a way how the define a support and a score of a pattern.

Support of a pattern usually means the number of sequences in which the pattern occurs. We can require that pattern should occur in all sequences or there is a minimum number of occurrences specified by user. In some cases the number of occurrences is not specified but it is part of a scoring function – longer pattern with fewer occurrences can be sometimes more interesting than shorter pattern with more occurrences. The situation is even more complicated in the case of probabilistic patterns, such as Hidden Markov models. Deterministic patterns either match sequence or not (zero or one), whereas probabilistic models give a probability between 0 and 1. Therefore there are different degrees of "matching". It is necessary to set some threshold on what should be considered a match or to include these matching probabilities to the score of the pattern.

Methods for scoring patterns also differ from paper to paper. Score can describe only the pattern itself (e.g. its length, degree of ambiguity etc.) or it can be based on the occurrences of the pattern (their number, how much these occurrences differ from the pattern). Scoring functions are sometimes based on statistical significance. For example we may ask, what is the probability that the pattern would have so many occurrences if the sequences were generated by random. If this probability is small, the pattern is statistically significant. More detailed discussion of statistical significance of patterns can be found in Section 4.

The goal of an algorithm may be to find the best (i.e. usually the highest scoring patterns), or to find several best scoring patterns, or all patterns with some predefined level of support and score.

Pattern discovery vs. pattern matching. So far we have discussed the problem of pattern discovery, i.e. the algorithm is supposed to discover pattern unknown in advance. However in biology many consensus sequences are known and it is important to have tools that allow to find occurrences of known patterns in new sequences. This problem will be called pattern matching. Program for pattern matching can be quite general, i.e. they get pattern as a part of input, or they can be built to recognize only one particular kind of pattern. In this case authors usually try to fine-tune the parameters of the system to get better sensitivity and specificity of the algorithm. From computer science point of view these programs are not so interesting, however they are very useful for biologists.

3.1.2 Input sequences

The input of pattern discovery programs usually consists of several sequences, expected to contain the pattern. We will denote Σ the alphabet of all possible characters occurring in the sequences. Thus $\Sigma = \{A, C, G, T\}$ for DNA sequences and Σ is a set of all 20 amino acids for protein sequences. Most of the algorithms can be easily adapted to work with any finite alphabet (this is true for algorithms, but not necessarily for their implementations). Thus the pattern finding algorithm can be used also outside bioinformatics, or on other types of biological data.

Some algorithms use not only sequences, but also other information. For example pattern discovery is much easier in aligned sequences. Also we may use information about secondary or tertiary structure, evolutionary relationships between sequences and so on. However most of the time we will concentrate on the discovery from unaligned sequences only.

3.1.3 Types of patterns

Different programs discover patterns of different kind. On the most general level patterns can be divided between deterministic and probabilistic. A deterministic pattern either matches given string or not. On the other hand probabilistic patterns are

usually probabilistic models that give to each sequence probability that this sequence is generated by the model. The higher is this probability, the better is the match between sequence and pattern.

Deterministic patterns. The simplest kind of a pattern is just a sequence of characters from alphabet Σ, such `TATAAAA`, the TATA box consensus sequence. We can also allow more complex patterns, adding some of the following frequently used features.

- **Ambiguous character** is a character corresponding to a subset of Σ. Ambiguous character then matches any character from this set. Such sets are usually denoted by a list of its members enclosed in square brackets e.g. `[LF]` is a set containing L and F. `A-[LF]-G` is a pattern in a notation used in PROSITE database. This patterns matches 3-character subsequences starting with A, ending with G and having either L or F in the middle.

  For nucleotide sequence there is a special letter for each set of nucleotides, where `R=[AG]`, `Y=[CT]`, `W=[AT]`, `S=[GC]`, `B=[CGT]`, `D=[AGT]`, `H=[ACT]`, `V=[ACG]`, `N=[ACGT]`.

- **Wild-card or don't care** is a special kind of ambiguous character that matches any character from Σ. Wild-cards are denoted N in nucleotide sequences, X in protein sequences. Often they are also denoted by dot '`.`'. Sequence of one or several consecutive wild-cards is called **gap** and patterns allowing wild-cards are often called gapped patterns.

- **Flexible gap** is a gap of variable length. In PROSITE database it is denoted by `x(i,j)` where i is the lower bound on the gap length and j is an upper bound. Thus `x(4,6)` matches any gap with length 4, 5, or 6. They also denote a fixed gap of length i as `x(i)` (e.g. `x(3)` = `...`). Finally `*` denotes gap of any length (possibly 0).

Following string is an example of a PROSITE pattern containing all mentioned features: `F-x(5)-G-x(2,4)-G-*-H`. Some programs do not allow all these features, for example they do not allow flexible gaps or they allow any gaps but do not allow ambiguous characters other than a wild-card.

Patterns with mismatches. One can further extend expressive power of deterministic patterns by allowing certain number of mismatches. Most commonly used type of mismatches are substitutions. In this case subsequence S matches pattern P with at most k mismatches, if there is a sequence S' exactly matching S that differs from S in at most k positions.

Sometimes we may also allow insertions or deletions, i.e. the number of mismatches would be an edit distance between the substring S and a closest string matching the pattern P.

Position weight matrices. Even the most complicated deterministic patterns cannot capture some subtle information hidden in a pattern. Assume we have a pattern that contains on the first position C in 40% cases and G in 60% cases. The ambiguous symbol `[CG]` gives the same importance to both nucleotides. It is so important in strong patterns, but it may be important in weak patterns, where we need to use every piece of information to distinguish the pattern from random sequence.

The simplest type of probabilistic pattern is position-weight matrix (PWM). PWMs are also sometimes called position-specific score matrix (PSSM), or a profile (however profiles are often more complicated patterns, allowing gaps). PWM is a simple ungapped pattern specified by a table. This table contains for each pair (position, character), the relative frequency of the character at that position of the pattern (see Figure 1 for an example).

Assume that the pattern (i.e. PWM) has lent k (number of columns of the table). The score of a sequence segment $x_1 \dots x_k$ of length k is

$$\prod_{i=1}^{k} \frac{A[x_i, i]}{f(x_i)}$$

where $A[c, i]$ is an entry of position weight matrix corresponding to position i of the pattern and character c and $f(c)$ is background frequency of character c in all considered sequences. This product represents odd-score that the sequence segment $x_1 \dots x_k$ belongs to the probability distribution represented by the PWM [Dorohonceanu and Nevill-Manning, 2000]. In order to simplify computation of the score we can store log-odd scores $\log A[c, i]/f(c)$ in the table, instead of plain frequencies $A[c, i]$. Then the following formula gives us log-odd score instead of odd score ($A'[c, i]$

PWM with relative frequencies

A	0.26	0.22	0.00	0.00	0.43	1.00	0.11
C	0.17	0.18	0.59	0.00	0.26	0.00	0.35
G	0.09	0.15	0.00	0.00	0.30	0.00	0.00
T	0.48	0.45	0.41	1.00	0.00	0.00	0.54

PWM with log-odd scores (using $f(c) = \frac{1}{4}$)

A	-3.94	-4.18	$-\infty$	$-\infty$	-3.22	-2.00	-5.18
C	-4.56	-4.47	-2.76	$-\infty$	-3.94	$-\infty$	-3.51
G	-5.47	-4.74	$-\infty$	$-\infty$	-3.74	$-\infty$	$-\infty$
T	-3.06	-3.15	-3.29	-2.00	$-\infty$	$-\infty$	-2.89

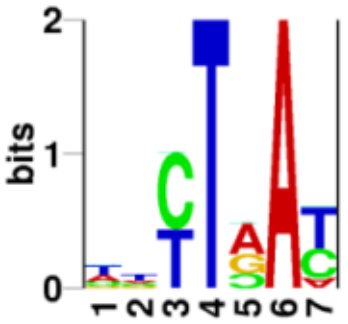

Figure 1: Position weight matrix of vertebrate branch point in form of a table and corresponding visual representation as a sequence logo. The sequence logo was created using on-line software at `http://www.cbs.dtu.dk/gorodkin/appl/slogo.html`

is an entry of the table containing log-odd scores):

$$\sum_{i=1}^{k} A'[x_i, i].$$

Position-weight matrices can be visualized in the form of sequence logos [Schneider and Stephens, 1990] (see Figure 1). Each column of a sequence logo corresponds to one position of the pattern. Relative heights of the characters in one column are proportional to the frequencies $A[c, i]$ at the corresponding position of the pattern. The characters are displayed sorted according to their frequency, with the most frequent character on top. Each column is scaled so that its total height is proportional to the information content of the position, computed as

$$\log_2 |\Sigma| + \sum_c A[c, i] \log_2 A[c, i].$$

Value $\log_2 |\Sigma|$ is added in order to obtain positive values. It depends on the size of alphabet Σ. Sequence logos were further improved by [Gorodkin et al., 1997a] to take background distribution into account, and displaying characters that occur less frequently than expected upside down.

Quick look at a sequence logo reveals most preserved positions, consensus characters at all positions etc. Notice, that the size characters in different columns cannot be directly compared.

Stochastic models. All types of patterns discussed so far are explicit in a sense that the user can easily see important characteristics of the occurrences of a pattern. Sometimes it is advantageous to represent a pattern in a more implicit form, usually as some discrimination rule, which decides whether a given sequence is an occurrence of the modeled pattern or not. Such a discrimination rule can be based on some stochastic model, such as hidden Markov model (HMM), or can employ machine learning methods, for example neural nets, and so on.

It is possible to argue whether such rules constitute a pattern at all, but obviously they can be trained (which corresponds to pattern discovery) and then they can be used for discrimination (which corresponds to pattern matching). Therefore they are applicable in pattern-related tasks such as protein family classification, binding sites discovery etc. In some cases (such as HMMs with simple topology) it is even possible to obtain some information about the pattern modeled, such as relative frequencies of characters at individual conserved positions.

3.2 Exhaustive search

Many computer science problems related to pattern discovery are provably hard, therefore one cannot hope to find fast algorithm which would guarantee to find the best possible solution. Therefore many approaches are based on exhaustive search. Although thus algorithms may run in exponential time in the worst case, programs often use sophisticated pruning techniques that make the search feasible for typical input data.

3.2.1 Enumerating all patterns.

The simplest approach to pattern discovery is to enumerate all possible patterns satisfying constraints given by the user, for each pattern find its occurrences in input sequences and based on this

occurrences assign score or statistical significance to each pattern. Then we may output patterns with highest score or all patterns with scores above some threshold.

For example if we want to find the most significant nucleotide pattern of length 10, allowing at most 2 mismatches, we can enumerate all possible strings of length 10 over the alphabet $\{A, C, G, T\}$ (there are $4^{10} = 1,048,576$ such strings). Each string is a potential pattern. We find all its occurrences with at most 2 mismatches in input sequences and to compute the score of the pattern. Then we report the pattern with the highest score.

This method is however suitable only for short and simple patterns, because the running time grows exponentially with the length of the pattern. The number of possibilities is even larger if we allow patterns containing wild-cards, ambiguous characters, gaps etc. On the other hand the running time grows usually linearly with increasing length of the input sequences. Therefore the enumeration approach may be suitable for finding short patterns in a huge amount of data.

The advantage of this method is that it is guaranteed to find the best pattern. We may easily output arbitrary number of high scoring patterns, we may also choose relatively complicated scoring functions, as far as they can be easily computed based on the pattern and its occurrences. Also we can allow mismatches, even insertions and deletions.

Application of enumerative method. Many protein binding sites in DNA are actually short ungapped motifs, with certain variability. They can be quite well modeled with simple patterns allowing small number of mismatches. Therefore we can apply exhaustive search to find this type of binding sites. Recent examples of this approach can be found in [van Helden et al., 1998, Tompa, 1999]. In both papers authors use straightforward enumeration of all possible patterns and concentrate more on estimating statistical significance of their occurrence. [van Helden et al., 1998] tries to find pattern that appears in several copies in most sequences (GATA box). Therefore they consider patterns consisting of 4-9 nucleotides not allowing mismatches and they try to find such patterns that occur more often than others, taking into account background distribution. [Tompa, 1999] allows mismatches and tries to find statistical significance of the given number of occurrences of the pattern with mismatches.

Enumerating gapped patterns. In some contexts it is more reasonable to search for patterns with gaps. Example of such system is MOTIF [Smith et al., 1990]. MOTIF finds patterns with 3 conserved amino acids, separated by two fixed gaps (for example `A...Q....I`). The gaps can have length $0, 1, \ldots, d$ where d is a parameter specified by user. The number of possible patterns is $20^3 d^2$. MOTIF does not allow any mismatches, however the pattern does not need to occur in all sequences. If the sequences contain a conserved region of more than 3 positions, then there will be many patterns, each containing different subset of conserved positions from this region. Therefore in the next step the algorithm removes patterns occurring close to each other. Then all matches of a particular pattern are aligned and based on this alignment the pattern is extended by finding consensus in the columns of the alignment. Pattern can be also extended to both sides, if possible.

Pruning pattern enumeration. If we want to find longer or more ambiguous patterns, we cannot use straightforward exhaustive search. Assume we want to find a long ungapped pattern occurring possibly with some mismatches in at least K sequences. The we may start from short patterns (for example patterns of length 1) that appear in at most K sequences and extend them until the support does not go below K. In each step we need to extend the pattern in all possible ways and check whether the new pattern still occur in at least K sequences. Once we get a pattern that cannot be extended without loss of support, this pattern is maximal and can be written to output. This search strategy is actually a depth first search of the tree of all possible sequences (see Figure 2). We prune branches that cannot yield any supported patterns.

This kind of improvements can work well in some real-life situations, however the theoretical worst-case time still remains exponential. Their main advantage is that they allow to search for longer and more complicated patterns than simple exhaustive search. Examples of this strategy include Pratt algorithm described in detail below and the first, scanning phase of TEIRESIAS algorithm [Rigoutsos and Floratos, 1998b] (see also part 3.3.1). Both programs find patterns allowing gaps.

Pratt software. Pratt [Jonassen, 1996] is quite advanced algorithm based on the idea of depth first search in a tree of patterns, as described above. Pratt discovers quite general patterns which contain

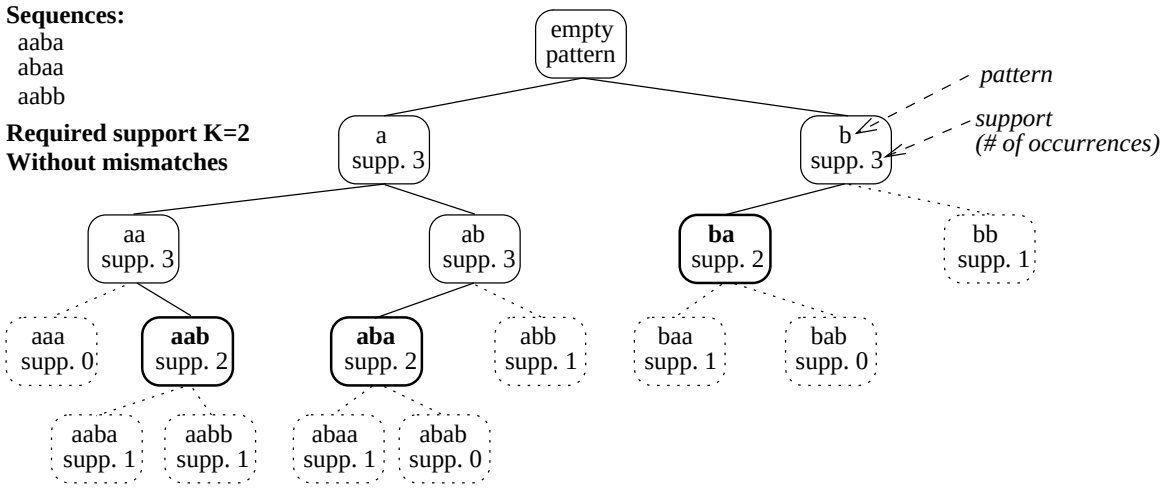

Figure 2: One way to improve exhaustive search is to search in a tree of all possible patterns. When we discover node corresponding to pattern that does not have enough support, we do not continue to search its children. Dashed nodes do not have enough support. Bold nodes are patterns that cannot be further extended.

flexible gaps (such gap has lower and upper bound of its length) and ambiguous symbols (i.e. symbols representing sets of symbols from Σ). Each discovered pattern is required to match exactly at least some predetermined number of sequences. The user has to specify number of parameters that restrict the type of patterns. These include the maximum total length of pattern, maximum number of gaps, maximum number of flexible gaps, allowed ambiguous symbols (i.e. allowed sets of related amino acids) and so on.

To reduce the size of output and the size of search space the program does not report patterns that are less specific than other discovered patterns. Here pattern A is more specific than pattern B if any sequence that matches A must also match B (for example B is less specific if it can be obtained from A by deleting some consensus characters, replacing non-ambiguous character with ambiguous, or making a gap more flexible). This is achieved by a special scoring function that gives higher score to more specific patterns than to less specific.

In each step of the depth first search we take an existing pattern with sufficient support and we add a gap (possibly of length 0) and another character or ambiguous character. All such possibilities are tried out. This means that if we allow flexible gaps, we try all possible lengths of these gaps, if we allow ambiguous symbols, we try all possible sets of symbols. Afterwards each new pattern is tested whether it has enough support. This is done by a special data structure which makes the search faster. Patterns without enough support are discarded.

There are also other optimizations. First, even in one expansion step some new patterns are less specific than others. If more specific pattern and less specific pattern have the same occurrences in sequences, the less specific pattern is not needed (it will not produce better patterns later). Therefore we can discard it.[1] Also for each pattern created so far it is possible to estimate the maximum score we can obtain by extending this pattern. If this estimated maximum is lower than the best score found so far we can discard the current pattern.

In case when no flexible gaps are allowed, the Pratt algorithm is guaranteed to find the pattern with highest score that has enough support. In case when we allow flexible gaps, one of the optimizations is only a heuristics, and therefore does not guarantee finding the highest scoring pattern. The program can also return the highest scoring pattern among those that start at a particular position in the sequence. In this way we get more than one high scoring pattern.

3.2.2 Exhaustive search on graphs

So far we have concentrated on enumerating all patterns or relevant part of the pattern space. Different idea is to search through all combinations of substrings of given sequences that can be possible

[1] Similar idea is used to much greater extend in TEIRESIAS algorithm.

occurrences of a pattern. Assume we have n sequences and we want to find pattern of given length L which occurs in all sequences with at most d mismatches. Then if we take two occurrences of such pattern, they will differ in at most $2d$ positions, because they both differ from pattern in at most d positions. The idea is to search for a group of n substrings of length L, each from different sequence such that any two differ in at most $2d$ positions [Pevzner and Sze, 2000].

Even if we find such combination of substrings it does not guarantee that we find a pattern. For example assume that we want pattern of length $L = 4$ with at most one mismatch and we have found the following 3 occurrences: `AAAA`, `BBAA` and `CCAA`. Any two of them differ in exactly $2 = 2d$ positions but there is no string such that they would all differ in at most one position from this string. However we may assume that this would not happen very often and that most found combination will actually correspond to a pattern.

Another problem is to find the actual pattern from the set of occurrences. Sometimes we do not need the pattern, only the occurrences (for example when we search for protein binding sites). In other cases we may enumerate all patterns that occur within distance d from one chosen occurrence (there are at most $\binom{L}{d}(|\Sigma|-1)^d$ such patterns, this number is exponential in d but not in L, and d is typically small). The search can be further pruned by using knowledge about other sequences. Different possibility is to use the set of occurrences as a starting point of Gibbs sampling or other iterative method (see part 3.4.1, 3.4.2). This is not guaranteed to really find the pattern with specified parameters.

Now let us return to the problem how to find the set of n occurrences, one from each string, so that each two differ in at most $2d$ positions. This can be formulated as a problem in graph theory. Each substring of length L will be a vertex of graph G. Vertices corresponding to two substrings will be connected by an edge if the substrings are taken from different sequences and differ in at most $2d$ positions (see Figure 3a). This graph is n-partite, which means that it can be partitioned to n partitions so that edges there is no edge between vertices in one partition. In this case partitions will corresponds to individual sequences. We want to find a set of n vertices such that any two vertices are connected by an edge. Such set of vertices is called clique.

Problem of finding clique is known to be NP-complete, i.e. we do not know any polynomial-time algorithm. One possibility is to search for clique using exhaustive search with careful pruning. Algorithm WINNOWER [Pevzner and Sze, 2000] vastly reduces the number of edges in the graph, removing only edges that cannot be part of any clique of size n. In this way we may obtain graph with less edges, that will be easy to search for clique.

The algorithm is based on the notion of expandable clique. It is clear that each vertex of a clique is in a different partition (because in clique all pairs of vertices are connected by an edge and vertices in one partition are not connect by an edge). We want to find a clique that has in each partition exactly one vertex. Vertex is called a neighbor of a clique if it connected by an edge with each vertex of a clique (i.e. by adding this vertex we get a bigger clique). Clique with $k < n$ vertices is called expandable if it has in each partition at one vertex or at least one neighbor.

If the graph contains a clique with n vertices, then when we take a subset containing k of these n vertices, this subset is an expandable clique. Therefore for any clique with n vertices there are $\binom{n}{k}$ expandable cliques with k vertices. Any edge belonging to a clique with n vertices is therefore member of $\binom{n-2}{k-2}$ expandable cliques. Therefore if we find an edge that is not a member of $\binom{n-2}{k-2}$ expandable cliques, than this edge cannot be part of n-vertex clique and it can be deleted. We can delete edges until there are no more edges that can be deleted.

In particular we choose k and find all expandable cliques and remove edges that are not in at least $\binom{n-2}{k-2}$ of them. This can destroy some expandable cliques and therefore we iterate (see Figure 3b). Probably entire process can be made more efficient than iteration, however this is not mentioned in [Pevzner and Sze, 2000]. For $k = 1$, each vertex is a clique. Vertex is an expandable clique if it is connected with at least one vertex in each other partition. Vertices without this properties can be deleted. For $k = 2$ clique is each edge (u, v). It is expandable, if there is a vertex w in each of the $n-2$ other partitions such that there are edges (u, w) and (v, w) (i.e. vertices u, v, w form a cycle of length 3). Each edge should be in $\binom{n-2}{k-2} = 1$ expandable clique. Similarly we can require each edge to be in at least $n - 2$ expandable cliques with 3 vertices and so on. The higher k we choose the more time it will take, but in on the other hand we can remove more and more edges.

[Pevzner and Sze, 2000] also give expected time for randomized graph, methods how to find the best value of d, how to extend the technique for a case with uneven nucleotide distribution etc.

$n = 4$, $d = 1$, $L = 3$
Sequences:
`abde`
`afcg`
`hbci`
`jbck`

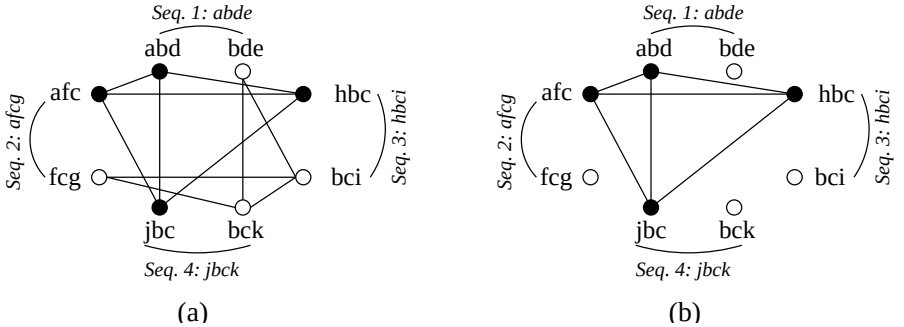

Figure 3: Part (a) shows the graph corresponding to the depicted set of sequences. Part (b) shows the same graph after removing edges with WINNOWER algorithm with $k = 1$. This graph contains exactly one clique corresponding to pattern `abc`.

3.3 Creating long patterns from short patterns

In order for a pattern to be significant, it must be sufficiently long. However long patterns are harder to find using enumerative techniques presented in the previous part. One possible approach to finding long patterns is to start with shorter patterns and to combine them together. Maybe the most elegant example of such algorithm is TEIRESIAS algorithm [Rigoutsos and Floratos, 1998b]. This algorithm is also based on some kind of well-organized exhaustive search, this time based on possible combinations of shorter patterns. In the worst case the algorithm is exponential, but works very well for usual inputs. Authors of TEIRESIAS discovered recently different algorithm which is guaranteed to run in polynomial time (see part 3.3.2).

3.3.1 TEIRESIAS algorithm

TEIRESIAS searches for patterns consisting of characters of the alphabet Σ and wild-card characters '.'. Moreover the patterns must satisfy certain density constraint, limiting the number of wild-cards occurring in any stretch of pattern. We will call such patterns $\langle L, W \rangle$ patterns, where L and W are constants specified by user.

Definition 1 *Pattern P is an $\langle L, W \rangle$ pattern if it meets the following rules:*

- *P is a string of characters from Σ and wild-cards '.'*
- *P starts and ends with a character from Σ*
- *Any subpattern of P (i.e. subsequence starting and ending with a character from Σ) containing exactly L non-wildcard characters has length at most W.*

Consider for example $L = 3$ and $W = 5$. String `AF..CH..E` is a valid $\langle 3, 5 \rangle$ pattern, however string `AF.C.H..E` is not.

TEIRESIAS discovers all $\langle L, W \rangle$ patterns that occur in at least K input sequences ($K \geq 2$ is also specified by the user). However out of several patterns having essentially the same set of occurrences it outputs only the one which is most specific. Pattern P is more specific than pattern Q if we can get Q from P by removing several (possibly 0) characters from both ends of P and replacing several (possibly 0) non-wildcards with wildcard. For example `AB.CD.E` is more specific than `AB..D`.

Maximal patterns. Now assume that pattern P is more specific than Q. It is clear that every occurrence of P is also occurrence of Q. If Q has the same number of occurrences as P than it is not useful to report both P and Q because they cover the same occurrences but P contains more information. Therefore the algorithm outputs only P. However if Q has more occurrences than P, we output also Q, because although it has smaller specificity, it has greater support. The patterns output by the TEIRESIAS are called maximal.

Algorithm. The basic idea of TEIRESIAS algorithm is that if a pattern P is a $\langle L, W \rangle$ pattern occurring in at least K sequences, then its subpatterns are also $\langle L, W \rangle$ patterns occurring in at least K sequences. Therefore the algorithm assembles the maximal patterns from smaller subpatterns.

TEIRESIAS works in two phases. In the first phase (called scanning phase) it finds all $\langle L, W \rangle$ patterns occurring in at least K sequences that contain exactly L non-wildcards. This is done by pruned exhaustive search (see 3.2.1). In the second, convolution phase we try to extend these elementary

patterns by gluing them together. The basic operation is to take two patterns P and Q created so far, take the suffix of P containing exactly $L-1$ non-wildcards, take prefix of Q containing exactly $L-1$ non-wildcards. If the suffix and the prefix are equal, P and Q can be glued together so that the $L-1$ non-wildcards overlap. The list of occurrences of the resulting pattern can be constructed from the lists of occurrences of P and Q (we do not need to scan all sequences). If the resulting pattern occurs at least K times, we keep it, otherwise we discard it.

For example let P = `AB.CD.E` and Q = `DFE.G` (with $L = 3$, $W = 5$). Then P and Q cannot be glued together, because `D.E` $\neq$ `DF`. However if Q = `D.E.G` we can glue them together obtaining `AB.CD.E.G`. If occurrences of P are $(1,1),(2,3),(2,6),(4,7)$ (each pair gives sequence and position in sequence) and occurrences of Q are $(1,5),(2,8),(2,10)$, then the list of occurrences for the new pattern is $(1,1),(2,6)$.

Convolution phase. In the convolution phase we produce all possible patterns in this way. We take each elementary pattern, and we try to extend it on both sides by gluing it with other elementary patterns in all possible ways (depth first search). Any pattern that cannot be extended without loss of support can be potentially maximal. However still we can obtain non-maximal patterns in the output and some patterns can be generated more than once. Therefore we keep a list of patterns written to output so far. We check any newly generated pattern (even if it can be further extended) with the list and if the list contains more specific pattern with the same occurrences we simply discard the new pattern. The search for new patterns is organized so that any maximal pattern P is written to output before any non-maximal patterns less specific than P. In this way we never need to remove pattern already written to the output.

This order of generating patterns is achieved by careful organization of the depth first search. For this purpose we define prefix and suffix ordering of the set of patterns.

Prefix ordering is defined as follows. Take both patterns and replace all wildcard characters with 0 and other characters with 1. Compare the resulting strings lexicographically. The suffix ordering is defined in similar way, except we compare reversed strings. For example `AB.C` is smaller than `AC.B.D` in prefix ordering but it is greater in suffix ordering[2].

[2]Please note, that for simplicity we have defined prefix and suffix ordering in reverse order compared to [Rigoutsos and Floratos, 1998b].

The convolution phase of the TEIRESIAS algorithm can be described as follows:

Convolution phase:

- For each elementary pattern P (starting with the largest pattern in prefix ordering), try to extend pattern P with other elementary patterns.

Extend pattern P:

- While there exist an elementary pattern Q, which can be glued to the left side of P:
  - Take such Q which is largest in suffix ordering.
  - Let R be the pattern resulting from gluing Q to the left side of P.
  - If pattern R has number of occurrences at least K and is maximal with respect to the set of already reported patterns:
    - Try to extend pattern R with other elementary patterns (using the procedure "Extend pattern" recursively).
    - If pattern R has the same number of occurrences as pattern P, then pattern P is not maximal and we do not need to search for other extensions of P (exit the procedure).

    otherwise pattern R is not significant pattern.
- Repeat the same process for the elementary patterns which can be glued to the right side of P (starting with the largest pattern in prefix ordering).
- Report pattern P.

Performance guarantees. It can be shown, that the TEIRESIAS algorithm produces all maximal $\langle L,W\rangle$ patterns. For details see [Rigoutsos and Floratos, 1998b].

Conclusions. The TEIRESIAS algorithm is an exact algorithm. It is guaranteed to find all $\langle L,W\rangle$ maximal patterns supported by at least K sequences.

However, number of such patterns can be very high. In particular, in [Parida et al., 2000] it is

shown, that the number of maximal patterns can be exponential. In such case TEIRESIAS will take exponential time. However, such situation is not likely to happen in the real data. For example, in entire GenPept database containing 120 mil. amino acids there exist only 27 mil. maximal patterns (see [Rigoutsos et al., 2000]). Experimental studies suggest that running time of TEIRESIAS algorithm is linear in the number of patterns on the output [Rigoutsos and Floratos, 1998a].

The other problem with TEIRESIAS algorithm is that the form of the patterns is not very flexible. First of all, the only allowed mismatches are wildcard characters. Newer versions of TEIRESIAS ([Rigoutsos et al., 2000]) allow also patterns containing ambiguous characters representing prespecified groups of characters from Σ. Second, TEIRESIAS patterns do not allow gaps with flexible length. This problem can be addressed by postprocessing phase, where we can combine patterns found into larger patterns separated by flexible gaps ([Rigoutsos and Floratos, 1998a]). However, such methods do not guarantee the performance of the algorithm, i.e. not necessarily all patterns of the specified form will be found.

3.3.2 Work related to TEIRESIAS algorithm

Irredundant patterns. One of the drawbacks of TEIRESIAS algorithm is potentially exponential size of the output and thus potentially exponential running time. This issue was recently addressed in [Parida et al., 2000] by a new algorithm. This algorithm imposes more restrictions on the set of patterns written to the output. They define a set of irredundant patterns such that all other patterns can be easily obtained from this set. The size of this set is at most $3n$ where n is the length of the input. Also they give an algorithm finding all irredundant patterns in $O(n^3 \log n)$ time. Following definition gives notion of irredundant patterns.

Definition 2 *A maximal pattern P is* redundant, *if there exists a set of maximal patterns $P_1, \ldots, P_l$, such that every occurrence of P is also occurrence of at least one of the patterns $P_1, \ldots, P_l$ and every occurrence of P_i (for $1 \leq i \leq l$) is also occurrence of P.*

For example, consider patterns $P_1 =$ `AB.D`, $P_2 =$ `A.DDE`, and $P =$ `A..D`. Let us have the following set of sequences: {`FABDDE`, `ABCDCCD`, `DDACDDE`}. List of occurrences of pattern P_1 in this case will be $L_1 = \{(1,2),(2,1)\}$[3] and list of occurrences of pattern P_2 will be $L_2 = \{(1,2),(3,3)\}$. We can get list of occurrences of pattern P as $L_1 \cup L_2$, therefore P is redundant in this case. However, if we add sequence `EAEDDD`, pattern P will no longer be redundant.

The list of occurrences of redundant pattern can be constructed from the list of occurrences of irredundant patterns without examining the original set of sequences. Therefore, we do not need to find and report these patterns explicitly. This feature allows us to guarantee polynomial running time of the algorithm finding all such patterns.

As far as we know, neither implementation of this algorithm, nor experimental study demonstrating application of this approach is available to date.

SPLASH algorithm. Different algorithm, which finds patterns of TEIRESIAS type is called SPLASH ([Califano, 2000]). As far as we know, the performance guarantees of both algorithms are the same and no comparative study involving both SPLASH and TEIRESIAS is available to date.

3.4 Iterative heuristic methods

So far we have described mainly algorithms guaranteed to find the best pattern. However for more complicated types of patterns we cannot hope to do so. We have to use heuristic approaches that do not necessarily find the best pattern, but may converge to a local maximum. The most important example of such technique is Gibbs sampling.

At the end of this part we also describe polynomial approximation scheme for finding certain kind of patterns. This is a kind of tool that might not find the best possible pattern, but it is guaranteed to find a pattern almost as good as possible. We include it here because it is also based on iterative ideas. Disadvantage of this approach is that the time complexity is too high to be of a practical value.

3.4.1 Gibbs sampling

[Lawrence et al., 1993] present a heuristic algorithm for pattern discovery based on so called Gibbs sampling method. In the simplest version, we are looking for the best conserved ungapped pattern of fixed length W in the form of position weight matrix. We assume, that the pattern occurs in all sequences.

The algorithm works in iterations. The result of each iteration is a set of subsequences of length W

[3] The first number in each tuple represent a sequence number, the second number is a position in the sequence

– exactly one from each sequence. This set of subsequences represents occurrences of the pattern in sequences. We can compute a position weight matrix characterizing the pattern from this set of occurrences. The algorithm works as follows:

- At the beginning select randomly one subsequence of length W from each input sequence. These subsequences will form our initial set of occurrences. Denote o_i occurrence in sequence i.
- **Iteration step.**
  - Pick randomly one sequence i.
  - Compute position weight matrix based on all occurrences except o_i. Denote this position weight matrix P.
  - Take each subsequence of sequence i of length W, and compute a score of this subsequence according to matrix P.
  - Choose new occurrence o'_i randomly among all subsequences of i of length W using probability distribution defined by the scores (higher score means higher probability).
  - Replace o_i with o'_i in the set of occurrences.
- Repeat iteration steps, until some stopping condition is met.

The Gibbs sampling algorithm does not guarantee that the position weight matrix and set of occurrences giving best score will be found. Instead, the algorithm can converge to a local maximum, rather than to the global one. On the other hand, method is fast, which makes it suitable for many applications.

Several problems related to Gibbs sampling and its patterns have been identified and addressed in subsequent work.

- **Phase shifts.** Assume, that optimal set of occurrences starts at positions 8, 14, 22 and 7 of corresponding sequences. When we get position 21, while processing the third sequence, position, the whole system is likely to converge to the set of occurrences 7, 13, 21 and 6 instead.

  The problem was addressed in [Lawrence et al., 1993] by introducing a new randomized step into the algorithm, which computes scores of occurrences shifted by several characters. In this step we compute scores for all possibilities and use random choice with corresponding probability distribution. Similar approach was taken in PROBE [Neuwald et al., 1997], where authors in similar way reduced or extended pattern on both sides.
- **Multiple patterns.** Sometimes it is appropriate to define a pattern as a sequence of several consecutive subsequences of fixed length separated by variable length gaps. It means, that occurrence in Gibbs sampling would be in this case represented by several short subsequences in the sequence rather than one. It is possible to modify Gibbs sampling in this way [Lawrence et al., 1993, Neuwald et al., 1997] using dynamic programming in the process of ranking and choosing a new candidate occurrence. Lengths of subsequences and their number is specified beforehand.
- **Pattern width.** So far we have assumed, that the pattern width is fixed and is given to us by user beforehand. Most of the time, it is not reasonable assumption, especially if we are looking for multiple patterns separated by variable length gaps.

  In PROBE [Neuwald et al., 1997] genetic algorithm is used to determine parameters of patterns (i.e. the number of subsequences and their lengths). When we have two sets of parameters, we can try to recombine them (take part of the first and part of the second set) and in this way we can sometimes obtain a better set of parameters. Sets of parameters for recombination are chosen by random with distribution proportional to their score (called *fitness*). Fitness of the set of parameters is determined by running Gibbs sampling procedure with the set of parameters.
- **Gapped patterns.** Not all positions within continuous block of length W are necessarily important for the function of this block. Rather we want to create a pattern, which is gapped, i.e. only $J < W$ positions are used to form the model.

  This issue was addressed in [Liu et al., 1995]. The authors suggest to introduce yet another randomized step, where we replace one of the J positions, which we take into account, by one of the $W - J + 1$ positions, which are not included in the pattern. The choice is again done

randomly with distribution of probabilities proportional to corresponding scores.

3.4.2 Other iterative methods

There several other approaches using iterative methods similar to Gibbs sampling. Typical approach is to start with some pattern. Then find occurrences of this pattern in the sequences. Based on this occurrences build a pattern that matches the occurrences best. Repeat this process with the new pattern until no improvement is obtained. The main difference between this approach and Gibbs sampling is that here we use all sequences for definition of the new pattern and then we refine position of the new pattern in all sequences. The process is completely deterministic, and of course has no guarantee to find the global optimum.

This kind of approach was used for example in [Pevzner and Sze, 2000] where authors want to find ungapped deterministic pattern of given length that matches all sequences with mismatches. We want to minimize the total number of mismatches. By using different methods they obtain a set of occurrences of some unknown candidate pattern and they refine it by iterative method. In each step they compute the new pattern by taking the most frequent character in each position (based on the frequencies in the occurrences). The method is further improved to remove non-significant columns from consideration, obtaining a gapped pattern. Also it is possible to use this method to find patterns which do not occur in all input sequences.

The goal of [Singh et al., 1998] is to detect coiled coil regions in histidine kinase receptors. Coiled coils were previously detected in other protein families, therefore the statistical properties of such regions are known, although they may be slightly different in this family. The goal is to find distribution of residues and pairs of residues in different distances apart in a sliding window of fixed length, provided that such window is from coiled coil region. The process start with taking known distribution from other families. Based on this signal we can assign score to each position of sliding window and the best scoring positions are the candidates for coiled coil region. We randomly choose sample of these candidates and based on this sample we compute a new distribution. This process is iterated. In each step we add a pseudocount from the known distribution of other families. In contrast to previous method, this one is randomized, and also cannot diverge too much from the original pattern due to the pseudocounts.

Finally, the iterative approach can be also used to improve position weight matrices, as shown in [Zhang, 1998]. Here the author starts from PWM computed for several signals from vertebrate genomes and refines them by iteration to obtain PWM specific for human.

In general it seems that simple iterative methods are suitable for improvement of patterns obtained by other methods or from different data. They are however not good enough to discover patterns without any prior knowledge.

3.4.3 From iteration to PTAS

Some problems associated with pattern discovery are NP-hard, which means that it is unlikely that any polynomial time algorithm for such problem exists. Example of such problem is Consensus Pattern problem, where we want to find pattern P (consisting only of characters form Σ) and one occurrence of P in each sequence so that the total number of mismatches over all occurrences is minimized (mismatch here means substitution).

Since there is not an algorithm guaranteed to find the best such pattern in reasonable time, we may wish to have a guarantee that the cost of the found pattern (i.e. the total number of mismatches) is guaranteed to be at most α times the cost of the optimal pattern. Value α is called approximation ratio. For example if $\alpha = 2$ we are guaranteed to find a pattern that has cost at most twice as many as mismatches as the best possible pattern for this set of input sequences.

For some problems it is possible to construct an algorithm that works for any α (that is the user can specify the desirable precision). However the trick is that the smaller is approximation ratio, the longer the algorithm runs. This type of algorithm is called polynomial approximation scheme, or PTAS. [Li et al., 1999] shows a PTAS for Consensus Pattern problem. It is based on a simple iterative idea repeated many times with different initial patterns.

The PTAS gets input sequences, the desired length L of pattern and parameter r. It finds all possible combinations of r substrings of length L taken from input sequences. Each combination may contain zero, one or several substrings from each sequence, some substrings may even repeat more than once. If the total length of all sequences is N, there are $O(N^r)$ combinations. For each combination of r substrings perform the following steps:

- Compute the majority pattern P of the r substrings. This pattern has in each position the

character occurring most frequently in this position in r substrings.

- For each input sequence find the best occurrence of P.
- Based on these occurrences compute a new majority pattern P'.
- Find the best occurrences of P' in all sequences and compute the number of mismatches (cost of P').

The result will be the pattern P' which achieves the minimum cost. Notice, that the algorithm performs one step of iteration with the pattern obtained from each possible combination. The running time of the algorithm is $O(N^{r+1}L)$ and its approximation ratio is $1 + (4|\Sigma|A - 4)/(\sqrt{e}(\sqrt{4r+1} - 3))$ for $r \geq 3$.

This result is very interesting from the point of view of theoretical computer science. However the algorithm is not very practical for real use. For example, if we choose $r = 3$ and $\Sigma = \{A, C, G, T\}$ the algorithm will find the pattern with at most 13 as many mismatches as the optimal pattern. For $r = 3$ the running time is $O(N^4 L)$, which is impractical for large inputs. In order to achieve $\alpha = 2$ we need $r = 20.7$ which gives algorithm with time complexity $N^{21.7}$. Of course, the approximation ratio is only upper bound of the possible error. The algorithm can for some inputs find even optimal or close-to-optimal results even for small r, however there is no guarantee.

There is a new program called COPIA [Liang et al., 2000] inspired by this PTAS, which is more practical, however it is heuristic without guaranteed performance. The enumeration of all possible combinations of r substrings is replaced by random sampling and one iteration of pattern improvement for each combination is replaced by iterating until there is no further improvement (similarly as in [Pevzner and Sze, 2000]).

Finally, [Li et al., 1999] also describe PTAS for the problem of finding the best consensus pattern under the information content measure. In addition they give an algorithm with a fixed approximation ratio which finds the pattern that has occurrence with at most d mismatches in each sequence (minimizing value d).

3.5 Machine learning methods

Sometimes a pattern cannot be well described by a simple deterministic pattern and one may wish to express it in form of stochastic model, such as Hidden Markov model, or position weight matrix (which is a simpler version of HMM). This kind of models is usually discovered using iterative expectation maximization techniques that do not necessarily converge to global maximum.

First we will consider simpler case of position weight matrices. Notice, that the algorithms presented here differs from those from previous part in a fact that it uses all possible occurrences of the pattern to obtain new matrix, instead of only one chosen occurrence in each string. In the second part we discuss Hidden Markov models, including issues that have to be addressed when we want to use already trained Hidden Markov model for pattern matching.

3.5.1 Expectation maximization

[Lawrence and Reilly, 1990] used simple learning algorithm called expectation maximization (EM) algorithm. The purpose of the algorithm is to estimate parameters of the stochastic model of the pattern, which occurs once at unknown position in each sequence from the given family of sequences. The position weight matrix is used as an underlying stochastic model in [Lawrence and Reilly, 1990]. However, the approach can easily be extended to more complicated models e.g. with flexible gaps, finite mixture model [Bailey and Elkan, 1994], etc. In some cases, we can choose underlying model using prior knowledge of the sequence family.

The algorithm is iterative, in each iteration performing two steps as follows:

- **E step.** For every sequence s and for every position in s compute the probability that the occurrence of the pattern in s starts at this position. Base the computation on the model of the pattern from previous iteration, or on the initial model parameters, if this is the first iteration. Initial model parameters are usually set randomly.
- **M step.** For every position in the pattern compute new probabilities of characters at this position. This is done based on all possible occurrences of the pattern weighted by probabilities computed in E step. These values will form new parameters of the model of the pattern.

As for the most of the iterative methods, the main problem of EM algorithm is, that instead of converging to global maximum likelihood, it will rather converge to local maximum depending on initial parameters of the model. The other

problem of EM algorithm, as it is described in [Lawrence and Reilly, 1990], is the assumption that every pattern occurs exactly once in every sequence.

[Bailey and Elkan, 1995] addressed these two problems. They developed MEME algorithm, which is modification of EM algorithm. Their algorithm is based on the assumption, that the pattern found should closely resemble at least one subsequence found in the dataset. They also modified formulas given in [Lawrence and Reilly, 1990] so, that they can handle several or none possible occurrences of the pattern in a sequence. The algorithm works as follows:

1. For each subsequence in the dataset, form initial model. The initial model is a position weight matrix, where for every position the character on corresponding position in the subsequence has probability p (p is usually between 0.5 and 0.8), and all other characters have probability $(1-p)/(|\Sigma|-1)$.
2. One iteration of EM algorithm is performed on each such initial model. Likelihood score is computed for resulting models.
3. Choose model with the largest likelihood score and use it as an initial model for EM algorithm.

The algorithm can be forced to report more patterns by erasing all occurrences of the found pattern from the dataset and rerunning the whole process again.

3.5.2 Hidden Markov models

Hidden Markov Models (HMMs) can be used as a model of a family of sequences. For detailed description of HMMs and related algorithms see [Durbin et al., 1998]. There are three issues, which need to be addressed, when using HMM as a representation of sequence family:

- **Topology of HMM.** Topology specifies general layout of the model, which we use to represent a sequence family.
- **Training process.** The learning process is needed to estimate the parameters of the model so that the sum of scores of sequences in the family is optimized.
- **Search for sequences.** The searching process should allow us to distinguish between the sequences, that belong to the family and sequences, which do not.

Topology of HMM. Commonly used HMM topology [Krogh et al., 1994, Hughey and Krogh, 1996] for sequence analysis is depicted in Figure 4.

The model consists of three types of states. **Match states** model conserved parts of the sequences (motifs). Match states specify probability distribution of characters on each conserved position. There can be any number of match states in the model, we assume that this number is given by the user beforehand. **Insertion states** model possible gaps in between match states. Gaps can be arbitrarily long. Probability assigned to a self-loop in an insertion state models probability distributions of possible gap lengths (the probability distribution is geometric, mean value can be easily computed). Finally, **deletion states** allow to bypass some of the match states.

Once the all parameters (probabilities) of the model are set, we can compute most probable path through the model for a given sequence (in $O(nm)$ time using Viterbi algorithm [Durbin et al., 1998], where n is the length of the sequence and m is the number of states in the model). Positions corresponding to match states in the model represent the most probable occurrence of the pattern in the sequence.

We can also compute the probability P that the sequence was generated by the model (in $O(nm)$ time using forward algorithm [Durbin et al., 1998]). We call value $-\log P$ a *NLL score of the sequence* with respect to the model. Higher probability of generating the sequence corresponds to lower NLL score of the sequence.

Training of the model. Now we have the topology of HMM. We also have a family of sequences which we want to characterize by the model. We need to estimate parameters of the model so that the model will generate sequences similar to those in the family with high probability.

There is a standard learning method to do this task (Baum-Welch algorithm, see [Durbin et al., 1998]). The method is iterative. First we start with an arbitrary model parameters (if we have some prior knowledge about the sequence family, we can use this knowledge to set initial parameters). Then in each step, probabilities of all paths for all sequences are computed, and the model parameters are reestimated in order to minimize the NLL score (maximize the probability) of the training sequences.

The algorithm does not guarantee finding of

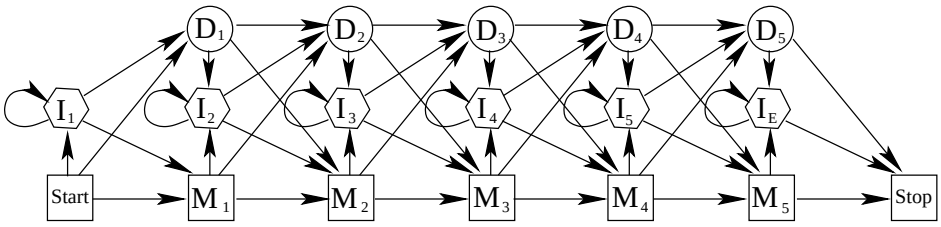

Figure 4: Commonly used HMM topology. States $M_1, \ldots, M_5$ are match states, $I_1, \ldots, I_E$ are insertion states, and $D_1, \ldots, D_5$ are deletion states.

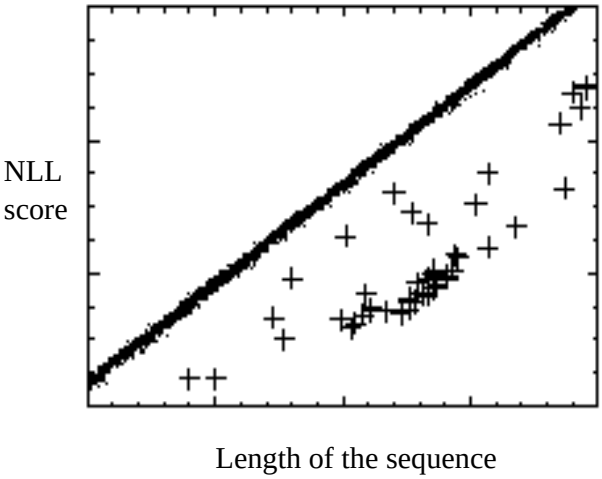

Figure 5: NLL score versus sequence length. Sequences from the domain, on which learning have been applied, are denoted by '+'. [Hughey and Krogh, 1996]

global optimum. It can happen that the algorithm will iterate to a local minimum. This depends on initial parameter settings.

Search for sequences. Search for the pattern in the form of HMM is more complicated, than in the case of simpler patterns. Given a sequence, we can compute the most probable alignment of the sequence to the pattern represented by HMM using Viterbi algorithm and compute its NLL score by forward algorithm (see [Durbin et al., 1998] for algorithms details). NLL score gives us a measure of how well the sequence can be aligned to the pattern.

However, NLL scores highly depend on a sequence length. In general, shorter sequences have smaller NLL scores than longer ones. Therefore, NLL score with certain threshold can be used for discrimination only if all the input sequences have approximately the same length.

Figure 5 shows typical example of NLL score dependency on sequence length. Sequences, which do not belong to the sequence family, form a line corresponding to a linear dependency. NLL scores for family members significantly drop below this line.

[Krogh et al., 1994, Hughey and Krogh, 1996] use simple technique for discriminating family members called windowing technique:

1. Take all intervals $[k, e_k]$ containing at least 1000 sequences. For every such interval compute the average sequence length t_k, the average NLL score μ_k and standard deviation σ_k.

2. Linear interpolation (and extrapolation on both ends) is used on points (t_k, μ_k) to get expected NLL score for every sequence length. Similarly, expected standard deviation for every sequence length is found.

3. For every sequence, z-score (the number of standard deviations from the expected value of NLL score) is computed. If z-score is more than given threshold ([Hughey and Krogh, 1996] use threshold 4), consider the sequence as a member of the sequence family.

4. Remove all sequence family members already identified and perform the whole routine again, until some stopping condition is met (typically, number of iterations is given beforehand).

[Hughey and Krogh, 1996] report that this method for sequence discrimination usually works, however, there is no guarantee of that. The other disadvantage of this technique is that the search must be performed on many sequences at once to get enough data (in other words, we need enough background sequences, which are not members of the sequence family).

3.5.3 Improvements of HMM models

Reducing number of parameters. The HMM model with the topology as it was presented in previous section has usually too many parameters with respect to amount of training data. For this reason, it cannot be trained properly and overfitting to training data is possible. There are several alternatives, how to solve this problem.

- **Model surgery.** The idea of the model surgery [Hughey and Krogh, 1996] is to adjust model topology during the training, reducing number of parameters of the model. In particular, authors observed, that following two problems arise during the training:
  - **Some match states are used by only few sequences.** If a match states is used only by number of sequences below the threshold (typically one half), the state is removed. In this way we force sequences to either use insertion state at this point, or change their alignment significantly.
  - **Some insertion state is used by too many sequences.** If an insertion state is used by more sequences than given threshold (typically one half), then the state is replaced by a chain of match states. The number of inserted match states is equal to expected number of insertions in the replaced insertion state.
- **Use different initial topology**. MetaMEME [Grundy et al., 1997] uses another program to report simple short patterns in the sequence. These patterns are transformed into matching states in HMM. Different patterns are combined together using insertion states as on Figure 6.

Discovering subfamilies. Sometimes the family of sequences can consists of several subfamilies. It means, that the family is represented by several motifs rather than by one. To handle this case, [Krogh et al., 1994] combine several HMMs with standard topology to one larger HMM as it is shown in Figure 7.

If we do not have any preliminary knowledge on base of which we can set initial parameters of such model, it might be hard to train the model accurately. Therefore, use of Viterbi training algorithm, which considers only one best path (in contrast to Baum-Welch algorithm which uses all paths weighted by their probabilities), is more appropriate in this case. The advantage of this algorithm is, that once some part of the model get biased to one subfamily, only sequences in this subfamily will be used to train this part of the model.

3.6 Methods using additional information

Pattern discovery is a difficult task. If we want to find a longer flexible pattern, exhaustive methods are no longer feasible and in many cases no methods with guaranteed performance are known. Often the pattern we need to find is very subtle, hardly distinguishable from random sequence. In this part we will show how to exploit other sources of information, other than bare sequences.

3.6.1 Finding motifs in aligned sequences

Finding pattern and finding local alignment are closely related tasks. Once we have a multiple sequence local alignment we can easily find a pattern. Local alignment already gives us position in each sequence which is a candidate occurrence of the pattern. From this we can easily obtain the pattern in form of consensus sequence, position-weight matrix etc. The question starts to be interesting, if we assume that the input can contain errors (i.e. some sequences are not aligned correctly or they are not members of family etc.). In this case we may try to find pattern which does not match all sequences, maybe even pattern characteristic for a subfamily. This task is addressed by EMOTIF software [Nevill-Manning et al., 1998].

EMOTIF searches for motifs containing characters, ambiguous characters (sets, e.g. `[ILMV]`) and wild-card characters '`.`'. The set of possible ambiguous characters is fixed. It is obtained by inspecting substitution frequencies between pairs of amino acids in a database of aligned protein families. After that clustering algorithm is applied, producing set of clusters such that amino acids within one cluster frequently substitute one for another and they substitute less frequently with amino acids outside this cluster. The resulting clusters are then used as possible ambiguous characters.

Each character (normal, ambiguous or wild-card) corresponds to one column of the local alignment. For each such pattern it is possible to compute its specificity (how likely it is to occur by random) and coverage (how many training sequences it covers). EMOTIF finds many motifs with different values of specificity and coverage (however lower bound of

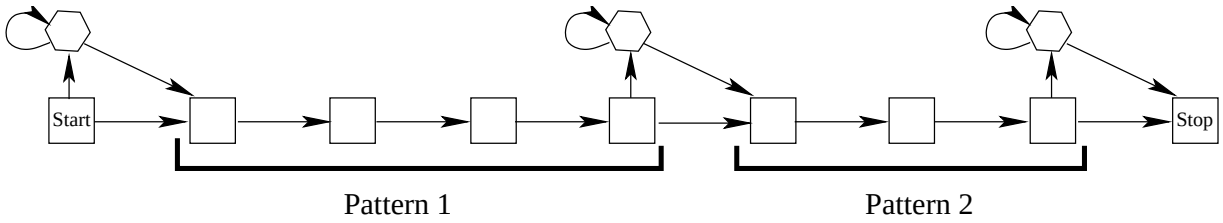

Figure 6: Meta-MEME uses much simpler initial topology. Patterns found by other programs are connected together by insertion states.

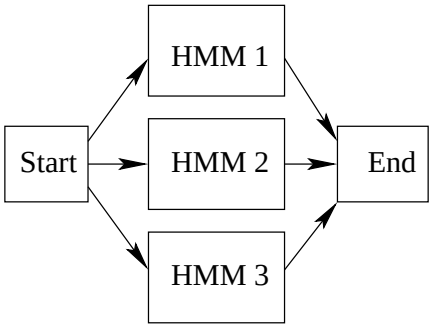

Figure 7: Topology of HMM suitable for representing families of sequences with several subfamilies.

coverage is specified by user). For several motifs with the same specificity only the one with greatest coverage is reported and vice versa. Motifs are found by pruned exhaustive search, trying to apply all applicable ambiguous characters in each column, and computing specificity and coverage.

3.6.2 Global properties of a sequence

As pointed out in [Pedersen et al., 1999], transcription factor binding site signals are often too weak to be distinguishable from random sequence. However the cell is able to recognize transcription start sites correctly. Therefore the lack of local information indicating transcription starting site implies that this information is somehow represented globally, in long stretches of DNA. One example of such phenomenon are CpG islands – most vertebrate genes have 1-2kb long regions with higher frequency of CpG than found elsewhere in the genome [Pedersen et al., 1999, Durbin et al., 1998]. [Pedersen et al., 1999] also shows that region downstream from transcription start point has usually low flexibility (flexibility, or bendability of DNA can be estimated from sequence-based models of DNA structure).

In pattern matching problem we can use global information, such as CpG islands and flexibility to distinguish random occurrences of pattern from those that have functional significance. In pattern discovery we may use this kind to prior knowledge to choose appropriate parts of genome as our input set in which we search for patterns.

3.6.3 Using phylogenetic tree

One of the basic assumptions in finding patterns in biological sequences is that regions conserved in evolution are functionally important. Therefore it is natural to use known phylogenetic relations among sequences to guide the pattern search.

Assume we want to find a regulatory element. Instead of using regulatory regions from many coregulated genes of the same species we will use regulatory regions of the same gene taken from many related species. We assume that the evolutionary tree of these species is known. Now we may try to find the short pattern best conserved in the evolution.

This approach is used in [Blanchette et al., 2000]. They use a parsimony measure to select the best conserved pattern, defined as follows. We are given length of pattern k. We want to associate a sequence t_w of length k with each node w. In each leaf, t_w is required to be a subsequence of the input sequence associated with this leaf (t_w corresponds to an occurrence of the pattern). We want to minimize the sum $d(t_v, t_w)$ over all tree edges (v, w). Here $d(t_v, t_w)$ is distance between two strings of length k. Authors consider Hamming distance (i.e. number of substitutions), however they note that the algorithm can be used also for edit distance. The problem to find such strings minimizing the parsimony score is NP-hard, therefore they devise an exhaustive search.

We first root the tree in an arbitrary internal node and then compute the scores for subtrees of the tree, starting from smaller subtrees. For node w and each possible string t of length k we compute the best

possible parsimony score $d^*_w(t)$ that can be achieved in the subtree rooted at w provided that the string t is stored in w. Let n be the number of the input sequences. Since there are $2n-2$ nodes in a tree and $|\Sigma|^k$ possible strings, we compute $(2n-2)|\Sigma|^k$ scores.

The scores are found easily for leaf nodes: if t is a substring of the input string associated with the leaf w, then $d^*_w(t) = 0$, otherwise the score will be ∞. Now, let w be an internal node with children w_1 and w_2. Assume that the node w_1 stores sequence t_{w_1}, w_2 stores t_{w_2} and w stores t_w. Then the parsimony of the subtree rooted at w will be

$$[d^*_{w_1}(t_{w_1}) + d(t_{w_1}, t_w)] + [d^*_{w_2}(t_{w_2}) + d(t_{w_2}, t_w)].$$

For each possible t_w we want to find t_{w_1} and t_{w_2} that minimize this sum. We can choose t_{w_1} and t_{w_2} independently of each other. Therefore we can minimize each of the square brackets separately and then just sum the result. Our task is therefore to find t_{w_i} that minimizes $d^*_{w_i}(t_{w_i}) + d(t_{w_i}, t_w)$. Simplest algorithm takes all possible values of t_{w_i} and computes the sum. This leads to time complexity $O(nk|\Sigma|^2 k)$ because for each edge between w and its child w_i we need to consider all possible pairs of t_w and t_{w_i} and to compute the distance in time $O(k)$.

We can further improve the running time to $O(nk|\Sigma|^k)$ by computing the value $\min_{t_{w_i}}[d^*_{w_i}(t_{w_i}) + d(t_{w_i}, t_w)]$ for all possible values t_w at once. In particular, for some fixed t_w this minimum is always either equal $d^*_{w_i}(t_w)$ (in case $t_{w_i} = t_w$) or it is one more than the minimum for some string t'_w within Hamming distance 1 from t_w.

Therefore we can construct a graph in which each value of t_w correspond to one vertex. Two vertices are connected by an edge with weight 1, if their corresponding strings have Hamming distance 1. We add additional, starting vertex, and connect it to all vertices. Weight of an edge from start to vertex corresponding to t_w will be $d^*_{w_i}(t_w)$. Now $\min_{t_{w_i}}[d^*_{w_i}(t_{w_i}) + d(t_{w_i}, t_w)]$ is equal to the length of the shortest path from the starting vertex to t_w. This values can be found by Dijkstra algorithm for all vertices at once in time $O(k|\Sigma|^k)$, using the fact that the distance is a small integer (at most $2nk$), and degree of each vertex other than start is $k+1$. This gives overall complexity $O(nk|\Sigma|^k)$. The algorithm can be further pruned by stopping the Dijkstra algorithm as soon as the distances considered are higher than certain threshold.

The described algorithm computes the best parsimony score for the root node. We can use stored intermediate results to reconstruct entire optimal solution in a root-to-leafs manner. Strings t_w stored in the leaves represent the occurrences of our pattern.

This method, based on phylogenetic trees and parsimony measure, has two advantages. First, it can be used to discover regulatory elements that regulate only very small number of genes in one genome. This cannot be done using only information from one genome. Second, pattern finding algorithms have often problems if the input contains a group of highly similar sequences. These sequences then may force the algorithm to find pattern that just characterize similarity of these sequences, but not features common to all sequences in the input. Therefore it is often necessary to find closely homologous groups of sequences and choose only one member from each. In the phylogenetic tree close homologs are grouped together their weight is not so great – the pattern still have to fit well also other parts of the tree.

3.6.4 Use of secondary/tertiary structure

The positions important for secondary and tertiary structure of proteins are usually well conserved. If we know the structure of proteins in question, we can try to locate regions important for achieving this structure. These regions are good candidates for occurrences of our pattern. This approach was used in [Ison et al., 2000]. Authors use points of contact between two secondary structure elements as candidate spots for conserved positions. They try to construct a sparse deterministic pattern containing ambiguous characters and flexible gaps. A pattern should cover entire length of a protein. They find their patterns mainly manually. First they use a software to find points of interaction between secondary structure elements (each point is given by a pair of interacting residues). Then they manually align sequences so that the points of contact align together, if possible. Based on this alignment they choose positions that are well-preserved (among those columns that contain many points of contact). This set of positions constitutes a pattern, which is then tested against a database of proteins. If the performance of the pattern is not satisfactory, it is further improved (manually). Although the authors in [Ison et al., 2000] create patterns manually, it is for sure possible to implement similar process as a program. The main difficulty would be to choose various scoring functions to incorporate biological knowledge and intuition used at various stages of the pattern discovery by authors.

Secondary structure and search for motifs are closely tight together also in algorithms from

[Gorodkin et al., 1997b]. Here authors search for conserved patterns in RNA sequences. These are however more related in their structure rather than in sequence. On the other hand finding secondary RNA structure of a set of related RNA sequences is best done by first aligning the sequences. Alignment in turn requires to discover similarities. Therefore their algorithm tries simultaneously discover alignment, secondary structure features and conserved pattern.

3.7 Finding Tandem Repeats

An interesting comparison of deterministic and probabilistic algorithms can be made for the problem of finding tandem repeats. [Coward and Drablos, 1998] present an iterative method for finding repeats that is based on self-alignment of a given genomic sequence, while [Benson, 1999] gives a probabilistic approach to the same problem.

The first phase of Coward and Benson's algorithm is a process of *phase alignment*: a sequence is divided into consecutive subsequences, each of length p, and a measure of the 'mutual agreement' between the subsequences is calculated. If the sequence is p-periodic, this measure will be 0. If the sequence is periodic except for a few substitutions, the measure of mutual agreement will still be small. However, an insertion or deletion in the sequence will have a large effect on the measure. In this case, the effect of an insertion or deletion in the sequence can be compensated in the succeeding subsequences by a right or left shift [Coward and Drablos, 1998, p.499]. The alignment algorithm will then iterate until an optimal or near-optimal alignment has been found.

The second phase of the algorithm involves finding a *consensus pattern*, a candidate for the underlying periodic pattern. The third, optional, stage is the adjustment of the phase alignment according to the consensus pattern. The final stage is the computation of the phase scores for different possible periodicity values. The authors report that their algorithm works best for tandem repeats of at least 12–15 bases, and that when there are few insertions and deletions, the phase agreement is a good indicator of the presence of a repeat. They add that "the sensitivity of the method is also good when there are many substitutions" (p.506). They acknowledge that their method of phase shifting, in which the original sequence is split into independent parts, may be artificial, as it does not necessarily correspond to our understanding of biological sequences, but, in adopting this approach, they have gained a significant simplicity and efficiency of implementation.

In contrast, [Benson, 1999] probabilistic method for finding tandem repeats is a good deal more complicated in design and implementation. It models the alignment of two tandem copies of a pattern of length n by a sequence of n-independent Bernoulli trials (coin tosses). The algorithm has two separate *detection* and *analysis* components: the detection component uses a set of several statistically-based criteria to find candidate tandem repeats; the analysis component tries to produce a potential alignment for each candidate, and, if successful, various statistics about the alignment (e.g., percent identity) and the nucleotide sequence (e.g., composition, entropy measure).

At present, a limitation of Benson's algorithm is its lack of a good statistical measure for tandem repeats. Right now, the authors rely on a cut-off alignment score based on simulations with random sequences. Future development of this probablistic algorithm may be more likely than Coward and Drablos's simpler method to address such complex issues as analyzing the mutational history of repeats, which will require describing how the interwoven progression of substitutions, indels, and duplication/excision events led from the original single sequence to the current sequence of approximate repeats.

4 Statistical Significance

Given the multitude of methods discovering conserved pattern in biological sequences, one needs measures to assess the quality of such motif. This is important in experimental comparison studies of various methods, in ranking found motifs for user, and even as a guide in the pattern discovery process itself. Many different approaches have been developed to determine when the number of occurrences of a given pattern in a set of sequences is significant. Here we will discuss in detail three approaches that have been widely used by many authors, others will be just mentioned.

4.1 z-score

Suppose we have 100 sequences of length L and a simple pattern s of length k, for $k \leq L$. Assume it has N_s occurrences in the sequences. Let S be the set of $|\Sigma|^k$ possible patterns, i.e. strings of length k. The number of possible occurrences of s in each

sequence is $L-k+1$, therefore the total potential occurrences will be $100(L-k+1)$. We want to be able to decide whether the number of occurrences N_s observed for pattern s is something that we would expect to happen if the sequences were drawn under random (background) conditions.

This problem is related to hypothesis testing in statistics, the significance value is helping to decide when to reject or accept the null hypothesis that the observed N_s is usual under random conditions. A very standard criterion is to reject the null hypothesis when a P value of 0.05 is observed. What this value is suggesting is that, if we do 100 experiments, in each drawing 100 random sequences, we would expect to see s occur in at least N_s sequences just in 5 experiments. Another interpretation of the P value is the probability of making a mistake rejecting the null hypothesis.

Different statistical measure which we can use to measure statistical significance of a pattern is z-score. Let $E(N_s)$ and $\sigma(N_s)^2$ be mean and variance of the number of occurrences of the pattern s in the sequences generated according to the background distribution. Then the z-score associated with s is given by the following formula

$$z_s = \frac{N_s - E(X_s)}{\sigma(X_s)}$$

Then z-score z_s has limiting normal distribution with mean 0 and variance 1. It is the number of standard deviations by which the observed value N_s differs from the expected value. Since it is normalized it is suitable for comparing different motifs. Using Chebyshev's inequality we can obtain bounds on P value from the computed z score.

Examples of methods using similar statistical techniques to assess statistical significance of patterns can be found in [Atteson, 1998, Tompa, 1999, Sinha and Tompa, 2000, van Helden et al., 2000, Gelfand and Koonin, 1997]. The approaches in these papers differ in biological motivation, type of patterns considered (allowance of gaps, exact matches versus approximate matches, etc.), and background distribution. Considered model for generating random sequences (background distribution) is usually either characters generated independently with given probability of individual characters, or sequences generated by Markov chain of order k. Transition probabilities of Markov chain are determined by the $(k+1)$-mer frequencies in the input set of sequences, or in some larger database of sequences.

Article [Cowan, 1991] describes exact formula to compute expected number of occurrences of a given pattern among sequences generated by the first order Markov chain with the same transition probabilities as the input sequence, and also the same starting character. They use Whittle's formula [Whittle, 1955] to achieve this. Means and variance of the distribution of the number of occurrences of given pattern are also obtained in [Nicodème et al., 1999], using generating functions theory.

[Sinha and Tompa, 2000] compute z-score of a pattern defined as a string of length k over the alphabet $\{A, C, G, T, R, Y, S, W, N\}$, where R means purine, Y pyrimidine, S strong bond and W weak bond nucleotides. Input consists of a set of upstream regulatory sequences each having the same length. As a background distribution they use Markov chain of order 3, in order to account for frequent strings, such as `TATA`, `AAAA`, etc, that occur often in regulatory sequences.

They show how to compute z-score under these assumptions in $O(c^2k^2)$ time where c is the number of ambiguous characters. The most difficult part of this computation is to determine $\sigma(X_s)$. Therefore they use the following expression

$$z_s^* = \frac{N_s - E(X_s)}{\sqrt{E(X_s) - E(X_s)^2}},$$

with property $z_s^* \geq z_s$. Hence, z_s^* is computed first to determine if it is worthwhile to go into the variance computation.

[Pesole et al., 2000] use the same approach but they performed a chi-square test to assess the statistical significance such that

$$\chi^2 = (N_s - E(N_s))^2 / E(N_s).$$

Since they consider a large class of patterns, they do not determine the expected value precisely, but obtain estimate by actually simulating Markov chain and generating sequences.

4.2 Information content

Another popular approach for determining significance of motifs is based on *information content* (also called relative entropy) of the motif. A good description can be found in [Tompa, 1999].

Again, suppose a motif s of length k, has approximate occurrences in a subset of the S input sequences. The relative entropy of this motif is defined to be

$$\sum_{j=1}^{k} \sum_{c \in \Sigma} p_{c,j} \log_2 \frac{p_{c,j}}{b_c}$$

where $p_{c,j}$ is the frequency with which character c occurs in position j among the motif occurrences in S, and b_c is the background frequency of the character c.

Relative entropy provides a measure of how well conserved and how likely a motif is with respect to the background distribution. In particular, the more different the distribution $p_{c,j}$ from the background distribution b_c, the higher the relative entropy of position j.

This measure is good for comparing two motifs that have the same number of occurrences (occur in equinumerous subsets of the N input sequences), but not if the two motifs occur in a vastly different number of sequences. The reason is that the absolute number of occurrences are not taken into account, depending instead on the relative frequency $p_{c,j}$ of occurrence of each of the characters. Most of the applications that use relative entropy depend on the fact the motif occurs in all, or nearly all, of the N sequences.

Information content is also used for position weight matrices [Hu et al., 2000] (see also part 3.1.3 of this report). Assume that matrix entry $A[c, j]$ contains relative frequency of character c at position j of the pattern. In this case, the entropy for a particular column j in the matrix is defined as follows

$$E_j = -\sum_{c \in \Sigma} A[c, j] \log_2 A[c, j].$$

The consensus quality of column j is $C_j = \log_2 |\Sigma| - E_j$ and the final consensus quality of a matrix A is defined as

$$con(A) = \frac{1}{k} \sum_{j=1}^{k} C_j,$$

where k is the width of the motif. The consensus quality guides the search for well-conserved motif candidates.

4.3 Sensitivity, specificity and related measures

One application of pattern discovery methods is to find patterns that characterize given family of related proteins. We want to measure suitability of the discovered motifs for this task by finding occurrences of the motif in the database of all known proteins. We may distinguish between *true positives* TP (proteins that belong to our family and contain the pattern), *true negatives* TN (sequences that do not belong to the family and do not contain the pattern), *false negatives* FN (sequences that belong to the family and do not contain the pattern), and *false positives* FP (sequences that do not belong to family and contain the pattern). Based on counts of TP, TN, FP, FN we can define various measures [Brazma et al., 1998]. *Sensitivity* (also called coverage) is defined as $TP/(TP + FN)$ and *specificity* is defined as $TN/(TN + FP)$. A pattern has maximum sensitivity, if it occurs in all pattern in the family and maximum specificity, if it does not occur in any sequence outside the family. If we want to combine these two measure to one score, we may use *correlation coefficient*

$$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FN)(TN + FP)}}$$

This expression has value 1 when there are no false positives or false negatives and decreases towards zero, as the number of false positives and false negatives grows.

5 Biological Verification

In this section we describe methods how the patterns discover by computer science tools can be verified experimentally. We describe several experimental approaches, suitable for different kinds of patterns.

Functional groups. Once proteins have been assigned to a functional group based on the presence of a motif(s) in their sequence, the putative function of the proteins can be verified by expressing, isolating and purifying the protein(s) and testing for their activity using an appropriate biological assay. For example, if a protein is thought to be a particular enzyme, one could add the purified protein to a solution containing the enzyme's predicted substrate and measure the appearance of expected reaction products. To substantiate the involvement of a motif in this activity, various mutations could be introduced into the conserved sequence at the nucleotide level by site-directed mutagenesis and disruption of the protein's activity could be measured. Chimeric proteins, created by fusing a motif-containing functional domain to another protein sequence, can be assayed for acquisition of the function specified by the motif. Again, all manipulations of protein sequence are performed at the nucleotide level.

Binding sites. Where the motif is thought to be involved in the binding of a protein to a specific

DNA sequence, DNA footprinting assays could be performed. Labeled DNA is incubated with the putative DNA binding protein and then digested with DNase. The resulting DNA fragments are separated by gel electrophoresis. Regions bound by the protein are protected from DNase digestion and show up as blank areas in the gel compared to samples not incubated with the protein. In this way, proteins with DNA binding motifs can be confirmed.

It may be possible to detect proteins with DNA binding motifs on a large, genomic scale using a modification of DNA microarray technology. DNA (oligonucleotide) substrates of known sequence are applied to a slide and labeled test proteins are then passed over the slide and allowed to bind to DNA. Unbound proteins are removed by washing and bound proteins are visualized, perhaps using labeled antibodies. In addition to verifying the presence of a DNA binding motif in the protein, its cognate DNA recognition site can easily be identified.

Cellular location. If a motif specifies the cellular location of a protein, it can be fused to a reporter protein and its location identified by microscopy. [Geraghty et al., 1999] tagged putative peroxisomal proteins with the reporter protein, green fluorescent protein (GFP), and visualized the subsequent subcellular distribution of the fusion proteins by fluorescence microscopy.

Protein structure motifs. Motifs involved in protein structure can be verified once the three dimensional structure has been confirmed by x-ray crystallography.

Finding sequences containing motif. Analogous to an approach used in silico to detect specific patterns in genomic sequences, short oligonucleotide primers designed to contain a conserved motif can be used to screen a genome(s) for other sequences containing the same motif via the polymerase chain reaction (PCR). The presence of conserved motifs can be confirmed by sequencing and alignment of the PCR amplified fragments. Methods described above may substantiate the functional relatedness of these proteins.

Promoter sequences. Putative promoter sequences can be tested for their ability to initiate gene expression in vivo using reporter genes. DNA libraries containing fragments of a prokaryotic genome inserted upstream of promoterless luxAB or lacZ reporter genes are assayed for light production or β-galactosidase activity (blue colonies), respectively, indicative of the presence of a promoter sequence. Specific sequences required for promoter activity can be further pinpointed by removing nucleotides from the upstream region (either as a block of nucleotides by restriction enzyme digestion or by systematic removal of single nucleotides by exonuclease digestion) and determining if the reporter gene is still expressed. Although in theory, all constitutive promoters could be found by this method, inducible promoters that are activated by regulatory factors will of course require that the activating factor be added. Transcription factor binding sites in eukaryotic sequences can be verified by a similar method; however, because regulatory factors may not be well conserved among eukaryotes, an appropriate expression system must be chosen to ensure that all components of the transcriptional machinery are present. In addition to assaying for reporter gene expression, transcription and subsequent synthesis of the natural protein resulting from promoter activation by inducing factors can be assayed by Northern and Western hybridization, respectively [Roulet et al., 2000].

Alternatively, the binding of a nucleotide motif such as a transcription factor binding sequence can be appraised in vitro by gel shift assay. [Roulet et al., 2000] designed a series of oligonucleotides with variant sequences and measured their affinity for CTF/NFI transcription factors by such a means. Labeled oligonucleotides consisting of the consensus binding site were incubated with purified transcription factor and increasing amounts of variant unlabeled "competitor" oligonucleotides. Binding was detected by the presence of a complex of higher molecular weight than unbound oligonucleotides as indicated by a shift in position following gel electrophoresis. The relative binding affinities were estimated from the relative amounts of competitor oligonucleotides required to inhibit binding of the transcription factor to the labeled consensus oligonucleotides.

Binding affinity can also be measured using PCR. To confirm the identify of a promoter element that stimulates promoter activity through strong interactions with an RNA polymerase subunit, [Estrem et al., 1998] incubated promoter fragments with RNA polymerase under increasingly stringent conditions (i.e. decreasing concentrations of RNA polymerase and decreasing duration of incubation). Bound promoters were then selected by PCR and sequenced to corroborate the presence of the promoter element.

6 Success Stories

In this section we introduce two examples were usage of pattern finding tools helped researchers to make new discoveries. There are of course many such examples. The two examples chosen here illustrate, types of discoveries that can be made and how these discoveries can be verified by biological experiment.

6.1 Tuberculosis

Proteins secreted by Mycobacterium tuberculosis, the causative agent of tuberculosis, are often targets of immune responses by an infected host that can lead to protection from the disease. Thus, the identification and study of these secretory proteins is a first step to the design of more effective vaccines targeted against this dreadful pathogen.

Proteins are directed to the outer membrane of bacterial cells by a short N-terminal sequence of amino acids called a signal peptide. Following secretion, the signal peptide, which remains attached to the internal face of the membrane, is cleaved to release the protein to the external environment. Conserved features of both the signal peptide and cleavage site, such as length and amino acid composition, make them amenable to identification by sequence analysis. The recently released nucleotide sequence of the M. tuberculosis genome was screened for these conserved features and a number of putative secretory proteins were detected [Gomez et al., 2000].

That these proteins are secreted was confirmed by fusing the gene for the secretory protein to the E. coli gene encoding alkaline phosphatase (PhoA) that had its own signal peptide removed. Clones carrying the fusions were evaluated for extracellular alkaline phosphatase activity. Ninety percent of the predicted secretory proteins were confirmed. A disadvantage of using this computer-based approach is that it is limited to those proteins secreted via the general export pathway. Evidence suggests the existence of other as yet undefined mechanisms for protein secretion in M. tuberculosis.

6.2 Coiled coils in histidine kinases

Histidine kinases are important proteins involved the responses of a bacterium to its environment. These proteins "report" to the cytoplasm changes in the physico-chemical conditions of the external world, and lead to control of gene expression through regulation of kinase activity (via phosphorylation). The identification and manipulation of histidine kinases can have several applications, including the control of biofilms, persistent bacterial community structures formed in response to environmental signals, and in the design of treatments to control bacterial infections in animals and plants.

Histidine kinases are transmembrane proteins with three domains: a sensor domain which is in contact with the outside environment, a kinase domain which is in contact with the cytoplasm, and a linker region which traverses the membrane and connects the two domains. [Singh et al., 1998] wanted to identify motifs common to histidine kinases that would allow rapid detection of these proteins in different bacteria. Studies of other kinases had revealed the presence of coiled coil domains involved in phosphorylation.

The authors utilized an iterative learning algorithm to detect potential coiled coils in histidine kinases. They used established sequence patterns for known coiled coil proteins and known histidine kinases. The assumption was that, despite limited sequence homology in the coiled coil region, there was likely to be some structural and mechanistic similarity among the kinases and therefore patterns that might not be readily apparent from multiple sequence alignment.

Coiled coil motifs were found to occur broadly in histidine kinases, appearing in all reported eukaryotic histidine kinases as well as in most of those from the bacteria E. coli and Salmonella typhimurium. The results were verified biologically by constructing chimeric proteins in which coiled coil sequences from one histidine kinase were fused to the cytoplasmic domain of another histidine kinase, and testing these for phosphorylation activity, which is indicative of kinase activity. The frequency of the motif was surprising, given the high diversity of histidine kinase primary sequences. This suggests that coiled coils are important for the function of histidine kinases.

6.3 Conclusion

Although a higher profile objective of pattern discovery is to annotate proteins in entire genomes, the examples above illustrate the usefulness of identifying specific motifs in smaller scale experiments. Discovering related proteins or understanding the function and regulation of a protein using biological techniques can be onerous, often involving much trial and error. Pattern detection algorithms will prove to be important tools for molecular biologists working in "wet labs", allowing them to design bet-

ter experiments.

7 Conclusion

Pattern discovery is an important area of bioinformatics. The algorithms for pattern discovery use wide range of computer science techniques, ranging from exhaustive search, elaborate pruning techniques, efficient data structures, to machine learning learning methods and iterative heuristics.

The tools developed by computer scientists are today commonly used in many biological laboratories. They are important to handle large scale data, for example in annotation of newly sequenced genomes, and organization of proteins into families of related sequences. They are also important in smaller scale projects, because they can be used to detect possible sites of interest and assign putative structure or function to proteins. Thus they can be used to guide biological experiments, decreasing the time and money spent in discovering new biological knowledge.

Many patterns are hard to find, because they are very sparse and/or variable. Also they might not be specific enough to distinguish them from random noise only based on local sequence information. Small conserved regions, that were before interspersed through protein, and thus hard to find, can occur close to each other in secondary or tertiary structure conformation. Also DNA in eukaryotic nucleus is closely packed in a kind of tertiary structure and this structure can have influence on position of regulatory element binding sites. Therefore it seems that the future approaches to pattern discovery will also use other information available about the sequences, such as three dimensional structure.

References

[Atteson, 1998] Atteson, K. (1998). Calculating the exact probability of language-like patterns in biomolecular sequences. In *Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 17–24.

[Bailey and Elkan, 1994] Bailey, T. L. and Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 28–36.

[Bailey and Elkan, 1995] Bailey, T. L. and Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1/2):51–80.

[Barker et al., 1996] Barker, W. C., Pfeiffer, F., and George, D. G. (1996). Superfamily classification in PIR-International Protein Sequence Database. *Methods in Enzymology*, 266:59–71.

[Benson, 1999] Benson, G. (1999). Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27(2):573–580.

[Blanchette et al., 2000] Blanchette, M., Schwikowski, B., and Tompa, M. (2000). An exact algorithm to identify motifs in orthologous sequences from multiple species. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 37–45.

[Bornberg-Bauer et al., 1998] Bornberg-Bauer, E., Rivals, E., and Vingron, M. (1998). Computational approaches to identify leucine zippers. *Nucleic Acids Research*, 26(11):2740–2746.

[Brazma et al., 1998] Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998). Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5(2):279–305.

[Califano, 2000] Califano, A. (2000). SPLASH: structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16(4):341–347.

[Cowan, 1991] Cowan, R. (1991). Expected frequencies of DNA patterns using Whittle's formula. *Journal of Applied Probability*, 28(4):886–892.

[Coward and Drablos, 1998] Coward, E. and Drablos, F. (1998). Detecting periodic patterns in biological sequences. *Bioinformatics*, 14(6):498–507.

[Dorohonceanu and Nevill-Manning, 2000] Dorohonceanu, B. and Nevill-Manning, C. G. (2000). Accelerating protein classification using suffix trees. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 128–133.

[Durbin et al., 1998] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis*. Cambridge University Press.

[Ermolaeva et al., 2000] Ermolaeva, M. D., Khalak, H. G., White, O., Smith, H. O., and Salzberg, S. L. (2000). Prediction of transcription terminators in bacterial genomes. *Journal of Molecular Biology*, 301(1):27–33.

[Estrem et al., 1998] Estrem, S. T., Gaal, T., Ross, W., and Gourse, R. L. (1998). Identification of an UP element consensus sequence for bacterial promoters. *Proceedings of the National Academy of Sciences of the United States of America*, 95(17):9761–9766.

[Fickett and Hatzigeorgiou, 1997] Fickett, J. W. and Hatzigeorgiou, A. G. (1997). Eukaryotic promoter recognition. *Genome Research*, 7(9):861–868.

[Gelfand and Koonin, 1997] Gelfand, M. S. and Koonin, E. V. (1997). Avoidance of palindromic words in bacterial and archaeal genomes: a close connection with restriction enzymes. *Nucleic Acids Research*, 25(12):2430–2439. Published erratum appears in Nucleic Acids Research, 25(24):5135-5136.

[Gelfand et al., 2000] Gelfand, M. S., Koonin, E. V., and Mironov, A. A. (2000). Prediction of transcription regulatory sites in Archaea by a comparative genomic approach. *Nucleic Acids Research*, 28(3):695–705.

[Geraghty et al., 1999] Geraghty, M. T., Bassett, D., Morrell, J. C., Gatto Jr., G. J., Bai, J., Geisbrecht, B. V., Hieter, P., and Gould, S. J. (1999). Detecting patterns of protein distribution and gene expression in silico. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2937–2942.

[Gomez et al., 2000] Gomez, M., Johnson, S., and Gennaro, M. L. (2000). Identification of secreted proteins of Mycobacterium tuberculosis by a bioinformatic approach. *Infection and Immunity*, 68(4):2323–2327.

[Gorodkin et al., 1997a] Gorodkin, J., Heyer, L. J., Brunak, S., and Stormo, G. D. (1997a). Displaying the information contents of structural RNA alignments: the structure logos. *Computer Applications in the Biosciences*, 13(6):583–586.

[Gorodkin et al., 1997b] Gorodkin, J., Heyer, L. J., and Stormo, G. D. (1997b). Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Research*, 25(18):3724–3732.

[Grundy et al., 1997] Grundy, W. N., Bailey, T. L., Elkan, C. P., and Baker, M. E. (1997). Meta-MEME: motif-based hidden Markov models of protein families. *Computer Applications in the Biosciences*, 13(4):397–406.

[Hu et al., 2000] Hu, Y. J., Sandmeyer, S., McLaughlin, C., and Kibler, D. (2000). Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics*, 16(3):222–222.

[Hudak and Mcclure, 1999] Hudak, J. and Mcclure, M. A. (1999). A comparative analysis of computational motif-detection methods. *Pacific Symposium on Biocomputing (PSB)*, pages 138–139.

[Hughes et al., 2000] Hughes, J. D., Estep, P. W., Tavazoie, S., and Church, G. M. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *Journal of Molecular Biology*, 296(5):1205–1214.

[Hughey and Krogh, 1996] Hughey, R. and Krogh, A. (1996). Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12(2):95–107.

[Ison et al., 2000] Ison, J. C., Blades, M. J., Bleasby, A. J., Daniel, S. C., Parish, J. H., and Findlay, J. B. (2000). Key residues approach to the definition of protein families and analys is of sparse family signatures. *Proteins*, 40(2):330–331.

[Jonassen, 1996] Jonassen, I. (1996). Efficient discovery of conserved patterns using a pattern graph. Technical Report 118, Department of Informatics, University of Bergen, Norway.

[Kochetov et al., 1999] Kochetov, A. V., Ponomarenko, M. P., Frolov, A. S., Kisselev, L. L., and Kolchanov, N. A. (1999). Prediction of eukaryotic mRNA translational properties. *Bioinformatics*, 15(7-8):704–712.

[Kono and Sarai, 1999] Kono, H. and Sarai, A. (1999). Structure-based prediction of DNA target sites by regulatory proteins. *Proteins*, 35(1):114–121.

[Krogh et al., 1994] Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1501.

[Lawrence et al., 1993] Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214.

[Lawrence and Reilly, 1990] Lawrence, C. E. and Reilly, A. A. (1990). An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7(1):41–51.

[Li et al., 1999] Li, M., Ma, B., and Wang, L. (1999). Finding Similar Regions in Many Strings. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing (STOC)*, pages 473–482, Atlanta.

[Liang et al., 2000] Liang, C., Li, M., and Ma, B. (2000). COPIA: A New Software for Finding Consensus Patterns in Protein Sequences. To appear.

[Linial et al., 1997] Linial, M., Linial, N., Tishby, N., and Yona, G. (1997). Global self-organization of all known protein sequences reveals inherent biological signatures. *Journal of Molecular Biology*, 268(2):539–546.

[Liu et al., 1995] Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1995). Bayesian Models for Multiple Local Sequence Alignment and Gibbs Sampling Strategies. *Journal of the American Statistical Association*, 90(432):1156–1170.

[Mcclure and Kowalski, 1999] Mcclure, M. A. and Kowalski, J. (1999). The effects of ordered-series-of-motifs anchoring and sub-class modeling on the generation of HMMs representing highly divergent protein sequences. In *Pacific Symposium on Biocomputing (PSB)*, pages 162–170.

[Mironov et al., 1999] Mironov, A. A., Koonin, E. V., Roytberg, M. A., and Gelfand, M. S. (1999). Computer analysis of transcription regulatory patterns in completely sequenced bacterial genomes. *Nucleic Acids Research*, 27(14):2981–2989.

[Neuwald et al., 1997] Neuwald, A. F., Liu, J. S., Lipman, D. J., and Lawrence, C. E. (1997). Extracting protein alignment models from the sequence database. *Nucleic Acids Research*, 25(9):1665–1667.

[Nevill-Manning et al., 1998] Nevill-Manning, C. G., Wu, T. D., and Brutlag, D. L. (1998). Highly specific protein sequence motifs for genome analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 95(11):5865–5871.

[Nicodème et al., 1999] Nicodème, P., Salvy, B., and Flajolet, P. (1999). Motif statistics. In Nesetril, J., editor, *Algorithms - ESA '99, 7th Annual European Symposium*, volume 1643 of *Lecture Notes in Computer Science*, pages 194–211, Prague. Springer.

[Parida et al., 2000] Parida, L., Rigoutsos, I., Floratos, A., Platt, D., and Gao, Y. (2000). Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and an efficient polynomial time algorithm. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 297–308.

[Pedersen et al., 1999] Pedersen, A. G., Baldi, P., Chauvin, Y., and Brunak, S. (1999). The biology of eukaryotic promoter prediction–a review. *Computers and Chemistry*, 23(3-4):191–207.

[Peltier et al., 2000] Peltier, J. B., Friso, G., Kalume, D. E., Roepstorff, P., Nilsson, F., Adamska, I., and van Wijk, K. J. (2000). Proteomics of the chloroplast: systematic identification and targeting analysis of lumenal and peripheral thylakoid proteins. *Plant Cell*, 12(3):319–321.

[Pesole et al., 2000] Pesole, G., Liuni, S., and D'Souza, M. (2000). PatSearch: a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics*, 16(5):439–440.

[Pevzner and Sze, 2000] Pevzner, P. A. and Sze, S. H. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 269–278.

[Rigoutsos and Floratos, 1998a] Rigoutsos, I. and Floratos, A. (1998a). Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67. Published erratum appears in Bioinformatics, 14(2):229.

[Rigoutsos and Floratos, 1998b] Rigoutsos, I. and Floratos, A. (1998b). Motif discovery without alignment or enumeration (extended abstract).

In *Proceedings of the second annual international conference on Computational molecular biology (RECOMB)*, pages 221 – 227, New York.

[Rigoutsos et al., 1999] Rigoutsos, I., Floratos, A., Ouzounis, C., Gao, Y., and Parida, L. (1999). Dictionary building via unsupervised hierarchical motif discovery in the sequence space of natural proteins. *Proteins*, 37(2):264–267.

[Rigoutsos et al., 2000] Rigoutsos, I., Floratos, A., Parida, L., Gao, Y., and Platt, D. (2000). The emergence of pattern discovery techniques in computational biology. *Metabolic Engineering*, 2(3):159–167.

[Roulet et al., 2000] Roulet, E., Bucher, P., Schneider, R., Wingender, E., Dusserre, Y., Werner, T., and Mermod, N. (2000). Experimental analysis and computer prediction of CTF/NFI transcription factor DNA binding sites. *Journal of Molecular Biology*, 297(4):833–838.

[Savoie et al., 1999] Savoie, C. J., Kamikawaji, N., Sasazuki, T., and Kuhara, S. (1999). Use of BONSAI decision trees for the identification of potential MHC class I peptide epitope motifs. In *Pacific Symposium on Biocomputing (PSB)*, pages 182–189.

[Schneider and Stephens, 1990] Schneider, T. D. and Stephens, R. M. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research*, 18(20):6097–6100.

[Singh et al., 1998] Singh, M., Berger, B., Kim, P. S., Berger, J. M., and Cochran, A. G. (1998). Computational learning reveals coiled coil-like motifs in histidine kinase linker domains. *Proceedings of the National Academy of Sciences of the United States of America*, 95(6):2738–2743.

[Sinha and Tompa, 2000] Sinha, S. and Tompa, M. (2000). A statistical method for finding transcription factor binding sites. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 344–344.

[Smith et al., 1990] Smith, H. O., Annau, T. M., and Chandrasegaran, S. (1990). Finding sequence motifs in groups of functionally related proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 87(2):826–830.

[Tavazoie et al., 1999] Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J., and Church, G. M. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285.

[Tompa, 1999] 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)*, pages 262–271.

[van Helden et al., 1998] van Helden, J., Andre, B., and Collado-Vides, J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5):827–832.

[van Helden et al., 2000] van Helden, J., del Olmo, M., and Perez-Ortin, J. E. (2000). Statistical analysis of yeast genomic downstream sequences reveals putative polyadenylation signals. *Nucleic Acids Research*, 28(4):1000–1010.

[Whittle, 1955] Whittle, P. (1955). Some distribution and moment formulae for the Markov chain. *Journal of the Royal Statistical Society. Series B.*, 17:235–242.

[Yada et al., 1997] Yada, T., Totoki, Y., Ishii, T., and Nakai, K. (1997). Functional prediction of B. subtilis genes from their regulatory sequences. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 354–357.

[Zhang, 1998] Zhang, M. Q. (1998). Statistical features of human exons and their flanking regions. *Human Molecular Genetics*, 7(5):919–922.

A Publicly Available Software Tools

This appendix gives a short overview of software tools, which are available for pattern discovery. The list is by no means complete, we included only tools which were accessible to us as of December 2000. There is no particular order in the list.

TEIRESIAS

IBM Research, Computational Biology Center
`http://www.research.ibm.com/bioinformatics/home.html`
Terms of use: 90 day academic trial

Availability:

- **Download:** `http://www.research.ibm.com/bioinformatics/download.phtml.html`
  Distribution: binary (linux, aix, win32, solaris)
- **On-line:** `http://www.research.ibm.com/bioinformatics/teiresias.html`

  Limits size of input sequences to 30Kb. Allows finding occurences of found patterns in SWISS-PROT.

TEIRESIAS finds all patterns with given density in unaligned set of sequences, which occur in at least specified number of sequences. TEIRESIAS does not score these patterns. It is based on exhaustive search for short patterns and on combining short patterns to longer ones.

Example of found pattern:

```
G.GA..GG.A
```

Splash: Structural Pattern Localization Analysis by Sequential Histogram

IBM Research, Computational Biology Center
`http://www.research.ibm.com/splash/`
Terms of use: 90 day academic trial

Availability:

- **Download:** `http://www.research.ibm.com/splash/Download/agreement.htm`
  Distribution: binary (aix, win32, solaris)

Splash is a deterministic pattern discovery algorithm, which can find sparse amino or nucleic acid patterns matching identically or similarly in a set of protein or DNA sequences. Sparse patterns of any length, up to the size of the input sequence, can be discovered without significant loss in performances.

Example of found pattern:

```
CTC..TCA..TCTT
```

Pratt

Department of Informatics, University of Bergen
`http://www.ii.uib.no/~inge/Pratt.html`
Terms of use: free

Availability:

- **Download:** `ftp://ftp.ii.uib.no/pub/bio/Pratt/`
  Distribution: source (unix C)
- **On-line:** `http://www.ii.uib.no/~inge/Pratt.html`

The Pratt program is able to discover patterns conserved in sets of unaligned protein sequences. It allows the user to define a class of patterns (e.g., the degree of ambiguity allowed and the length and number of gaps), and the method is then guaranteed to find the conserved patterns in this class scoring highest according to a significance measure defined. Pratt uses carefully pruned exhaustive search method.

Example of found pattern:

```
C-T-x(4,5)-T-G-x(4)-G-x(1,2)-A-x(4)-G
```

MEME: Multiple EM for Motif Elicitation

UCSD Computer Science and Engineering
`http://meme.sdsc.edu/`
Terms of use: academic free, commercial available

Availability:

- **Download:** `ftp://ftp.sdsc.edu/pub/sdsc/biology/meme/`
  Distribution: source (ansi C)
- **On-line:** `http://meme.sdsc.edu/meme/website/meme.html`
  Limits size of input sequences to 100Kb.

MEME is a tool for discovering motifs in a group of related DNA or protein sequences. MEME represents motifs as position weight matrices. Individual MEME motifs do not contain gaps. Patterns with variable-length gaps are split by MEME into two or more separate motifs. The algorithm is based on expectation maximization tehnique. MEME can report patterns as PWM, log-odds matrix or in BLOCKS format.

Example of found pattern:

```
Multilevel        CCGGGAGTCCAGCCCCGGCCTG
consensus          AA CGCGA TT AGA   G
sequence            T  T         T
```

Meta-MEME

`http://metameme.sdsc.edu/`
Terms of use: academic free, commercial available

Availability:

- **Download:** `http://metameme.sdsc.edu/mhmm-download.html`
  Distribution: source (ansi C), binary (solaris, DEC)
- **On-line:** `http://metameme.sdsc.edu/cgi-bin/submit-verify.cgi`

  Allows running of estimated model against major databases.

Meta-MEME is a software toolkit for building and using motif-based hidden Markov models of biological sequences. The input to Meta-MEME is a set of similar DNA or protein sequences, as well as a set of motif models discovered by MEME. Meta-MEME combines these models into a single, motif-based hidden Markov model and uses this model to produce a multiple alignment of the original set of sequences and to search a sequence database for homologs.

SAM: Sequence Alignment and Modeling Software System

Baskin Center for Computer Engineering and Science
`http://www.cse.ucsc.edu/research/compbio/sam.html`
Terms of use: academic free, commercial available

Availability:

- **Download:** `http://www.cse.ucsc.edu/research/compbio/sam2src/`
  Distribution: binary (DEC, Linux, SGI, solaris)
- **On-line:** `http://www.cse.ucsc.edu/research/compbio/HMM-apps/tuneup-alignment.html`

  Allows use of SAM for multiple sequence allignment. No access to intermediate results (e.g. model built, etc.).
- **On-line:** `http://bioweb.pasteur.fr/seqanal/motif/sam-uk.html`

  Allows fairly complex analysis of set of sequences, starting with building and training HMM model. All intermediate results are stored on server and can be easily downloaded.

SAM is a set of software tools for building and using linear HMMs. Each node has a match state, insert state and delete state. Each sequence uses a series of these states to traverse the model from start to end. In many ways, these models correspond to profiles. For set of DNA or protein sequences, SAM allows to build HMM model, train it and use it for finding other similar sequences in databases. From HMM model itself, pattern in more conventional form (PWM) can be extracted.

Gibbs Motif Sampler

Biometrics Laboratory of Wadsworth Center
`http://bayesweb.wadsworth.org/gibbs/gibbs.html`

Availability:

- **On-line:** `http://bayesweb.wadsworth.org/gibbs/gibbs.html`

  Limits size of the input sequences to 10Kb.

Gibbs Motif Sampler is a program, which can find patterns in set of DNA or protein sequences. The program is an implementation of Gibbs sampling method. Motifs are represented by position weight matrices, program also perform local multiple sequence alignment.

Example of found pattern:

```
Motif probability model
-----------------------------------------
Pos. #    a     t     c     g
-----------------------------------------
   1 |  0.138 0.024 0.020 0.818
   2 |  0.024 0.024 0.020 0.932

   4 |  0.592 0.024 0.361 0.023
   5 |  0.024 0.138 0.815 0.023

   7 |  0.024 0.024 0.020 0.932
   8 |  0.024 0.024 0.020 0.932
   9 |  0.138 0.138 0.020 0.705
  10 |  0.933 0.024 0.020 0.023
  11 |  0.479 0.024 0.020 0.478
  12 |  0.819 0.024 0.020 0.137
```

Gibbs Sampler

National Center for Biotechnology Information
Terms of use: academic free

Availability:

- **Download:** `ftp://ncbi.nlm.nih.gov/pub/neuwald/gibbs9_95/`
  Distribution: source (ansi C)

- **On-line:** `http://copan.cifn.unam.mx/~jvanheld/rsa-tools/demo.gibbs.html`

The Gibbs sampler detects motifs shared by a set of functionally related sequences. The program stochastically examines candidate alignments in an effort to find the best alignment as measured by the maximum a posteriori (MAP) log-likelihood ratio. Another implementation of Gibbs sampling method.

Consensus

Department of Molecular, Cellular, and Developmental Biology, University of Colorado
Terms of use: non-commercial free, commercial available

Availability:

- **Download:** `ftp://beagle.colorado.edu/pub/consensus/`
  Distribution: source (gcc)
- **On-line:** `http://bioweb.pasteur.fr/seqanal/interfaces/consensus-simple.html`

The CONSENSUS programs are a collection of programs for determining and analyzing DNA and protein patterns describing functional elements. Patterns are described by position weight matrices.

Example of found pattern:

```
# MATRIX 1
# number of sequences = 6
# unadjusted information = 11.2299
# sample size adjusted information = 8.17697
# ln(p-value) = -43.9126   p-value = 8.49177E-20
# ln(expected frequency) = -4.96433   expected frequency = 0.00698264
#   1|2   :   2/303   GGCACAGGGA
#   2|4   :   3/162   GCACCAGGGA
#   3|5   :   4/76    GGCACAGGGA
#   4|3   :   5/244   GGACCAGGGA
#   5|6   :   6/372   GCCCCAGGGT
#   6|1   :   7/299   CGCCCAGGGA
A |   0   0   2   2   0   6   0   0   0   5
C |   1   2   4   4   6   0   0   0   0   0
G |   5   4   0   0   0   0   6   6   6   0
T |   0   0   0   0   0   0   0   0   0   1
```

COPIA: COnsensus Pattern Identification and Analysis

Availability:

- **On-line:** `http://dna.cs.ucsb.edu/copia/copia_submit.html`

Software finds consensus patterns in protein sequences.

AlignACE: Aligns Nucleic Acid Conserved Elements

`http://arep.med.harvard.edu/mrnadata/`
Terms of use: academic free

Availability:

- **Download:** `http://arep.med.harvard.edu/mrnadata/mrnasoft.html`
  Distribution: binary (linux, win32)

AlignACE is a program which finds sequence elements conserved in a set of DNA sequences. It uses a Gibbs sampling strategy. An iterative masking procedure is used to allow multiple distinct motifs to be found within a single data set. The output are motifs (aligned sites) sorted according to MAP score.

ASSET: Aligned Segment Statistical Evaluation Tool

Terms of use: free

Availability:

- **Download:** `ftp://ncbi.nlm.nih.gov/pub/neuwald/asset/`
  Distribution: source (unix C)

The Asset program produces scan file of the locally aligned segment blocks. It is possible to specify the percentage of sequences in the input file that are required to contain a motif before the corresponding motif block can be included in the scan file.

SDISCOVER

`http://www.cis.njit.edu/~discdb/`
Terms of use: academic free

Availability:

- **Download:** `http://www.cis.njit.edu/~discdb/`
  Distribution: source (unix C)

- **On-line:** `http://www.cis.njit.edu/~discdb/`

SDISCOVER takes a set of related proteins and produces a collection of active motifs in the set. Algorithm first finds candidate segments among a small sample of sequences and then combines the segments to form candidate motifs and checks, whether these motifs meet specified requirements.

Blocks Maker

Fred Hutchinson Cancer Research Center
`http://www.blocks.fhcrc.org/blockmkr/`
Terms of use: non-profit free

Availability:

- **Download:** `ftp://ncbi.nlm.nih.gov/repository/blocks/unix/protomat/`
  Distribution: source (unix C), binary (linux)

- **On-line:** `http://www.blocks.fhcrc.org/blockmkr/`

  Limits number of sequences to 250 and total length of sequences to 100Kb.

The BLOCK MAKER SERVER finds blocks in a group of related protein sequences. Blocks are short multiply aligned ungapped segments corresponding to the most highly conserved regions of proteins. Typically, a group of proteins has more than one region in common and their relationship is represented as a series of blocks separated by unaligned regions. Block Maker will run PROTOMAT twice, first using Smith's MOTIF and second using a modification of Lawrence's Gibbs sampler as motif-finding algorithms, and then it will report both sets of blocks.

Example of found pattern:

```
      unknownA, width = 42
HBP1_CASGL     8 ALLKQSWEVLKQNIPAHSLRLFALIIEAAPESKYVFSFLKDS
HBP2_CASGL    15 ALVVKSWSAMKPNAGELGLKFFLKIFEIAPSAQKLFSFLKDS
HBPL_PARAD    15 ALVVKAWAVMKKNSAELGLQFFLKIFEIAPSAKNLFSYLKDS
HBPL_TRETO    16 ALVVKSWAVMKKNSAELGLKFFLKIFEIAPSAKNLFSYLKDS
LGB1_LUPLU     9 ALVKSSFEEFNANIPKNTHRFFTLVLEIAPGAKDLFSFLKGS
LGB1_MEDSA     9 ALVNSSWEAFKQNLPRYSVFFYTVVLEKAPAAKGLFSFLKNS
LGB1_MEDTR     9 ALVNSSYEAFKQNLSGYSVFFYTVILEKAPAAKGLFSFLKDS
  LGB1_PEA     7 EALVNSSSEFKQNLPGYSILFYTIVLEKAPAAKGLFSFLKDT
```

---

eMOTIF

The Brutlag Bioinformatics Group, Stanford University
`http://motif.stanford.edu/emotif/`

Availability:

- **On-line:** `http://motif.stanford.edu/emotif/emotif-maker.html`

eMOTIF finds significant similarities in aligned set of sequences.

Example of found pattern:

`[ilv]..[iv]....g[filvy].....f...[fy].....[ast]p...`

B Databases of Patterns and Motifs

This appendix gives a short overview of databases related to pattern discovery. They can be divided into two groups:

- **Databases of protein domains** contain individual functional or structural domains usually characterized by pattern or local allignment.
- **Databases of transcription factors** contain information concerning transcription factors themselves and their binding sites in regulatory regions.

B.1 Databases of protein domains

PROSITE

Swiss Institute of Bioinformatics, Expert Protein Analysis System (ExPASy)
`http://www.expasy.ch/prosite/`
Terms of use: academic free, commercial available
Status (as of Dec. 2000): Release 16.30, of 09-Dec-2000 contains 1076 documentation entries that describe 1455 different patterns, rules and profiles/matrices.

Availability:

- **Download:** `ftp://ftp.expasy.ch/databases/prosite`

  Last release not avaliable.

- **On-line search:** `http://www.expasy.ch/prosite/`

  Search by text, author, etc, plus scan sequence against PROSITE and search pattern/profile agains SWISS-PROT.

PROSITE is a database of protein families and domains. It consists of biologically significant sites, patterns and profiles that help to reliably identify to which known protein family (if any) a new sequence belongs. Patterns are deterministic patterns with ambiguous characters and flexible gaps. profiles are extended position weight matrices with gaps. Entries include description of the family/domain, pattern, list of true hits, false hits, false misses, etc.

Example of pattern:

```
Peroxidases proximal heme-ligand signature:
 [DET]-[LIVMTA]-x(2)-[LIVM]-[LIVMSTAG]-[SAG]-[LIVMSTAG]-H-[STA]-[LIVMFY].

Src  homology  3 (SH3) domain profile (only parts are shown):
 /GENERAL_SPEC: ALPHABET='ABCDEFGHIKLMNPQRSTVWYZ'; LENGTH=62;
 /DISJOINT: DEFINITION=PROTECT; N1=6; N2=57;
 /NORMALIZATION: MODE=1; FUNCTION=LINEAR; R1=0.9383; R2=0.016376; TEXT='OrigScore';
 /CUT_OFF: LEVEL=0; SCORE=462; N_SCORE=8.5; MODE=1;
 /DEFAULT: D=-20; I=-20; B1=-60; E1=-60; MI=-105; MD=-105; IM=-105; DM=-105;
 /I: B1=0; BI=-105; BD=-105;
 /M: SY='P'; M=-4,-8,-26,-7,-1,-19,-11,-11,-15,-3,-16,-9,-6,6,-3,-6,-2,-3,-14,-25,-15,-3;
 /M: SY='E'; M=-5,-3,-25,0,4,-18,-8,-9,-17,-5,-15,-11,-4,-4,-3,-8,-1,-6,-14,-26,-14,0;
 /M: M=-5,-8,-25,-8,-4,-17,-4,-14,-14,-5,-13,-8,-7,-6,-7,-6,-4,-7,-11,-24,-15,-6;
```

```
...(skipped 9 lines)...
 /M: SY='E'; M=-6,5,-23,6,11,-22,-15,-3,-19,2,-18,-10,2,-7,6,-3,0,-4,-16,-28,-14,8;
 /I: I=-6; MI=0; IM=0; B1=-50;
 /M: SY='A'; M=16,-9,-20,-12,-5,-22,-2,-16,-17,-4,-18,-12,-6,2,-6,-8,5,-1,-11,-24,-20,-7;
 /M: SY='E'; M=-4,-2,-25,-3,8,-23,-13,-6,-19,5,-19,-10,1,-7,7,6,2,-2,-16,-26,-15,6;
...(skipped 52 lines)...
 /M: SY='S'; M=-1,-4,-21,-5,-4,-18,-9,-6,-14,-7,-14,-10,-1,-7,-4,-5,4,-1,-11,-29,-14,-5;
 /I: E1=0; IE=-105; DE=-105;
```

BLOCKS

Fred Hutchinson Cancer Research Center in Seattle, Washington, USA
`http://blocks.fhcrc.org/blocks/`
Terms of use: free
Status (as of Dec. 2000): Version 12.0, June 2000, 4071 blocks representing 998 groups documented in InterPro 1.0

Availability:

- **Download:** `ftp://ncbi.nlm.nih.gov/repository/blocks/unix/`
- **On-line search:** `http://blocks.fhcrc.org/blocks/`

  Search by keyword, ID, scan sequence agains blocks, compare blocks, display sequence logos, links to other databases.

The blocks for the Blocks Database are made automatically by looking for the most highly conserved regions in groups of proteins documented in the Prosite Database. Blocks are created by PROTOMAT system using the MOTIF algorithm as implemented in Block Maker. Patterns are gapless alignemnts of several segments of proteins, highly similar segments are grouped together eabd each segment is followed by score how much it differes from other blocks.

Example of pattern:

```
Block IPB000889B
ID   GSHPx; BLOCK
AC   IPB000889A; distance from previous block=(11,96)
DE   Glutathione peroxidase
BL   NAT;  width=25; seqs=64; 99.5%=1220; strength=1316
GSHY_ARATH|P52032  (  89) GKDVALNKFKGKVMLIVNVASRCGL  27
GSHZ_CITSI|Q06652  (  19) GQDVDLSIYKGKLLLIVNVASQCGL  32
GSHZ_HELAN|O23970  (  19) GNDVDLSVYKGKVVLIVNVASKCGL  20
GSHY_HELAN|O23968  (  32) GQDVELSKYKGKVLLIVNVASQCGF  22
GSHZ_NICSY|P30708  (  21) GNDVDLSIYKGKVLIIVNVASQCGL  18
GSHC_CAEEL|O02621  (  14) GDDVSLSDYKGKVLIIVNVASQCGL  21
GSHD_CAEEL|O62327  (  14) GEDTPLSNYQGKVLIIVNVASQCGL  33
O19985             (  89) GKDVALNKFKGKVMLIVNVASRCGL  27
O49069             (  21) GNDVDLSIYKGKVLIIVNVASQCGL  18
O65156             (  97) GKDVSLSKFKGKVLLIVNVASRCGL  19
O48646             (  20) GNDVDLSIYKGKVLLIVNVASQCGL  17
O22850             (  58) GKDVSLSKFTGKVLLIVNVASKCGL  28
O24296             (  89) KKDVSLSKFKGKVLLIVNVASRCGL  26
O23814             (  22) GNDVDLSIYKGKVLLIVNVASQCGL  17
O24031             (  21) GKDVDLSIYKGKVLIIVNVASQCGL  18
O81717             (  89) GKDVALNKFKGKVMLIVNVASRCGL  27
```

```
GSHU_CAEEL|Q95003  (  51) GEYTDLSQYRGKVILLVNVATFCAY  43
Q93204             (  50) GEYTDLSQYRGQVLLMVNVATFCAY  54
...(plus several more clusters)...
```

PRINTS: Protein Fingerprints database

University of Manchester (UK), Bioinformatics unit
`http://www.bioinf.man.ac.uk/dbbrowser/PRINTS/`
Terms of use: public domain database
Status (as of Dec. 2000): Release 28.0, September 25, 2000 contains 1410 entries, encoding 8550 individual motifs.

Availability:

- **Download:** `ftp://bioinf.man.ac.uk/pub/prints/`
- **On-line search:** `http://www.bioinf.man.ac.uk/dbbrowser/PRINTS/`
  Search by keyword, ID, text, etc., sequence against fingerprints, provides graphical display of search results.

PRINTS is a compendium of protein fingerprints. A composite or multiple-motif fingerprint contains a number of aligned motifs taken from different parts of a multiple alignment. Discrimination power is increased in these systems because the recognition of individual elements of the fingerprint is mutually conditional. True family members are then easy to identify by virtue of possessing all elements of the fingerprint, while subfamily members may be identified by possessing only part of it. Fingerprints are generated automatically based on multiple sequence alignment. Entries are annotated.

Example of pattern:

```
SH3DOMAIN2         Length of motif = 16   Motif number = 2
SH3 domain motif II - 9
                                   PCODE         ST     INT
DDLSFHKGEKFQILNS                   FYN_HUMAN     98     3
DDLSFHKGEKFQILNS                      Q62844     99     3
DDLSFHKGEKFQILNS                      Q16248     99     3
DDLSFHKGEKFQILNS                   FYN_MOUSE     98     3
DDLSFQKGEKFQILNS                   FYN_XENLA     98     3
DDLSFKKGERFQIINN                   YES_CHICK    106     3
TDLSFKKGERLQIVNN                  SRC1_XENLA     96     3
DDLSFHKGEKFQILNS                   FYN_CHICK     98     3
DDLSFKGGERFQIINN                      Q85466    382     3
DDLSFKGGERFQIINN                   YES_AVISY     98     3
DDLSFRKGERFQILNS                   FYN_XIPHE     98     3
DDLSFRKGDRFQIINN                   YES_XIPHE    109     3
...(plus many more alignment lines like these)...
```

PFAM: Protein families database of alignments and HMMs

Washington University in St. Luis, and Sanger Centre, Cambride, UK, and Center for Genomics Research, Karolinska Institutet, Sweden
`http://pfam.wustl.edu/`
Terms of use: GNU Library general public license
Status (as of Dec. 2000): Version 5.5 (Sept 2000) contains alignments and models for 2478 protein families

Availability:

- **Download:** `ftp://ftp.genetics.wustl.edu/pub/eddy/pfam-5.5/`
- **On-line search:** `http://pfam.wustl.edu/hmmsearch.shtml`

  Search by keyword, text, sequence against database, graphical display of domains in proteins.

Pfam is a database of multiple alignments of protein domains or conserved protein regions. Profile hidden Markov models (profile HMMs) built from the Pfam alignments can be very useful for automatically recognizing that a new protein belongs to an existing protein family, even if the homology is weak. Pfam-A are fairly accurate human crafted multiple alignments whereas Pfam-B is an automatic clustering of the rest of SWISS-PROT.

Example of pattern:

SH3 domain (part of the alignment)

```
ABL1_HUMAN/64-119     NLFVALYDFVASG..DNTLSIT.KGEKLRVLGYNHNGE..WCEAQTKNG.....QGWVPSNYITPV
ABL_DROME/207-263     QLFVALYDFQAGG..ENQLSLK.KGEQVRILSYNKSGE..WCEAHSDSGN....VGWVPSNYVTPL
MYS3_YEAST/1124-1181  PKFEAAYDFPGSG.SSSELPLK.KGDIVFISRDEPSG...WSLAKLLDGS...KEGWVPTAYMTPY
CSK_CHICK/12-68       TECIAKYNFHGTA..EQDLPFS.KGDVLTIVAVTKDPN..WYKAKNKV..G..REGIIPANYVQKR
DRK_DROME/1-56        MEAIAKHDFSATA..DDELSFR.KTQILKILNMEDDSN..WYRAELDGK.....EGLIPSNYIEMK
SEM5_CAEEL/1-56       MEAVAEHDFQAGS..PDELSFK.RGNTLKVLNKDEDPH..WYKAELDGN.....EGFIPSNYIRMT
SPCA_DROME/973-1027   ECVVALYDYTEKS..PREVSMK.KGDVLTLLNSNNK.D..WWKVEVN...D..RQGFVPAAYIKKI
SPCA_HUMAN/980-1034   QRVMALYDFQARS..PREVTMK.KGDVLTLLSSINK.D..WWKVEAA...D..HQGIVPAVYVRRL
BLK_MOUSE/54-109      RFVVALFDYAAVN..DRDLQVL.KGEKLQVLRST.GD...WWLARSLVTG...REGYVPSNFVAPV
HCK_HUMAN/81-136      IIVVALYDYEAIH..HEDLSFQ.KGDQMVVLEES.GE...WWKARSLATR...KEGYIPSNYVARV
LYN_HUMAN/65-120      DIVVALYPYDGIH..PDDLSFK.KGEKMKVLEEH.GE...WWKAKSLLTK...KEGFIPSNYVAKL
LCK_CHICK/62-117      KLVVALYDYEPTH..DGDLGLK.QGEKLRVLEES.GE...WWRAQSLTTG...QEGLIPHNFVAMV
FGR_HUMAN/80-136      TLFIALYDYEART..EDDLTFT.KGEKFHILNNTEGD...WWEARSLSSG...KTGCIPSNYVAPV
STK_HYDAT/62-118      TIFVALYDYEARI..SEDLSFK.KGERLQIINTADGD...WWYARSLITN...SEGYIPSTYVAPE
```

ProDom: The Protein Domain Database

INRA centre in Toulouse, France
`http://www.toulouse.inra.fr/prodom/doc/prodom.html`
Status (as of Dec. 2000): ProDom 2000.1 (February 2000) 391 ProDom families.

Availability:

- **Download:** `ftp://ftp.toulouse.inra.fr/pub/prodom/current_release/`
- **On-line search:** `http://protein.toulouse.inra.fr/prodom.html`

  Search by keyword, ID, sequence against against database, graphical display of domains in proteins.

The ProDom protein domain database consists of an automatic compilation of homologous domains. Current versions of ProDom are built using a novel procedure based on recursive PSI-BLAST searches. Strong emphasis has been put on the graphical user interface which allows for interactive analysis of protein homology relationships. Almost no annotation. Links to other databases.

Example of pattern:

```
Example of alignment (part of sequences):
 ABL1_CAEEL-120-184     .....FVALY DF.....HGV ...GEEQLSL RKGD..QV.. ..RILGYNKN
 ABL1_HUMAN-66-116      .....FVALY DF.....VAS ...GDNTLSI TKGE..KL.. ..RVLGYNHN
 ABL_FSVHY-15-65        .....FVALY DF.....VAS ...GDNTLSI TKGE..KL.. ..RVLGYNHN
 ABL_MOUSE-66-116       .....FVALY DF.....VAS ...GDNTLSI TKGE..KL.. ..RVLGYNHN
 ABL2_HUMAN-112-162     .....FVALY DF.....VAS ...GDNTLSI TKGE..KL.. ..RVLGYNQN
 ABL_DROME-209-260      .....FVALY DF.....QAG ...GENQLSL KKGE..QV.. ..RILSYNKS
 DRK_DROME-3-53         .....AIAKH DF.....SAT ...ADDELSF RKTQ..IL.. ..KILNMEDD
 GRB2_CHICK-3-53        .....AIAKY DF.....KAT ...ADDELSF KRGD..IL.. ..KVLNEECD
 GRB2_HUMAN-3-53        .....AIAKY DF.....KAT ...ADDELSF KRGD..IL.. ..KVLNEECD
 GRB2_MOUSE-3-53        .....AIAKY DF.....KAT ...ADDELSF KRGD..IL.. ..KVLNEECD
 Q63059_RAT-3-53        .....AIAKY DF.....KAT ...ADDELSF KRGD..IL.. ..KVLNEECD
```

DOMO

Resources Centre INFOBIOGEN, France
`http://www.infobiogen.fr/~gracy/domo/home.htm`
Status (as of Dec. 2000): Current release (03-Dec-2000) has 8877 entries.

Availability:

- **Download:** `ftp://ftp.infobiogen.fr/pub/db/domo/`
- **On-line search:** `http://www.infobiogen.fr/srs6bin/cgi-bin/wgetz?-page+LibInfo+-lib+DOMO+-newId`
  Search by keyword, ID, sequence

DOMO is a database of homologous protein domain families. It was obtained from fully automated successive sequence analysis steps including similarity search, domain delineation, multiple sequence alignment and motif construction. Major fields of each entry provide information about the related proteins, their functional families, domain decomposition, multiple sequence alignment, conserved residues, and evolutionary classification tree.

Example of pattern:

```
PEROXIDASE LIGAND PROXIMAL HEME (not all the sequences are shown in the example)
 #  access dom  beg                                                                      end
 al S46504   1   26 [LNFYAKSCP KAEKIIKDAA ILRMHFHDCF VRGCDGSVLL DLVLLS.... ..GAHTIGVS RCCAFVNSX> ----------   87
 al S55035   1   62 [VVRKHLKKV FKEDVGQAAG LLRLHFHDCF VQGCDASVLL DGSASGPSEQ DAPPNLSLRS KAFEIIDDL] ----------  129
 al JQ2252   1   56 [IVRTELKKV FQSDIAQAAG LLRLHFHDCF VQGCDGSVLL DGSASGPSEK DAPPNLTLRA EAFRIIER]- ----------  122
 al JC1249   1   59 [IVRKFVQDA VRKD....KG LLRLHFHDCF VQGCDASVLL HGSAAEPGEQ QAPPNLTLRP SALKAIDN]- ----------  121
 al S34355   1   59 [IVRKFVQDA VRKD....KG LLRLHFHDCF VQGCDASVLL HGSAAEPGEQ QAPPNLTLRP SALKAIDN]- ----------  121
 al S40268   1   40 [IITEEIDRA IRVAPSIGGP LLRLFFHDCF VRGCDASLLL NAT.SSSNPT EKDAPPNQFL RGFALIDR]- ----------  105
 co               1  ivr#==##a =#*-*r==a* llRlhFHDCF V#GCDas=Ll d**.**=**# ##==*****= rgf*=id#.              69
```

InterPro: Integrated Resource of Protein Families, Domains and Sites

European Bioinformatics Institute (EBI)
`http://www.ebi.ac.uk/interpro/`
Terms of use: free
Status (as of Dec. 2000): InterPro release 2.0 (October 2000) contains 3204 entries, representing 767 domains, 2372 families, 50 repeats and 15 post-translational modification sites.

Availability:

- **Download:** `ftp://ftp.ebi.ac.uk/pub/databases/interpro/`
- **On-line search:** `http://www.ebi.ac.uk/interpro/`

  Text search and sequence search, returns annotation and links to signature databases

InterPro provides an integrated view of the commonly used signature databases (provides data from databses SWISS-PROT, TrEMBL, PROSITE, PRINTS, Pfam, and ProDom), and has an intuitive interface for text- and sequence-based searches.

SBASE

The International Centre for Genetic Engineering and Biotechnology, Trieste Italy
`http://www3.icgeb.trieste.it/~sbasesrv/`
Terms of use: free
Status (as of Dec. 2000): Release 7.0 (6 October, 1999) contains 237.937 annotated segments of proteins clustered into over 1811 groups

Availability:

- **Download:** `ftp://ftp.icgeb.trieste.it/pub/SBASE`
- **On-line search:** `http://www.icgeb.trieste.it/sbase/`

  Search by BLAST or browse a list domains

SBASE protein domain library contains annotated structural, functional, ligand-binding and topogenic segments of proteins, cross-referenced to all major sequence databases and sequence pattern collections. The entries are clustered into over 1811 groups and are provided with two www-based search facilities for on-line use. Domains are given as list of occurences in proteins (no alignment found). They are constructed by BLAST and clustering methods.

HSSP: Database of Homology-derived Secondary Structure of Proteins

`http://www.sander.ebi.ac.uk/hssp/`
Status (as of Dec. 2000): Current release has 12037 entries

Availability:

- **Download:** `ftp://ftp.ebi.ac.uk/pub/databases/hssp/`
- **On-line search:** `http://srs.ebi.ac.uk/srs6bin/cgi-bin/wgetz?-page+LibInfo+-lib+HSSP+-newId`

  Search through SRS system

HSSP is a derived database merging structural (2-D and 3-D) and sequence information (1-D). For each protein of known 3D structure from the Protein Data Bank, the database has a file with all sequence homologues, properly aligned to the PDB protein. Homologues are very likely to have the same 3D structure as the PDB protein to which they have been aligned. As a result, the database is not only a database of sequence aligned sequence families, but it is also a database of implied secondary and tertiary structures.

TRANS-FAC: The Transcription Factor Database

AG Bioinformatik, GBF, Germany
`http://transfac.gbf.de/TRANSFAC/`
Terms of use: free for non-profit use
Status (as of Dec. 2000): Release 4.0 (12-Dec-1999)

Availability:

- **Download:** `http://transfac.gbf.de/cgi-bin/download/download.pl`
- **On-line search:** `http://transfac.gbf.de/TRANSFAC/`
  Search or browse by keyword, gene name, etc. No sequence input.

TRANSFAC database compiles data about gene regulatory DNA sequences and protein factors binding to and acting through them. It contains eukaryotic cis-acting regulatory DNA elements and trans-acting factors. It covers the whole range from yeast to human. The TRANSFAC data have been generally extracted from the original literature.

EPD: Eukaryotic Promoter Database

Bioinformatics Group, Swiss Institute for Experimental Cancer Research
`http://www.epd.isb-sib.ch/`
Status (as of Dec. 2000): Release 64

Availability:

- **Download:** `ftp://ftp.epd.unil.ch/pub/databases/epd/`
- **On-line search:** `http://www.epd.isb-sib.ch/epd_query_form.html`
  Search by keyword, sequence etc. Links to TRANSFAC and other databases.

The Eukaryotic Promoter Database is an annotated non-redundant collection of eukaryotic POL II promoters, for which the transcription start site has been determined experimentally. Access to promoter sequences is provided by pointers to positions in nucleotide sequence entries. The annotation part of an entry includes description of the initiation site mapping data, cross-references to other databases, and bibliographic references.

Example of pattern:

```
tcaggtccgcagaaggtctatttaaagggaagcttgctttcttcccttgACTTTTACTCA
```

COMPEL

Laboratory of Molecular Genetic Systems, Institute of Cytology and Genetics, Siberian Branch of Russian Academy of Sciences, Russia
`http://compel.bionet.nsc.ru/`
Terms of use: free for non-commercial users
Status (as of Dec. 2000): COMPEL 3.0, January 1999, contains 178 composite elements

Availability:

- **Download:** `http://compel.bionet.nsc.ru/cgi-bin/download/download.pl`
- **On-line search:** `http://compel.bionet.nsc.ru/`

  Search by gene name, factor name, text, etc. Search for possible elements in given sequence. Links to other databases.

COMPEL collects information about composite regulatory elements (CEs) - pairs of closely situated sites and transcription factors binding to them. We define a composite element as a minimal functional unit within that both protein-DNA and protein-protein interactions contribute to a highly specific pattern of gene transcriptional regulation. Composite regulatory elements contribute to the one of the fundumental principles of genome functioning - combinatorial nature of gene transcriptional regulation.

Example of pattern:

```
TGAGTCAggcttcCCCTTCCTGCC
```

TRRD: Transcription Regulatory Regions Database

Institute of Cytology and Genetics, Siberian Branch of Russian Academy of Sciences, Russia
`http://dragon.bionet.nsc.ru/trrd/`
Status (as of Dec. 2000): Release 4.1 comprises the description of 514 genes, 717 regulatory units (432 promoters, 139 enhancers, 34 silencers, 74 composite elements), and 2472 transcription factor binding sites. 1700 scientific publications.

Availability:

- **On-line search:** `http://dragon.bionet.nsc.ru/trrd/`

  Browse only

The Transcription Regulatory Regions Database (TRRD) is a curated database designed for accumulation of experimental data on extended regulatory regions of eukaryotic genes, the regulatory elements they contain, i.e., transcription factor binding sites, promoters, enhancers, silencers, etc., and expression patterns of the genes.

SCPD: The Promoter Database of Saccharomyces cerevisiae

Cold Spring Harbor Laboratory, USA
`http://cgsigma.cshl.org/jian/`

Availability:

- **On-line search:** `http://cgsigma.cshl.org/jian/`

  Search by motif, gene name (displays regulatory elements), name of regulatory element, etc.

Example of pattern:

```
TATAWAW
```

Bio-dictionaries(TM)

IBM Research, Computational Biology Center
`http://www.research.ibm.com/bioinformatics/metadata.phtml.html`
Status (as of Dec. 2000): Bio-dictionaries of 17 species are provided.

Availability:

- **Download:** `http://www.research.ibm.com/bioinformatics/metadata.phtml.html`

  Each file contain long list of patterns, as shown in the example. No other data are given.

Bio-Dictionary(TM) is a collection of recurrent amino acid combinations, called seqlets. Seqlets capture both functional and structural signals that have been reused during evolution both within as well as across families of related proteins. Provided Bio-DictionariesTM have been compiled by processing individual complete genomes. They that should greatly facilitate comparative genomics and other studies.

Example of pattern:

```
A...P.YD.EQY......NPREAD.L.VTG.VT...AE.L..IYEK.PEPK.VVAVGACAL.GG..K......G
```