Chapter 7
Methods for discovering conserved patterns in protein sequences and structures

Inge Jonassen
Department of Informatics, University of Bergen HIB, 5020 Bergen, Norway

1 Introduction

The amount of available biomolecular data is exploding: the number of known protein sequences is increasing rapidly while the number of known structures is also increasing, though not as rapidly. A very useful observation in this situation is that common features among proteins can be used to group them into families, and then study the proteins on a family level. By a family we mean a set of proteins sharing some definite biological properties in terms of common function and/or structure, often implying that the proteins have evolved from a common ancestor, i.e. that they are homologous.

When studying a family, one can compare the sequences and structures (if known) of the proteins in the family in order to find what sequence or structure properties are shared by the family members and how these could explain the biological properties shared by the proteins in the family. A description of sequence properties is called a *sequence pattern*, and a description of structure properties is called a *structure pattern*. If a pattern is common to a family of proteins, it is called a *motif* for the family.

An example protein family is the set of proteins containing the classical zinc-finger DNA-binding domain. Most of the sequences in this family match the pattern C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H, thus this is a sequence motif for this protein family. A sequence matches this pattern if it contains a C followed by 2–4 arbitrary letters followed by C and 3 arbitrary letters and one of L, I, V, M, F, Y, W, or C, and so on (this pattern notation is described in detail below). This particular pattern describes two cysteines and two histidines that are needed for coordinating the zinc ion in the classical zinc-finger domain, which means that this particular pattern has a direct biological interpretation. Additionally, the pattern can be used for classification, since not only do most sequences in the family match the pattern, but also very few sequences outside

the family match the pattern. So, if one finds a match to this pattern in a new sequence, the chances are good that this new protein contains a zinc finger domain and binds to DNA.

A large number of patterns have been compiled and collected in different protein family databases. An example is the PROSITE database (1) which contains more than 1000 protein families and for most of these it gives a pattern which occurs in most of the sequences in the family. The patterns are regular expressions (like the zinc finger pattern given above) or profiles (position specific scoring matrices). Other databases also use local alignments, profiles, and Hidden Markov models (HMMs). We describe briefly some of these databases and how they can be used later in the chapter.

When one knows the structures of some of the proteins in the protein family under study, the structures can be compared and similarities described as patterns or motifs. In the same way that protein structures can be described at different levels (e.g. atom, residue, secondary structure, element level), structure patterns can describe structure properties at different levels. For example, one pattern could describe the packing of four alpha-helices and another pattern could describe the relative position of the cysteines and histidines in the classical zinc finger.

In this chapter we will use a very broad definition of patterns including both sequence and structure patterns and all the ways in which these can be defined. When going into more detail we will focus on sequence patterns which are of the regular expression type and on one particular type of structure patterns which describes packing of individual residues. In the following we discuss how existing databases of patterns can be used for analysing a new protein query sequence and how to assess the output of such a search. Later we describe different approaches to finding respectively sequence and structure patterns for a family.

2 Pattern descriptions

A very simple type of patterns is substring patterns—a sequence matches a substring pattern if it contains the substring (contiguous word in the sequence). For example, the substring pattern CDEC is matched by all sequences containing CDEC as a substring. This very simple type of patterns can sometimes be useful in the analysis of protein or nucleotide sequences. We will first define the concept of approximate pattern matching and then describe different generalizations of substring patterns.

2.1 Exact or approximate matching

When matching a sequence against a substring pattern one may allow for approximate matching. In this case, a sequence matches the pattern if it contains a substring approximately equal to the pattern. In practice, one defines a measure of distance between a pattern and a substring and sets an upper limit

on the distance to be allowed. One simple way to measure the distance between two strings (or a pattern and a string) is to count the number of character changes needed to transform one into the other. This is called the number of mismatches or Hamming distance and can measure the distance between two strings only if they have equal length. For example, the sequence AGCDFCALKW approximately matches the substring pattern CDEC since the substring CDFC can be transformed into CDED by substituting the F with an E.

More general distance measures allow for insertion and deletion of characters in addition to substitutions. The edit distance between two strings (sequences) is the minimum number of single character insertions/deletions and substitutions needed to transform one into the other. For example, the sequence AGCDDALKW approximately matches the pattern CDED when one allows for an edit distance of more than one, but it does not match if one allows only for one mismatch. When comparing protein sequences, one may also penalize different substitutions differently since some amino acid replacements are found more often in equivalent positions in homologous (evolutionarily related) proteins. For example substitutions can be penalized using a substitution matrix, e.g. PAM-matrices (2) or BLOSUM matrices (3).

In order to find the edit distance between two sequences, one can use the dynamic programming algorithm (4) which can also be used when substitution matrices are used. For substring patterns, the matching problem is very similar to local pairwise alignment and database searching where speed-ups can be used, e.g. BLAST (5), Fasta (6).

2.2 PROSITE patterns

Using substring patterns and approximate matching one cannot specify that some pattern positions are compulsory (for instance, that there must be cysteines or histidines for binding zinc) while other positions are allowed to vary more freely. *Figure 1* illustrates this by showing four segments (substrings) containing the zinc finger motif and a consensus sequence (substring pattern) for the four. If this substring pattern is to match the four segments shown, one needs to allow up to 17 mismatches. A better way would be to allow no mismatches in the conserved positions, including the cysteines and histidines, and allow the remaining positions to be filled by any amino acid.

Some pattern languages allow a description of which amino acids are allowed at each position. For instance, in the PROSITE database (1) one uses a pattern language which is a subtype of regular expressions. Each pattern consists of a sequence of pattern elements that are of the following types:

(a) Single residue: matches one letter identical to the residue in the sequence, e.g. R matches an R in the sequence.

(b) Set of residues given in square brackets: matches any one sequence letter contained in the set, e.g. [KER] matches any one of K, E, or R in the sequence.

Local sequence alignment	Distance to consensus
EKPFACDFCGRKFARSDERKRHTKIHLRQKE	17
HKPFQCAICMRNFSRSDHLTTHIRTHTGEKP	1
HKPFQCRICMRNFSRSDHLTTHIRTHTGEKP	0
HKPFQCRICMRNFSRSDHLTTHIRTHTGEKP	0
HKPFQCRICMRNFSRSDHLTTHIRTHTGEKP	

Consensus sequence

Figure 1 Example of a local alignment of sequence segments containing the classical zinc-finger motif. The consensus sequence below shows for each position which amino acid is the most frequent, and to the right it is shown for each segment the number of mismatches between the segment and the consensus. Grey shading marks the positions conserved in all four segments. The positions of the conserved cysteines and histidines of the zinc-finger motif are underlined in the consensus sequence.

(c) Set of residues given in curly brackets: matches any one sequence letter not in the set, e.g. {KER} matches any letter except K, E, and R in the sequence.

(d) Wildcard x: matches any one letter in a sequence

Additionally:

(a) Single pattern elements can be followed by parentheses (i, j) which means that the sequence can contain between i and j (inclusive) letters each matching the preceding pattern element. For example, x(3,5) matches between 3 and 5 arbitrary sequence letters.

(b) The pattern can start with '<' meaning that the pattern should match from the beginning of a sequence.

(c) The pattern can end with '>' meaning that the pattern should match until the end of a sequence.

(d) The pattern elements are separated by hyphens '-'.

Consecutive pattern elements should be matched by consecutive sequence symbols, so for example the pattern C-x(2,3)-[DE] matches any sequence containing a C followed by two or three arbitrary letters followed by a D or an E.

2.3 Alignments, profiles, and hidden Markov models

Alternatives to regular expression type patterns are alignments, profiles, and HMMs. These can also be seen as generalizations of substring patterns. Effectively, for each position in the pattern, one now assigns a score (or a probability) to each of the 20 amino acids. Additionally, one assigns penalties (or probabilities) to insertions or deletions in each pattern position. Since alignments, profiles, and HMMs assign a score (probability) to a match to a sequence, they can be called *probabilistic*. Regular expression type patterns can be called *deterministic patterns*—they are deterministic in the sense that a sequence either matches or does not match the pattern (7).

A probabilistic pattern is normally constructed from a local alignment of a set of sequences from the family. A local alignment contains one (or more) segments from each sequence put on top of each other so as to align (put on top of each other) corresponding sequence positions. One may allow for insertion of special 'gap-characters' if needed to align corresponding positions. An example of a local sequence alignment (without gaps) is shown in *Figure 1* (see also Chapter 8). Normally, one would make an alignment of the parts of the sequences that are the most similar; for example in *Figure 1* the zinc-finger domains are aligned. Local alignments without gaps are used for example in the BLOCKS (8) and the PRINTS (9) databases. A sequence can be matched with (aligned to) an existing alignment using dynamic programming to optimize a score. A number of methods exist for scoring the match between an alignment of a single sequence letter to an alignment column and for penalizing gaps that may be inserted. Taking one alignment and a scoring scheme, one can make a scoring matrix giving a position specific scoring for each amino acid and also position specific gap penalties. Such descriptions are called profiles and were initially suggested by Gribskov *et al.* (10). An alternative approach is to use the framework of Hidden Markov Models (HMMs) (11), which, in the way used in sequence analysis, provides a scoring scheme similar to that of profiles, but based on probabilities (see *Chapter 4*).

This means that the information contained in an alignment is often represented as a weight matrix, a profile, or a HMM which specifies for each column in the alignment a score for each of the 20 amino acids when it is aligned with this position. The scores can be calculated from the distribution of amino acids in the column as well as using external information from substitution matrices or Dirichlet mixtures (12). Schemes have been developed to weight the sequences so as to adjust for biases in the input set (13).

PROSITE is also using profiles as a supplement to regular expression type patterns, since for some families it is not possible to define one single regular expression type pattern which matches all family sequences while avoiding matches in unrelated sequences. In such cases it can be possible to define a profile matching the sequences in the family with a higher score than all (known) sequences outside the family. This may be possible since profiles have more expressive power than the deterministic patterns as they can assign different scores to each amino acid when matched to each pattern position and also position-specific gap-penalties. On the other hand, profiles and other probabilistic patterns contain many more parameters to be estimated, and to estimate their values one needs a large number of examples (family members). Also, if one is to learn patterns from noisy examples (including unrelated sequences), the large number of parameters makes it easier to adapt the patterns to match unrelated sequences. In the context of learning patterns from noisy examples, therefore, the deterministic patterns can be more appropriate. Also, deterministic patterns are very simple, mathematically pure, and the human mind finds them easy to interpret. Whichever patterns one choose to use, an apparent problem is how to assess the patterns, to decide between alternative patterns

which is the best one, and to find whether the identified pattern could be the result of chance.

2.4 Pattern significance

When the quality of a pattern is to be assessed, it depends on what is the purpose of the pattern. One typical application of patterns is classification; that is, the patterns are to be used for discriminating between family members and non-members. Another is to find patterns that describe biologically important features. Below we discuss some ways of assessing the quality of patterns with respect to each of the two applications (see also ref. 7).

2.4.1 Characterization

Patterns can be used to describe biologically important features of the proteins, that is, one wants to describe which features are compulsory in order for the protein to belong to a particular family and which features are optional. If one has available a set of sequences (or structures) from the same family, these can be compared, and it can be determined which residues are conserved through evolution and therefore likely to be important to the proteins' function and structure. If the available proteins have undergone little evolution since their last common ancestor (for example if their sequences are 90% identical), it will not be easy to find which of the conserved residues are most important for the biological function of the proteins. Therefore one should try to collect proteins that are as diverse as possible while avoiding inclusion of unrelated proteins.

Having developed a pattern conserved in a set of sequences, one should find whether such a pattern is likely to be conserved by chance and therefore not necessarily biologically important. One common method is to calculate an estimate of the probability that a set of random sequences (equal in number and lengths to the sequences under analysis) would share a pattern of the same strength as the identified pattern, as a result of chance. If this probability is very low, one has a better reason for believing that the identified pattern has some biological meaning. When evaluating the significance of the discovered pattern, one should also take into account the number of patterns that have been considered in the pattern discovery phase (see e.g. ref. 14). For example, a pattern with probability 10^{-6} is expected to be found if one million patterns are considered.

An alternative to calculating pattern probabilities is to measure the patterns' information content (15). The higher the information content the pattern possesses, the less likely a random sequence is to match it. The measure was designed for ranking patterns matching the same number of sequences. Using the principle of minimum description length (MDL) from machine learning (16), this has been extended to also take into account the number of sequences matching each sequence (17).

An alternative approach is to do a series of pattern discovery experiments on sets of sequences with characteristics similar to the sequences in which the

patterns were found. The sequences should be chosen so that they share no significant patterns. The result of the pattern discovery experiments will give information about what type of patterns can be found by chance. For example, one can repeat x times: shuffle all the sequences, and check which patterns can be found to match at least the same number of (the shuffled) sequences as the pattern under analysis. An advantage of this approach is that in assessing the 'background probability' one can use sequences which have the same characteristics as the original similar sequences (local sequence composition, etc.) for example by using special shuffling operations.

When evaluating discovered patterns, it is important to take into consideration whether (some of) the sequences under analysis are very closely related. When calculating the probabilities, the model normally assumes that the sequences are independently generated by some probabilistic model and the shuffling would normally also extinguish any close similarities between the sequences. If some of the sequences are very similar, they will contain many common patterns, and any pattern matching one of them will probably match all, and is therefore likely to be deemed as more significant. A scoring scheme taking this into account, has been proposed (18).

2.4.2 Classification

When a pattern is to be used for classification, it should ideally match all family members and no other sequences. Most often, however, the pattern fails to match some member sequences (called false negatives), and it may match some sequences outside the family (false positives). For an illustration, see *Figure 2*. The fewer false negatives, the more sensitive the pattern is said to be, and the fewer false positives, the more specific it is. Ideally, a pattern should have zero false positives and negatives.

An estimate of the number of matches in a sequence database can be found by multiplying the probability that one random sequence matches the pattern by the number of sequences in the database. In order to calculate the probability we assume that random sequences are generated using a specific probabilistic model. Sternberg (19) did this for all the patterns in the PROSITE database and showed a clear correlation between the expected number of false positives and

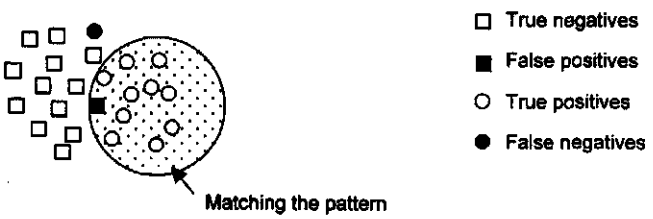

Figure 2 Illustration of the concepts of true positives, true negatives, false positives, and false negatives. The circles are the family members and the squares are non-family members. A pattern matches the encircled objects, and the status of each object is shown by its colour (unfilled means 'true' and filled means 'false').

the actual number, i.e. the number of unrelated sequences in the SWISS-PROT database (20) matching the pattern.

Denoting the number of true positives (sequences in the family matching the pattern) by TP and the number of false negatives by FN, the *sensitivity* of a pattern (21) can be defined as

$$Sensitvity = TP/(TP+FN)$$

and measures of how big a proportion of the family sequences are 'picked up by' (matched by) the pattern. Similarly, the *specificity* of the pattern can be defined as

$$Specificity = TN/(TN+FP)$$

(where TN and FP are respectively the number of true negatives and the number of false positives) which measures of how big a proportion of the sequences outside the family are not matched by the pattern. Yet another useful number is the positive predictive value (PPV) which says how big a proportion of the sequences matching the pattern are actually in the family,

$$PPV = TP/(TP+FP)$$

The value range for all three is from zero to one, one being the best possible. When evaluating patterns to be used for classification, one needs to use more than one of the measures. This can be illustrated by two degenerate cases, (1) the empty pattern matching any protein, and (2) a pattern matching one single protein being member in the family. Pattern (1) has perfect sensitivity, but very bad specificity and PPV, while pattern (2) has perfect specificity and PPV, but bad sensitivity. For a concrete example of the use of these equations, see below. In practice one often needs to make a trade-off between sensitivity and specificity when choosing which pattern to use for a family. One way to evaluate a probabilistic pattern's ability to discriminate between family members and other sequences is to find a cut-off on the score that gives the same number of false positives and false negatives. Tatusov *et al.* (22) evaluated alternative ways of finding weight matrices from local ungapped alignments using this approach.

2.4.3 Discussion

Often the patterns that describe biologically important features will also be good for classification purposes and *vice versa*. However, it is possible that a pattern that gives perfect discrimination can be derived and yet lacks any biological interpretation. Also, it may be that the features described by a pattern are important in the family, but not unique to the family, so that it is not specific enough to be used for classification purposes.

2.5 Pattern databases

A number of different databases for storing information about protein families and motifs have been established during the last ten years. They differ in a number of ways. Firstly, they differ in how they represent the patterns for each family. Secondly, some of them are constructed manually (both sequence group-

Table 1 Summary information about some protein family databases

Database	Pattern type	URL
PROSITE	Reg.exp and profiles	`Http://expasy.hcuge.ch/sprot/prosite.html`
BLOCKS	Blocks (ungapped local alignments)	`Http://www.blocks.fhcrc.org/`
Prints	Blocks	`Http://www.biochem.ucl.ac.uk/bsm/dbbrowser/PRINTS/`
Identify	reg. exp	`Http://motif.stanford.edu/identify/`
Pfam	HMMs	`Http://www.sanger.ac.uk/Pfam/`

ing and pattern definition) while others are made (to a varying degree) automatically. For some summary information about a few family databases, see *Table 1*.

2.5.1 PROSITE

One of the most widely used databases is PROSITE (1), in which, or each family, one or several patterns and/or profiles are given in the format described above (*Section 2.2*). Profiles are described using the generalised profile syntax (23). Statistics are given which describe the patterns' ability to discriminate between family members and other sequences given in the SWISS-PROT protein sequence database (20), in the form of the number of false positives and false negatives. Also, a number of unknowns is given which is the number of sequences in SWISS-PROT which match the pattern, but for which it is not yet known whether it belongs to the family or not.

Figure 3 shows a PROSITE entry giving a signature pattern (motif) for the actinin-type actin-binding domain. We will explain the most important (for our purpose) parts of the entry. First, the ID and AC lines give, respectively, the name and the accession number of the PROSITE entry. The pattern is given on the PA line (the pattern can continue over several PA lines, then end of the pattern being marked by a period sign). The NR lines give statistics about the pattern's discriminatory power with respect to the SWISS-PROT database. The first NR line says that the statistics are with respect to release 35 of SWISS-PROT, which has 69113 sequence entries. The next NR line gives the number of matches of different categories (true positives, false positives, false negatives, etc.) in the SWISS-PROT database. Each is given as $x(y)$ meaning that there are x matches to the pattern in y different sequences.

The DR lines give references to the corresponding SWISS-PROT entries both by their names and accession numbers. Each reference is on the form 'AC, ID, Status;' where Status is one of T (true positive), N (false negative), F (false positive), P (partial), and ? (unknown). Finally, the 3-D line gives names of PDB (Protein Data Bank) entries containing structures of proteins in the family, and the DO line gives the accession number of the entry in the PROSITE documentation part corresponding to this entry.

```
ID   ACTININ_1; PATTERN.
AC   PS00019;
DT   APR-1990 (CREATED); NOV-1997 (DATA UPDATE); NOV-1997 (INFO UPDATE).
DE   Actinin-type actin-binding domain signature 1.
PA   [EQ]-x(2)-[ATV]-[FY]-x(2)-W-x-N.
NR   /RELEASE=35,69113;
NR   /TOTAL=55(46); /POSITIVE=35(28); /UNKNOWN=0(0); /FALSE_POS=20(18);
NR   /FALSE_NEG=0; /PARTIAL=1;
CC   /TAXO-RANGE=??E??; /MAX-REPEAT=2;
DR   P12814, AAC1_HUMAN, T; P35609, AAC2_HUMAN, T; Q08043, AAC3_HUMAN, T;
DR   Q99001, AACB_CHICK, T; Q90734, AACN_CHICK, T; P20111, AACS_CHICK, T;
DR   P05094, AACT_CHICK, T; P05095, AACT_DICDI, T; P18091, AACT_DROME, T;
DR   P21333, ABP2_HUMAN, T; P11533, DMD_CHICK , T; P11532, DMD_HUMAN , T;
DR   P11531, DMD_MOUSE , T; P19179, FIMB_CHICK, T; P54680, FIMB_DICDI, T;
DR   P32599, FIMB_YEAST, T; P13466, GELA_DICDI, T; Q14651, PLSI_HUMAN, T;
DR   P13796, PLSL_HUMAN, T; Q61233, PLSL_MOUSE, T; P13797, PLST_HUMAN, T;
DR   Q63598, PLST_RAT  , T; Q00963, SPCB_DROME, T; P11277, SPCB_HUMAN, T;
DR   P15508, SPCB_MOUSE, T; Q01082, SPCO_HUMAN, T; Q62261, SPCO_MOUSE, T;
DR   P46939, UTRO_HUMAN, T;
DR   P11530, DMD_RAT   , P;
DR   P13688, BGP1_HUMAN, F; P06731, CCEM_HUMAN, F; P31997, CGM6_HUMAN, F;
DR   P40782, CYP1_CYNCA, F; P10474, GUNB_CALSA, F; P40199, NCA_HUMAN , F;
DR   P11462, PBG1_HUMAN, F; P11463, PBGC_HUMAN, F; P11464, PBGD_HUMAN, F;
DR   Q00887, PSGB_HUMAN, F; P35853, PUR1_LACCA, F; P14410, SUIS_HUMAN, F;
DR   P23739, SUIS_RAT  , F; P28668, SYEP_DROME, F; P07814, SYEP_HUMAN, F;
DR   P42954, TAGH_BACSU, F; P09301, UL07_VZVD , F; P52583, VGR2_COTJA, F;
3D   1KSR; 1DRO; 1BTN; 1MPH;
DO   PDOC00019;
```

Figure 3 Example of a PROSITE entry taken from release 14.0 (November 1997). See text, for a detailed explanation.

Referring to the pattern quality measures discussed earlier, this particular pattern has sensitivity 28/(28 + 0) = 1, specificity (69 113 − 28–20)/(69 113 − 28) = 0.9997 (69 113 − 28 is the number of sequences in SWISS-PROT outside the family), and PPV 28/(28 + 18) = 0.608. The last number means that if a sequence from SWISS-PROT matches the pattern, the probability that it belongs to the family is 60.8%. For this particular application, the PPV measure seems more meaningful than specificity since the number of false positives do not affect the measure of specificity very much as the number of true negatives most often will be very much bigger than the number of false positives. Each entry in the PROSITE documentation part gives a description of the family and explains the biological significance of the signature patterns. It also gives literature references and one or several experts that can be contacted for more information about the family. The PROSITE database is largely maintained manually; new families, profiles and patterns are carefully scrutinized.

2.5.2 BLOCKS

Another protein family database is BLOCKS (8) which contains the same families as PROSITE, but instead of giving patterns or profiles, it gives a set of blocks for each family. A block is an ungapped local multiple alignment. The database is constructed fully automatically. For each family the member sequences (as given in PROSITE) are subjected to pattern discovery and local alignment methods. The blocks can also be linked in chains. It is recommended that a query sequence is matched against both PROSITE and BLOCKS, because even though they describe

the same families, the patterns and the blocks in a sense have complementary strengths when used as classifiers.

2.5.3 PRINTS

The PRINTS database is a collection of fingerprints and is constructed semi-automatically (9). A fingerprint is defined as a list of motifs, each motif being a local ungapped alignment. The fingerprints are made by first manually making an alignment of some family members and then iteratively scanning a database of protein sequences, adding new members, updating the fingerprint, scanning again until convergence (no new family members are found). Each entry in PRINTS, gives the local alignments corresponding to the motif and information about partial matches etc. On the website of PRINTS, a tool FingerPRINTScan, is available for scanning a query sequence against the database. The output can be visualized showing the position of the motif matches in the query sequence.

2.5.4 PFam

Pfam is a database of multiple sequence alignments and HMM-profiles of protein domains (24). It is partly manually curated. Seed alignments for each family are made semi-automatically, and these are extended using HMM methods. Version 3.1 (August 1998) contains 1313 families.

On the Web site there are available tools for matching a query sequence against the database. Also, a special database SWISSPFam is made which shows the domain organization (according to Pfam) or the sequences in SWISS-PROT and TrEMBL.

2.5.5 Identify

Identify (25) is a database of patterns of the same form as used in PROSITE but without flexible length wildcards. It contains patterns for the families in the BLOCKS and PRINTS databases. The patterns were constructed from the ungapped alignments (blocks) in the BLOCKS and PRINTS databases by using a pattern finding program (EMOTIF). For each block, there can be several patterns so that each pattern matches a subset of the sequences. The patterns were generated to have a certain specificity (calculated as the probability that a random sequence matches the pattern by chance, cf. ref. 19), and patterns were generated for different specificity levels. The World Wide Web server allows the user to input a query sequence which then is matched against the pattern collection and the user is given the list of matching patterns together with links to the corresponding BLOCKS and PRINTS database entries.

2.6 Using existing pattern collections

Most of the family databases are available on the World Wide Web, and they also provide on-line tools for (1) matching a sequence against the pattern in the database, and sometimes (2) matching a new (user-defined) pattern against a sequence database. For example, on the PROSITE website, the search engine

ScanProsite that can be used for both (1) and (2) with regular expression type patterns. For scanning against the profiles in PROSITE, the tool ProfileScan can be used. The ScanProsite program does not return any information about the probability that the match between the sequence and the pattern could be by chance, and it does only allow for exact matching between the pattern and the sequence.

Another very useful tool is PdbMotif (26), which takes as input a protein structure and finds all matches between the protein's sequence and patterns in the PROSITE database. It generates a script that can be input to RasMol (27) to highlight the pattern matches. PdbMotif also outputs a probability that the sequence should match the PROSITE pattern by chance, which can help to identify possible false positives.

3 Finding new patterns

Analysing a set of proteins believed to be related one may want to find a motif describing the set. The goal may be to find which features are common to the proteins under study helping to better understand the relationships between sequence, structure, and function of the proteins. Motifs can also help to identify new possible family members. We may also want to find motifs to be included in protein family databases where the motif may later be used for classification.

One may develop motifs semi-manually using knowledge about the proteins under study and for example alignments generated either manually or automatically using a multiple (local or global) sequence alignment program. If the alignment programs were guaranteed to find the correct alignments, this approach would be all you needed. However, multiple alignments are often difficult to obtain and interpret. Therefore, in many cases, direct methods for finding motifs directly from the sequences or structures can lead to better results. In such cases, the motifs found can also be used to guide the alignment of the sequences or structures in the family.

3.1 A general approach

There exist a large number of methods for the discovery of patterns from protein (and DNA) sequences. A survey of these is given by Brazma *et al.* (7) who propose a general three-step approach to pattern discovery:

(a) Choose a **solution space**, i.e. set of patterns that the method potentially can discover.

(b) Define a **fitness function** reflecting how well a pattern fits the input sequences.

(c) Develop an **algorithm**, which given a set of input sequences, returns the pattern(s) from the solution space with high (highest) fitness.

The different methods can be classified according to their approach to each of the three steps. This three-step approach can be used for the discovery of all

types of sequence patterns (including regular expression type patterns, profiles, and Hidden Markov models) and also for the discovery of motifs in protein structures. An additional criterion is whether a method is guaranteed, for any input set, to find the best (as measured by the fitness function) pattern in the solution space. In the following sections we will illustrate the three steps approach by describing in some detail some representative methods, especially focusing on the Pratt and SPratt methods.

3.2 Discovery algorithms

Previous sections have discussed in some detail different solution spaces and principles of fitness functions. Here we discuss in some more detail the third step, namely that of finding the patterns in the solution space that have high, and possibly the highest, fitness. Brazma et. al. (7) identified two main algorithmic approaches to this problem:

(a) Pattern Driven (PD) methods.

(b) Sequence Driven (SD) methods.

3.2.1 Pattern driven methods

In the simplest form, PD methods enumerate all patterns in the solution space calculating each pattern's fitness so that the best ones can be output. For example, one step of the algorithm by Smith *et al.* (28) works by enumerating all patterns of the form A1-x(d1)-A2-x(d2)-A3 having single amino acid symbols and d values up to some maximum (e.g. 10). For each pattern, the number of matching input sequences were counted, and the patterns matching the most sequences were subjected to further analysis.

Some methods have mechanisms for avoiding looking at all patterns in the solution space, for example by not searching parts of the solution space that cannot possibly contain patterns scoring higher than already analysed patterns. Often heuristics are used to discard parts of the solution space that are unlikely to contain high scoring patterns. Examples of this will be seen when the Pratt method is described in some more detail below (see *Figure 4*).

3.2.2 Sequence driven methods

These methods work by comparing (normally pairs of) the input sequences and expressing local similarities as patterns. By repeating this for different pairs of sequences, or by subjecting the patterns themselves to pairwise comparison, patterns are gradually built up that match many or all of the input sequences. Some of these methods are very closely related to methods for sequence alignment.

For example, the pattern discovery method proposed by Smith and Smith (29) first computes the similarity between all pairs of input sequences. The most similar pair is input to a local pairwise alignment method that is based on dynamic programming (30). Instead of producing a pairwise alignment, this algorithm outputs a pattern common to the two sequences. The pattern is basic-

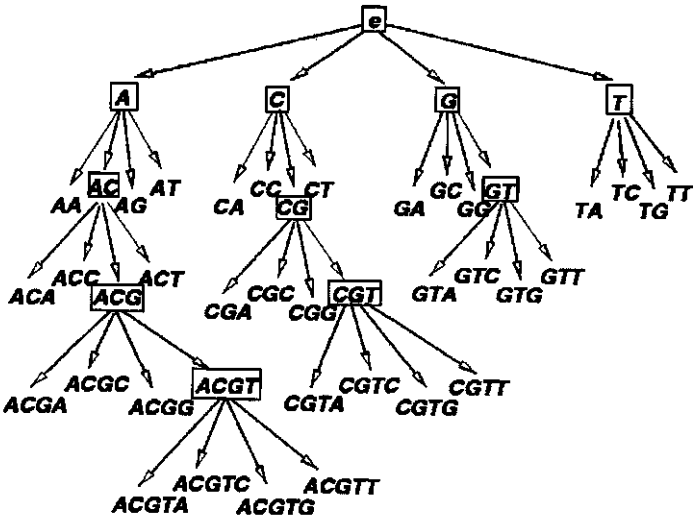

Figure 4 Example of a pattern-driven discovery method applied to a set of four sequences: ACGTT, TACGTA, GCACGT, and TTACGTAA. The solution space consists of all words over the DNA alphabet (no wildcards or group characters), and patterns are sought that match all four sequences. Patterns matching all four are boxed, and only these are extended. The longest pattern found to match all four sequences is ACGT, The 'e' at the top is the empty pattern which matches any sequence.

ally of the PROSITE type, but the set of possible amino acid groups is limited and given by an Amino Acid Class Covering (AACC) hierarchy. The two sequences aligned are now replaced by their common pattern, and the procedure is repeated until there remains only one pattern matching all of the input sequences. For an example, see *Figure 4*. Note that this approach is very analogous to the progressive multiple alignment methods used for example by Thompson *et al.* (31) and Taylor (32).

4 The Pratt programs

The Pratt programs take as input a set of (un-aligned) sequences and finds patterns matching at least a (user-defined) minimum number of the sequences. The patterns are of the type used in PROSITE and can include both character groups and flexible length wildcards. For instance, Pratt is able to automatically rediscover the pattern C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H from the set of over 300 classical zinc-finger-containing protein sequences in SWISS-PROT in less than a minute on a modern workstation. No prior information about the pattern is given to Pratt and the sequences are input unaligned.

The user can input constraints on the patterns to be considered, effectively defining the solution space to be used (cf. three-step approach). The user also chooses how many of the input sequences a pattern should match, in order to be output. We will call any pattern matching at least this number of sequences,

a conserved pattern. Pratt uses a two-step search for finding conserved patterns from the chosen solution space having maximum fitness. The fitness is normally defined as the information content of the pattern (see below).

In the following sections we give a practical guide to how Pratt should be used and then some details about the algorithms used in Pratt. For more detailed technical descriptions, see the original papers (15, 33) and the Pratt home page on the World Wide Web: http://www.ii.uib.no/~inge/Pratt.html.

4.1 Using Pratt

When using Pratt from a command line environment, it is started using the command

```
pratt <format> file [options]
```

where `format` is one of `fasta` or `swissprot` and `file` is a file containing the sequences in the indicated format. Optionally, one can specify options on the command-line. If no options are given, a menu appears. *Figure 5* shows the menu of Pratt when run on a file MUTT containing 26 unaligned sequences. The menu can be used to choose values for a number of parameters and also to obtain help. The parameters fall within a few main classes:

(a) **Pattern conservation.** Using options CM (respectively, C%) one can set the minimum number (respectively, percentage of the input sequences) of sequences a pattern should match.

(b) **Pattern restrictions.** A number of parameters are used to constrain the patterns to be considered. For example PL can be used to set the maximum length of a pattern, PN to set the maximum number of non-wildcard symbols, PX to set the maximum length of a wildcard region. For constraining the flexibility to be allowed in wildcard regions, FN is used to set the maximum number of flexible wildcards and FL to set the maximum flexibility of a

```
Pratt version 2.2:  Analysing 26 sequences from file MUTT

 PATTERN CONSERVATION:                  SEARCH PARAMETERS:
 CM: min Nr of Seqs            26       G:  Pattern graph from          seq
 C%: min Percentage         100.0
                                        E:  Search greediness             3
 PATTERN RESTRICTIONS :                 R:  Pattern Refinement           on
 PP: pos in seq               off       RG: Generalise                  off
 PL: max Length                50
 PN: max Nr of Symbols         50       OUTPUT:
 PX: max Nr of x's              5       OF: Output filename   MUTT.26.pat
 FN: max Nr of flex.            2       OP: PROSITE pat notation         on
 FL: max Flexibility            2       ON: max nr patterns              50
 BI: Symbol File              off       OA: max nr alignments            50
                                        M:  Match summary                on
 BN: Initial Search             20       MR: Ratio                        10
                                        MV: Vertical summary            off
 PATTERN SCORING:
 S:  Scoring                  info

 X: eXecute program       Q: Quit       H: Help

 Command:
```

Figure 5 Pratt menu. The options are explained in the text.

wildcard. Note that the PX constraint also applies to patterns to be found during the initial search. For some examples, see *Table 2*. Also, one can choose which pattern symbols should be used during initial pattern search and during pattern refinement using options BI (BF) and BN.

(c) **Pattern scoring**. By default patterns are scored by their information content. Optionally one can use a scoring function derived from the Minimum Description Length (MDL) principle. Patterns can also be ranked by their positive predictive value (PPV)—in this case the name of a file containing a sequence database in flat file format must be given using option SF.

(d) **Search parameters**. By default, the shortest sequences will be used for deriving a pattern graph (see below). Optionally this can be generated instead from a special query sequence or from an alignment, in which case only patterns that match the query sequence respectively are consistent with the alignment, will be considered in the search.

(e) **Output format**. Filename for output can be chosen using option OF, format of patterns using option OP, if pattern matches should be shown (option OA), etc.

(f) **Help and control**. Help can be obtained by typing in option H, the search started by using option X, or abandoned using option Q.

All parameters that can be set using the menu can alternatively be set from the command line by adding '-<menu option> <value>' to the command. For instance

```
pratt fasta seqs -cm 20
```

tells Pratt to analyse the sequences in the file `seqs` (fasta format) to find patterns matching at least 20 of the sequences. When command line options are used, the menu will not appear unless the option `-menu` is used.

For example, if the file c2h2 contains the sequences (in fasta format) of the proteins in the classical c2h2 zinc finger family, these can be analysed using the command:

```
pratt fasta c2h2 -px 15
```

Table 2 The table shows some example patterns, and for each example, the minimum values to be used for some Pratt parameters if the pattern is to be discovered. For example, in order to discover the bottom-most pattern, one needs to increase the value of the PX parameter to at least 12

	Minimum parameter values				
Pattern	**PL**	**PN**	**PX**	**FL**	**FN**
C-x(3)-C	5	2	3	0	0
C-x(2,5)-C	7	2	5	3	1
C-C-D-E-x(7)-C	12	5	7	0	0
C-x(2,4)-C-x(12)-H-x(3,5)-H	25	4	12	2	2
Default values	50	50	5	2	2

using the option -px 15 to allow long wildcards (if we want to re-discover the known motif, we need to set PX to at least 12 since during the initial pattern search, a spacing of 12 is needed between the last conserved cysteine and the first histidine). Also, if one wants to find all patterns matching a minimum 90% of these sequences, one can use the command

```
pratt fasta c2h2 -px 15 -c% 90
```

4.2 Pratt: Internal search methods

The search for conserved patterns is done in two phases:

(a) **Initial pattern search**. Search for patterns having only single character elements and wildcards (possibly of flexible length). For example, in this phase the pattern Pratt can discover that the pattern `A-x(4)-D-x-E` is conserved. Optionally, group characters can be allowed also in this step, at the cost of increased computing time.

(b) **Pattern refinement**. Take each of the best patterns from phase 1, collect the matching sequence segments and check if wildcard positions can be replaced by group characters so that the pattern remains conserved. Also, the pattern may be extended to the right (but not to the left) in this phase. For example, the pattern above can now be refined to

```
A-x-[KER]-x(2)-D-[ILV]-E-x(4)-[KR].
```

4.2.1 Initial pattern search

In the first version of Pratt (15), in the first phase a search tree containing all patterns to be considered, is explored. The label of the root node is the empty pattern. A node having the pattern P as label has children with labels $P\text{-}x(i,j)\text{-}A$, that is P extended with a wildcard region (empty if $i=j=0$) and a single character A, for all allowable values of i, j, and A.

The search starts at the root of the tree. At each step of the search a node in the tree having label P is considered. It is assumed that P is conserved. Then, all children of P are generated, and for each of these it is checked whether the corresponding pattern is conserved (whether it matches the minimum number of sequences). If a pattern is conserved, this is recursively analysed using the same procedure. If no extension of P is conserved, then, depending on the score of P, it is included in the list of the best patterns that are subsequently input to the refinement algorithm.

In the second version of Pratt (33), instead of considering the full tree of all patterns in the solution space, a pattern graph is used to define the set of patterns to be considered. The pattern graph is a node- and edge-labelled directed acyclic graph. The nodes are labelled with non-wildcard pattern elements and the edges have as labels wildcard lengths. A path in the graph defines a pattern and from this pattern, more generalized versions will be generated. An example of a pattern graph is shown in *Figure 6*.

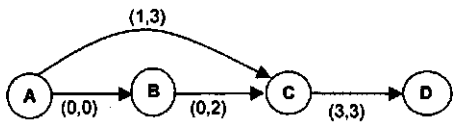

Figure 6 Example of a pattern graph. The paths in the graph define the patterns A-B-x(0,2)-x(3,3)-D, A-B-x(0,2)-C, A-B, B-x(0,2)-C-x(3,3)-D, B-x(0,2)-C, C-x(3,3)-D, A-x(1,3)-C-x(3,3)-D, A-x(1,3)-C.

The initial search explores all patterns that can be derived from paths in the pattern graph and that are contained in the class of patterns defined by the user. The search is focused on finding only the highest scoring patterns. Branch-and-bound techniques are used to avoid considering parts of the search space that cannot possibly contain patterns with higher scores than already identified patterns.

Also, heuristics have been implemented that effectively avoids exploring search paths unlikely to produce patterns scoring higher than patterns already found. The user can adjust the greediness of the search. Setting the E parameter to zero gives non-heuristic search (guaranteed to find highest-scoring patterns), setting E to 1 gives the same guarantee in cases where no flexibility is allowed in wildcard regions, and E values above *1* gives increasingly greedy search. The default value is *3*. The more greedy the search, the faster it will be, and the more likely Pratt is to not find the highest scoring patterns. Experiments have shown that $E = 3$ gives a good compromise between speed and accuracy for protein sequences, while for DNA a lower value should be used (for instance, $E = 1.5$).

4.2.2 Pattern refinement

During refinement, each position in the fixed-length wildcard regions (excluding regions x(i, j) where j > i) is analysed and it is checked whether replacing the wildcard with an allowed group character (the allowed groups are given as a list) so that the resulting pattern remains conserved. It might be that there are several wildcard positions that can be replaced by group characters, so that replacing any one of these gives a conserved pattern, but if all are replaced simultaneously, the resulting pattern will not be conserved. As there are exponentially many (in the number of positions that can be replaced) subsets of replacements that could be done, it is computationally expensive to consider all. Therefore, in the second version of Pratt (33), a heuristic refinement algorithm is used. The degree of greediness is adjusted using the same E-parameter as for the initial search.

4.2.3 Block data structure

To find all matches to each pattern quickly, a special data structure initially proposed by Neuwald and Green (14) is used. Assume that the input sequences are $S = \{S1, S2, \ldots, Sn\}$. For some fixed parameter w (typically 50), let B be the set of all w-segments (consecutive substrings of length w) from all of the sequences in S. Also construct w-segments in the end of the sequences by appending to

Input sequences:

```
MDNVVDPWYI
MANVEKPND
MMHIKSLPHAHH
```

The corresponding 5-segments:

```
B={MDNVV,DNVVD,NVDDP,VVDPW,VDPWY,DPWYI,PWYI-,WYI--,YI---,I----,
   MANVE,ANVEK,NVEKP,VEKPN,EKPND,KPND-,PND--,ND---,D----,
   MMHIK,MHIKS,HIKSL,IKSLP,KSLPH,SLPHA,LPHAH,PHAHH,HAHH-AHH--,HH---,H----}
```

Figure 7 Example of the set B of w-segments made for a set of sequences. The B segments are used in the block data structure.

each sequence *w-1* dummy symbols '-'. For an example of the *w*-segments made for a set of sequences, see *Figure 7*.

Now, for each amino acid symbol *a*, and for each *i* between *1* and *w*, construct the set $b_{i,a}$ that is the set of all *w*-segments having character *a* in position *i*. These sets can be used to quickly find the set of *w*-segments matching any pattern considered by Pratt not having length exceeding *w*. For instance, the set of segments matching `A-x(2)-B` is $b_{1,A} \cap b_{4,B}$. In the recursive search, the set of segments matching P is used together with the block data structure to find the segments matching each extension P-x(i,j)-A of P. For a more detailed description, see Jonassen *et al.* (15).

4.3 Scoring patterns

Pratt can score discovered patterns using different fitness functions. By default the patterns are evaluated by their information content (15). This measure depends only on the pattern itself and is only appropriate for ranking patterns matching the same number of sequences. The information content of a pattern is a measure of the information gained about an unknown sequence when one is told that the sequence matches the pattern. It increases with the number of single character elements (any one single character contributing the same) and with the number of group character elements (more ambiguous contributing less). A penalty is subtracted for flexible wildcard regions, more specifically *x(i,j)* is penalized by *c(j-i)* where *c* is a parameter whose default value is *0.5*.

As an alternative, Pratt can score the discovered patterns using the minimum description length (MDL) principle based fitness measure described by Brazma *et al.* (17). This assigns a score to a pattern that depends on the pattern's information content and on how many sequences it matches. The fitness measure definition contains some parameters that can be used to slant the optimum towards strong patterns matching few sequences or towards weaker patterns matching many sequences. Brazma *et al.* (17) used the measure in a method for simultaneously finding subfamilies and patterns in a set of unaligned (and unlabelled) sequences. It was required that each pattern matches all sequences in one of the subfamilies. The method uses Pratt to find patterns matching different sized subsets of sequences. Next it selects in a greedy fashion a collection of the patterns that cover the input sequence and has a high fitness value.

If the aim is to find patterns to be used for classification, Pratt can evaluate the discovered patterns by their positive predictive value (PPV, see *Section 2.4.2*, Classification). It is assumed that the sequences under analysis are all in the SWISS-PROT database (20). The number of false positives for each pattern is found by matching the patterns against the SWISS-PROT database that must be locally available in flat file format.

5 Structure motifs

Finding recurring patterns (motifs) in protein structures help to better understand the rules underlying the formation of protein structures. Since structure is better conserved during evolution than sequence, structural similarities can also help to identify remote evolutionary relationships. Structure motifs can help in approaching the structure prediction problem and in assigning function to proteins. Structure motifs can represent common structural features at different levels. For example, they can represent packing of secondary structure elements, local packing of residues, and atom coordinates of binding atoms in active site (or ligand binding) residues. Structure motifs describing functional sites in proteins, have been developed by, for example, Wallace *et al.* (34). They call their motifs 'templates' and suggest that they can be used for finding functional sites in proteins. They have also developed a database PROCAT of such templates (35) which allows the user to search for 3-D enzyme active sites in a protein structure.

In order to find recurring patterns in protein structures one can use methods for the comparison of protein structures. A number of such methods have been developed, most of them for comparing pairs of structures, but also some for multiple structure comparison. The methods differ in what similarities they are able to find. Some represent the structures as composed of secondary structure elements (alpha helices, beta strands, and loops) and have provided methods for finding patterns of conserved patterns at this level. Other methods find patterns of residues (or atoms) that have similar configurations in space. Brown *et al.* (36) gives a survey of a large number of different methods focusing on their way of representing similarities. An important difference between methods is whether they require matched elements to be in the same order along the proteins' primary structure. A number of different methods have been used, including extensions of the dynamic programming algorithms used for pairwise sequence (37), use of graph-theoretic methods (38), and methods from computer vision (39).

Also, some more direct methods for structure motif discovery have been suggested. Here we will describe the SPratt program, a more detailed description of which can be found in (40).

5.1 The SPratt program

The idea behind the SPratt program was to use the Pratt method developed for sequence motif discovery, to discover structure motifs. This can be done by

encoding structural features in the form of strings and input these to Pratt producing patterns common to the strings. Next one needs to check if the patterns found in the structure description strings correspond to similarities between the structures. The encoding of structural properties that we adopted was one described by Karlin and Zhu (41). In their method, they make one string per residue in each structure. The strings contain information about the spatial neighbourhood of each residue. Karlin and Zhu describe alternative methods for making these strings, and we chose one of them to be used in SPratt.

Overview of the method:

1. For each residue *a*, we make two neighbour strings *Ca* and *Na*. The string *Ca* starts with *a* followed by all the residues C-terminal to *a*, whose spatial distance to *a* is below a user-chosen threshold *dmax*. The residues are ordered in N-to-C chain order. Analogously, the string *Na* starts with *a* and contains in C-to-N order the residues preceding a in the chain which are spatially close to a (distance below *dmax*).
2. Run Pratt twice, once on the complete set of *C* and once on the *N* strings. Search for patterns whose matches start from the beginning of the matched residue strings and that match residue strings from at least the minimum number of structures chosen by the user.
3. For each pattern, consider the neighbourhood string matches and retrieve the substructures (list of residues) corresponding to each such match. Compare the spatial geometry of the substructures by calculating the root mean square deviation (RMSD) when superposing each pair of substructures.
4. Output the patterns for which all pairwise RMSD values are below some upper limit, and rank them by *I/R* where *I* is the information content of the neighbourhood string pattern and *R* is the maximum RMSD for any pair of matching substructures.

The method differs from other structure comparison methods in that it considers information from all structures simultaneously in step 2. Other methods perform a number of pairwise structure comparisons and combine the results to find motifs shared by all or most of the structures under study. By utilizing information from all structures simultaneously, SPratt avoids considering patterns common to pairs of structures but not shared by the others. On the other hand, when Pratt is used to analyse the neighbourhood strings, many patterns can be found which later prove not to reflect similar substructures (i.e. they do not superpose very well). It might be advantageous to use additional structural information in this step to avoid considering such patterns. We explore extending the encoding of the neighbourhoods to include information about secondary structure and restrict Pratt to match only residues from the same sort of secondary structure (alpha-helix, beta-strand, or loop).

The patterns found by SPratt are evaluated using a very simple function, which basically rewards patterns containing more residues and imposing stricter restrictions on the amino acids allowed for each residue, and penalizes patterns

for which the occurrences do not superpose very well using a measure of RMSD. However, the RMSD value for superposing the coordinates of a small number of residues, for example 4 is not very informative. The significance of the patterns found by Pratt, can be further assessed using the structure alignment program SAP (42). We have done this by rewarding alignment of residues in agreement with the motif, and in this way checking whether the matching of the few residues described by the motif can be extended to an alignment of larger parts of the structures.

6 Examples

In (15) we described the application of the first version of Pratt to the analysis of some protein families in PROSITE. For example, we analysed the Snake toxin family (PS00272 in PROSITE) containing 164 sequences of average length 64. We retrieved the sequences from the SWISS-PROT database and input them (unaligned) to the Pratt program. Using default parameters (which requires patterns to match all the sequences), no patterns were found. However, using Pratt to discover patterns matching at least 155 out of the 164 sequences, we got the pattern G-C-x(1,3)-C-P-x(8,10)-C-C-x(2)-[PDEN]. This pattern turned out not to match any sequences in SWISS-PROT apart from the family members and was since included in the PROSITE database as the pattern for this family.

Using the SPratt program we analysed a set of cupredoxin protein structures (40). The proteins were selected from the cupredoxins super-family in SCOP (43) so that all pairwise sequence similarities were 30% or less. The 10 structures were input to the SPratt program and it was instructed to search for patterns containing single residue elements and also allowing the match-set [MLQ] (since the methionine ligand-binding residue is known to be substituted by L or Q in some proteins). SPratt used two minutes and identified three patterns all matching around the copper binding sites. The substructure occurrences of the pattern superpose with very low RMSD values (0.7 Å or less). We also used the motif identified by SPratt to guide the structure alignment program SAP using the pairs of equivalenced residues as extra constraints on the alignment. This resulted in a greatly improved alignment with RMSD of 1.56 Å over 63 pairs of residues (as compared to the alignment with RMSD of 5.1 Å over 26 residues when no SAP was run without any motif information).

7 Conclusions

Patterns and pattern discovery tools can help in the analysis of protein families, i.e. sets of proteins believed to share structural and/or functional properties. The most well conserved parts of the sequences or structures can be identified and it can be analysed whether the conserved patterns are statistically significant and therefore likely to have biological importance. Furthermore, the patterns can be used to identify additional related proteins. Patterns can describe protein prop-

erties at sequence level (sequence patterns) or at structure level (structure patterns). There exist a number of databases of protein families that give for each family one or several sequence patterns that can be used to identify more family members. We described in some detail the PROSITE, BLOCKS, Identify, PRINTS, and the Pfam databases. Structure pattern databases are starting to appear, for instance PROCAT is a database of 3-D enzyme active site templates (35).

For sequence patterns we made a distinction between deterministic and probabilistic pattern. Regular expression type patterns fall into the first class and profiles and HMMs fall into the second class. The different types of patterns used each have their strengths and weaknesses. For example, regular expression patterns are easily interpreted, but provide less expressive power than profiles or HMMs.

A number of methods have been developed for the automatic discovery of conserved patterns in protein sequences. We summarized a framework comprising a three-step approach that can be used to better understand the myriad of different methods. One sequence motif discovery program, Pratt, was described in some more detail, focusing on the more practical aspects of how it can be used to find conserved regular expression type patterns in unaligned protein sequences.

Structure motifs can describe properties at different levels, e.g. at atom group or residue level or at secondary structure level. Most structure comparison methods are for comparing pairs of structures, but some of these have been extended to the comparison of several by combining the results of a set of pairwise comparisons. We described in some detail a tool SPratt (Structure Pratt) which is able to find patterns of locally packed residues whose spatial positions are similar in a set of protein structures. This method utilizes information from all structures simultaneously in the search for conserved motifs.

Motifs provide a powerful classification tool for determining structure and function of proteins coming out of the numerous genome projects. Also, methods for finding conserved patterns in structures and sequences can be used to extract information from the large amounts of biomolecular data. In this chapter we have only considered patterns and motifs in proteins, but pattern discovery methods can also be used in other applications. For example, recently, a pattern discovery method was used to find putative regulatory elements in sets of gene upstream regions from the Yeast genome (44).

References

1. Bairoch, A., Bucher, P., and Hofman, K. (1996). *Nucleic Acids Res.*, **24**, 189.
2. Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978). In *Atlas of protein sequence and structure* (ed. M. O. Dayhoff), Vol. 5, Suppl. 3, p. 345. National Biomedical Research Foundation, Washington DC.
3. Henikoff, S. and Henikoff, J. G. (1992). *Proc. Natl. Acad. Sci. USA*, **89**, 10915.
4. Needleman, S. B. and Wunsch, C. D. (1970). *J. Mol. Biol.*, **48**, 443.
5. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). *J. Mol. Biol.*, **215**, 403.

6. Lipman, D. J. and Pearson, W. R. (1985). *Science*, **277**, 1435.
7. Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998). *J. Comp. Biol.*, **5**, 279.
8. Pietrokovski, S., Henikoff, J. G., and Henikoff, S. (1996). *Nucleic Acids Res.*, **24**, 197.
9. Attwood, T. K., Beck, M. E., Bleasby, A. J., Debtyarenko, K., and Smith, D. J. P. (1996). *Nucleic Acids Res.*, **24**, 182.
10. Gribskov, M., McLachland, A. D., and Eisenberg, D. (1987). *Proc. Natl. Acad. Sci. USA*, **84**, 4355.
11. Krogh, A., Brown, M., Miah, I. S., Sjoelander, K., and Haussler, D. (1994). *J. Mol. Biol.*, **235**, 1501.
12. Brown, M., Hughey, R., Krogh, A., Mian, I. S., Sjoelander, K., and Haussler, D. (1993). In *Proc. 1st Int. Conf. On Intell. Systems for Mol. Biol.*, pp. 47–53. AAAI Press.
13. Altschul, S. F., Carroll, R. J., and Lipman, D. J. (1989). *J. Mol. Biol.*, **207**, 309.
14. Neuwald, A. F. and Green, P. (1994). *J. Mol. Biol.*, **239**, 698.
15. Jonassen, I., Collins, J. F., and Higgins, D. G. (1995). *Protein Sci.*, **4**, 1587.
16. Rissanen, J. (1978). *Automatica-J.IFAC*, **14**, 465.
17. Brazma, A., Jonassen, I., Ukkonen, E., and Vilo, J. (1996). In *Proc. 4th Int. Conf. On Intell. Systems for Mol. Biol.*, pp. 34–43. AAAI Press.
18. Jonassen, I., Helgesen, C., and Higgins, D. (1996). Reports in Informatics, report no. 116. Dept. of Informatics, Univ. of Bergen, Norway.
19. Sternberg, M. J. E. (1991). *Nature*, **349**, 111.
20. Bairoch, A. and Boeckmann, P. (1992). *Nucleic Acids Res.*, **20**, 2019.
21. Lathrop, R., Webster, T., Smith, R., Winston, P., and Smith, T. (1993). In *Artificial intelligence and molecular biology* (ed. L. Hunter), pp. 211–58. AAAI Press/The MIT Press.
22. Tatusov, R. L., Altschul, S. F., and Koonin, E. V. (1994). *Proc. Natl. Acad. Sci. USA*, **91**, 12091.
23. Bucher, P.and Bairoch, A. (1994). In *Proc of 2nd Int. Conf. On Intell. Systems for Mol. Biol.*, pp. 53–61. AAAI Press.
24. Sonnhammer, E. L., Eddy, S. R., Birney, E., Bateman, A., and Durbin, R. (1996). *Nucleic Acids Res.*, **26**, 320.
25. Nevill-Manning, C. G., Wu, T. D., and Brutlag, D. L. (1998). *Proc. Natl. Acad. Sci. USA*, **95**, 5865.
26. Saqi, M. A. S. and Sayle, R. (1994). *Comput. Appl. Biosci.*, **10**, 545.
27. Sayle, R. A. and Milner-White, E. J. (1995). *Trends Biochem. Sci.*, **20**, 374.
28. Smith, H. O., Annau, T. M., and Chandrasegaran, S. (1990). *Proc. Natl. Acad. Sci. USA*, **87**, 826.
29. Smith, R. F. and Smith, T. F. (1990). *Proc. Natl. Acad. Sci. USA*, **87**, 118.
30. Smith, T. F. and Waterman, M. S. (1981). *J. Mol. Biol.*, **147**, 195.
31. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). *Nucleic Acids Res.*, **22**, 4673.
32. Taylor, W. R. (1988). *J. Mol. Evol.*, **28**, 161.
33. Jonassen, I. (1997). *Comput. Appl. Biosci.*, **13**, 509.
34. Wallace, A. C., Borkakoti, N., and Thornton, J. M. (1997). *Protein Sci.*, **6**, 2308.
35. http://www.biochem.ucl.ac.uk/bsm/PROCAT/PROCAT.html
36. Brown, N. P., Orengo, C. A., and Taylor, W. R. (1996). *Comput. Chem.*, **20**, 359.
37. Taylor, W. R. and Orengo, C. A. (1989). *J. Mol. Biol.*, **208**, 1.
38. Artymiuk, P. J., Porrette, A. R., Grindley, H. M., Rice, D. W., and Willett, P. (1994). *J. Mol. Biol.*, **243**, 327.
39. Nussinov, R. and Wolfson, H. J. (1991). *Proc. Natl. Acad. Sci. USA*, **88**, 10495.
40. Jonassen, I., Eidhmmer, I., and Taylor, W. R. (1999). *Proteins: Struct. Funct. Genet.*, **34**, 206.
41. Karlin, S. and Zhu, Z.-Y. (1996). *Proc. Natl. Acad. Sci. USA*, **93**, 8344.
42. Taylor, W. R. (1999). *Prot. Sci.*, **8**, 654
43. Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). *J. Mol. Biol.*, **247**, 536.
44. Brazma, A., Jonassen, I., Vilo, J., and Ukkonen, E. (1998). *Genome Res.*, **8**, 1202.