REPORTS
IN
INFORMATICS

ISSN 0333-3590

Structure Comparison and Structure Patterns

Ingvar Eidhammer, Inge Jonassen,
William R. Taylor

REPORT NO 174 July 1999

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

Structure Comparison and Structure Patterns

Ingvar Eidhammer, Inge Jonassen
[ingvar,inge]@ii.uib.no,
Dept. of Informatics,
University of Bergen,
Norway

William R. Taylor
wtaylor@nimr.mrc.ac.uk
Division of Mathematical Biology,
National Institute for Medical Research,
London, United Kingdom

July 2, 1999

Abstract

This article investigate different aspects regarding pairwise and multiple structure comparison, and the problem of automatically discover common patterns in a set of structures. Descriptions and representation of structures and patterns are investigated, as well as scoring and algorithms for comparison and discovery. A framework and nomenclature is developed, and a lot of methods are reviewed and placed into this framework.
Key words: structure comparison, structure motifs

1 Introduction

Sequence and structure patterns can be used for characterising families of proteins which are defined to be sets of functionally or structurally related proteins. The discovery of such patterns can help in the understanding of relationships between sequence, structure, and function of proteins and in a bigger context help to understand the working of living organisms. Sequence patterns can be used to predict, from sequence information only, the structure and function of new proteins. Having found the sequence of a new protein, a database similarity search can identify proteins with similar sequences, and if the similarity is sufficiently strong, one can assume that the new protein has the same structure and sometimes function as the protein in the database. Sets of proteins having the same structure and or function can be analysed and one can find which residues are most informative about the structure/function, which are allowed to vary more freely etc. Generally one can define sequence patterns characteristic of each family. Then comparing a new sequence to a collection of patterns (one for each structure family) is more efficient and also provides a more sensitive and specific test of family membership.

However, even if the two proteins have similar structures, they can have different functions, as a small number of amino acid changes may be responsible for changing the function of the protein. Families of proteins having the same function (represented by the same or different structures) may be formed, and it may be possible to find a signature (pattern) which is characteristic of the function so that the new protein can be matched against, and the outcome of the matching may be used to suggest presence or absence of the particular function.

In this way patterns can play a major role in understanding relationships between different structures (to find recurring structural patterns and rules) and relationships between structure

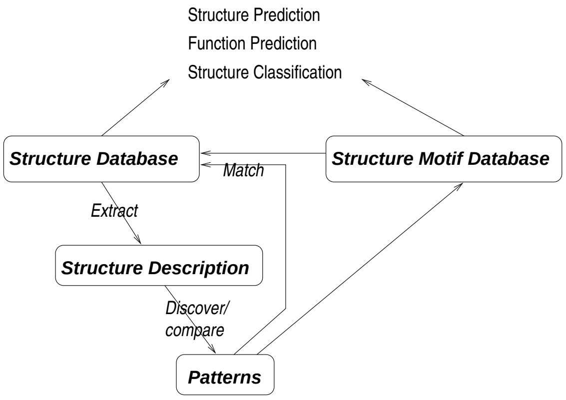

Figure 1: Protein comparison and motif discovery in a context.

and function.

Sequence patterns can be used for characterising proteins with structural similarity, for example the PROSITE entry for the SH3 domain which is a small protein domain of about 60 amino acid residues in two tightly packed anti-parallel beta sheets occurring in a variety of intracellular and membrane-associated proteins. The PROSITE entry gives a profile for this family. The function of the domain is not well understood, but if one finds that a new protein matches the profile with a sufficiently high score, one can be reasonably sure that the protein contains a SH3 domain. Patterns can also be used in relation to functional properties. For example, PROSITE gives a pattern characteristic for the sequences of a certain class of aminoacyl-tRNA syntethetases (a group of enzymes which activate amino acids and transfer them to specific tRNA molecules as the first step in protein biosynthesis). If a new sequence matches this pattern, one can hypothesise that it is an enzyme having this function.

As the number of known protein structures increases (10213 in the PDB as of June 1999) [BKW$^+$77], there is need for methods describing and revealing common functionally important units in (related) structures. Such units could be described by structure patterns, analogous to the sequence patterns. Such patterns can be used to classify protein structures into structure families, for example by description of common arrangements of secondary structure elements in the core of proteins, as described by [KLW96]. Structure patterns can also be used to make a hypothesis of the function of a protein, for example the "corrdinate templates" for finding Ser-His-Asp catalytic triads in the serine proteinases and lipases as reported in [WLT96]. See Figure 1.

The terminology used in the literature for describing common similarities between sequences or structures is quite confusing. The words pattern, motif, fingerprint, template, fragment, core and site are all used. By a pattern we here mean any description of sequence or structure properties for which one can either (1) decide whether a protein matches it or not, or (2) assign a value to the match between the protein and the pattern. In case (1) it will be called a *deterministic pattern* and in case (2) a *probabilistic pattern*. A pattern may or may not match any sequence or structure and it may or may not have a biological meaning. Different formalisms for describing patterns exist. Following the definition above, for any pattern formalism there is a mechanism for deciding whether or not a sequence or a structure matches a pattern. In most cases, a pattern is allowed to match part of a sequence or a structure. The part of a sequence or a structure (e.g., a subsequence or a substructure) that matches a pattern, is called an *occurrence* of (or a match to) the pattern.

For some pattern classes, one can define a partial ordering of the patterns with respect to their generality. If it can be proven that any structure that matches a pattern P_1 will also match pattern P_2, then P_2 is a *generalisation* of P_1 (or subsumes P_1) which we can denote by $P_1 \prec P_2$ (or $L(P_1) \subseteq L(P_2)$. This can also be extended to include structure descriptions, so that if pattern P matches structure S, then $S \prec P$ (or $S \in L(P)$). This is easiest to do formally for deterministic patterns (see e.g., [SA95, Con95]). For probabilistic patterns it is not that straightforward, but for example for alignments, a subalignment (consecutive columns) can be seen as a generalisation of an alignment.

By a *motif* we mean a pattern which has a biological meaning – i.e., it can be used to predict functional or structural properties of the protein, or it describes (non-trivial) features common to biologically (structually or functionally) related proteins. Having found a pattern, one wants to evaluate whether it is a motif, i.e., if it has a biological meaning. Since in automatic methods there is no way of assessing the biological meaning of a pattern directly, alternative, mathematical methods for pattern evaluation have been developed. We call them *fitness* or *scoring* functions. The term motif can include sites (a small part of the structure having a specific functional or structural role e.g., the active site in enzymes or metal binding sites), cores (an often bigger part of the structure in the interior of the protein e.g. the hydrophobic core), secondary structures and supersecondary motifs (constituted of secondary structures).

The **PDB** (Protein Data Bank, http://www.pdb.bnl.gov/) [BKW+77] at Brookhaven National Laboratory) is the international structure database. It is an archive of experimentally determined three-dimensional structures of biological macromolecules, together with an extensive annotation. A lot of other structural databases exists, see [dat99].

In the following sections we investigate different aspects regarding pairwise and multiple structure comparison, and the problem of automatic discovery of common patterns in a set of structures. Descriptions and representation of structures and patterns are investigated, as well as scoring and algorithms for comparison. A framework and nomenclature is developed, and a number of methods are reviewed and placed into this framework. The current article partly builds on and extends two newly published survey articles; [BOT96], which discuss structure comparison, and [BJEG98], which deal with the discovery of sequence patterns. Some method papers also include small reviews that have been useful in this work [GMB96, God96].

2 Framework

Figure 2 shows the over-all components of discovering structure motifs and matching new structures against them. The different steps involve:

1. *Feature extraction* – in this step the features to be used in the comparison of the structures or in the pattern discovery method, are extracted. This might include comprehensive computing, e.g. assigning secondary structures to the residues. Not all methods perform this step explicitely, but all take into account a well-defined set of structural features and conceptually this can be seen as if they first extract the relevant features and then subsequently works on the resulting structure descriptions.

2. *Comparison* – takes as input a pair of structure descriptions (or a pair of description/pattern) and finds (local or global) similarities between the two optimising a similarity measure and outputing a score. The similarity may also be represented as a pattern.

3. *Discovery* – patterns matching many or all of the input structures are found. The patterns are chosen from a solution space so that their fitness with respect to the input structures is as high as possible.

4. *Matcher* – takes as input one pattern and one structure and evaluates the match between the two, the output is 'yes' or 'no' if the pattern is deterministic or a score if the pattern is probabilistic. Note the similarity between the matching and comparison operations; they may some times be implemented using the same method.

The steps involved in these operations will be discussed in more detail in the following sections.

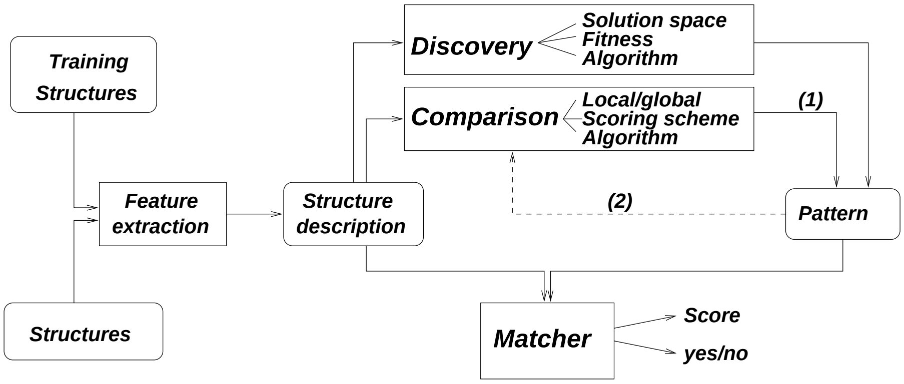

Figure 2: Overview of the process of discovering patterns, either by direct discovery method or using a comparison algorithm, and pattern matching. The edge marked (1) indicates that the result of a comparison method may need some further processing in order to be expressed as one or several patterns. Furthermore, if the result of comparing two structure descriptions can be represented as a pattern and then compared with another structure (or another pattern), the edge (2) can be used to find patterns matching progressively bigger sets of structures. Note that "Matcher" and "Comparison" are very closely related and may in fact be implemented using the same method.

3 Descriptions and representations

3.1 Feature extraction

This step is used to extract the features from the structure file which are relevant for pattern discovery, learning, matching and comparison.

When deciding which features should be extracted and represented, certain aspects must be considered. First, one needs to decide on which structure level similarities are sought (e.g., atom group, residue, secondary structures). Also, one needs to decide whether the similarities should require sequence order to be preserved, a reasonable requirement if we assume that the proteins are evolutionarily related, but not if common features have arisen independently. The structure description to be used as input to the comparison or pattern discovery algorithms should contain only the features which we would like to compare and/or to describe as patterns. A structure description will consist of geometry, topology and properties, and for each of these we need to decide which features to include:

1. Geometry – e.g., coordinates or relative positions of atoms, residues, fragments, or SSEs.

2. Topology – e.g., the elements' order along the backbone.

3. Properties – e.g., physico-chemical properties of the elements (e.g., residues)

In our framework, patterns are to be found from structure descriptions so that they represent features common to a set of such descriptions. Patterns will therefore be generalisations of structure descriptions, and are limited to features included in these.

In order to provide the comparison (pattern discovery) algorithms with a good starting point, the structure descriptions should ideally satisfy the following properties:

1. *Invariant to trivial changes*, such as translation, rotation

2. *Robust* – the description should not change drastically due to minor changes in the structure (e.g., experimental errors, hydrogen-bonding definition)

3. *Similar* structures should get similar descriptions

4. *Different* structures should get different descriptions

Loosely, the definition of 'similar' and 'different' will depend on what aspects of protein structure one wants to compare or capture in a pattern. For example, if the comparison is to be done at the level of packing of secondary structure element, the descriptions of structures with similar packing should be similar.

3.2 Structure descriptions

A natural way to describe a complex object like a protein structure is to break it into pieces (units) and to describe each unit separately and (most often) the relationship between the units. For structure comparison, the units are chosen in two ways:

1. *Element based* – natural structure elements (atoms, residues, fragments, secondary structures (SSEs)) are used as the basic units.

2. *Space based* – the space in which the structure is located is divided into (possibly overlapping) geometrically defined cells, e.g., by using a grid, or shells around center points.

3.2.1 Element based descriptions

Element based are the clearly most used type of descriptions. Brown et al. ([BOT96]) give an overview of structure comparison methods using such description and present a nomenclature for use in comparison of structures. Following this, three concepts can be used for description: *element class, properties,* and *relation* (Brown et al. use the word 'feature', we use 'property' to avoid confusion with our use of 'feature' at the higher level).

- *Element class* – the level of the description varies: atom (group), residue, backbone fragment, and secondary structure element.

- *Property* – is used for specifying the properties of each element, such as three-dimensional coordinates, physico-chemical properties, amino acid type, secondary structure type, curvature and torsion.

- *Relation* – is used for describing the relation between the elements. In practice the relations are binary, such as geometrical distances, difference in orientation, and bonding.

Depending on what element class is used, one might say that the description is on a coarse or a fine level. Secondary structure is on a coarse level, and the common similarities in this case are often common folds. Atom (group) and residue level are fine, while backbone fragment level might in some cases be considered as low, in other cases as high.

The two levels have different goals, the high level aims to classify the whole overall structure, but the low level aims to be used more directly in determining/finding active/binding sites.

3.2.2 Space based descriptions

In space based descriptions, the 3D space in which a structure is located, is divided into (possible overlapping) cells. The part of the structure falling in each cell is described, for example by the number of residues of each type, or the number of residues with certain property values. Examples of space based descriptions are in [BA95] and [KKS98]. Figure 3(a) shows cells as shells around points in the structure space.

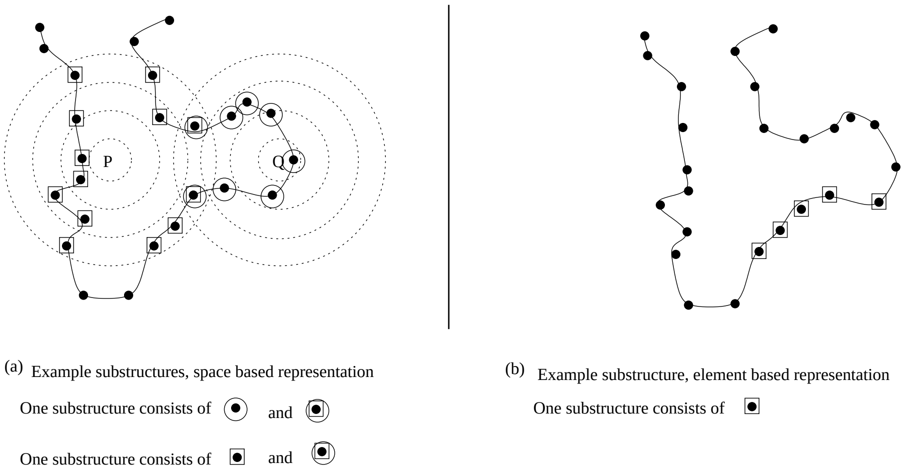

Figure 3: Substructures illustrated in 2D. The structures are shown by solid curves, the dots represent elements. (a): Two substructures in a space based description, one is the part of the structure falling in the shells around P, the other is the part falling in the shells around Q. (b): A substructure in an element based description, where a substructure is a fragment (consequent residues)

3.2.3 Substructure descriptions

Some methods divide the structures into substructures, describe each substructure, and then find (local) patterns common to substructures from different structures. This initial step is often followed by a step where the identified local patterns are combined to form larger patterns. For example, Escalier et al. use this approach and element based structure descriptions. They define a substructure to consist of elements where the distance between any two elements is below a given threshold [EPSV98].

The approach can also been used together with space based structure descriptions. For example, Bagley and Altman [BA95] define a substructure as the part of the structure that falls within a certain distance from a specified point. Similar sites constitute a pattern, and no combination of (local) patterns are made. Figure 3 illustrates substructures.

3.3 Structure representation

For each protein, the selected features are represented in a data structure. Typically each unit (element or cell) is represented separately and combined to form a representation of the whole structure. For example, each residue can be represented by its coordinates and amino acid, and the whole protein structure can then be represented as a list (ordered) or set (un-ordered) of residue representations.

We have grouped the different representations into five groups:

1. *Strings* – It is possible to represent structural features as one or several strings, for instance [MTH97] where residue i is represented by a letter showing the C_α's position relative to the positions of the C_α of residue $i-2, i-1, i+1$. The representation is appropriate only for matching of (consecutive) substrings.

2. *List of unit descriptions* – For example extract one or several coordinate per residue: the coordinates of the C_α atom ([TO89]), mean coordinates of side chain ([JET99]), use two pseudo atoms ([APG$^+$94]), or unit vectors (the vectors between succeeding C_α atoms, [CHKK99]). In addition to the coordinates, each unit can have associated additional properties, such as amino acid type, physico-chemical properties, degree of burial, SSE status, etc.

3. *Set of unit descriptions* – Same as list, but the units are unordered. Alexandrov et al. [ATG92] in their SARF program use a set of backbone fragments.

4. *Graphs* – A labelled graph can be used to represent all element based feature descriptions, nodes representing the elements and the edges represent the relations. For instance, identify secondary structure elements and label the edges with the distance and angular relationships between the SSEs of the nodes ([GARP93]), or label them with the type of parallelism ([KLW96]).

5. *Feature arrays* – Features of the structure can be represented by fixed length arrays. This can be used for both space based and element based description. [BA95] have a space based description where each cell is described by an array summing up the properties of it's residues.

3.3.1 External/internal representations

For reason of efficiency most of the methods use internal representation of the structures, hidden from the user. Internal representations can for example fascilitate more direct comparison of spatial similarity. In the "double dynamic programming" method by Taylor and Orengo [TO89], a local coordinate system is constructed for each residue into which the remaining residues' coordinates are transformed. Then, assuming a particular residue pair is aligned, the spatial similarity of the remaining residues can be assessed and scores assigned to be used in a (lower level) dynamic programming alignment step (see Section 4.4).

The same approach is used in geometric hashing [FWLN94, WLT96, ANW96]; a local coordinate system is constructed for each residue, but in geometric hashing normally the coordinates of only the neighbouring residues are transformed. Then for a pair of residues (one from each structure), their spatial neighbourhoods can be compared simply by counting the number of neighbouring residues with similar position (in the respective local coordinate systems). Special hash tables are used to speed up the computations (see Section 4.6).

Another approach is taken by Jonassen et al [JET99] who represent the spatial neighbourhood of each residue by a string of the spatially close residues (having spatial distance below some threshold). The strings do not contain information about the exact geometry of the neighbours, but do give the sequence order (along the backbone) of the neighbouring residues. Patterns shared by sets of these strings can be discovered using efficient sequence pattern discovery methods (see Section 5.5).

The external representations can be generalised to be useful for pattern descriptions, which is not the case for the internal representations. When algorithms using an internal representation are to present the result to the user as a pattern, it must be described in another form. For example, each pattern found by the SPratt method [JET99] is presented as a list of residues, for each residue is given the allowed amino acids and the 3D coordinates.

4 Pairwise structure comparison

In this section we discuss methods for comparing pairs of structures, focusing on element based structure descriptions. The most natural way to compare two objects each represented by a collection of elements, is to try to find element correspondences between the two. More formally, for two objects A and B having elements $a_1, a_2 \ldots, a_m$, and $b_1, b_2, \ldots, b_n$, respectively, we define an *equivalence* as a set of pairs $L(A,B) = (a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), \ldots, (a_{i_l}, b_{j_l})$. The equivalence is called an *alignment* if the elements of A and B are ordered and if the pairs in $L(A,B)$ are co-linear, i.e., if $i_1 < i_2 < \ldots < i_l$ and $j_1 < j_2 < \ldots < j_l$. Many different equivalences exist and a scoring function is needed to rank them and to discriminate good equivalences from bad ones. We first briefly review different scoring schemes proposed for pairwise structure comparison. Then we give an overview of different algorithms, which given two structure descriptions and a scoring scheme, aims to find the equivalences giving the highest score. A schematic overview of the steps involved is given in Figure 4.

The general problem is NP-hard ([Lat94]), therefore some simplifications have to be done, either in the search or in the scoring.

4.1 Scoring equivalences

Assuming that one want to assign high values to 'good' equivalences the comparison problem is one of finding an equivalence with as high as possible score (maximization problem). The score of an equivalence will be high if the pairs are between elements with similar properties (for coordinates; if they can be superpositioned well – see below) and if relations between pairs of paired elements are similar.

When comparing two sequences, the traditional way is to align the sequences to pair up identical or similar residues. The score of the alignment is assigned using the similarity of the aligned (equivalenced) residues (as measured by a scoring matrix, e.g., [Day78, HH92]) and as a function of the gaps inserted in order to obtain the alignment [SK83]. Different scoring schemes apply depending on whether the alignment is to be local or global. Pairwise sequence comparison is very widely used, in particular in database similarity searches. For each similarity found, the alignment and its score should be given and also the statistical significance of the hit. Most often this is given as an estimate of the probability of finding a hit at least this strong (having the same score or higher) by chance [LP85, AGM$^+$90].

When comparing structures, an intuitive way is to try to put the structures on top of each other so that the equivalenced elements come as close as possible. The obtained distances can be used to quantify the similarity and to score the equivalence. This is called *superposition* of structures and if the geometry of the structures are not changed in the process, it is refered to as rigid-body superposition.

Algorithms exist for superposing structure A on structure B by finding the superposition (*translation*, 3 distances and *rotation*, 3 angles) to minimize the *coordinate* root mean square deviation ($RMSD_C$) given by

$$RMSD_C = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(x_i^A - x_i^B)^2} \tag{1}$$

where $(x_1^A, x_1^B), \ldots (x_N^A, x_N^B)$ are the coordinates (after superpositioning) of the equivalenced elements. For residue level structure descriptions, there may be one coordinate set per residue (e.g., C_α atom) or a number of coordiante sets per residue. An alternative measure is *distance* RMSD ($RMSD_D$). This alleviates the need for finding a translation and rotation of one of the structures and is given by.

$$RMSD_D = \frac{1}{N}\sqrt{\Sigma_{i=1}^{N}\Sigma_{j=1}^{N}(d_{ij}^A - d_{ij}^B)^2} \tag{2}$$

where each d_{ij}^T is the spatial distance between elements i and j in structure T. The two measures are linearly related except that $RMSD_D$ is invariant under reflection.

It is observed [BOT96], that the translation is effected by relocating the origin of the coordinate system of each protein to the centre of mass of its equivalenced elements. Finding the best superposition is easily done by standard pairwise least square fitting algorithms (e.g. [Kab78], finding eigenvalues in a matrix constructed from the two coordinate sets).

Most scoring schemes for evaluating equivalences between structure descriptions contain factors related to the $RMSD_C$ or $RMSD_D$. Related measures include distance map similarity [HS93] or contact map overlap [GS94]. Many structure comparison programs give as output an equivalence and the resulting $RMSD_D$ (or a closely related measure such as weighted $RMSD_D$ or maximum $RMSD_D$) even if they internally do not use $RMSD_D$ to score equivalences (e.g., [TO89]). Internal measures can use information about for example amino acid types, physico-chemical properties, exposure/buriedness, secondary structure elements, and chemical bonds. Not all methods specify explicitly a scoring function to be optimised but is rather based on a combination of (often heuristic) steps [TO89].

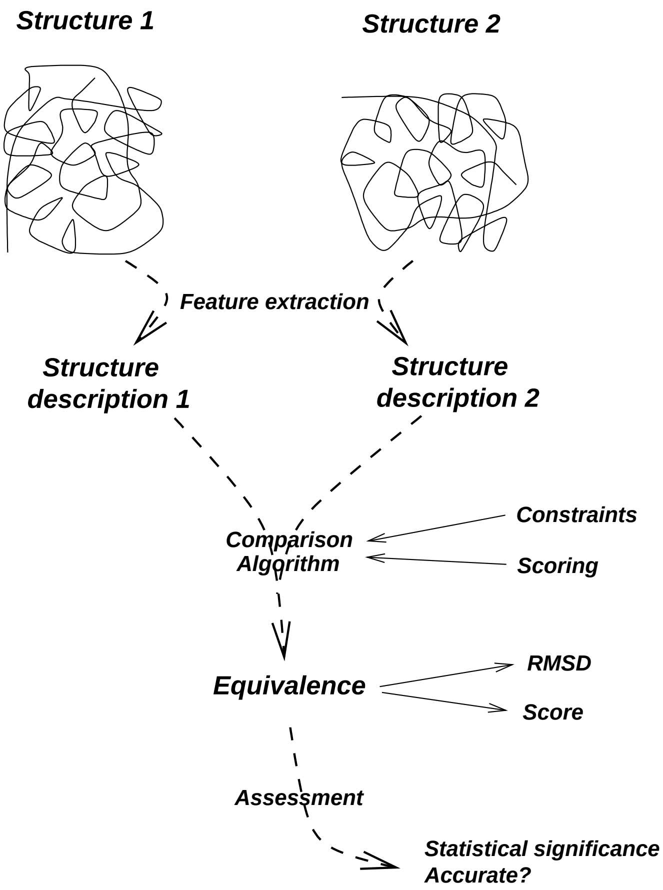

Figure 4: Framework for pairwise structure comparison. First the relevant features are extracted and represented in structure descriptions. These are input to a comparison algorithm which finds an equivalence obeying certain constraints with as high scores as possible. The equivalence is output together with RMSD value. It can be assessed with respect to whether it is accurate (if a standard of truth is available) and to whether it is statistically significant – i.e., if the similarity is stronger than could be expected by chance.

Falicov and Cohen [FC96], use a scoring based on minimal surface metric. Given a superposition of two structures, a surface between them can be described by a list of triangles, of which there are two types. Let a_i be the C_α of i'th residue in one of the structures, and b_j of the other. Then the triangles are either (a_i, a_{i+1}, b_j) or (a_i, b_j, b_{j+1}). An atom might be vertex in several consecutive triangles, thus allowing for insertion/deletion in a corresponding alignment. The triangulation corresponding to the minimal surface can be found by a dynamic programming procedure, and this gives the score for the transformation.

NEW! Another alternative scoring is URMS introduced by Chew et al. [CHKK99]. A structure is represented by unit vectors $\{v_i\}$, starting in a common point, where v_i is the unit vector along the vector $C_{\alpha_i}, C_{\alpha_{i+1}}$. URMS is then the minimum RMS distance for a pair of sets in that unit-vector representation. They claim that this measure is effective for finding small substructures shared by two structures.

Assessing the significance of RMSD values obtained for candidate equivalence sets, one first of all needs to consider how many elements were equivalenced since for random comparisons the expected RMS value is proportional to the square root of the number of equivalenced residues. Often rules of thumb are used. See [LG98, GL98] and Section 6.

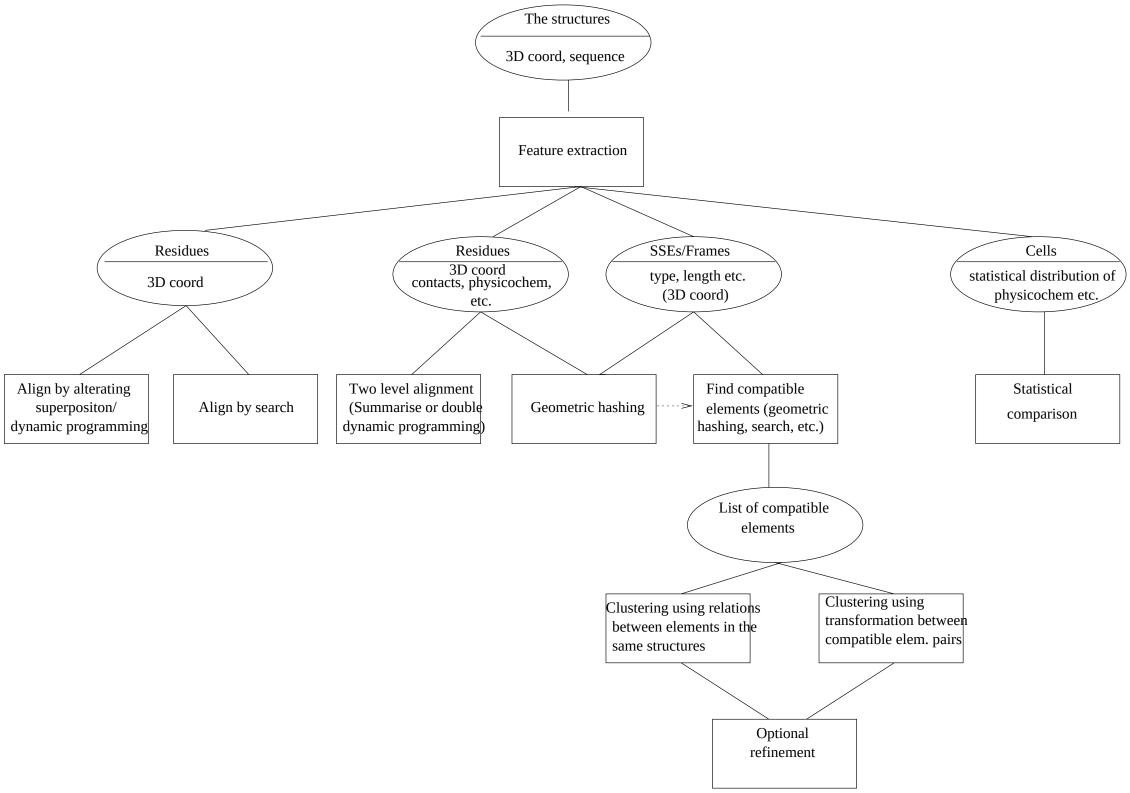

Figure 5: Schematic diagram of the different approaches for pairwise comparisons. The ellipses representing information, the rectangles processing. The dotted line indicate that one process might be used by the other

4.2 Comparison algorithms

A large number of methods exist for comparing pairs of sequences, pairs of structures, and also pair of sequence and structure. The methods are used not only for comparing a given pair of objects, but also in database searches where the objects most similar to a query object are to be found as quickly as possible. A classification of the methods can be done along different axis, for example the level of representation (atom, residue, frames or secondary structure element), the constraints on topology (content, sequential, non-sequential [BOT96]), or techniques (superposition, dynamic programming, graph theory, geometric hashing etc).

In this survey we focus on the *algorithms*, and make a classification of *approaches*. Our classification of the pairwise comparison methods comprises alignment based methods, search methods, geometric hashing, statistical approaches, and clustering.

For each approach we explain the fundamental principles, and also explain (on a coarse level) some of the published *methods* in which the approach is used, focusing on the most recently published methods. Figure 5 shows a schematic overview and connections of the different approaches.

Pairwise structure alignment includes methods for aligning sequences where structural information is used, and the colinearity is maintained. The most fundamental techniques for comparing structures are superpostion and dynamic programming.

Dynamic programming (DP) [NW70] for finding the best alignment of two sequences or structures uses a scoring matrix with a score for each pair of residues. One normally starts from the beginning of each structure extending the alignment towards the end, filling in a DP matrix. An important assumption is that the optimality of earlier made choices is not affected by decisions made later on, that is, when filling in an entry in the DP matrix, one does not have to go back into already filled-in table entries to change the values. This assumption holds for sequence alignment

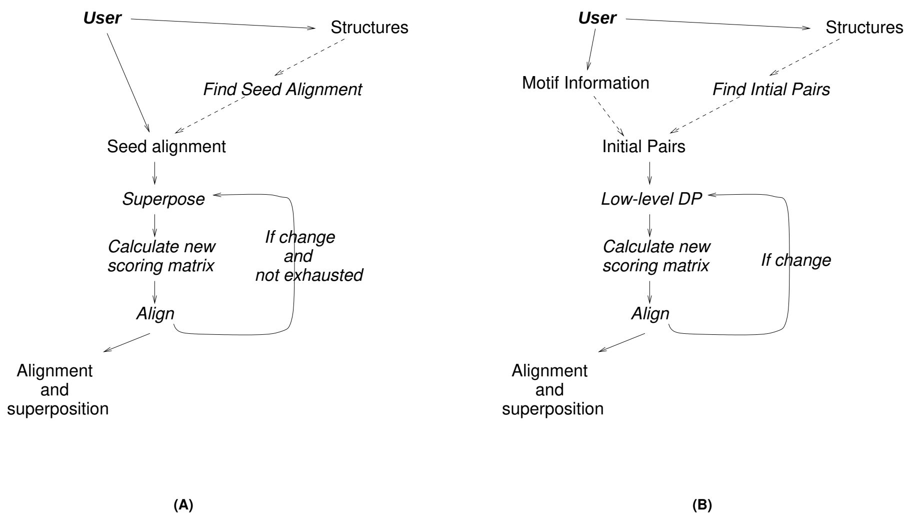

Figure 6: (A): Outline of algorithm alternating between alignment and superpositioning. (B): For comparison, outline of the SAP method.

using typical scoring schemes, e.g., edit distance or sum-of-pairs, but typically not for structure alignment. In the words of [GL98], "structural alignment fails to converge globally because the possible matches for different segments are tightly linked as they are part of the same rigid 3D structure".

4.3 Iteration of alternating superposition - aligning

Given an equivalence of atoms from each structure, E_0, a superposition algorithm can be used to find a transformation T_0 minimising an RMSD measure. When the whole structures are superposed using this transformation, the distances between all pairs of atoms (residues) from the two structures, can be used to define a new scoring matrix. This matrix can be used for a new alignment, and the equivalences (pair of residues) with least distances can be chosen for a new superposition. This iteration continues until convergence (the alignment is not changed) or some maximum number of iterations is done. Figure 6 illustrates the approach.

Rao and Rossmann [RR73] and Rossmann and Argos [RA75, RA76] first used this approach to align protein structures. Initial equivalences are found by searching for pairs of similar short fragments (3-4 residues). In the alignment phase, a constraint is used to assure co-linearity (dynamic programming is not used). The scoring is determined by a probability function of the distances between the residues, and a measure of their conformational similarity determined by use of the coordinates of the two (sequence) neighbouring residues. In a related method, Cohen and co-workers [SCPD86, Coh97] use dynamic programming in the alignment step. This is also used by Russel and Barton in the program STAMP [RB92], where the scoring matrix is the probability function of Rossmann and Argos. An analog iteration is used in the program CONTRUST of Ding et al. [DQF94], where the scoring between two residues consists of three components: sequence, local structure, and global structure components. The local structure component depends on the difference between the curvature and torsion calculated for the residues, and the global structure component depends on the distance between the two residues, after superposition is done.

The idea of Rossman et al. of automatically finding an initial equivalence is also used by Zu-Kang and Sippl [ZKS96]. Recognizing that the initial seed alignment is critical to the final result, they try different possible seed alignments and conclude that depending on the seeds chosen, the alignments can be quite different (not share many residue pairs) while still giving similar quality measures (RMSD and number of aligned residues).

Petitjean [Pet98] uses iteration to find the "best" common subset of C_α coordinates of two structures, measured by RMSD. The initial alignment is chosen randomly, and as in the above methods, iterations with alternating superpositioning and alignment is performed to maximise the number of C_α coordinates aligned. An analog iteration is also used by Gerstein and Levitt [GL98] in an extension of the ALIGN program [Coh97].

The method of performing alternating superpositiong and alignment has also been used for refining (post-processing) the results found by other methods (e.g., [HS95], Section 4.6).

4.4 Two-level alignment

Traditional dynamic programming (DP) guarantees to find an optimal alignment of sequences or structures if the scoring of matched pairs of elements are independent of eachother, and if gaps inserted can be penalised independently. When one assumes that the structures are already superposed, reasonable scoring schemes can be devised which allows the structures to be aligned optimally using DP (see Section 4.3). However, ideally, one might wish to simultaneously align and superpose the structures to optimise a score depending on how well aligned substructures superpose. Since any choice to align two substructures affect the scoring of the alignment of the complete structures, the independency requirement is violated and DP no longer can guarantee an optimal solution. Several heuristic algorithms trying to extend DP to solve this problem have been proposed.

A heuristic method was presented by Sali and Blundell [SB90]. Several alignments are made, one for each type of relation, each alignment is obtained using stochastic optimisation (simulated annealing). The results are summarised into a residue-by-residue scoring matrix U where a residue pair is assigned a high score if the two residues are aligned in many of the relationship based alignments. Finally U is combined with property information to obtain a new scoring matrix, which is used in an alignment using dynamic programming.

Another method based on DP has been used in the program SSAP by Taylor and Orengo [TO89]. The method is called double dynamic programming since DP is used at two different levels. At the lower level a series of DP matrices are calculated. For each the alignment of one particular pair of residues is fixed. The highest scoring path from each lower level DP matrix is propagated to the higher level "summary" DP matrix in which DP is done to find the over-all best alignment.

We will describe this algorithm in a bit more detail. Let i, k be residues in the two structures A, B, respectively, and let ${}^{ik}R$ be a lower level score matrix. The element ${}^{ik}R_{jl}$ is a score showing the goodness of aligning A_j, B_l given that A_i, B_k is aligned. For finding the score, local reference systems are defined at A_i and B_k and the coordinates of the remaining residues in each structure are transformed into these coordinate systems (see Section 3.3.1). The score of aligning A_j, B_l depends on the distance between A_j and B_l in the respective coordinate systems defined by A_i and B_k, respectively.

The idea of Taylor and Orengo is similar to that used in the methods alternating between superpositioning and alignment (Section 4.3). While the latter methods find the DP matrix using an optimal superposition for the current equivalence, Taylor and Orengo do not need to decide (assume) one exact alignment to calculate the higher-level DP matrix. Instead the residues pairs that participate in high-scoring lower-level alignments receive high values in the higher-level DP matrix and are likely to be included in the final alignment.

More recently, an iterative version of the SSAP algorithm (called SAP) has been proposed by Taylor [Tay97] which is even more closely related to the methods of Section 4.3 (see Figure 6). In this version, lower level DP matrices are only calculated for a subset of the possible residue pairs. In the first iteration, an initial set of residue pairs is used (randomly chosen or using local seed as secondary structures, burial or motifs [JET99]), and the set to be used in each subsequent iteration is calculated using the result of the last (high-level) alignment. The procedure can be described as below, where $DP(R)$ is the result of aligning A, B using the matrix R, where R_{ik} is the score for aligning A_i, B_k. $DP(R)$ therefore returns a list of indices.

$Q := 0$ *Set high level score matrix to zero*
$I :=$ list of seed pairs
iter
  $S := \{0\}_{m\times n}$ *Set accumulated low level score matrix to zero*
  for each pair $(ik) \in I$ **do**
    $L := DP(^{ik}R)$ *Low level dynamic programming*
    for each pair $(r,s) \in L$ **do** $S_{rs} := S_{rs} +^{ij} R_{rs}$ **end**
  end
  $Q := \frac{1}{2}Q + \log(1 + S/10)$ *Update high level score matrix*
  $L = DP(Q)$ *High level dynamic programming*
  $I :=$select new pairs
until convergence

4.5 Search

Several methods use general techniques for search in a well-defined search space. For example, May and Johnson [MJ95] use a genetic algorithm to find the superposition giving the best alignment. The search space is a set of transformations. Each transformation is scored by the score of the best alignment of the two structures under this transformation. The alignment is computed using dynamic programming where the score for a residue pair (one residue from each structure) is defined depending on the inter-residue distance under the transformation.

A related method presented by Diederichs [Die95] also performs a search for the best transformation. The search space is defined by a "Lattmann" angle space as a set of rotations associating equal volumes of angle space. For each rotation, a "translation grid" with 1Å spacing is defined and for each grid point it is counted how many C_α pairs (one from each structure) superpose well. This is used as one component of the scoring of the transformation. Optionally, the similarity of aligned residues and topological matching can be included in the scoring. Each good superposition is checked for local or global sequentiality. Local sequentiality is a measure of how frequently contiguous (backbone) fragments are superposed onto residues having the same sequence order. Global sequentiality is measured by the correlation between the indices i, j of matching residues. Different scoring schemes are used depending on whether, in addition to similarity in secondary structure architecture, one also wants topological similarity and whether direction of SSEs are taken into account.

Falicov and Cohen [FC96], also search in the space of transformations. Each transformation is scored by the minimal surface metric, see Section 4.1. Three techniques were used for the search: Downhill Simplex, Powell's Method and Conjugate Gradient.

Holm and Sander [HS96], describe a method based on a threading algorithm for sequence-structure comparison by Lathrop and Smith [LS96]. The sequence of their method is replaced by the structure (residue centers). The method is a branch-and-bound search method, where the solution space consist of all possible placements of the residues in a structure B relative to the SSE-segments of residues of another structure A. The structures are represented by distance matrices, and the upper bounds are calculated using distance matrices.

4.6 Geometric hashing

The aim of this technique is to discover common substructures, i.e. set of elements from the two structures with the same mutual spatial relations. A highly-redundant representation of the structures is used, which is *independent of rotation, translation, and sequence order.* This is normally obtained through the definition of local coordinate (reference) systems into which the coordinates of all (or a selected subset of) the elements (atom groups/residues/SSEs) are transformed. The methods use hash tables for storing and comparing local geometrical information (hence the name geometric hashing). The local reference systems are defined by use of 3D coordinate frames, and three points are needed for each frame (xyz). The principle for the technique will be explained by

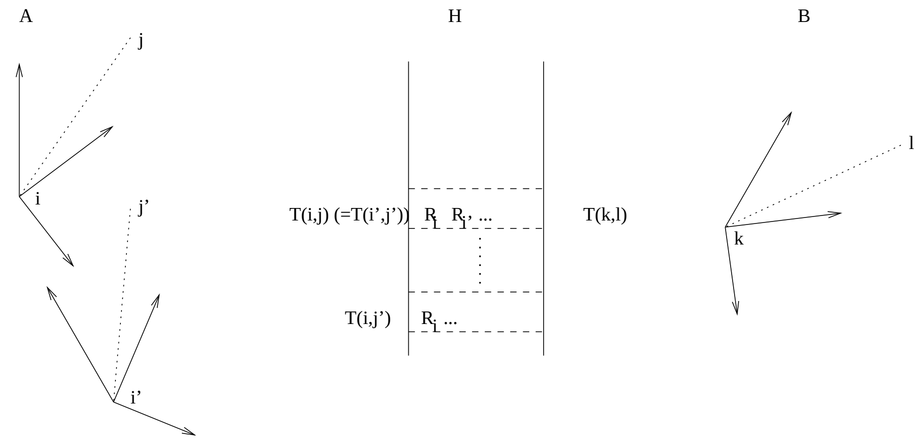

Figure 7: A simple model for geometric hashing. The hash table is filled from structure A using the preprocess algorithm. When looking at the pair (k, l) when B is prosessed by the recognize algorithm, both $S(i)$ and $S(i')$ are increased by one.

a simple example using residue based representation, where each residue is represented by the 3D coordinates of an atom (e.g., C_α).

Let A (the target) and B (the query) be two structures with m and n residues, respectively. For each $i, i = 2, ..., m-1$, define a reference system (R_i) by use of the $i-1, i, i+1$'th residues of A, and let $T(i,j)$ be a transformed representation of the position of residue j in reference system R_i. Further, let H be a (one dimentional for simplicity) hash table, indexed by $T(i,j)$, such that an entry in H contains a list of reference systems. The hash table is filled by using all reference systems from A, by the following procedure:

Preprocess:
for $i = 2, ..., m-1$ **do**
  calculate the reference system R_i
  for $j = 1..m, j \neq i$ **do** calculate $T(i,j)$; $H(T(i,j)) := H(T(i,j)) \cup (R_i)$ **end**
end

For each pair of elements i in A and k in B, we count the number of element pairs (j, l) such that j has the same spatial relation to i as l has to k, and retain those for which this number is above a threshold (t). Then the pair of elements (i, k) is likely to be part of a common structure. The following procedure can be used (for illustration see Figure 7):

Recognize:
for $k = 2, ..., n-1$ **do**
  calculate the reference system R_k
  set $S(i)$ equal 0 for all $i, i = 2, ..., m-1$ *For counting*
  for $l = 1..n, l \neq k$ **do**
    calculate $T(k,l)$; **for** each $(R_i) \in H(T(k,l))$ **do** $++S(i)$ **end**
  end
  $C = C \cup \{(R_i, R_k) | S(i) > t\}$
end

The pair of elements (i, k) in C are likely to be part of a common substructure. To find the actual substructures, clustering can be performed (see Section 4.7).

All triples of points can be used for defining reference systems. In the above example, triples of residues were used. Since the number of reference systems can be very large, it may be desirable to use heuristics to reduce the number of triples to be considered. Also, labels can be assigned to the elements (Nussinov and Wolfson [NW91]), and corresponding elements can be required to have equal labels (implemented by including the label in hashing $T(i,j,label)$). Also it is used to simultaneously compare a structure to a set of structures $A_1, A_2, ..., A_n$. The hash table is filled with data from each of the structures, but each data marked with the corresponding structure [NW91].

Holm and Sander have proposed a method where geometric hashing is used with SSEs as elements [HS95]. The SSEs are represented as vectors, and right-handed coordinate systems are defined by using ordered pairs of SSEs and assigning the origin to be at the midtpoint of the first vector, and directing the y-axis along that vector. By definition, the midtpoint of the second vector lies in the z-positive yz halfplane. (A similar reference frame was used by Orengo et al. [OBT92].) The SSE vectors are stored at the location of their midtpoint coordinates, for every coordinate system. Each cell in the hash table contains (a pointer to) a list holding the explicitly transformed coordinates of the SSE vectors, the elements constituting the coordinate system, and sequential number and type of the stored element. When deciding a match (i.e. element q in the query structure has the same relationship to two other elements (k,l) as an element p in the target has to (i,j)), the position of the midtpoint of q and p, and the directions, must be similar to a specified degree, as well as equal SSE type, and sequential position (before or after). The hash table contains additional information to be used under the comparison, such that finding the common substructures can be done without clustering. A refinement procedure analog to the iteration described in Section 4.3 is then performed.

Geometric hashing is often used for finding compatible elements in methods using clustering, see below.

4.7 Clustering

The result of a pairwise structure comparison is often a set or list of element pairs. Each pair contains one element from each structure and may have a score associated with it. The elements in each pair must be *compatible*. The requirement for compatibility varies. Residues may be defined to be compatible if they have the same amino acid or have amino acids in the same group. Secondary structure elements may be compatible if they are of the same type, or have equal internal distances. Two or several pairs of compatible elements are *consistent* if the substructures consisting of the elements satisfy come constraints (e.g., the substructures are considered as similar).

The methods using clustering follows the scheme 1) Find pairs of compatible elements 2) Cluster using consistent compatible pairs to find the (k) substructure(s) with highest score 3) Optional refinement.

Let (A_i, B_k) and (A_j, B_l) be two compatible pairs. They can be joined (are consistent) if the substructure consisting of (A_i, A_j) is "similar" to the substructure consisting of (B_k, B_l). To decide this, either *relations* between elements of the same structure are used, or *transformations* between compatible elements from different structures. Figure 8 illustrates this, with examples. Transformation in the most cases mean the transformation for an optimal superposition, consistency exists if two transformations are similar enough.

The clustering is then done based on either transformation or relation, or in some cases on both. In addition, an overall score might be used. The clustering problem is NP-hard, and heuristics are often used in practice. Several clustering techniques for use in comparing structures have been proposed in the literature. Each aims to group element pairs together so that the groups make up as big or as high scoring as possible similar substructures. The methods differ in the following aspects:

- how the joining is done, if only a cluster and a pair can be joined, or also two clusters
- how to test for consistency, is the new pair tested against the cluster as a whole, or against each pair in the cluster. Let a cluster $C = ((A_{i_1}, B_{k_1})(A_{i_2}, B_{k_2})...(A_{i_m}, B_{k_m}))$, and a pair $P = (A_i, B_k)$. If the consistency is transitive, it is enough to test P for consistency against one of the pairs in C, if not it must be tested against all.

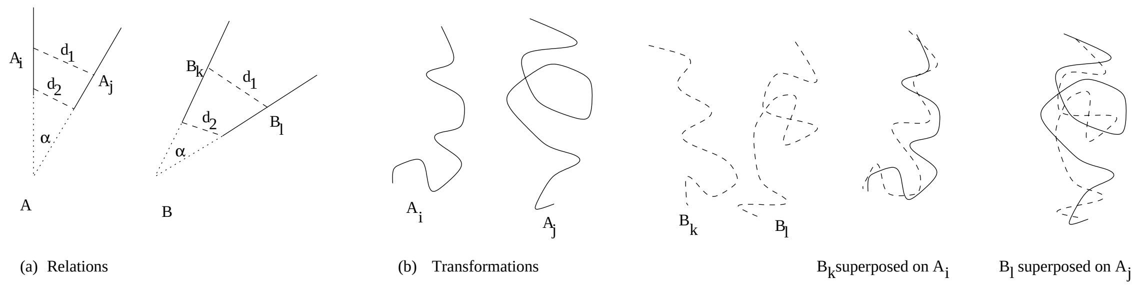

Figure 8: Examples of relation and transformation. A_i, A_j are two elements from a structure A, and B_k, B_l the same for structure B. Assume (A_i, B_k) and (A_j, B_l) each are compatible. (a) Relations, the elements represented as vectors. The relation is between elements from the same structure, here represented by two distances and an angle (found by projection on a plane). The substructure (A_i, A_j) is consistent with (B_k, B_l) if the relation $(d_1, d_2, \alpha)_A$ is similar to the relation $(d_1, d_2, \alpha)_B$ (b) Transformation, the elements represented by the 3D coordinate for each residue. The transformation is between elements from different structures, in the example it is the transformation for the best superposition. If the transformation for the best superposition of B_k onto A_i is similar to the one for B_l onto A_j, then (A_i, A_j) is consistent with (B_k, B_l).

- how is the next pair for clustering choosen
- can *parts* of elements be joined to clusters
- are the resulting clusters disjoint or overlapping

The clustering methods generally allow for insertions and deletions and topological permutations (but there exist exceptions).

4.7.1 Clustering by use of transformations

The methods discussed in this section follow the scheme: 1) Find compatible pair of elements 2) Find the optimal transformation between the compatible pairs 3) Cluster compatible pairs using similar transformations 4) Possibly perform final refinements.

The most widely used method for finding the compatible elements is geometric hashing. For each pair of compatible elements, a transformation describes how to transform the first element onto the second such that the "distance" between them becomes minimal. Often *incremental* clustering is performed where it is tested for one pair at a time whether it can be included in the cluster by comparing its transformation to that of the first pair in the cluster.

Using geometric hashing (see Section 4.6), Nussinov and Wolfson first find pairs of compatible triplets. For each pair of compatible triplets, a transformation is calculated from one of the reference systems to the other, and in a clustering step, pairs with similar transformations are grouped together. The best superposition for the joined element sets is found.

Vriend and Sander [VS91] cluster pairs of fragments. The fragments have at least some minimum length, and only fragments of equal length can be compatible. Compatibility is decided by first using distance geometry (internal C_α-distances). Those having similar internal geometry are superposed and two fragments are compatible if the RMSD, and the longest $C_\alpha - C_\alpha$ distance between equivalent alpha-carbon are below specified upper limits. The compatible fragments might be elongated by adjacent residues. Then an incremental clustering is done, where a new fragment is compared to the first member of the cluster. To decide if the pair (A_i, B_k) is consistent with (A_j, B_l), first the distance between the center of mass of A_i and A_j is compared to that between B_k and B_l. If this is satisfactory, the rotation of the best superposition of (A_i, B_k) and (A_j, B_l) are compared. After clustering, a final adjustment is done. In principle each fragment pair should be used to start a new clustering process.

The SARF program by Alexandrov et al. [ATG92] also use fragments as elements. Compatibility is decided by superposition (of fragments of equal length), and a compatible pair is given a score depending on the RMSD and the length. The clustering procedure is 'best fit': a list of temporary clusters exists, initially every compatible fragment pair is a cluster. At each iteration in the clustering, all pairs of current clusters are temporarily grouped, and a score is calculated for each group. The group with the highest score is kept permanently. This continues as long as a clustering increases the score. In this manner each fragment pair does belong to only one cluster.

Another example of using geometric hashing to find pairs of compatible elements is the method of Fischer et al. [FWLN94]. They define spheres around each C_α atom. Geometric hashing [NW91] is used to find the number of C_α atoms inside two spheres (one from each structure) with "similar" coordinates, and the two C_α atoms are defined to be compatible if there are at least 20. Pairs of compatible spheres having similar transformations are clustered by similarity of transformation. Later the method has been modified to also use secondary structure constraints [FTNW95]. Spheres are constructed only around residues assigned α-helices or β-strands, and corresponding residues must have the same type.

Geometric hashing followed by clustering has also been used for analysing structures described at secondary structure level. For example, Alesker et al. [ANW96], use geometric hashing for finding pairs of compatible secondary structure elements (SSEs). Then, for each pair, the 6 parameters describing the best superposition transformation are computed. For pairs of (compatible) pairs a distance measure is defined depending on the similarity of the transformations. Clustering is performed in a iterative manner. In each iteration the compatible pair with the minimal distance from one of the clusters is joined to this cluster. Pennec and Ayache [PA98], also use SSEs and geometric hashing. They define an *information measure* for the transformations using different types of informations, and cluster using "similar" transformations with decreasing information. When no more transformations can be joined to the current cluster, a new clustering is started on the rest transformations. The remaining transformations are then clustered using the same algorithm.

Verbitsky et al. [VNW99], use geometric hashing when comparing two structures, where one (the model) is allowed to contain a hinge point (the other is the target). The hash table is filled by use of the model, only saving the transformation between the coordinate frame at the hinge and the one at each of the other residues (C_α). The task is then to find the best candidates for the position (C_α) of the target corresponding to the hinge, together with two transformations, one for each part of the model. This is done using hash tables and clustering. Finally the candidates are evaluated by RMSD. The complexity is of $O(n^2)$, where n is the number of C_α atoms compared.

Clustering by transformation is also used by Chew et al. [CHKK99]. Let $R_{i,j}$ be the rotation matrix that best superpose two succeeding unit vectors $(i, i+1)$ from structure A with two $(j, j+1)$ from B (see Section 4.1). Two compatible connected substructers (elements) of length $k+1$ are found if $R_{i,j} \approx R_{i+1,j+1}, ..., \approx R_{i+k,j+k}$ where $R \approx S$ if the two are approximately equal. Ordinary transformations are then used for clustering of compatible substructures.

Depending on the clustering procedure, a final adjustment or refinement might be worth while ([VS91], [HS95]). This is either an extension by searching for additional matching pairs of elements [FWLN94], or an iterative procedure, similar to the one described in section 4.3.

4.7.2 Clustering by use of relations

Clustering methods compare element based structure descriptions and build up shared substructures by first finding pairs of compatible elements (one element from each structure) and then progressively building up larger correspondences. In the previous section, we discussed methods where the consistency between substructure correspondences was checked by comparing transformations. In this section we discuss methods where the consistency is assured by requiring similar relations between corresponding elements in the two structures.

Analysing structure A and B, a list of all compatible elements (A_i, B_k) is generated. Each pair of compatible pairs is then tested for consistency. Let $\rho(A_i, A_j)$ denote the relation between any two elements (from the same structure). The pair (A_i, B_k) is consistent with (A_j, B_l) if and only if $\rho(A_i, A_j)$ and $\rho(B_k, B_l)$ satisfy some requirements, and $\rho(A_i, A_j)$ is similar to $\rho(B_k, B_l)$. The problem is then to find the set of compatible pairs with highest score, such that any two of the compatible pairs are consistent. Note that this implies that testing a pair for joining a cluster requires testing for consistency against all the pairs in the cluster, this is due to non-transitivity.

In most methods, the elements are fragments or SSEs, represented as vectors. The relations considered when checking consistency include distances and angles (projected on specific planes) between the two vectors.

The problem can be formulated with help of graph theory in the following way: the structures are represented as graphs, where the nodes are the elements, and there is an edge between nodes A_i, A_j if the relation $\rho(A_i, A_j)$ satisfies some constraints. The nodes are labelled with element properties, and the edges with relations. The problem is then, given two graphs, find a subgraph of each so that the subgraphs are similar and have maximum score. When the scoring is the number of compatible pairs, the problem can easily be solved by finding the maximal clique in a product graph (or connection graph) (Grindley et al. [GARP93]), which can for example be done by an algorithm by Bron and Kerbosch [BK73]. The (node) product graph is constructed by creating a node for each compatible node pair, and an edge from node (A_i, B_k) to node (A_j, B_l) if they are consistent.

For example, Grindley et al. [GARP93] use SSEs, represented as vectors. SSEs of the same type are compatible, and the relation for deciding consistency is described by three parameters, distances and torsion angle between the elements, and the algorithm by Bron and Kerbosch is used. Rufino and Blundel [RB94], use the same algorithm as Grindley, but the constraint for being compatible, and the relations, are a bit stronger. A final dynamic programming step is performed for refinement.

Koch et al., [KLW96], also use graph theory and SSEs, but for making an edge between A_i, A_j they require that some of the van der Waals volumes around the atoms of A_i overlaps with some of the van der Waals volumes around the atoms of A_j. The relations (edge labels) are "parallel", "antiparallel" or "mixed" (additional information can be added easily). They argue that it is sufficient to find *connected* subgraphs, thus reducing the problem. The algorithm is very similar to that of Grindley et el. but they define an *edge* product graph, in contrast to the *node* product graph. A node in an edge product graph consists of two edges (one from each structure). The difference is illustrated in figure 9.

The method of Grindley et al has been generalised by Mizuguchi and Gō [MG95] who assign a a *continuous number* to the edges of the product graph (thus generalising from consistent/not consistent to giving a score of the consistency). In that way each cluster is given a score $-\sum_{i \neq j} d((A_i, B_i), (A_j, B_j))$, where d is a measure of dissimilarity. They then search for the highest scoring clusters using a *parallel* iterative clustering procedure, where (the best) m new clusters are generated at each iteration, with the combination of two existing clusters.

A version of the SARF program, SARF2, [AF96] uses SSEs instead of backbone fragments. Compatibility is decided by SSEs of same type. The relation consists of five parametres describing distances and angle, and upper limits on these are used to decide if there will be an edge A_i, A_j. A recursive incremental clustering procedure is used. After finding the clusters, they apply an iterative procedure for refinement.

The DALI program of Holm and Sander [HS93] starts by using overlapping hexapeptide fragments. The decision of compatibility is done by use of distance matrices ($C_\alpha - C_\alpha$ distances), and the relations are also distance matrices. List of *contact patterns* (pairs of compatible fragments, 2 from each structure) are formed (which in effect is the product graph). The contact patterns are used for finding *seeds* for starting the clustering, which is done by a simulated annealing (Monte Carlo) search. An elastic score for the similarity of corresponding distances is used. Each step in the search is either adding or removing pair of tetrapeptides (one hexapeptide generating 3 overlapping tetrapeptides). Several searches are done in parallel so that one can obtain different solutions at the end. A number of techniques are used for speeding up the computations.

Kleywegt and Jones [KJ97], describe a program DEJAVU which searches for similarities in structures where the elements are SSEs. The elements are described by type, number of residues and the C_α coordinates of the first and last residue. Two elements are compatible if they have the

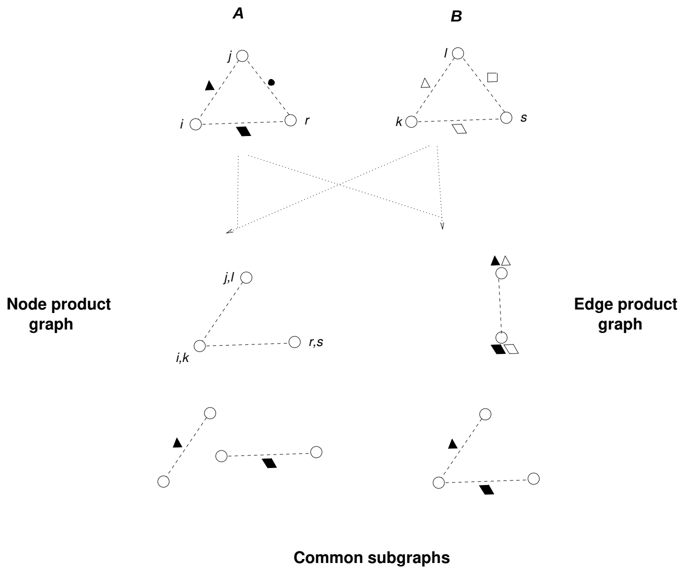

Figure 9: The difference between node product graph and edge product graph is illustrated by an example with three nodes in each substructure. $(A_i, B_k), (A_j, B_l), (A_r, B_s)$ are compatible, and the relations $\rho_{ij} \sim \rho_{kl}$ and $\rho_{ir} \sim \rho_{ks}$ but not $\rho_{jr} \sim \rho_{ls}$. The node product graph contains two cliques and hence result in two common substructures, each with two nodes, and edge product graph in the common substructures $(\rho_{ij}, A_i, \rho_{ir}), (\rho_{kl}, B_k, \rho_{ks})$. The edge product graph contains only one clique corresponding to one shared subgraph containing two edges.

same type. The user can specify which type of distances are to be used (center-to-center etc), as well as if directional and topological constraints are to be included. Weights to a scoring scheme can also be given. The clustering algorithm is a depth-first search. In a final step, elaborate iterative refinements to optimise the superpositioning of the two structures, is performed.

Escalier et al [EPSV98] use fragments, and compatibility is decided by structure similarity. A pattern driven (simultaneous) searching procedure is used for finding compatible fragments from two structures, based on the 3D coordinates. One node in the search tree represent a subfragment (by 3D coordinates) from one of the structures (query), and every occurrences of this in the other structures (scene) is saved in a list. Two search nodes of equal length (k atoms), and with $k-1$ common atoms, are then used to generate a search node of $k+1$ atoms. The relations used for clustering are the positions of corresponding atoms (residues), and the distances between the atoms in two fragments: the pairs (A_i, B_k) and (A_j, B_l) is consistent if 1) every atom a appearing in both A_i and A_j should correspond to the same atom in B_k and B_l, and 2) the distance between atoms in A_i, A_j should not deviate much from the distance between corresponding atoms in B_k, B_l. The score of the clustering problem is the number of corresponding atoms. The clustering is performed by a branch-and-bound search procedure.

Russel [Rus98] describes a method for finding similarities at residue level possibly related to functional sites (active and binding sites). The method finds sets of residue pairs (one from each structure) with amino acids from the same group, and with similar orientiation of the side chains. Sequence order need not be preserved. Only conserved residues are considered (requiring a multiple *sequence* alignment), and amino acids with only carbon and hydrogen in their side chains, and residues involved in disuplphide bindings, are excluded. Compatibility between the remaining residues are decided by amino acids from the same group (he defines 9 groups). Each residue is represented by three atoms, and for consistency it is required that the distances between corresponding atoms in two residues (in the same structure) is below a limit, and that corresponding interatomic distances in the two structures are equal enough. The clustering is incremental, implemented as a recursive depth-first search. A new pair is tested for consistency against all pairs in the running cluster. A weighted RMSD measure is calculated, and the statistical significance of the similarity is assigned from analysis of randomly generated side-chain patterns.

4.8 Statistical approach

These methods are used when the proteins are characterised by some distributions of biological attributes, and compared by using a (protein length independent) function that computes a distance between the proteins. The methods classified as Content methods in Brown et al. [BOT96] (3 methods) are all statistical.

The methods using space based description are all statistical (section 3.2). A common pattern can be found comparing sets of corresponding cells. This is used by Bagley and Altman [BA95], where the cells are shells around a common centre. The substructures are described by the set of atoms within the cells, along with their 3D coordinates. In addition they include user-defined properties, such as the types of atoms, chemical groups, amino acids, secondary structures, charge, polarity, mobility and solvent accessibility.

The statistical approach is also used by Kastenmüller [KKS98], to compare structures as a whole. Histograms are used for representing the distribution of the number of atoms in each cell, and the belonging properties.

5 Multiple structure comparison - Pattern discovery

The problem of discovering a pattern from a set of biological objects, (structures or sequences) can be looked upon as a machine learning problem (e.g., [BJEG98, Con95]).

Brazma et al. [BJEG98] classifies the methods for discovering deterministic sequence pattern in *Sequence driven* and *Pattern driven*. Another concept for Sequence driven could be *Comparison based*:

1. *Pattern driven* – methods that search a solution space for patterns with a high fitness value with respect to the input (training) objects.

2. *Comparison based* – methods that find common patterns through (pairwise) comparisons of the input objects.

A survey treating both pattern driven and comparison driven methods for sequences is presented in [BJEG98]. For structures, the majority of the methods are comparison based.

Extending a pairwise method to the multiple case ($n > 2$ structures) can be done by mainly three approaches (see also [EPSV98]):

PI – Pivot – use one object as pivot, and compare it successively to all the other objects. The results are then compared to find the part of the pivot that have similarities in all the other objects, or in at least $k-1$ of them.

LP – Linear progressive – start with one object, and compare successively the other objects to the results. The results are descriptions of the similarities of the objects involved.

TP – Tree progressive – compare (results) using a (possible implicit) tree, where the leafs are objects.

All pairwise methods can be extended to the multiple case by the PI approach. Note however that a description of the similarities is not neccessarily given, only the part of the pivot which occurs in $(k-1)$ other objects, and where. It may be possible to describe the similarities as a pattern (for different pattern descriptions, see Section 5.1). Ideas from the pivot approach can be combined with pattern driven, search based methods to obtain a pattern common to the pivot and at least $k-1$ other objects simultanously (e.g., [Jon97]).

If a pairwise comparison method can express each of the (one or several) best local similarities as a pattern, and if these patterns can be compared to other objects, then the method can be extended to the multiple case by using the LP approach. If also patterns can be compared to find common generalisations, the TP approach can be used.

As noted in for example [BJEG98], methods for multiple sequence comparison based on pairwise methods have the weakness that in each pairwise comparison, only information about the two sequences/structures is available and so alignments/patterns optimal for the whole set can be missed if they are not also optimal for every pair. Therefore pattern driven methods may first be used to find elements common to all structures and next use these to guide the pairwise comparisons in a more detailed structure based method For example, output of the local motif discovery tool SPratt [JET99] has been used to guide the multiple alignment program MSAP [TFO94].

An alternative to PI, LP, and TP is the following:

AA – All_by_all – all $n(n-1)/2$ pairwise comparisons are performed to find common pairwise similarities. Comparasions (e.g. intersections) are done on these results, to find similarities occurring in for example k of the objects (a clique of similarity relations).

Using AA, the result of all pairwise similarities can be used to identify similarities between all objects (or some minimum proportion of the objects). In a sequence comparison method by Vingron and Argos [VA91], the pairwise similarities are represented by dot matrices (reflecting residue similarities) and they are combined using matrix multiplication to filter out similarities not shared by all objects. We are not aware of methods using this approach for comparison of structures.

In principle, PI and AA are able to give the same results if the similarity relation used is transitive. Transitivity exists if similarity between A and B and between B and C implies that A is similar to C. However, the result of PI and AA approaches will depend on the exact implementations, for example how similarities are scored and how many are included in the analysis.

The structure of the rest of this section is as follows. First it is discussed how patterns can be described. Then follows an overview of methods for multiple structure analysis, first superpositioning methods, then alignment and clustering based methods and finally search methods which take into account information from all structures simultaneously.

5.1 Pattern representation

In this work, a pattern is an object to which a structure description can be matched. If the pattern is deterministic, the structure description either matches or does not match a pattern; if the pattern is probabilistic, the result of the matching is a number (e.g., a probability). In a pattern discovery method, one needs to define the class of patterns to be considered. For a structure comparison method it is possible to define pattern classes which allow one to describe the outcome of the comparison, e.g., a consensus representation of the aligned substructures.

5.1.1 Sequence patterns

If only primary structure is considered, the patterns will be sequence patterns. A common form of deterministic sequence patterns are regular expression type patterns such as those used in the PROSITE database. A simple example taken from this database is the pattern used for the classical zinc finger C2H2 motif, 'C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H' that matches all sequences containing a C followed by between 2 and four arbitrary residues followed by a C and three arbitrary residues, one of L, I, V, M, F, Y, W, or C, etc. For a fuller treatment of regular expression type sequence patterns, see [BJEG98]. Probabilistic sequence patterns include profiles, hidden Markov models, neural networks, and stochastic context free grammars. The Probabilistic pattern formalisms give greater expressive power, but on the other hand have more parameters to be estimated, and are less easily interpreted by humans.

Some protein families are difficult to describe adequately using sequence patterns, that is (1) the pattern does not perfectly discriminate between family members and non-members, and (2) it does not give a complete description of the features critical to the function/structure class associated with the family. When structural information is available, it is possible to define a structure pattern.

5.1.2 Structure patterns describing sites

Kasuya and Thornton [KT99], have studied the three-dimensional structure of protein fragments matching PROSITE patterns. They matched each PROSITE pattern against all structures in the PDB databank and studied cases where PROSITE patterns give false positives (matches in unrelated proteins). By comparing the structure of each matching fragment to that of a true matching fragment and requiring similarity to accept the match, they achieved increased specificity. Jonassen et al [JETG], have in an independent, but closely related study arrived at equivalent results. Additionally they found that by relaxing the PROSITE pattern by allowing for approximate matching, but requiring structural similarity to a known positive, sometimes also can give improved sensitivity.

The functionality of protein active sites arise from their three-dimensional structures. The residues forming a site need not be close in sequence, and the sequences of the proteins sharing an active site are sometimes extremely divergent. For example, the bacterial and eukaryotic serine proteases share a very similar active site while having very different structures. Especially when the residues involved in the site are non-local, sequence patterns (deterministic and probabilistic) cannot recognize such structures. However, one can define structure motifs describing the spatial configuration of the active site residues.

Wallace and Thornton have constructed the PROCAT database containing manually defined patterns called "3D coordinate templates" describing active sites (http://www.biochem.ucl.ac.uk/bsm/PROCAT/PROCAT.html, [WLT96, WBT97]). Each template specifies the spatial position of a set of atoms in each residue making up the site and the amino acids allowed for each residue. The sequence position of the residues are not constrained. A similar description (called local packing patterns) is proposed by Jonassen et al [JET99] where for each residue the allowed amino acids are given together with one coordinate set (mean side chain atom). The order (but not distance) of residues along the sequence is given. Similar patterns discarding the sequence order of the aligned residues can be discovered by the methods using geometric hashing ([NW91, FWLN94, FTNW95], Section 4.6).

Yet another similar description formalism named "Fuzzy Functional Forms" (FFFs) is proposed by Fetrow and Skolnick [FS98a]. FFFs specify for a set of residues, which amino acids are allowed, the C_α-C_α distance (average and variance) for each pair of residues, and constraints on the sequence distance between the residues. For example, they specify a FFF for the disulfide oxidoreductase activity of the glutareoxin/thioredoxin protein family: two cystein residues separated by two residues and a α-carbon distance of 5.5Å (±0.5Å) close to a proline residues (distance requirements included).

5.1.3 Structure patterns describing folds or architectures

We say that two proteins (or domains) share the same architecture if they have secondary structure elements (SSEs) with similar spatial relative positions. If, additionally the SSEs are connected in the same way along the sequence, the two proteins have the same fold. The shared architecture or fold of a set of proteins can be described using structure patterns. This can be done at residue, fragment, or SSE level.

Gerstein and Altman [GA95] define the core of a family of aligned protein structures as a collection of residues for each of which position (mean and variation) information is given. The descriptions are made from multiple structure alignments and therefore co-linearity of aligned residues is maintained. This is equivalent to the internal representation in the extension of the SSAP comparison method to multiple structures, [TFO94].

A number of methods for structure comparison perform their analysis at SSE level. Typically the structures are described using graphs (section 4.7) and the common substrucures are found as subgraphs common to the graphs representing the individual structures. When finding common subgraphs it can be chosen whether the solutions should be restricted to obey the sequence order of the SSEs. Different methods have been reported using similar structure representations and matching algorithms, see for example [KLW96, AF96, WIS94, GWNT99].

5.2 Superposition

Several methods for multiple superposition based on a correspondance (alignment) exists. To avoid biasing the superposition towards a specific (pivot) structure, a simultaneously superposition should be used, resulting also in an average structure.

MNYFIT is a program by Sutcliffe et al. [SHCT87] that uses pairwise superposition to iterate to a common average structure. Weights can be used, giving more strength to topologically equivalent positions. Kearsley [Kea90], build on the work by Gerber and Müller [GM87], optimise simultaneously the superposition of all possible pairs, by minimising the sum $\sum_i^{n-1} \sum_{j=i+1}^{n} \sum_k^{ns} (Q^i x_k^i - Q^j x_k^j)^2$, where Q^i is the rotation of structure i (weights can be allowed for). One structure is selected as static, and all the centroids are shifted to the origin of coordinates. Minimising is done by Rational Function Optimisation, using iteration for convergence. Shapiro et al. [SJPL92], and Diamond [Dia92], describe other methods for optimally superposing all pairs of structures simultaneously. The method of Diamond has time complexity $O(n)$, and the optimalisation is done without referring to the original coordinates of the structures.

5.3 Multiple structure alignment

5.3.1 Progressive alignment

The multiple alignment of Sali and Blundell [SB90], Russel and Barton [RB92], Ding et al. [DQF94] and May and Johnson [MJ95] all follow the same algorithmic scheme:

- perform all $n(n-1)/2$ pairwise alignments, and give a similarity score to each of them
- use the scores to generate a dendrogram
- perform alignments following the dendrogram from the leaves to the root. When aligning one structure with a subalignment, or two subalignments, the same procedures as for the pairwise alignments are used. The score when aligning two subalignments is *in principle* done in the same manner for all the methods:

$$W_{ij} = \frac{1}{n_1 n_2} \sum_{l=1}^{n_1} \sum_{k=1}^{n_2} \begin{cases} W_{i'j'}^{lk} & \text{if two residues are compared} \\ \text{some value} & \text{if a residue and a gap are compared} \\ \text{some value} & \text{if two gaps are compared} \end{cases}$$

n_1 and n_2 are number of structures in the subalignments, and i, j are the positions (columns) compared. i', j' are the true indices for residues from structures l, k, occupying the columns i, j.

Taylor and co-workers [TFO94] have also adopted the progressive approach, in a fusion of the pairwise method SSAP (see Section 4.4) and the multiple sequence alignment program MULTAL [Tay88]. First, each pair of structures are compared using SSAP to assess all pairwise similarities. The most similar structure pairs are aligned independently into a consensus structure and then the pairwise similarities between all single and consensus structures are calculated, to progressively bring together the most similar of these at each stage. Note that this differs from the TP approach in that the progression of the pairwise alignments is not determined by a pre-computed dendrogram; instead at each stage, the similarities are re-computed and the most similar objects aligned first.

The main problem when going from sequence to structure is to define the structural equivalent of a consensus sequence (or profile), for use when aligning alignments. In SSAP interatomic vectors are used, and the consensus used is an average vector and its variance (error measure). The alignment process results in multiple sets of interatomic vectors (one vector set for each residue) that are not necessarily mutually consistent (due to, say, relative domain movements). Procedures for making them consistent, and from that calculating the 3D coordinates of the consensus are incorporated.

Gerstein and Levitt [GL98] use a similar, but simpler method. All pairs of structures are aligned, and the structure that is on average closest to all other structures, "the median structure", is identified. A multiple alignment is formed by the alignment of all the other structures onto this median structure by consistently combining the alignments. They argue that this simple approach is adequate when the number of structures is small.

5.3.2 Search for global multiple alignment

All types of searching can in principle be used to search for multiple alignment of structures, though not guarantee finding the best. Each point in the search space represent an alignment, and is evaluated by use of a scoring scheme. Godzik and Skolnick [GS94] use a multidimensional lattice chain and a Monte Carlo (simulated annealing) approach for making a multiple alignment using different scoring schemes: Dayhoff substitution matrix, RMSD, $C_\alpha - C_\alpha$ distance difference, and contact map overlap. Hence, the algorithm can be used for both sequence and structure comparison. An alignment is represented as a lattice chain, which for d structures consists of a list of elementary vectors of length d: $[1, 0, ..., 0], [0, 1, ..., 0], ..., [0, 0, ..., 1]$. There is an one-to-one correspondance between an alignment and such a chain. In a single step of the search algorithm a chain fragment is chosen at random, and replaced by another.

5.3.3 Finding patterns from multiple alignment

When a multiple alignment is done (establishing correspondance between residues), superpositions can be performed to find those residue positions with small spatial variations, possible being a *core*. This can be done as in [GA95]: the structural variation of each atom (residue) is computed, and a cutoff value is used to remove atoms with high structural variation. Then they iterate by making a new multiple superposition on the atoms not removed. New atoms can be added if they now satisfy the cutoff value, and this removal/adding can be done until convergence.

Rinaldis et al. [dRACHC98] develop *profiles* representing the most conserved residues on the surfaces. A multiple superposition is done, and the structures are represented in a 3D grid, whose cells have 2 Å sides, a residue represented by a pseudo atom averaging the side chain. Only exposed residues are retained. For each cell is constructed a column in a profile, as described in Gribskov et al. [GME87], with the 20 values for each amino acid, and the 3D coordinates of an average pseudo atom.

5.4 Clustering

To our knowledge, none of the clustering methods using transformations (see Section 4.7.1) have been extended to the analysis of multiple structures, but it could be done easily by a comparison based method. The result of a pairwise comparison is a list of corresponding residues, and extending by a pivot method is straighforward. Extending by a progressive method require calculating an average structure at each iteration, representing the result of the comparison.

Two of the methods using relations (section 4.7.2) are extended to multiple case, both using the pivot based extension.

Koch et al. [KLW96], first compute all sets $E(G_1, G_i), i = 2, ..., n$, where $E(G_1, G_i)$ is the set of all connected maximal common substructures in G_1 and G_i, represented with the edges. Then they successivily intersect each edge set from $E(G_1, G_2)$ with each edge set from $E(G_1, G_i), i = 3, ..., n$, to find subgraphs of G_1 that are also contained in one or more $G_2, ..., G_n$. These subgraphs may be disconnected. Other types of analyses can be done.

Escalier et al [EPSV98] extend to the multiple case in a way analog to that of Koch et al. The same procedure as for the pairwise case is used, but $n - 1$ lists of occurrences are kept in the step for finding compatible elements (one list for each structure compared to the pivot).

5.5 Simultaneous search

A few methods using simultaneous search are published. The method SPratt of Jonassen et al [JET99] is a pattern driven method, the elements are residues. The method finds patterns which occur in at least k of the structures. Each residue's 3D coordinate is represented by a pseudo atom, calculated from the side chain atoms. For each residue is generated a string of amino acids, from the residues falling inside a sphere around the pseudo atom, satisfying the sequential order. A sequence pattern discovery method Pratt [JCH95, Jon97] is then used to discover common patterns in these strings, using a simultaneously heuristic depth-first search method. The occurrences are further constrained to have all pairwise RMSD values below a given threshold. The patterns are described in a PROSITE-like manner, and gives constraints on a set of residues. The constraints are on the coordinates, the allowed amino acid types and on their relative ordering along the backbone.

Wako and Yamato [WY98], divide the space around a structure into non-overlapping tetrahedrons, (using Delaunay tessellation). The tetrahedrons are constructed using C_α atoms as vertices. On the bases of each tetrahedron and its up to four neighbouring tetrahedrons, constructed from 20 atoms, a code of up to 32 digits is found (each digit being between 1 and 8). Thus, the codes represent non-sequential overlapping substructures. For each structure a list of such codes is found. A search procedure then searches for codes occurring in all or several of the structures. Each such code constituts a potential motif. The codes do not represent the 3D structure explicitly, and superposition of the substructures is neccessary to assess the degree of resemblance.

Su et al. [SCH99, CHD96], represent the structures as linear graphs (hence sequantial), one version on residue level, and one on SSE-level. The SSEs constitute the nodes of the graph, labeled with type, length and direction (right- or left- handed helix, and parallel or antiparallel sheets). The linear graphs representing the set of structures are concatenated, and the edges inside a structure is represented by a label. The procedure for finding patterns (substructures) is a beam search, of width k, and the evaluation of each node in the search tree is done by use of minimum description length: each occurrences of a pattern is replaced by a link, and the length of this compressed representation is found. The search starts with k search nodes representing one element each. At each iteration each of the k nodes are extended by all possible neighbours (one from each occurrence), and the k best are kept for the next iteration. The matches may be inexact, and by combining the search with hill climbing, they obtain a polynomial run time.

Gilbert et al. [GWTV], have developed a program for discovering supersecondary motifs in TOPS-cartoons [FMT94], by a pattern driven approach. The TOPS-cartoons are two dimentional representations of protein folds by showing the secondary structure elements, and illustrate the relative spatial position and direction of these elements.

5.6 Learning the description of known sites

Fetrow et al. [FS98b] represent active sites by three dimensional descriptors termed fuzzy functional forms (FFF), based on the geometry, residue identity and conformation of protein active sites. A FFF contains the mean and allowed variations of the internal distances between the functionally important residues. They start by multiple sequence alignment and available information about residues important for functions. Then they superpose these, and iterate to learn a specific and unique form.

Bagley and Altman [BA95] use a machine learning system, FEATURE, with sites and nonsites, to learn description of sites (the distribution of the properties over the cells). The method then learns the description of *known* sites. In [WAC97] there is an application to calcium binding sites.

Generally finding motifs in statistically described structures can be done by characterising a set of structures by some "middle" distribution, dividing the structure space into subspaces, and compare the subspaces from the different structures. Since the number of subspaces is indefinite (if real values), a finite number must be chosen, and the goodness of such a method would depend on how the subspaces are chosen.

6 Assessment

When comparing structures it is widely known from practical experience (e.g., [TO89]), and from more systematic investigations [God96, May96], that, beyond close similarity, there is no uniquely correct structural alignment of two proteins. Different alignments are achieved depending on which biological properties and relations are emphasised in the comparison. This add a complicated element to the assessment of the result of a comparison.

When new methods are presented, the authors most often assess their methods by comparing its results to manually determined alignments. This is systematically done by Gerstein and Levitt [GL98], where an assessment of the ALIGN-program ([Coh97]) is done using the SCOP classification ([MBHC95]) as a "gold standard". The program was tested for 2107 pairs of structures, where each structure in a pair was from the same superfamily, hence similar structures. They made a plot of the results, plotting a normalised RMSD against the number N of matched residues. The RMSD was normalised by $225RMSD/(N+135)$, and a horizontal demarcation line was drawn at 4. Only 32 of the 2107 pairs got a normalised RMSD value above the demarcation line. They also assessed their multiple method against 9 SCOP superfamilies.

6.1 Statistical assessement

Several methods are assessed statistically. The scoring found for an alignment of native structures is then compared against what is expected by chance, most often implemented as what is expected by aligning random structures, often using non-related proteins.

Random structural models are discussed by Taylor in [Tay97]. The choice however, of the best random model against which native/native comparison scores should be compared is not simple and depends on the degree to which the inherent nonrandom features of protein structure in general should be considered significant. The best suitable of the random models would be those generated with secondary structures. Ideally, these models should be calculated for each comparison to match the length of the native comparison and the secondary structure composition. However, this models are complex to generate and cannot be "tailor-made" for each individual comparison without excessive computation. Taylor describes several random models: constrained random walk, random models from distance geometry (DRAGON, [AT97]), combinatorial models, combinatorial reconnection, chain reversal, and chain reflection. The closer the random model is to preserving the properties of the native proteins, the more difficult it becomes to generate plausible alternatives. Taylor's approach is to expand the population of random models at the level of the alignment – generating from each individual model structure a family of near-optimal subalignments. An advantage of this approach is that it can be applied not only to structures belonging to the set of randomised models but also to the native structure itself.

RMSD is used alone or as part of most scoring schemes, hence knowing the behaviour of RMSD of random structures is important. Alexandrov and Go made an analysis for finding the significance of similar SARFs [AG94]. For a fixed length L, they picked up all fragment pairs of this length in two unrelated structures, and found the value R_L from the condition that only 1% of pairs have smaller RMSD. Assuming normal distribution, this correspond to the deviation of 4.3σ. They then plotted R_L against L, and show that the result could be approximated by the curve $R_L = 1.37 + (1.16L - 15.1)^{1/2}$. Russel [Rus98], did an analysis using distance RMSD, related to his method for detecting side-chain patterns. Random pairs of structures with different folds were chosen, and random patterns of two to six patterns were derived, claiming amino acid type conservation. For $P(x)$ being the proportion of the results with RMSD$< x$ (for a fixed pattern length) it was found that $\log P$ was linearly related to $1/x$ in the upper part of $P(x)$. Maiorov and Crippen [MC94] have proposed a self-referential non-statistical definition of the significance of RMSD. They take two conformers to be intrinsically similar if their RMSD is smaller than that when one of them is mirror inverted.

When comparing two sequences, P-value, E-value and Z-value are used for significance of a score S. The P-value give the probability of achieving a similar score by chance, the E-value is the expected number of comparison achieving the same score by chance in a data base search. Z-value is the number of standard deviations S is from the mean value in an appropriate probability distribution. Alexandrov and Fischer [AF96] use Z-value for the statistical significance. Comparing a structure A with all others in a structural database they get a distibution of the scores. From this they can, for each structure B, use the Z-value Z(A,B). Z(A,B) might be unequal Z(B,A), and they define ZAB=Z(A,B)+Z(B,A), resulting in a symmetrical significance value. Holm and Sander [HS93, HS96], also express the significance of an alignment in terms of Z-value, using the distribution from an all-vs.-all comparison. Gibrat et al. [GMB96] in their VAST program computes a P-value for an alignment based on how many secondary structure elements are aligned as compared with the chance of aligning so many elements randomly.

6.2 Comparing scoring schemes

Levitt and Gerstein [LG98], made a comparison of RMSD to the scoring scheme used in their iterated dynamic programming/superposition program. The P-value of a score S for a fixed N (number of matched residues) can be found by fitting a function $Z(S)$ to an extreme-value distribution. Unrelated structures from the SCOP database are used. The same statistics can be developed for RMSD. These two can then be compared by a method of Brenner et al. [BCH98]. The structure pairs were sorted by their E-values, and the E-value giving 1% false positives was identified and the number of true positives above this E-value was recorded. Levitt and Gerstein's scoring scheme shows a much better E-value ($\log(E)$ equal -1.58 compared to -32.8 for RMSD) and larger number of true positives (627 compared to 202). This confirms that the scoring scheme is much better than RMSD for this purpose.

Taylor also compared the scoring scheme used in his SAP program to weighted RMSD [Tay99]. It showed some correlation of high SAP score with low wRMSD, but the highest scoring alignments did not have the lowest wRMSD, even when the length component is eliminated. He also compared other results from the literature (DALI server, http://ebi.ac.uk/DALI, and those in [God96]), and found that most methods do not attain the minimum RMSD for the number of residues that they align, with DALI as an exception.

7 Conclusion

The aim of this work has been to establish some systematisation to the field of structure comparison and structure pattern discovery, and to give a survey of methods focusing on the most recent ones. For this purpose, a framework has been designed and a number of methods have been surveyed and placed into this framework. A lot of methods exist, using different algorithmic approaches, from which we have been able to identify some basic principles. These principles form the foundation for the framework.

We think and hope that this survey with its framework gives a contribution to the work of understanding and explaining the different methods for structure comparison, and being a foundation for developing new and more powerful methods.

Acknowledgments

IJ was supported by grants from the Norwegian Research Council.

References

[AF96] N.N. Alexandrov and D. Fischer. Analysis of topological and nontopological structural similarities in the PDB: new examples with old structures. *Proteins: Struct., Func. Gen.*, 25:354–365, 1996.

[AG94] N.N. Alexandrov and N. Go. Biological meaning, statistical significance, and classification of local spatial similarities in nonhomologous proteins. *Prot. Sci.*, 3:866–875, 1994.

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Bio.*, 215:403–410, 1990.

[ANW96] V. Alesker, R. Nussinov, and H. J. Wolfson. Detection of non-topological motifs in protein structures. *Protein Eng.*, 9:1103–1119, 1996.

[APG+94] P.J. Artymiuk, A.R. Porrette, H.M. Grindley, D.W. Rice, and P. Willett. A graph-theoretic approach to the identification of three-dimensional patterns of amino acid side-chains in protein structures. *J. Mol. Biol.*, 243:327–344, 1994.

[AT97] A. Aszdi and W.R. Taylor. Hierarchic inertial projection: A fast distance matrix embedding algorithm. *Comp. Chem.*, 21:13–23, 1997.

[ATG92] N.N. Alexandrov, K. Takahashi, and N. Go. Common spatial arrantements of bacbone fragments in homologous and non-homologous proteins. *J. Mol. Biol.*, 225:5–9, 1992.

[BA95] S.C. Bagley and R.B. Altman. Characterizing the microenvironment surrounding protein sites. *Prot. Sci.*, 4:622–635, 1995.

[BCH98] S.E. Brenner, C. Chothia, and T.J.P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci*, 95(11):6073–6078, 1998.

[BJEG98] A. Brazma, I. Jonassen, I. Eidhammer, and G. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *J. Comp. Biol.*, 5(2):279–305, 1998.

[BK73] C. Brown and J. Kerbosch. Algorithm 457 - finding all cliques of an undirected graph. *Comm ACM*, 16:575–577, 1973.

[BKW+77] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol*, 112:535, 1977.

[BOT96] N.P. Brown, C.A. Orengo, and W.R. Taylor. A protein structure comparison methodology. *Comp. Chem.*, 20:359–380, 1996.

[CHD96] D.J. Cook, L.B. Holder, and S. Djoko. Scalable discovery of informative structural concepts using domain knowledge. *IEEE EXPERT*, 11(5):59–68, 1996.

[CHKK99] L.P. Chew, D. Huttenlocher, K. Kedem, and J. Kleinberg. Fast detection of common geometric substructure in proteins. In *Proceedings of the third international conference on computational molecular biology, RECOMB'99*, 1999.

[Coh97] G. H. Cohen. ALIGN: a program to superimpose protein coordinates, accounting for insertions and deletions. *J. Appl. Cryst.*, 30:1160–1161, 1997.

[Con95] D. Conklin. Machine discovery of protein motifs. *Machine Learning*, 21:125–150, 1995.

[dat99] Table of contents. *Nucleic Acids Res.*, 27(1), 1999.

[Day78] M. O. Dayhoff. *Atlas of protein sequence and structure*, volume 5. National Biomedical Research Foundation, 1978.

[Dia92] R. Diamond. On the multiple simultaneous superposition of molecular structures by rigid body transformations. *Prot. Sci*, 1:1279–1287, 1992.

[Die95] K. Diederichs. Structural superposition of proteins with unknown alignment and detection of topologically similarity using a six-dimensional search algorithm. *Proteins: Struct., Func., Gen.*, 23:187–195, 1995.

[DQF94] D.F. Ding, J.A. Qian, and Z.K. Feng. A differential geometric treatment of protein structure comparison. *Bull. Math. Biol.*, 56:923–943, 1994.

[dRACHC98] M. de Rinaldis, G. Ausiello, G. Cesareni, and M. Helmer-Citterich. Three-dimensional profiles: A new tool to identify protein surface similarities. *J. Mol. Biol.*, 284:1211–1221, 1998.

[EPSV98] V. Escalier, J. Pothier, H. Soldano, and A. Viari. Pairwise and multiple identification of three-dimensional common substructures in proteins. *J. Comp. Biol.*, 5:41–56, 1998.

[FC96] A. Falicov and F.E. Cohen. A surface of minimum area metric for the structural comparison of proteins. *J. Mol. Biol.*, 258:871–892, 1996.

[FMT94] T.P. Flores, D.M. Moss, and J.M. Thornton. An algorithm for automatically generating protein topology cartoons. *Protein Eng.*, 7(1):31–37, 1994.

[FS98a] J.S. Fetrow and J. Skolnick. Method for prediction of protein function from sequence... *J. Mol. Biol.*, 281:949–968, 1998.

[FS98b] J.S. Fetrow and J. Skolnick. Method for prediction of protein function from sequence using the sequence-to-structure-to-function paradigm with application to glutaredoxins/thioredoxins and t_1 ribonucleases. *J. Mol. Biol.*, 281:949–968, 1998.

[FTNW95] D. Fischer, C.J. Tsai, R. Nussinov, and H. Wolfson. A 3D sequence-independent representation of the protein data bank. *Protein Eng.*, 8:981–997, 1995.

[FWLN94] D. Fischer, H. Wolfson, S.L. Lin, and R. Nussinov. Three-dimensional, sequence order-independent structural comparison of a serine protease against the crystallographic database reveals active site similarities: Potential implications to evolution and to protein folding. *Prot. Sci.*, 3:769–778, 1994.

[GA95] M. Gerstein and R.B. Altman. Using a measure of structural variation to define a core for the globins. *CABIOS*, 11(6):633–644, 1995.

[GARP93] H.M. Grindley, P.J. Artymiuk, D.W. Rice, and P.Willett. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *J. Mol. Biol.*, 229:707–721, 1993.

[GL98] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Prot Sci*, 7:445–456, 1998.

[GM87] P.R. Gerber and K. Mller. *Act. Cryst.*, A43:426, 1987.

[GMB96] J.F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Curr. Op. Struct. Biol.*, 6:377–385, 1996.

[GME87] M. Gribskov, A.D. McLachlan, and A.D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proc. Natl Acad. Sci. USA*, 84:4355–4358, 1987.

[God96] A. Godzik. The structural alignment between two proteins: is there a unique answer? *Prot. Sci.*, 5:1325–1338, 1996.

[GS94] A. Godzik and J. Skolnick. Flexible algorithm for direct multiple alignment of protein structures and sequences. *CABIOS*, 10(6):587–596, 1994.

[GWNT99] D. Gilbert, D. Westhead, N. Nagano, and J. Thornton. Motif-based searching in tops protein topology databases. *Bioinformatics*, 15:317–326, 1999.

[GWTV] D. Gilbert, D. Westhead, J. Thornton, and J. Viksna. Tops cartoons: formalisation, searching and comparison. Poster at RECOMB 99.

[HH92] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:100915–100919, 1992.

[HS93] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, 1993.

[HS95] L. Holm and C. Sander. 3-d lookup: Fast protein structure database searches at 90reliability. In Chris Rawlings et al., editor, *Proceedings of the Third Iternational Conference on Intelligent Systems for Molecular Biology*, pages 179–187. AAAI, 1995.

[HS96] L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.

[JCH95] I. Jonassen, J. F. Collins, and D. G. Higgins. Finding flexible patterns in unaligned protein sequences. *Prot. Sci.*, 4(8):1587–1595, 1995.

[JET99] I. Jonassen, I. Eidhammer, and W. R. Taylor. Discovery of local packing motifs in protein structures. *Proteins: Struct., Func., Gen.*, 34:206–219, 1999.

[JETG] I. Jonassen, I. Eidhammer, W.R. Taylor, and S.H. Grindhaug. Extending prosite patterns with coordinate information. *In preparation.*

[Jon97] I. Jonassen. Efficient discovery of conserved patterns using a pattern graph. *CABIOS*, 13:509–522, 1997.

[Kab78] W. Kabsch. *Acta Cryst.*, A32:922–923, 1978.

[Kea90] S.K. Kearsley. An algorithm for the simultaneous superpostion of a structural series. *J. Comp. Chem.*, 11(10):1187–1192, 1990.

[KJ97] G.J. Kleywegt and T.A. Jones. Detecting folding motifs and similarities in protein structures. In *Methods in Enzymology*, pages 525–545. Academic Press, 1997.

[KKS98] G. Kastenmller, H. P. Kriegel, and T. Seidl. Similarity search in 3d protein databases. In O. Zimmermann and D. Schomburg, editors, *German conference on bioinformatics, GCB 98*. Cologne Univerity, 1998.

[KLW96] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *J. Comp. Biol.*, 3(2):289–306, 1996.

[KT99] A. Kasuya and J.M. Thornton. Three-dimensional structure analysis of prosite patterns. *J. Mol. Biol.*, 286:1673–1691, 1999.

[Lat94] R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Eng.*, 7:1059–1068, 1994.

[LG98] M. Levitt and M. Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proc Natl Acad Sci USA*, 95:5913–5920, 1998.

[LP85] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.

[LS96] R.H. Lathrop and T.F. Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *J. Mol. Biol.*, 255(4):641–665, 1996.

[May96] A.C.W. May. Pairwise iterative superposition of distantly related proteins and assessment of the significance of 3-D structural similarity. *Protein Eng.*, 9:1093–1101, 1996.

[MBHC95] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. Scop: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[MC94] V.N. Maiorov and G.M. Crippen. Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins. *J. Mol. Biol.*, 235:625–634, 1994.

[MG95] K. Mizuguchi and N. Go. Comparison of spatial arrangements of secondary structural elements in proteins. *Protein Eng.*, 8:353–362, 1995.

[MJ95] A. C. W. May and M. S. Johnson. Improved genetic algorithm-based protein structure comparisons: pairwise and multiple superpositions. *Protein Eng.*, 8:873–882, 1995.

[MTH97] H. Matsuda, F. Taniguchi, and A. Hashimoto. An approach to detection of protein structural motifs using an encoding scheme of backbone conformation. In *Pacific Symposium on Biocomputing'97*, 1997.

[NW70] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–454, 1970.

[NW91] R. Nussinov and H.J. Wolfson. Efficient detection of three-dimensional structural motifs in biolog ical macromolecules by computer vision techniques. *Proc. Natl. Acad. Sci. U.S.A.*, 88:10495–10499, 1991.

[OBT92] C.A. Orengo, N.P. Brown, and W.R. Taylor. *Protein: Struct., Funct., Gen.*, 14:139, 1992.

[PA98] X. Pennec and N. Ayache. A geometric algorithm to find small but highly similar 3D substructures in proteins. *Bioinformatics*, 14:516–522, 1998.

[Pet98] M. Petitjean. Interactive maximal common 3d substructure searching with the combined sdm/rms algorithm. *Comp. Chem.*, 22(6):463–465, 1998.

[RA75] M. G. Rossmann and P. Argos. A comparison of the heme binding pocket in globins and cytochrombe b5*. *J. Biol. Chem.*, 250:7523–7532, 1975.

[RA76] M. G. Rossmann and P. Argos. Exploring structural homology of proteins. *J. Mol. Biol.*, 105:75–96, 1976.

[RB92] R.B. Russel and G.J. Barton. Multiple protein sequence alignment from tertiary structure comparison: Assignment of global and residue confidence levels. *Proteins: Struct., Func., Gen.*, 14:309–323, 1992.

[RB94] S.D. Rufino and T.L. Blundell. Structure-based identification and clustering of protein families and superfamilies. *J. Computer-Aided Molecular Design*, 8:5–27, 1994.

[RR73] S. T. Rao and M. G. Rossmann. Comparison of super-secondary structures in proteins. *J. Mol. Biol*, 76:241–256, 1973.

[Rus98] R.B. Russel. Detection of protein three-dimensional side-chain patterns: New examples of convergent evolution. *J. Mol. Biol*, 279:1211–1227, 1998.

[SA95] T. Shinohara and S. Arikawa. Pattern inference. In K. P. Jantke and S. Lange, editors, *Algorithmic learning for knowledge-based systems, GOSLER final report*, pages 259–291. Springer-Verlag, 1995.

[SB90] A. Sali and T.L. Blundell. Definition of general topological equivalence in protein structures: A procedure involving comparison of properties and relationships through simulated annealing and dynamic programming. *J. Mol. Biol.*, 212:403–428, 1990.

[SCH99] S. Su, D.J. Cook, and L.B. Holder. Applications of knowledge discovery to molecular biology: Identifying structural regularities in proteins. In *Pacific Symposium on Biocomputing'99*, 1999.

[SCPD86] Y. Satow, G. H. Cohen, E. A. Padlan, and D. R. Davies. *J. Mol. Biol*, 190:593–604, 1986.

[SHCT87] M.J. Sutcliffe, I. Haneef, D. Carney, and T.L.Blundell. Knowledge based modeling of homologous proteins 1. 3-dimensional frameworks derived from the simultaneous superposition of multiple structures. *Prot. Eng.*, 1:377, 1987.

[SJPL92] A. Shapiro, J.D.Botha, A. Pastore, and A.M. Lesk. A method for multiple superposition of structures. *Acta. Cryst.*, A48:11–14, 1992.

[SK83] D. Sankoff and J. B. Kruskal. *Time warps: string edits, and macromolecules: the theory and practice of sequence compar ison*. Addison-Wesley, 1983.

[Tay88] W.R. Taylor. A flexible method to align large numbers of biological sequences. *J. Mol. Evol.*, 28:161–169, 1988.

[Tay97] W.R. Taylor. Random structural models for double dynamic programming score evaluation. *J. Mol. Evol.*, 44:174–180, 1997.

[Tay99] W.R. Taylor. Protein structure comparison using iterated double dynamic programming. *Prot. Sci.*, 8:654–665, 1999.

[TFO94] W.R. Taylor, T.P. Flores, and C.A. Orengo. Multiple protein structure alignment. *Prot. Sci.*, 3:1858–1870, 1994.

[TO89] W.R. Taylor and C.A. Orengo. Protein structure alignment. *J. Mol. Biol.*, 208:1–22, 1989.

[VA91] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, 218:33–43, 1991.

[VNW99] G. Verbitsky, R. Nussinov, and H. Wolfson. Structural comparison allowing hinge bending, swiveling motions. *Proteins: Struct., Func., Gen.*, 34(2):232–254, 1999.

[VS91] G. Vriend and C. Sander. Detection of common three-dimensional substructures in proteins. *Proteins: Struc., Func., Gen.*, 11:52–58, 1991.

[WAC97] L. Wei, R. B. Altman, and J. T. Chang. Using the radial distribution of physical features to compare amino acid environments and align amino acid sequences. In *Pacific Symposium on Biocomputing'97*, 1997.

[WBT97] A.C. Wallace, N. Borkakoti, and J.M. Thornton. TESS: A geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. Applications to enzyme active sites. *Prot. Sci.*, 6:2308–2323, 1997.

[WIS94] J.V. White, I.Muchnik, and T.F. Smith. Modeling protein cores with markov random fields. *Math. Biosc.*, 124:149–179, 1994.

[WLT96] A.C. Wallace, R.A. Laskowsi, and J.M. Thornton. Derivation of 3D coordinate templates for searching structural databases: Applications to ser-his-asp catalytic triads in the serine proteinases and lipases. *Prot. Sci.*, 5:1001–1013, 1996.

[WY98] H. Wako and T. Yamato. Novel method to detect a motif of local structures in different protein conformations. *Protein Eng.*, 11(11):981–990, 1998.

[ZKS96] F. Zu-Kang and M. J. Sippl. Optimum superimposition of protein structures: Ambiguities and implications. *Folding and Design*, 1:123–132, 1996.