Theoretical and computational aspects of protein structural alignment

Paweł Daniluk and Bogdan Lesyng

Abstract Computing alignments of proteins based on their structure is one of the fundamental tasks of bioinformatics. It is crucial in all kinds of comparative analysis as well as in performing evolutionary and functional classification. Whereas determination of sequence relationships is well founded in statistical models, there is still considerable uncertainty over how to describe geometric relationships between proteins. Continuous growth of structural databases calls for fast and reliable algorithmic methods, enabling one to effectively compute alignments of pairs and larger sets of protein molecules. Although such methodologies have been developed over the past two decades, there exist so-called "difficult similarities" which may include repeats, insertions or deletions, permutations and conformational changes. A brief overview of existing methodologies with emphasis on the different approaches to decomposition of structures into smaller fragments is followed by a presentation of a formalism of local descriptors of protein structures. A formal definition of the problem of computing optimal alignments accommodating aforementioned difficulties is presented along with an analysis of the computational complexity of its important variants. Examples of "difficult similarities" and practical aspects of protein structure comparison are discussed.

Paweł Daniluk
Department of Biophysics, Faculty of Physics, University of Warsaw, Żwirki i Wigury 93, 02-892 Warsaw, Poland
Bioinformatics Laboratory, Mossakowski Medical Research Centre, Pawinskiego 5, 02-106 Warsaw, Poland
e-mail: pawel@bioexploratorium.pl

Bogdan Lesyng
Department of Biophysics, Faculty of Physics, University of Warsaw, Żwirki i Wigury 93, 02-892 Warsaw, Poland
Bioinformatics Laboratory, Mossakowski Medical Research Centre, Pawinskiego 5, 02-106 Warsaw, Poland
e-mail: lesyng@imdik.pan.pl

1 Introduction

Proteins are biopolymers comprising one or more polypeptide chains. There exist twenty amino-acid residues which occur in proteins encountered in living organisms. Thus a first approximation (*primary structure*) of a protein is its sequence, normally represented as a string of letters from a 20 letter alphabet. Sequences may be compared to reveal genetic, evolutionary relationships between proteins. Sequence comparison is a variant of a well researched string matching problem which is usually solved with an ubiquitous Needleman-Wunsch algorithm[35] or its heuristic counterparts[39, 2]. A polypeptide chain of a protein after synthesis undergoes a process of folding in which it obtains a well defined characteristic spatial conformation (*tertiary structure*). Structure is instrumental to the role a given protein performs in a living organism. With some simplification, one may assume that a residue sequence determines a spatial structure, which in turn determines a function. Due to the nature of evolutionary processes it can be observed that structure is a much more conserved property than sequence. Even remotely homologous proteins usually have similar tertiary structure. Therefore, comparison of structures, although more difficult, may provide more information on evolutionary and functional relationships than sequence analysis alone.

Although several methods for protein structure comparison have been developed during the past two decades, no single "best of all" method exists, and there are many known cases of so-called difficult similarities, which cannot be correctly solved by most methods. Relatively little effort has been put into development of formal theories of this problem, which would enable a thorough analysis of its properties.

The purpose of this study is to give a brief overview of the existing approaches and methodologies followed by a formal analysis of several variants of the problem of computing alignments based on a set of local similarities. Description of a method based on presented theoretical principles along with a few practical aspects of comparing protein structures are also provided.

This study is organized as follows. The introduction covers basic definitions, contains a brief overview of popular methods, outlines potential pitfalls and gives a short introduction into theory of computational complexity. In the following section the most popular approaches to defining and comparing structural fragments are presented. The third and fourth sections are devoted to the problems of computing an optimal alignment of two or more protein structures and include an analysis of the computational complexity of several variants of these problems. In the fifth section we present practical but rarely used techniques which may be useful in similarity analysis, as well as several case studies.

1.1 Alignments and superpositions

The notion of an alignment in the context of biological sequences originates from the concept of introducing gaps into sequences written one below the other, to maximize the number of columns with identical or similar residues. Alternatively, one may view an alignment as a renumbering of residues such that equivalent residues have the same number. In actual applications an alignment with sequence identity of 30% may be considered significant. Thus, under typical conditions it is impossible to consider alignments which do not preserve order of residues. They would introduce more noise (false positives) than can be coped with. It is well known, however, that structures containing segment swaps or circular permutations may have similar shapes and perform related functions[30, 18]. Similarities of this kind are virtually undetectable by conventional sequence analysis.

In this study we place emphasis on this particular issue, and diverge from the traditional understanding of an alignment. We assume that an *alignment* may be any correspondence between residues of aligned structures. We will avoid the use of terms such as "sequence" or "structure" alignment, which may indicate the basis of the similarity which an alignment is supposed to maximize, but which have no connection to the mapping of residues once it has been computed.

Whenever spatial objects like protein structures are being compared, visual representation of similarity plays a significant role for the end user. Similar molecules can usually be isometrically transformed, so that distances between corresponding residues are minimized and superimposed. In this respect *superposition* is secondary to the alignment, since it may be computed only after corresponding residues have been identified. In extreme cases it may even happen that a biologically correct alignment results in a visually poor superposition.

1.2 Existing methods

Methodologies of protein structure comparison may be classified into two major categories – global and local. In the first one an alignment and superposition of molecules are iteratively improved. Starting with a given alignment, an optimal superposition is computed, then a new alignment is extracted from the superposition by identifying pairs of residues spatially close to each other. Such methods are effective, assuming conformational variability is limited and similarity is significant enough for the process to converge quickly.

Alternatively, computing an alignment may start with identifying a set of local similarities, which afterwards serve as building blocks for the global alignment. There are several methods of decomposing structures into smaller fragments. The most popular are inter-residue distances (SSAP[37], DALI[21], PAUL[50]), single continuous segments of the main chain (CE[46]) or secondary structure elements (SSEs) (VAST[17], SARF[1], MATRAS[26], GANGSTA[19]). Less popular include Delaunay triangulation (TOPOFIT[22]), spherical polar Fourier rep-

resentations (3D-BLAST[31]), and geometric hashing (C_α-match[4]). Local descriptors of protein structures (see section 2.2) have also been successfully applied (DEDAL[10]). Global alignment is computed by selecting the largest consistent set of local similarities. Definitions of consistency and methods for searching the solution space vary. Usually it is required that correspondences between residues given by two consistent alignments have to agree on all residues common to both of them. Sometimes additional criteria are used, such as the similarity of transformations required to superimpose fragments[4] or the ordering in the protein sequence are used. The search of the solution space is performed using algorithms for finding isomorphic subgraphs or cliques, clustering, dynamic programming or other techniques. Some methods use a one-dimensional representation of structure – where each residue is substituted with a characterization of its local features – and use dynamic programming to align such artificial sequences (e.g. SHEBA[23]). Due to the computational complexity caused by the combinatoric size of the solution space, solutions containing circular permutations or segment swaps are disregarded even if the method could find them in theory. Such a situation takes place with the DALI method and its publicly available implementation DaliLite[21, 20]. Sometimes spatial distortions are accommodated by introducing "hinges" (FATCAT[52], FlexProt[43], ProtDeform[41], FlexSnap[42]).

Method name	Year	Authors	Flexible	Segment swaps
SSAP[37]	1989	Orengo and Taylor	No	No
C_α-match[4]	1993	Bachar et al	No	Yes
DALI[21]	1993	Holm and Sander	No	No[1]
VAST[17]	1996	Gibrat et al	No	No
SARF[1]	1996	Alexandrov	No	Yes
CE[46]	1998	Shindyalov and Bourne	No	No
SHEBA[23]	2000	Jung and Lee	No	No
MATRAS[26]	2000	Kawabata and Nishikawa	No	No
FATCAT[52]	2003	Ye and Godzik	Yes	No
TOPOFIT[22]	2004	Ilyin et al	No	No
FlexProt[43]	2004	Shatsky et al	Yes	No
GANGSTA[19]	2008	Guerler and Knapp	No	Yes
ProtDeform[41]	2009	Rocha et al	Yes	No
3D-BLAST[31]	2010	Mavridis and Ritchie	No	No
FlexSnap[42]	2010	Salem et al	Yes	Yes
PAUL[50]	2010	Wohlers et al	No	No
DEDAL[10]	2011	Daniluk and Lesyng	Yes	Yes

Table 1: Selected methods for computing alignments of two protein structures.

The problem of computing multiple alignments of protein structures is much harder and less popular. There are two basic approaches to defining and computing a multiple alignment – searching for a substructure common to all structures com-

[1] DALI in principle is capable of computing alignments with segment swaps, but the publicly available implementation (DaliLite) lacks this feature.

pared, or searching for all similarities as long as equivalences between residues are unambiguous (see section 4.1). Existing methods are often generalizations of methods of computing pairwise alignments. Based on the similarity of all pairs a binary tree is built. Its leaves correspond to structures, while nodes to multiple alignments of structures in its descendants, which are computed in a manner similar to aligning two structures. When computation ends, the root node contains a multiple alignment of all structures (MUSTANG[28], POSA[53]). Sometimes a strategy similar to hierarchical clustering is used. Starting with single structures, at each step the two most similar multiple alignments (or structures) are combined (Matt[33]). There also exist methods where all structures are considered at the same time. MASS[13] is based on searching for maximal correspondences between SSEs assuming rigid global superpositions. On the other hand, MultiProt[44] attempts to align a chosen pivot structure with all others. This process is repeated for all selections of pivot, and the best multiple alignment is returned. DAMA[9] – an extension of the DEDAL method employing an evolutionary algorithm is currently under development.

1.3 Difficult similarities

In many cases, the similarity between protein structures is either obvious or nonexistent. Nevertheless, there exists a "grey area" of so-called difficult similarities. It comprises cases where similarity between sequences cannot be detected or is misleading, the evolutionary relationship is not obvious, or where there exist significant distortions that obscure the similarity. These distortions may include repeats, insertions or deletions, permutations or substantial conformational changes.

Repeating motifs involve a significant combinatorial burden, because in principle all assignments between occurrences of such a motif should be assessed. This is particularly challenging in case of the so-called propeller folds, which contain structures similar to a marine propeller. They are composed of 4 to 8 blades resulting in up to 8! possible assignments of blades and at least 8 equivalent alignments.

Insertions and deletions may be a result of genomic rearrangements. After losing a segment of a significant length a protein may retain its conformation. Nevertheless the similarity is obfuscated by size differences, and the fact that some fragments of the smaller structure usually have a different conformation to fill the gap after missing residues.

Permutations probably pose the most fundamental challenge since the whole concept of an alignment has to be readjusted. Circular permutations are the most common example. They may be caused by gene duplication or rearrangements of the protein chain during folding[49]. Two protein chains are circular permutations of each other if they can be divided into two subunits (A_1-B_1 and A_2-B_2 respectively), such that structures A_1-B_1 and B_2-A_2 are similar in the traditional sense (without permutations). More complex rearrangements (e.g. caused by changes of the number of residues in loop regions) have been observed[18]. Oligomeric structures are another example of sequence rearrangements. Sometimes proteins com-

posed of several chains are similar despite the fact that chain boundaries are placed differently or that numbers of chains differ (in such a case, chains cannot be compared separately).

Finally protein structures are not rigid. Many functions they perform involve conformational changes[16, 12]. Furthermore experimental methods used to determine tertiary structure usually involve changing environmental conditions to nonphysiological, which may distort the studied structure (see section 5.3 for an example). Conformational variability is especially difficult, because assessing structural similarity relies on geometrical data. Distinguishing between "natural" flexibility and dissimilarity may be challenging even to experts.

1.4 Computational complexity

This study presents several results concerning the computational complexity of protein structure alignment. In this section we provide a brief introduction to the theory of computational complexity.

Traditionally computational complexity theory is applied to so-called decision problems, which originate from the formalism of recognizing languages by finite-state automata or Turing machines. In this formalism, instances of a problem are encoded as words over a certain alphabet, and words corresponding to instances with positive answers belong to a language recognized by a machine. Decision problems have a strict form – *"For a given instance I determine whether I satisfies a predicate* $P(I)$*."*, which is quite different to an open form of optimization problems which can be stated as follows *"For a given instance I find a solution S which has a maximal value of property p from all valid solutions of I."*. Any optimization problem, however, can be transformed to a decision problem of the form *"For a given instance I and a value v does there exist a solution with value of property p greater than or equal to v."*.

There are two fundamental classes of decision problems. The first one (P) contains problems which can be solved by a Turing machine in polynomial time. This in practice means, that a polynomial time algorithm for solving such a problem exists, and can be implemented on any computer. Such problems are considered tractable or efficiently solvable since computation time for any instance is limited by a polynomial function of its size. The second major class (NP) comprises problems which can be solved in polynomial time by a non-deterministic Turing machine. This informally means, that given a potential solution to the problem, it is possible to check if it is valid in polynomial time. All problems from P belong to NP, because if the solution can be computed in polynomial time, it can also be efficiently checked. There are, however, problems in NP for which a polynomial time algorithm is not known. Some of them belong to a subclass of NP-complete problems, which may be deemed as a collection of the "hardest" problems in NP. It can be proven that, if there exists a polynomial time solution for any NP-complete problem, all problems

in NP also have a polynomial time solution, and thus P=NP. Until now finding such a solution, or proving that it does not exist, remains an open problem.

Problems in NP can be "ranked" by their "difficulty", with NP-complete problems being the hardest. In order to prove that a given problem P_1 is NP-complete, it is enough to prove that it belongs to NP and that it is "harder" than a known NP-complete problem (P_2). This is performed by constructing a so-called *reduction* of P_2 into P_1. A reduction is a recipe for converting all instances of P_2 into instances of P_1 preserving the decision result (i.e. accepted instances of P_2 are converted to accepted instances of P_1 and *vice versa*). The reduction has to be performed in polynomial time. This proves that if P_1 is tractable, any instance of P_2 also can be solved in polynomial time by converting it into an instance of P_1 and applying an algorithm for P_1. Therefore, if P_1 belongs to P, P_2 does also along with all problems in NP.

More information on computational complexity may be found in the seminal book[15].

2 Fragment-based methods

2.1 Continuous segments, segment pairs

Continuous backbone segments are tremendously popular in all computational applications of protein structure analysis. They have been successfully applied in protein structure prediction[8] and are instrumental in structure comparison. They are small and easy to compare, and thus are a good choice for detecting local similarities which might serve as starting points for a global alignment. Generally only segments of a certain length (varying from 5 to 15) are considered.

As long as two segments of the same length are compared, there is no need to consider any non-trivial mapping between their residues. It is sufficient to apply a distance measure defined on sets of points. *Root Mean Square Distance* (RMSD) is the metric of choice. For two sets of points $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ of size n ($a_i, b_i \in \mathbb{R}^3$) it is defined as:

$$RMSD(A,B) = \min_{\substack{R \text{ - rotation in } \mathbb{R}^3 \\ T \in \mathbb{R}^3}} \sqrt{\frac{\sum_{i=1}^{n} |a_i - (Rb_i + T)|^2}{n}}$$

This formula corresponds to the process of isometric transformation of B such that the sum of squares of distances between respective points in A and transformed points in B is minimized. Although it seems at first glance that computing RMSD is a difficult optimization problem, there exists an efficient algorithm for this (Kabsch algorithm [24, 25]). Although, reproducing it in detail is beyond the scope of this study, we would like to identify one of its relevant features. The preliminary step of the Kabsch algorithm involves computing geometrical centers of A and B as well as a matrix M:

$$M_{ij} = \sum_{k=1}^{n} a_{ki} b_{kj}$$

where a_{ki} (b_{ki}) denotes the ith element of vector a_k (b_k). One can easily see that M and geometrical centers can be recycled. Extending sets A and B after computing their RMSD can be easily implemented, greatly reducing computational complexity (e.g. computing RMSD of segments of length n requires $O(n)$ time, just like computing distances between all prefixes of A and B). A pair of similar segments is usually called an *aligned fragment pair* (AFP).

To our best knowledge all alignment methods using AFP employ some sort of a global similarity measure. It is necessary because the fact that alignment is built from APFs does not imply actual similarity of aligned substructures. The inability to capture spatial relationships between residues distant in the sequence but neighboring in space is the main drawback of continuous segments. It can be amended by using fragments encompassing at least two disjoint pieces of backbone.

DALI[21], a popular and highly regarded method, uses pairs of continuous segments of length 6. A similarity measure is based on the distances between points representing residues. If $A = \{a_1, \ldots, a_{12}\}$ and $B = \{b_1, \ldots, b_{12}\}$ are residues belonging to certain pairs of hexapeptides in structures A and B, similarity is computed as follows:

$$S = \sum_{i=1}^{12} \sum_{j=1}^{12} \theta - \left| d(a_i, a_j) - d(b_i, b_j) \right|$$

where $d(a,b)$ is an Euclidean distance between points a and b, and θ is the parameter determining a zero level of similarity. The distance based approach is appealing because distance maps are invariant under isometric transformations, hence there is no need to search for a transformation giving the optimal superposition. It is also easy to implement and fast for small fragments (although its computational complexity is bound by $O(N^2)$).

2.2 Local descriptors of protein structure

Local Descriptor of Protein Structure[2] is a small fragment of a protein structure, which encompasses a physico-chemical environment of a given residue. In principle it can be defined for any residue in a protein molecule. It is built by identifying residues the selected (*central*) residue is in contact with. Then, for each of the identified residues a 5-residue continuous piece of backbone (*element*) is added to a descriptor. Overlapping elements are combined into *segments* (see Fig. 1).

Substructures chosen with this method may comprise several disjoint segments. Thus, a descriptor may be viewed as a subset of residues enclosed in an irregular surface corresponding to the range of physico-chemical interactions of the central residue with its molecular environment. The radius of a descriptor approximates

[2] In the following text we will simply call it a descriptor.

the range of residue-residue interactions. In contrast to continuous segments, which are limited to one-dimensional neighborhoods along the protein sequence, local descriptors contain information about the spatial environments of residues. They are complete, in a sense that they contain all interacting residues, not some arbitrarily chosen ones. The actual shape (content) of a descriptor depends on a definition of a contact.

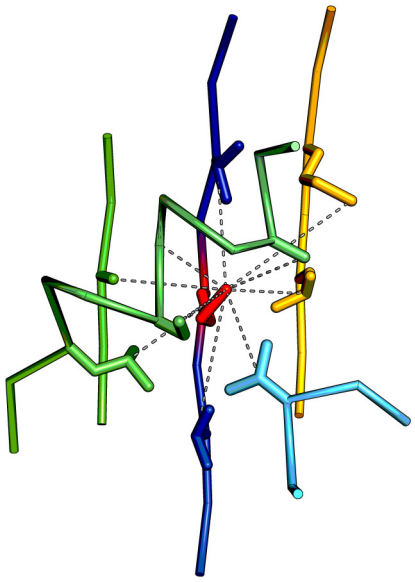


Fig. 1: A descriptor built for residue `MET70` of an ASTRAL domain `d1lg7a_` comprises 9 contacts (dotted lines) between the central residue (red) and residues being centers of elements. Some of the 5-residue elements ovelap and constitute longer segments (two β-strands and an α-helix).

2.2.1 Definition

We consider two residues to be in contact, if one of the following conditions is satisfied:

1. $d_\alpha \leq 6.5$Å,
2. $d_{\beta_x} \leq 8$Å and $d_\alpha - d_{\beta_x} \geq 0.75$Å.

In the above d_α denotes the distance between C^α atoms, and d_{β_x} – a distance between C^{β_x} points (see Fig. 2), which are computed by extending a $C^\alpha - C^{\beta_x}$ vector by 1Å[3]. Such definition of a contact favors residues whose sidechains "point" to-

[3] For glycine we assume $C^{\beta_x} = C^\alpha$; for alanine $C^{\beta_x} = C^\beta$.

wards each other, and is convenient to compute, as it does not depend on sidechain atoms, which may often be missing.

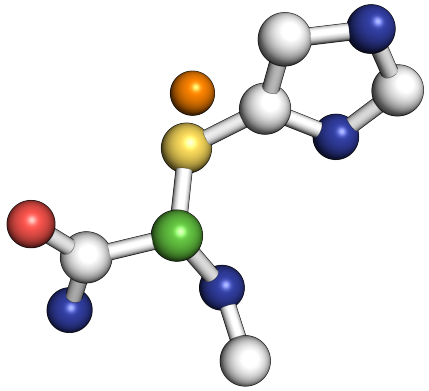


Fig. 2: Histidine with point C^{β_x} (orange); remaining atoms: C^{α} – green, C^{β} – yellow, O – red, N – blue, remaining coal atoms – white.

A protein structure can be viewed as a sequence of residues $(a^{(1)}a^{(2)}\ldots a^{(N-1)}a^{(N)})$. For a given residue $a^{(p)}$ a *descriptor element for* $a^{(p)}$ ($El(a^{(p)})$) is a 5 residue long segment of a backbone around a^p:

$$El(a^{(p)}) = a^{(p-2)}a^{(p-1)}a^{(p)}a^{(p+1)}a^{(p+2)}$$

It is convenient to view a descriptor of a residue a as a triple $\langle a,C,R\rangle$, where C is a set of residues which are in contact with a, and R is a set-theoretical sum of descriptor elements for a and residues in C. We will say that C is a *contact pattern* of a descriptor. One should note that according to this definition residues which are located close to backbone terminals or gaps do not have descriptor elements, and thus cannot belong to a contact pattern of a valid descriptor. Nevertheless, such residues may belong to the set R.

2.2.2 Similarity of local descriptors

Let $D_1 = \langle a_1,C_1,R_1\rangle$ and $D_2 = \langle a_2,C_2,R_2\rangle$ be two descriptors. We will call any partial function $\varphi\colon C_1 \to C_2$ a *mapping of contact patterns* C_1 *and* C_2. A mapping of contact patterns is valid if it can be unambiguously extended to a function $\psi\colon R_1 \to R_2$ such that:

1. If $\varphi(a^{(i)}) = b^{(j)}$, then

$$\psi(El(a^{(i)})) = \psi(a^{(i-2)})\psi(a^{(i-1)})\psi(a^{(i)})\psi(a^{(i+1)})\psi(a^{(i+2)}) = \\ = b^{(j-2)}b^{(j-1)}b^{(j)}b^{(j+1)}b^{(j+2)} = El(b^{(j)})$$

2. $\psi(El(a_1) = \psi(El(a_2))$

In simple terms, a mapping contains pairs of corresponding contacts. It does not necessarily cover all contacts in both descriptors, but each contact may have only one corresponding counterpart in the other descriptor. To be valid a mapping has to preserve overlapping of elements. Contacts with overlapping elements can be mapped only to contacts with the same overlap, while non-overlapping contacts may have only non-overlapping counterparts. We will say that a valid mapping constitutes an *alignment* of descriptors. One should note that under this definition an alignment may contain so-called segment swaps (i.e. aligned segments may have different order in structures they originate from). This is a fundamental difference between traditional understanding of alignment and our definition.

For two descriptors to be similar, an alignment between them has to exist and satisfy requirements imposed on its size and the spatial similarity of aligned substructures. The size can be measured with the number of aligned residues, elements or segments, while spatial similarity may be assessed using a Root Mean Square Distance (RMSD). This is a two-objective optimization problem, since extending alignment will most likely increase RMSD between substructures and *vice versa*.

To reliably solve this problem, we use an extensive search algorithm that finds alignments satisfying the following conditions:

1. the RMSD of aligned elements must not exceed 1.5Å,
2. for each pair of aligned elements, the RMSD of substructures consisting of these elements and respective central elements must not exceed 2.5Å (i.e. elements should have the same position relative to the central element),
3. at least half of the segments must be aligned,
4. the RMSD of aligned residues must not exceed 2.5Å.

The algorithm searches through all alignments satisfying the above conditions. First, all pairs of elements satisfying conditions 1 and 2 are identified. Then, all possible assemblies of those pairs are checked for condition 4. If it is not met, they are reduced by removing the least fitting pairs of elements, until either condition 4 is met or condition 3 is no longer satisfied.

2.2.3 Computational complexity of descriptors comparison

In section 1.4 we have briefly explained the main ideas behind the theory of computational complexity. We will demonstrate that the problem of assessing descriptor similarity is NP-complete. We start by providing a formal definition of the decision problem for finding an optimal descriptor alignment. The definition will be slightly simpler than the one used in the previous section in order to avoid technical difficulties.

Definition 1. For two descriptors D_1, D_2 and constants n and T the **Optimal Alignment of Descriptors** (**OAD**) problem is to determine whether there exists an alignment of D_1 and D_2 covering no less than n residues such that the RMSD between aligned residues is not greater than T.

Theorem 1. ***OAD*** *is NP-complete.*

Proof. First we notice that it is enough to prove that **OAD** is NP-complete for one particular value of T, since, if a problem contains an NP-complete sub-problem it is NP-complete itself. Thus we will assume that T is large enough for any alignment to be structurally acceptable (e.g. $T = \infty$).

The most common way of proving NP-completeness is to define a so-called reduction of a known NP-complete problem to the one being considered. In this case we will use a well known **3-PARTITION** problem[15, problem SP15].

Definition 2. For a given set A containing $3m$ elements, a positive integer B and a function $s\colon A \to \mathbb{N}$ such that:

$$\bigwedge_{a\in A} \frac{1}{4}B < s(a) < \frac{1}{2}B,$$

$$\sum_{a\in A} s(a) = mB,$$

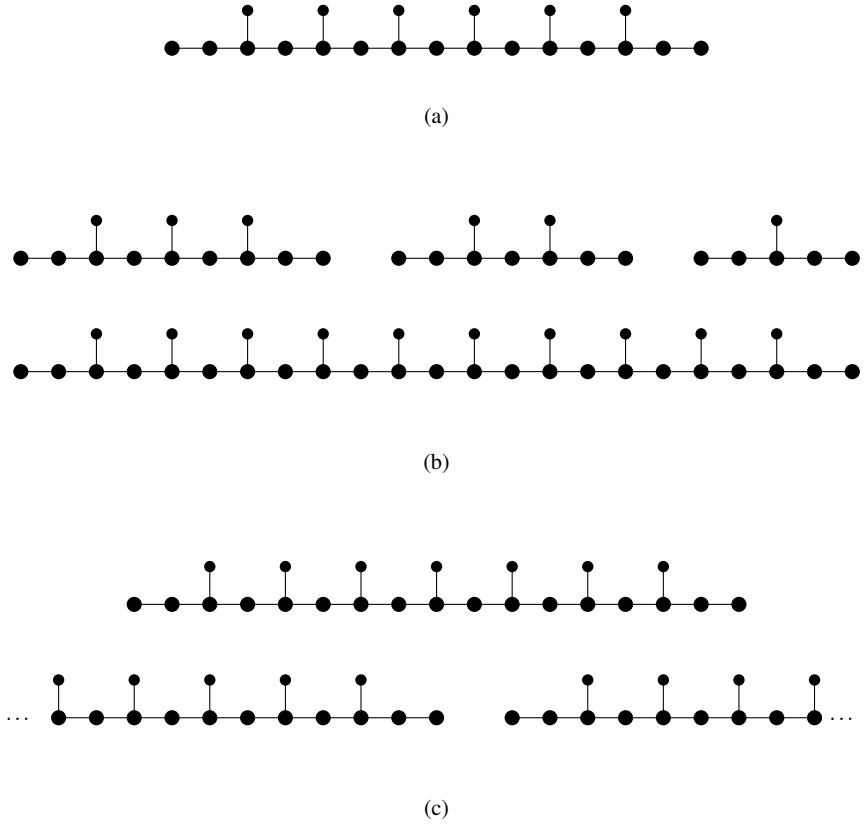
the problem of **3-PARTITION** is to determine whether there exists a partition of A into m disjoint subsets $A_1, A_2, \ldots, A_m$ such that:

$$\bigwedge_{1\le i\le m} \sum_{a\in A_i} s(a) = B$$

It is easy to see that any subset in such partition must contain exactly three elements. Our reduction will assign an instance of **OAD** to any instance of **3-PARTITION**. We will show a method of constructing descriptors D_1 and D_2 for any m, B and s. Because we have assumed that the threshold for the RMSD of aligned residues is infinitely large, we don't have to deal with providing coordinates. It is enough to give contact patterns.

A *comb of length k* (Fig. 3a) is a contact pattern which contains k residues such that subsequent residues lay one residue apart in the sequence . Let D_1 contain $3m$ combs (one for each element of A) of lengths given by values of s for corresponding elements of A. Let D_2 contain m combs of length $B+4$. Finally, let n be equal to the number of residues in $D_1 - m(2B+9)+5$.

To prove that this reduction is correct, we have to show that D_1 and D_2 have a sufficiently large alignment, if and only if there exists a 3-partition of A. Indeed, if such a partition exists, each subset of A can be mapped to a set of three combs in D_1, which can be aligned to a comb of length $B+4$ in D_2 (see Fig. 3b). Conversely, if there exists an alignment of D_1 and D_2, such that all residues in D_1 are aligned, each comb in D_1 is aligned to a part of exactly one comb in D_2. If a comb is aligned



(a)

(b)

(c)

Fig. 3: (a) A comb of length 6. (b) An alignment of three combs from D_1 to a comb in D_2. (c) If one comb is aligned to two combs, at least one residue remains unaligned.

to two separate combs, at least one residue in the first comb has to remain unaligned (see Fig. 3c), which leads to a contradiction.

Presented reduction can be computed in polynomial time with respect to mB, which is acceptable because **3-PARTITION** is strongly NP-complete (NP-completeness is preserved regardless of the method of encoding numbers).

This, and the fact that given an alignment of two descriptors it is possible to compute its size and RMSD in polynomial time, proves that **OAD** is NP-complete.

The significance of the presented theorem should be properly understood. The NP-completeness of aligning two descriptors means that most likely[4] any algorithm will require an exponential time depending on the size of descriptors being compared. Due to limits imposed by the physics of protein molecules the size of a de-

[4] Unless P=NP.

scriptor is of course strictly limited. Descriptors having more than 15 elements do not exist or are extremely rare. Therefore, computation time is limited by a constant. We have also omitted structural aspects of the comparison by choosing an infinite RMSD threshold. This, to our best knowledge, was required to formulate a theorem, which could be proven. Encoding the property of being a protein[5] in strict mathematical terms is beyond our capacity. Nevertheless this theorem is useful because it reflects the fact that without identifying a more complex internal structure for a descriptor the extensive search is justified.

3 From local similarities to global alignments

In this section we describe a generic paradigm of computing alignments of protein structures based on a set of local similarities. We assume that local similarities, regardless of their particular definition, constitute mappings between the residues they encompass. As in the previous chapter we apply the term "alignment" to any such mapping, despite the fact that it may not preserve ordering of residues in their respective sequences. We also assume that local alignments should be treated as indivisible and immutable blocks and may only be included in the resulting solution as a whole. All mappings in the solution have to belong to at least one block included in it.

By these assumptions we simplify the problem, without risking loss of generality. In all actual applications one can safely assume that the size of a local similarity is bound by a constant. The number of subsimilarities which would have to be considered if local alignments were divisible is therefore also bound. The same applies to the majority of useful mutations which one might care to explore. Thus the problem with mutable, divisible blocks, can be converted to a problem where blocks are immutable, multiplying their number by a constant factor.

The difficulty in building a global alignment in this paradigm lies in the fact that local similarities cannot be used in the same alignment, if there exists a residue which they map differently. An alignment may be built from local alignments, which are all pairwise consistent. This brings into mind the well known NP-complete clique finding problem, but a careful analysis taking into account actual properties of local similarities used is required to correctly assess the computational complexity.

The score of a resulting global alignment may solely depend on its size (quality being assured by local similarities), or there may be some more complex scoring function (e.g. RMSD). We examine the case where alignment size is to be maximized regardless of its quality.

For the purpose of this and subsequent sections let S_1 and S_2 be certain protein structures, and Φ be the set of local similarities represented by mappings of residues.

Let $\xi : S_1 \to S_2$ be a mapping of residues. A *support of* ξ *in* Φ ($Supp(\xi, \Phi)$) is a subset of mappings from Φ which are completely included in ξ.

[5] A protein is a polypeptide which under physiological conditions assumes and maintains a certain native conformation.

Definition 3. An *alignment of S_1 and S_2 with support in Φ* is a mapping of S_1 and S_2, which has a support in Φ covering all its residues.

According to this definition, every element of Φ is an alignment of structures itself. A support of the alignment contains the set of local similarities, which have to be combined to build this alignment. They are all consistent, because they are all completely covered by the same alignment.

Definition 4. For given structures S_1 and S_2, set Φ and number k the *Optimal Structure Alignment Problem* (**OSA**) is to determine, whether there exists an alignment of S_1 and S_2 with support in Φ covering at least k residues.

3.1 Computational complexity of Optimal Structure Alignment Problem

Theorem 2. ***OSA*** *is NP-complete.*

Proof. To prove that **OSA** is NP-complete we will provide a reduction of another NP-complete problem to **OSA**. There is a well known NP-complete problem called **3-DIMENSIONAL MATCHING** (**3DM**)[15, problem SP16].

Definition 5. For a given set $M \subseteq W \times X \times Y$, where W, X i Y are disjoint sets of size q, the **3-DIMENSIONAL MATCHING** problem is to determine whether there exists a subset $M' \subseteq M$ such that $|M'| = q$ and elements of M' are disjoint.

In other words, the task is to choose from a given set of triples (M) a subset in which every element from sets W, X and Y occurs exactly once. There also exists a two dimensional version of this problem – **2-DIMENSIONAL MATCHING** (**2DM**, also called a marriage matching problem) where $M \subseteq X \times Y$. Although very similar surprisingly it can be solved in polynomial time. We will use a slightly modified version of this problem.

Definition 6. For given sets $M \subseteq X \times Y$ and $G \subseteq P(M)$[6], where X and Y are disjoint sets of size q, the **RESTRICTED 2-DIMENSIONAL MATCHING** (**R2DM**) problem is to determine, whether there exists a subset $M' \subseteq M$ such that, $|M'| = q$, elements of M' are disjoint and there exists $G' \subseteq G$ such that $\bigcup G' = M'$.

R2DM may be viewed as a case of the marriage matching where would-be wives set conditions of the kind: "I will marry you, if Mr. X marries Ms. W and Mr. Q marries Ms. S". Such conditions are encoded as elements of the set G. To prove, that **R2DM** is NP-complete, we may use a simple reduction of **3DM**. Each triple from M in **3DM** is encoded as two pairs in M and a set containing these pairs in G. Figure 4 contains an example of such transformation. One can easily establish that

[6] $P(M)$ denotes a power set of M, i.e. a family of all subsets of M.

a solution of an instance of **R2DM** obtained from **3DM** can always be converted to a solution of the original problem. Furthermore, if the original **3DM** instance has a valid solution, the corresponding instance of **R2DM** is always solvable.

R2DM is very convenient for proving that **OSA** is intractable. In the conversion of an **R2DM** instance to **OSA** sets X and Y will correspond to sets residues of S_1 and S_2; set M – to the set of all pairs of mapped residues in alignments from Φ and finally set G – to Φ itself. Elements of G are sets of pairs from M which have to be picked together. In the case of alignment with support in Φ, each pair of aligned residues has to belong to a local alignment from Φ. Let S be a set of pairs from G. S is converted to an alignment, which for each pair in S maps together residues corresponding to its elements. A subset A' being a solution of **R2DM** corresponds to the alignment covering whole structures. A set G' corresponds to the support of this alignment in Φ.

$$
\begin{aligned}
W &= \{\mathrm{a},\mathrm{b},\mathrm{c}\} \\
X &= \{\mathrm{A},\mathrm{B},\mathrm{C}\} \\
Y &= \{1,2,3\} \\
M &= \{\langle \mathrm{a},\mathrm{A},1\rangle, \langle \mathrm{b},\mathrm{B},2\rangle, \langle \mathrm{c},\mathrm{C},3\rangle, \langle \mathrm{a},\mathrm{B},3\rangle\}
\end{aligned}
\qquad
M' = \{\langle \mathrm{a},\mathrm{A},1\rangle, \langle \mathrm{b},\mathrm{B},2\rangle, \langle \mathrm{c},\mathrm{C},3\rangle\}
$$

(a)

$$
\begin{aligned}
X &= \{\mathrm{a},\mathrm{b},\mathrm{c},\mathrm{A}',\mathrm{B}',\mathrm{C}'\} \\
Y &= \{\mathrm{A},\mathrm{B},\mathrm{C},1,2,3\} \\
M &= \left\{ \begin{array}{l} \langle \mathrm{a},\mathrm{A}\rangle,\ \langle \mathrm{A}',1\rangle, \\ \langle \mathrm{b},\mathrm{B}\rangle,\ \langle \mathrm{B}',2\rangle, \\ \langle \mathrm{c},\mathrm{C}\rangle,\ \langle \mathrm{C}',3\rangle, \\ \langle \mathrm{a},\mathrm{B}\rangle,\ \langle \mathrm{B}',3\rangle \end{array} \right\} \\
G &= \left\{ \begin{array}{l} \{\langle \mathrm{a},\mathrm{A}\rangle, \langle \mathrm{A}',1\rangle\}, \\ \{\langle \mathrm{b},\mathrm{B}\rangle, \langle \mathrm{B}',2\rangle\}, \\ \{\langle \mathrm{c},\mathrm{C}\rangle, \langle \mathrm{C}',3\rangle\}, \\ \{\langle \mathrm{a},\mathrm{B}\rangle, \langle \mathrm{B}',3\rangle\} \end{array} \right\}
\end{aligned}
\qquad
\begin{aligned}
M' &= \left\{ \begin{array}{l} \langle \mathrm{a},\mathrm{A}\rangle,\ \langle \mathrm{A}',1\rangle, \\ \langle \mathrm{b},\mathrm{B}\rangle,\ \langle \mathrm{B}',2\rangle, \\ \langle \mathrm{c},\mathrm{C}\rangle,\ \langle \mathrm{C}',3\rangle \end{array} \right\} \\
G' &= \left\{ \begin{array}{l} \{\langle \mathrm{a},\mathrm{A}\rangle, \langle \mathrm{A}',1\rangle\}, \\ \{\langle \mathrm{b},\mathrm{B}\rangle, \langle \mathrm{B}',2\rangle\}, \\ \{\langle \mathrm{c},\mathrm{C}\rangle, \langle \mathrm{C}',3\rangle\} \end{array} \right\}
\end{aligned}
$$

(b)

Fig. 4: (a) Sample instance of **3DM**. (b) The same instance converted to **R2DM**. Each triple is converted to two pairs and an element in set G. Primes are added to element names to comply with the requirement that sets X and Y are to be disjoint.

To make the reduction possible, local similarities have to allow for sequence swaps. Otherwise, instances of **R2DM** for which sets X and Y cannot be ordered in such a way that all elements in G are ordered on both positions, could not be converted to an instance of **OSA**.

3.2 Important variants of Optimal Structure Alignment problem and their complexity

In the previous section we have proven the intractability of the Optimal Structure Alignment problem. The proof presented applies to the most generic version of **OSA** where local similarities may be any arbitrary mappings between residues. All "real" approaches known to us employ local similarities having a well defined structure (e.g. continuous segments, pairs of segments, local descriptors). If we look back to the proof of NP-completeness of **OSA**, it is evident, that it cannot be applied in the case of local similarities in the form of single continuous segments. It also remains to be seen whether **OSA** would remain intractable if it was restricted to alignments without permutations.

Definition 7. An *Optimal Straight Structure Alignment problem* (**OSSA**) is a variant of **OSA** in which an alignment satisfying the size threshold cannot contain segment swaps.

Theorem 3. *If the set of local similarities in an instance of **OSSA** contains only matchings of single continuous segments, the computation required to solve **OSSA** can be performed in polynomial time.*

Proof. These sort of cases can easily be solved with the modified Smith-Waterman algorithm. Algorithms of this sort based on dynamic programming usually have polynomial complexity. In this particular case a pessimistic estimate of computation time linearly depends on the number of residues in aligned structures and the size of Φ ($O(|S_1||S_2||\Phi|)$).

Theorem 3 encourages further questions regarding the intractability of **OSSA**. Perhaps it is also easy for more complex elements of Φ.

Theorem 4. *A variant of **OSSA** where the set of local similarities may contain matchings of three separate continuous segments is NP-complete.*

Proof. We will prove NP-completeness by reduction of the popular **3SAT** problem [15, problem LO01].

Definition 8. For a given collection of clauses C on a set of variables U, where each clause is an alternative of exactly three literals from U (positive or negative) and each variable is used exactly three times, the **3SAT** problem is to determine whether there exists a truth assignment for U which satisfies all clauses in C.

3SAT is one of the oldest known NP-complete problems. It is useful in proving the intractability of problems where one can identify a set of "switches" and a certain pattern they have to achieve. The fact that each variable appears no more than three times in the collection of clauses is instrumental to our proof. We begin with the observation that, without any loss of generality, we may assume that every variable appears at least once in a positive and negative literal. Otherwise (in the case of only

positive or only negative appearances), such variables can be easily eliminated by setting their value to true or false respectively. Let k be the number of variables and l be the number of clauses in a certain instance of **3SAT**. Below, we demonstrate a conversion of such an instance to an instance of **OSSA**.

Let S_1 be a protein structure which contains segments: $v_1, v_2, \ldots, v_k, c_1, c_2, \ldots, c_l$; and S_2 a protein structure containing segments: $t_1, f_1, t_2, f_2, \ldots, t_k, f_k, c'_1, c'_2, \ldots, c'_l$. In both these structures segments appear in the order given above. All segments in S_1 are disjoint, whereas in S_2 t_i overlaps f_i, and all other segment pairs are disjoint. All segments are of equal length.

Segments v_i correspond to variables, segments t_i and f_i correspond to assignment of true and false values to respective variables, and segments c_i and c'_i correspond to clauses. For each variable we define two local similarities φ_i and $\overline{\varphi}_i$:

$$\varphi_i(s) = \begin{cases} t_i & s = v_i \\ c'_p & s = c_p \text{ and clause } p \text{ contains a positive appearance of } i\text{th variable} \end{cases}$$

$$\overline{\varphi}_i(s) = \begin{cases} f_i & s = v_i \\ c'_p & s = c_p \text{ and clause } p \text{ contains a negative appearance of } i\text{th variable} \end{cases}$$

Example 1. Let C be a collection of clauses on $U = \{u_1, u_2, u_3\}$:

$$C = \{\{\neg u_1, u_2, u_3\}, \{u_1, \neg u_2, u_3\}, \{u_1, u_2, \neg u_3\}\}$$

equivalent to the following formula:

$$(\neg u_1 \vee u_2 \vee u_3) \wedge (u_1 \vee \neg u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee \neg u_3)$$

This instance of **3SAT** could be converted to the following instance of **OSSA** (assuming that all segments are of length 5) ($\varphi(a) = \perp$ means that $a \notin Dom(\varphi)$):

$$S_1 = \overbrace{a^{(1)}\ a^{(2)}\ a^{(3)}\ a^{(4)}\ a^{(5)}}^{v_1}\overbrace{a^{(6)}\ a^{(7)}\ a^{(8)}\ a^{(9)}\ a^{(10)}}^{v_2}\overbrace{a^{(11)}a^{(12)}a^{(13)}a^{(14)}a^{(15)}}^{v_3}$$
$$\overbrace{a^{(16)}a^{(17)}a^{(18)}a^{(19)}a^{(20)}}^{c_1}\overbrace{a^{(21)}a^{(22)}a^{(23)}a^{(24)}a^{(25)}}^{c_2}\overbrace{a^{(26)}a^{(27)}a^{(28)}a^{(29)}a^{(30)}}^{c_3}$$

$$S_2 = \overbrace{b^{(1)}\underbrace{b^{(2)}\ b^{(3)}\ b^{(4)}\ b^{(5)}}_{f_1}\ b^{(6)}}^{t_1}\overbrace{b^{(7)}\underbrace{b^{(8)}\ b^{(9)}\,b^{(10)}b^{(11)}}_{f_2}b^{(12)}}^{t_2}\overbrace{b^{(13)}\underbrace{b^{(14)}b^{(15)}b^{(16)}b^{(17)}}_{f_3}b^{(18)}}^{t_3}$$
$$\overbrace{a^{(19)}a^{(20)}a^{(21)}a^{(22)}a^{(23)}}^{c'_1}\overbrace{a^{(24)}a^{(25)}a^{(26)}a^{(27)}a^{(28)}}^{c'_2}\overbrace{a^{(29)}a^{(30)}a^{(31)}a^{(32)}a^{(33)}}^{c'_3}$$

$$\varphi_{\mathbf{1}}(S_1) = \overbrace{\underbrace{b^{(1)}\ b^{(2)}\ b^{(3)}\ b^{(4)}\ b^{(5)}}_{t_1}}^{v_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_3}$$
$$\overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_1}\ \overbrace{\underbrace{b^{(24)}b^{(25)}b^{(26)}b^{(27)}b^{(28)}}_{c_2'}}^{c_2}\ \overbrace{\underbrace{b^{(29)}b^{(30)}b^{(31)}b^{(32)}b^{(33)}}_{c_3'}}^{c_3}$$

$$\overline{\varphi}_{\mathbf{1}}(S_1) = \overbrace{\underbrace{b^{(2)}\ b^{(3)}\ b^{(4)}\ b^{(5)}\ b^{(6)}}_{f_1}}^{v_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_3}$$
$$\overbrace{\underbrace{b^{(19)}b^{(20)}b^{(21)}b^{(22)}b^{(23)}}_{c_1'}}^{c_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_3}$$

$$\varphi_{\mathbf{2}}(S_1) = \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_1}\ \overbrace{\underbrace{b^{(7)}\ b^{(8)}\ b^{(9)}\ b^{(10)}b^{(11)}}_{t_2}}^{v_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_3}$$
$$\overbrace{\underbrace{b^{(19)}b^{(20)}b^{(21)}b^{(22)}b^{(23)}}_{c_1'}}^{c_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_2}\ \overbrace{\underbrace{b^{(29)}b^{(30)}b^{(31)}b^{(32)}b^{(33)}}_{c_3'}}^{c_3}$$

$$\overline{\varphi}_{\mathbf{2}}(S_1) = \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_1}\ \overbrace{\underbrace{b^{(7)}\ b^{(8)}\ b^{(9)}\ b^{(10)}b^{(11)}}_{f_2}}^{v_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_3}$$
$$\overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_1}\ \overbrace{\underbrace{b^{(24)}b^{(25)}b^{(26)}b^{(27)}b^{(28)}}_{c_2'}}^{c_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_3}$$

$$\varphi_{\mathbf{3}}(S_1) = \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_2}\ \overbrace{\underbrace{b^{(13)}b^{(14)}b^{(15)}b^{(16)}b^{(17)}}_{t_3}}^{v_3}$$
$$\overbrace{\underbrace{b^{(19)}b^{(20)}b^{(21)}b^{(22)}b^{(23)}}_{c_1'}}^{c_1}\ \overbrace{\underbrace{b^{(24)}b^{(25)}b^{(26)}b^{(27)}b^{(28)}}_{c_2'}}^{c_2}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_3}$$

$$\overline{\varphi}_{\mathbf{3}}(S_1) = \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{v_2}\ \overbrace{\underbrace{b^{(13)}b^{(14)}b^{(15)}b^{(16)}b^{(17)}}_{f_3}}^{v_3}$$
$$\overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_1}\ \overbrace{\perp\ \perp\ \perp\ \perp\ \perp}^{c_2}\ \overbrace{\underbrace{b^{(29)}b^{(30)}b^{(31)}b^{(32)}b^{(33)}}_{c_3'}}^{c_3}$$

All clauses are satisfied by the assignment:

$$u_1 \to 0, u_2 \to 1, u_3 \to 1$$

Therefore, there exists an alignment with support $\{\overline{\varphi}_1, \varphi_2, \varphi_3\}$:

$$\xi(S_1) = \overbrace{\underbrace{b^{(2)}\ b^{(3)}\ b^{(4)}\ b^{(5)}\ b^{(6)}}_{f_1}}^{v_1}\overbrace{\underbrace{b^{(7)}\ b^{(8)}\ b^{(9)}\, b^{(10)} b^{(11)}}_{t_2}}^{v_2}\overbrace{\underbrace{b^{(13)} b^{(14)} b^{(15)} b^{(16)} b^{(17)}}_{t_3}}^{v_3}$$
$$\overbrace{\underbrace{b^{(19)} b^{(20)} b^{(21)} b^{(22)} b^{(23)}}_{c'_1}}^{c_1}\overbrace{\underbrace{b^{(24)} b^{(25)} b^{(26)} b^{(27)} b^{(28)}}_{c'_2}}^{c_2}\overbrace{\underbrace{b^{(29)} b^{(30)} b^{(31)} b^{(32)} b^{(33)}}_{c'_3}}^{c_3}$$

ξ is a straight alignment of size 30.

An alignment covering at least $(k+l)r$ residues, where r is the length of segments, corresponds to an assignment for which all clauses are true. To prove this, we note that no alignment with support in $\Phi = \{\varphi_1, \overline{\varphi}_1, \ldots, \varphi_k, \overline{\varphi}_k\}$ can contain both φ_i and $\overline{\varphi}_i$, because segment v_i cannot be aligned to both t_i and f_i at the same time. This, in the context of the **3SAT** problem, establishes that in no assignment can a variable be both true and false. If an alignment is supposed to cover $(k+l)r$ residues it has to cover all segments in S_1. This means that all variables will have an assignment.

Each clause is an alternative of three literals. It is sufficient for one of them to have a truth value for the clause to be satisfied. In our reduction this is expressed by aligning segments c_p and c'_p. All clauses have to be satisfied for the alignment to be sufficiently large.

The remaining technical details are omitted for reasons of brevity.

To conclude we demonstrate that **OSA** is intractable if local similarities are allowed to comprise at least two continuous segments.

Theorem 5. *A variant of **OSA** where the set of local similarities may contain matchings of two separate continuous segments is NP-complete.*

Proof. To prove this theorem it suffices to note that a variant of **R2DM** in which each element of the set G contains exactly two pairs from A is NP-complete. This follows directly from the reduction we have used to prove the intractability of **R2DM**. All instances of **R2DM** resulting from it have this feature. Therefore, if we assume that structures are divided into non-overlapping segments of the same length, any local similarity in the reduction from **R2DM** to **OSA** will consist of no more than two segments.

This series of theorems is briefly summed up in Fig. 5. Computing straight alignments is easier than computing alignments with permutations. Nevertheless, in the case of moderately complex local similarities with three non-overlapping segments,

both problems are intractable. The case of **OSSA** with 2-segment similarities is particularly interesting since it is the basis of a popular structure alignment method DALI[21]. Unfortunately, we are not aware of any results concerning its computational complexity.

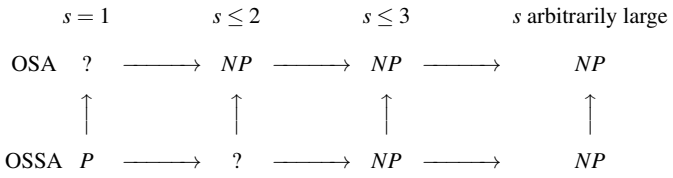


Fig. 5: Computational complexity of **OSA** and **OSSA** depending on the maximal number of segments in local similarities s.

OSA and **OSSA** are theoretical approximations of actual problems. As in the previous section, we have disregarded the fact that protein structures have several properties, which are hard or impossible to describe with a mathematical formalism. Therefore, results shown here may not reflect the actual complexity of these problems applied to real proteins. Nevertheless, they are useful, since if the fact of a polypeptide being a protein cannot be described formally, it cannot be used in the process of designing and proving properties of an algorithm. Knowledge that a certain variant of **OSA** is NP-complete indicates that it is very unlikely for an accurate, polynomial time algorithm to exist and thus favors the application of extensive search or heuristic solutions.

Among numeorus simplifications, we have also disregarded the issue of assessing alignment quality. Introducing a restriction that an alignment has to satisfy some quality requirement (e.g. RMSD) apart from the size criteria, doesn't necessarily make the problem harder. For example, if aligned parts were to have identical shapes (RMSD equal to zero), only local similarities with zero RMSD could be used, thereby drastically limiting the size of set Φ. Nevertheless, theoretical complexity, which should normally be assessed for all threshold values, would not change.

3.3 Solving Optimal Structure Alignment Problem with local descriptors

Local descriptors are particularly useful in computing protein structure alignments[10]. Local similarities defined by pairs of descriptors are usually significant, so that alignments with support in the set of such similarities do not require further verification of quality. Therefore, computing an alignment of structures using descriptors amounts to solving an instance of **OSA**. Since the problem is NP-complete, there

is no disadvantage in applying formalism and experience from another well-known NP-complete problem.

All similarities in the support of an alignment are required to be consistent. Conversely, if all local similarities in a certain set are consistent with each other, such a set is a support of some valid alignment. The consistency of local similarities can be described by a graph, with nodes representing similarities. Nodes in this graph are connected by an edge, if corresponding similarities are consistent. A clique[7] in such a graph may be interpreted as a valid alignment between the structures. As long as the function used to score the alignments does not decrease with the clique growth, maximal alignments can be found by looking for the maximal cliques.

Accurate solution – extensive search

Clique searching algorithms are usually designed to find the largest clique in terms of the number of nodes. In solving the **OSA**, this might not be enough. The goal is to find an alignment covering as many residues as possible. The largest clique does not necessarily have this property, since local similarities in a clique usually overlap, contributing to the "thickness" of coverage instead its "breadth". Most likely "thickest" coverages should correspond to the best alignments, nevertheless an algorithm to guarantee the optimal solution has to compute and assess all maximal cliques.

Applying a branch-and-bound strategy to build all possible cliques, while preserving a required number of the highest scoring alignments, is a viable solution. Each node either belongs to the clique or not. Such decisions can be made separately for each node in an arbitrary order assuming that choices which would violate the clique condition are disallowed. Thus, building all maximal cliques can be performed by traversing a decision tree in which nodes at the kth level correspond to the decision of including the kth graph node in the subset. It is well known, that such an approach can be vastly improved by employing a branch-and-bound strategy. In order to make this computation feasible we introduced two optimizations (cuts).

If a clique in a given branch can be unequivocally expanded with a previously rejected node, it is abandoned, because it does not contain any maximal cliques (maximal cliques containing those abandoned belong to another branch of a tree). This ensures that only maximal cliques are obtained and each is constructed precisely once.

Since the goal is to compute the largest alignments, branches which do not contain cliques corresponding to sufficiently large alignments may be abandoned. The lower bound of the size of alignments of interest may be given as a parameter or gradually increased as alignments are being computed. An upper bound of the size of a maximal alignment in a given branch can be computed as a sum of the size of the alignment being constructed, and a number of residues outside this alignment covered by descriptor pairs which are yet to be considered. This is not an exact size of the best solution in the branch, because some descriptor pairs are contradictory

[7] Clique in a graph is a subset of nodes such that every two nodes in the subset are connected by an edge.

and cannot be combined in one alignment, but still such an upper bound is frequently low enough to discard significant portions of a decision tree.

This strategy can be modified to search for alignments which have a certain property. For example, in order to find optimal alignments comprising only one structurally continuous fragment a variant may be used which extends the clique only if the subalignment which is being added has common residues with the alignment being extended.

Monte-Carlo approximation

In certain cases extensive search even with the cuts described above is infeasible. This is caused by a large number of suboptimal alignments which cannot be pruned from the decision tree. Large structures with a high degree of self-similarity (i.e. recurring structural motifs) are especially affected. Nevertheless, in these cases correct alignments are most likely easily identifiable by visual inspection. Therefore, one might speculate it should be possible to easily detect them without a systematic search of the overwhelming solution space. Monte-Carlo methods[34] have a huge potential in finding most probable states of complex systems. In particular, a widely recognized Replica Exchange Monte Carlo algorithm[48] can be used to search for high score alignments. Here we will mention the algorithm for generating transitions between states, and the energy function. Let $C_n = \{d_1, d_2, \ldots, d_n\}$ be the clique defining a state at the nth step. The clique C_{n+1} describing the state in the next step is generated as follows:

1. randomly pick a graph node d not belonging to C_n,
2. take a set C_{n+1} containing d and elements from C_n which are connected to d (one sees it is a clique),
3. if there are graph nodes which belong to every maximal clique containing C_{n+1}, add them to C_{n+1}.

Such parameters as number of steps, number of replicas, their temperatures, and exchange frequency should be adjusted to reproduce accurate results in the shortest time.

4 Alignments of multiple structures

In contrast to alignments of pairs of structures, which can be defined as mappings between respective sets of residues, alignments of multiple structures do not have one canonical definition. Intuitively multiple alignment is some sort of correspondence between residues in several structures. The fundamental question is whether such correspondence is transitive, i.e. if residue a is structurally equivalent to b, which is equivalent to c, does it imply that a is equivalent to c. There are three

common strategies which are used to compute a score of the multiple alignment of sequences:

- sum of pairs (*SP-score*),
- star alignment,
- alignment according to a given phylogenetic tree (tree alignment).

Each of the above has different properties and assumptions. Star alignment, which aims at finding a sequence most similar to a given set (which may be viewed as their average) may be interpreted in the context of structure alignment as searching for a core common to all structures. Maximization of the SP-score, on the other hand, may be understood as searching for similarities within all subsets of structures. Tree alignment is somewhat in between and can be applied, if there exists a hypothetical phylogenetic hierarchy based on some external knowledge. No matter which of these three strategies is used, the problem of multiple alignment of sequences is NP-hard[14].

When comparing protein structures it is easier to detect a conserved core common to all structures (if only it exists) than to identify all similarities. Nevertheless, we focus on the sum of pairs strategy, since it gives more complete information. We assume that a multiple alignment can be described with a set of alignments of all pairs of structures. Naturally, not every set of pairwise alignments describes a multiple alignment. We set out to establish the necessary and sufficient condition for this.

4.1 Optimal Structural Multiple Alignment Problem

We will define a multiple alignment in a similar fashion as pairwise alignment. Firstly, we develop a notion of *multiple mapping*. Let $\mathscr{S} = \{S_1, S_2, \ldots, S_N\}$ be a set of structures. A *multiple mapping* of structures in $\mathscr{S}$ is a symmetric reflective relation between all residues belonging to structures in $\mathscr{S}$ in which under no circumstances can two different residues in one structure be transitively aligned with each other. This property is crucial, since it distinguishes the problem of computing an optimal multiple alignment from computing optimal alignments between all pairs of structures. It may happen, that residue $a_1 \in S_1$ is mapped to $b \in S_2$ in the optimal alignment between these structures, $b \in S_2$ is mapped to $c \in S_3$, which in turn is equivalent to $a_2 \in S_1$. In such cases pairwise alignments cannot be directly merged into a valid multiple alignment, because both a_1 or a_2 would have to be aligned with b and c.

Definition 9. An *multiple alignment of structures in set $\mathscr{S}$ with support in Φ* is a multiple mapping of structures in $\mathscr{S}$, such that pairwise alignments derived from it for each pair of structures in $\mathscr{S}$ have support in Φ covering all their residues. We assert that the aforementioned set of pairwise alignments *induces* the multiple alignment.

This definition is a direct generalization of definition 3. It is easy to see, that if $\mathscr{S}$ contains two structures, their multiple alignment is equivalent to pairwise alignment[8]. For a given multiple alignment there exists exactly one set of pairwise alignments inducing it. If a set of pairwise alignments induces a multiple alignment, we will say that it is *consistent*. The *size of a multiple alignment* is the average number of residues covered by the pairwise alignments which induce it.

Definition 10. For a given set of structures $\mathscr{S}$, set Φ, and number k the *Optimal Structure Multiple Alignment Problem* (**OSMA**) is to determine whether there exists a multiple alignment in $\mathscr{S}$ with support in Φ of size no less than k.

4.2 Analysis of computational complexity

Theorem 6. *Optimal Structure Multiple Alignment Problem is NP-complete.*

Proof. It is easy to see that **OSMA** contains the problem of computing optimal pairwise alignments (**OSA**). Therefore it is NP-hard. It belongs to the NP class because computing the size of a given multiple alignment can be readily performed in polynomial time.

Theorem 6 is rather obvious and unfortunately gives little insight into the complexity of the problem. We have already established that although **OSA** is NP-complete it can be effectively solved either by accurate algorithms or Monte-Carlo approximations. If computing multiple alignments was not harder than **OSA**, most likely we would be able to use similar approaches to solve it. On the other hand, computing multiple sequence alignments is NP-complete, although pairwise sequence alignments can be performed in polynomial time. Intuitively, one might argue that "multiplicity" of the alignment introduces intractability. If this was the case, computing multiple structural alignments would require a different approach. It may seem that assuming that **OSA** belongs to P and analyzing the complexity of **OSMA** under this assumption would be the simplest way to check this hypothesis. Unfortunately, assuming that a certain NP-complete problem can be solved in polynomial time is equivalent to assuming that P equals NP (i.e. all NP problems can be solved in polynomial time). Therefore, we must analyze the complexity of such variants of **OSMA** for which all relevant **OSA** can be solved in constant time.

Theorem 7. *For any value of k, a variant of **OSMA** in which the number of local similarities in Φ for each pair of structures does not exceed k is NP-complete.*

Before we prove this theorem, let us note that if size of set Φ in **OSA** is limited by a constant, then the number of possible alignments with support in Φ is limited by

[8] We deliberately skip over the fact that multiple alignment is a relation while a pairwise alignment is a function. The property of being a multiple alignment guarantees that it can be converted to a function in a trivial way.

2^k. This means that an extensive search algorithm for finding an optimal alignment will have a computational complexity of $O(2^k) = O(1)$ (because k is constant). In layman's terms theorem 7 establishes that pessimistic computation time for **OSMA** is exponential with respect to the number of structures[9].

Proof. We once more use **3SAT** (see Def. 8). Let $U = \{u_1, \ldots, u_k\}, C = \{C_1, \ldots, C_l\}$ be an instance of **3SAT**. We will construct three sets of structures corresponding to: variables (set V), clauses (set K) and assignment of values (set L):

(1) $V = \{V_1, \ldots, V_k\}$, gdzie $V_i = a_1^{V_i} a_2^{V_i} \ldots a_{19}^{V_i} a_{20}^{V_i}$[10]
(2) $K = \{K_1, \ldots, K_l\}$, gdzie $K_i = a_1^{K_i} a_2^{K_i} \ldots a_{14}^{K_i} a_{15}^{K_i}$
(3) $L = \{L_0\}$, gdzie $L_0 = a_1^{L_0} a_2^{L_0} \ldots a_{20}^{L_0} a_{21}^{L_0}$

To simplify the notation we will define the following segments:

(1) $v_i = a_1^{V_i} \ldots a_5^{V_i}$, $v_i' = a_6^{V_i} \ldots a_{20}^{V_i}$
(2) $k_{i1} = a_1^{K_i} \ldots a_5^{K_i}$, $k_{i2} = a_6^{K_i} \ldots a_{10}^{K_i}$, $k_{i3} = a_{11}^{K_i} \ldots a_{15}^{K_i}$
(3) $t = a_1^{L_0} \ldots a_5^{L_0}$, $f = a_2^{L_0} \ldots a_6^{L_0}$, $l' = a_7^{L_0} \ldots a_{21}^{L_0}$

Having structures defined we construct four sets of local similarities corresponding to: assignment of values to variables (sets Φ^T and Φ^F), occurrences of variables in clauses (set Φ^K) and kinds of literals(positive vs. negative) occurring in clauses (set Φ^L):

(1) $\Phi^T = \left\{ \varphi_i^T : V_i \to L_0 \middle| 1 \le i \le k \wedge \varphi_i^T(v_i) = t \wedge \varphi_i^T(v_i') = l' \right\}$
(2) $\Phi^F = \left\{ \varphi_i^F : V_i \to L_0 \middle| 1 \le i \le k \wedge \varphi_i^F(v_i) = f \wedge \varphi_i^F(v_i') = l' \right\}$
(3) $\Phi^K = \left\{ \varphi_{ij}^K : K_i \to V_p \middle| \begin{array}{l} 1 \le i \le l \wedge 1 \le j \le 3 \wedge 1 \le p \le k \wedge \varphi_{ij}^K(k_{ij}) = v_p \wedge \\ \wedge\, u_p \text{ or } \neg u_p \text{occurs at } j\text{th position in clause } C_i \end{array} \right\}$
(4) $\Phi^L = \left\{ \varphi_{ij}^L : K_i \to L_0 \middle| \begin{array}{l} 1 \le i \le l \wedge 1 \le j \le 3 \wedge \\ \\ \wedge\, \varphi_{ij}^L(k_{ij}) = \begin{cases} t & j\text{th position in clause } C_i \text{ is a positive literal} \\ f & j\text{th position in clause } C_i \text{ is a negative literal} \end{cases} \end{array} \right\}$

Finally, a set of local similarities Φ is the sum of the following:

$$\Phi = (\Phi^T \cup \Phi^F \cup \Phi^K \cup \Phi^L) \cup (\Phi^T \cup \Phi^F \cup \Phi^K \cup \Phi^L)^{-\mathbf{1}}$$

where $\Phi^{-\mathbf{1}}$ denotes the set of alignments inverse to those in Φ[11].

An assignment of values which satisfies all clauses exists if and only if there exists an alignment of $\mathscr{S} = \{L_0, V_1, \ldots, V_k, K_0, \ldots, K_l\}$ with support in Φ of size $\frac{2(20k+10l)}{(k+l)(k+l+1)}$. Furthermore, if Φ' is a support of such an alignment, for every i either $\varphi_i^T \in \Phi'$ or $\varphi_i^F \in \Phi'$ and an assignment:

---

[9] Unless P=NP.

[10] In this proof we abstain from giving residue numbers in upper index.

[11] If $\varphi : A \to B$ in an alignment between structures A and B, an inverse alignment is a function $\varphi^{-1} : B \to A$ derived from the same mapping of residues.

$$u_i \to \begin{cases} 1 & \varphi_i^T \in \Phi' \\ 0 & \varphi_i^F \in \Phi' \end{cases}$$

satisfies all clauses.

Example 2. Let C be a set of clauses over $U = \{u_1, u_2, u_3\}$:

$$C = \{\{\neg u_1, u_2, u_3\}, \{u_1, \neg u_2, u_3\}, \{u_1, u_2, \neg u_3\}\}$$

corresponding to the formula:

$$(\neg u_1 \vee u_2 \vee u_3) \wedge (u_1 \vee \neg u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee \neg u_3)$$

The reduction presented above yields the following instance of **OSMA**:

$$L_0 = a_1^{L_0} \overbrace{a_2^{L_0} a_3^{L_0} a_4^{L_0} a_5^{L_0}}^{f} a_6^{L_0} \overbrace{a_7^{L_0} a_8^{L_0} a_9^{L_0} a_{10}^{L_0} a_{11}^{L_0} a_{12}^{L_0} a_{13}^{L_0} a_{14}^{L_0} a_{15}^{L_0} a_{16}^{L_0} a_{17}^{L_0} a_{18}^{L_0} a_{19}^{L_0} a_{20}^{L_0} a_{21}^{L_0}}^{l'}$$

$$V_1 = \overbrace{a_1^{V_1} a_2^{V_1} a_3^{V_1} a_4^{V_1} a_5^{V_1}}^{v_1} \overbrace{a_6^{V_1} a_7^{V_1} a_8^{V_1} a_9^{V_1} a_{10}^{V_1} a_{11}^{V_1} a_{12}^{V_1} a_{13}^{V_1} a_{14}^{V_1} a_{15}^{V_1} a_{16}^{V_1} a_{17}^{V_1} a_{18}^{V_1} a_{19}^{V_1} a_{20}^{V_1}}^{v_1'}$$

$$V_2 = \overbrace{a_1^{V_2} a_2^{V_2} a_3^{V_2} a_4^{V_2} a_5^{V_2}}^{v_2} \overbrace{a_6^{V_2} a_7^{V_2} a_8^{V_2} a_9^{V_2} a_{10}^{V_2} a_{11}^{V_2} a_{12}^{V_2} a_{13}^{V_2} a_{14}^{V_2} a_{15}^{V_2} a_{16}^{V_2} a_{17}^{V_2} a_{18}^{V_2} a_{19}^{V_2} a_{20}^{V_2}}^{v_2'}$$

$$V_3 = \overbrace{a_1^{V_3} a_2^{V_3} a_3^{V_3} a_4^{V_3} a_5^{V_3}}^{v_3} \overbrace{a_6^{V_3} a_7^{V_3} a_8^{V_3} a_9^{V_3} a_{10}^{V_3} a_{11}^{V_3} a_{12}^{V_3} a_{13}^{V_3} a_{14}^{V_3} a_{15}^{V_3} a_{16}^{V_3} a_{17}^{V_3} a_{18}^{V_3} a_{19}^{V_3} a_{20}^{V_3}}^{v_3'}$$

$$K_1 = \overbrace{a_1^{K_1} a_2^{K_1} a_3^{K_1} a_4^{K_1} a_5^{K_1}}^{k_{1,1}} \overbrace{a_6^{K_1} a_7^{K_1} a_8^{K_1} a_9^{K_1} a_{10}^{K_1}}^{k_{1,2}} \overbrace{a_{11}^{K_1} a_{12}^{K_1} a_{13}^{K_1} a_{14}^{K_1} a_{15}^{K_1}}^{k_{1,3}}$$

$$K_2 = \overbrace{a_1^{K_2} a_2^{K_2} a_3^{K_2} a_4^{K_2} a_5^{K_2}}^{k_{2,1}} \overbrace{a_6^{K_2} a_7^{K_2} a_8^{K_2} a_9^{K_2} a_{10}^{K_2}}^{k_{2,2}} \overbrace{a_{11}^{K_2} a_{12}^{K_2} a_{13}^{K_2} a_{14}^{K_2} a_{15}^{K_2}}^{k_{2,3}}$$

$$K_3 = \overbrace{a_1^{K_3} a_2^{K_3} a_3^{K_3} a_4^{K_3} a_5^{K_3}}^{k_{3,1}} \overbrace{a_6^{K_3} a_7^{K_3} a_8^{K_3} a_9^{K_3} a_{10}^{K_3}}^{k_{3,2}} \overbrace{a_{11}^{K_3} a_{12}^{K_3} a_{13}^{K_3} a_{14}^{K_3} a_{15}^{K_3}}^{k_{3,3}}$$

$$\varphi_1^T(e) = \begin{cases} t & e = v_1 \\ l' & e = v_1' \end{cases} \qquad \varphi_2^T(e) = \begin{cases} t & e = v_2 \\ l' & e = v_2' \end{cases} \qquad \varphi_3^T(e) = \begin{cases} t & e = v_3 \\ l' & e = v_3' \end{cases}$$

$$\varphi_1^F(e) = \begin{cases} f & e = v_1 \\ l' & e = v_1' \end{cases} \qquad \varphi_2^F(e) = \begin{cases} f & e = v_2 \\ l' & e = v_2' \end{cases} \qquad \varphi_3^F(e) = \begin{cases} f & e = v_3 \\ l' & e = v_3' \end{cases}$$

$$\varphi_{1,1}^K(e) = \begin{cases} v_1 & e = k_{1,1} \end{cases} \quad \varphi_{1,2}^K(e) = \begin{cases} v_2 & e = k_{1,2} \end{cases} \quad \varphi_{1,3}^K(e) = \begin{cases} v_3 & e = k_{1,3} \end{cases}$$

$$\varphi^K_{2,1}(e)=\begin{cases}v_1 & e=k_{2,1}\end{cases}\quad \varphi^K_{2,2}(e)=\begin{cases}v_2 & e=k_{2,2}\end{cases}\quad \varphi^K_{2,3}(e)=\begin{cases}v_3 & e=k_{2,3}\end{cases}$$
$$\varphi^K_{3,1}(e)=\begin{cases}v_1 & e=k_{3,1}\end{cases}\quad \varphi^K_{3,2}(e)=\begin{cases}v_2 & e=k_{3,2}\end{cases}\quad \varphi^K_{3,3}(e)=\begin{cases}v_3 & e=k_{3,3}\end{cases}$$
$$\varphi^L_{1,1}(e)=\begin{cases}f & e=k_{1,1}\end{cases}\quad \varphi^L_{1,2}(e)=\begin{cases}t & e=k_{1,2}\end{cases}\quad \varphi^L_{1,3}(e)=\begin{cases}t & e=k_{1,3}\end{cases}$$
$$\varphi^L_{2,1}(e)=\begin{cases}t & e=k_{2,1}\end{cases}\quad \varphi^L_{2,2}(e)=\begin{cases}f & e=k_{2,2}\end{cases}\quad \varphi^L_{2,3}(e)=\begin{cases}t & e=k_{2,3}\end{cases}$$
$$\varphi^L_{3,1}(e)=\begin{cases}t & e=k_{3,1}\end{cases}\quad \varphi^L_{3,2}(e)=\begin{cases}t & e=k_{3,2}\end{cases}\quad \varphi^L_{3,3}(e)=\begin{cases}f & e=k_{3,3}\end{cases}$$

The following assignment satisfies all clauses:

$$u_1\to 0, u_2\to 1, u_3\to 1$$

Therefore, there exists an alignment with support containing $\{\varphi^F_1,\varphi^T_2,\varphi^T_3\}$ and size $\frac{180}{21}$. The support of this alignment also contains $\left\{\varphi^K_{1,1},\varphi^K_{2,3},\varphi^K_{3,2},\varphi^L_{1,1},\varphi^L_{2,3},\varphi^L_{3,2}\right\}$, and an alignment is induced by the following (see also Fig. 6) ($\varphi(a)=\perp$ means, that $a\notin Dom(\varphi)$):

$$\xi_{V_1L_0}(V_1)=\overbrace{\underbrace{a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}a_6^{L_0}}_{f}\underbrace{a_7^{L_0}a_8^{L_0}a_9^{L_0}a_{10}^{L_0}a_{11}^{L_0}}_{l^1}\underbrace{a_{12}^{L_0}a_{13}^{L_0}a_{14}^{L_0}a_{15}^{L_0}a_{16}^{L_0}}_{l^2}\underbrace{a_{17}^{L_0}a_{18}^{L_0}a_{19}^{L_0}a_{20}^{L_0}a_{21}^{L_0}}_{l^3}}^{v_1}$$
$$\xi_{V_2L_0}(V_2)=\overbrace{\underbrace{a_1^{L_0}a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}}_{t}\underbrace{a_7^{L_0}a_8^{L_0}a_9^{L_0}a_{10}^{L_0}a_{11}^{L_0}}_{l^1}\underbrace{a_{12}^{L_0}a_{13}^{L_0}a_{14}^{L_0}a_{15}^{L_0}a_{16}^{L_0}}_{l^2}\underbrace{a_{17}^{L_0}a_{18}^{L_0}a_{19}^{L_0}a_{20}^{L_0}a_{21}^{L_0}}_{l^3}}^{v_2}$$
$$\xi_{V_3L_0}(V_3)=\overbrace{\underbrace{a_1^{L_0}a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}}_{t}\underbrace{a_7^{L_0}a_8^{L_0}a_9^{L_0}a_{10}^{L_0}a_{11}^{L_0}}_{l^1}\underbrace{a_{12}^{L_0}a_{13}^{L_0}a_{14}^{L_0}a_{15}^{L_0}a_{16}^{L_0}}_{l^2}\underbrace{a_{17}^{L_0}a_{18}^{L_0}a_{19}^{L_0}a_{20}^{L_0}a_{21}^{L_0}}_{l^3}}^{v_3}$$
$$\xi_{K_1V_1}(K_1)=\overbrace{\underbrace{a_1^{V_1}a_2^{V_1}a_3^{V_1}a_4^{V_1}a_5^{V_1}}_{v_1}}^{k_{1,1}}\overbrace{\perp\perp\perp\perp\perp}^{k_{1,2}}\overbrace{\perp\perp\perp\perp\perp}^{k_{1,3}}$$
$$\xi_{K_2V_3}(K_2)=\overbrace{\perp\perp\perp\perp\perp}^{k_{2,1}}\overbrace{\perp\perp\perp\perp\perp}^{k_{2,2}}\overbrace{\underbrace{a_1^{V_3}a_2^{V_3}a_3^{V_3}a_4^{V_3}a_5^{V_3}}_{v_3}}^{k_{2,3}}$$
$$\xi_{K_3V_2}(K_3)=\overbrace{\perp\perp\perp\perp\perp}^{k_{3,1}}\overbrace{\underbrace{a_1^{V_2}a_2^{V_2}a_3^{V_2}a_4^{V_2}a_5^{V_2}}_{v_2}}^{k_{3,2}}\overbrace{\perp\perp\perp\perp\perp}^{k_{3,3}}$$

$$\xi_{K_1L_0}(K_1) = \overbrace{\underbrace{a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}a_6^{L_0}}_{f}}^{k_{1,1}}\overbrace{\bot\bot\bot\bot\bot}^{k_{1,2}}\overbrace{\bot\bot\bot\bot\bot}^{k_{1,3}}$$

$$\xi_{K_2L_0}(K_2) = \overbrace{\bot\bot\bot\bot\bot}^{k_{2,1}}\overbrace{\bot\bot\bot\bot\bot}^{k_{2,2}}\overbrace{\underbrace{a_1^{L_0}a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}}_{t}}^{k_{2,3}}$$

$$\xi_{K_3L_0}(K_3) = \overbrace{\bot\bot\bot\bot\bot}^{k_{3,1}}\overbrace{\underbrace{a_1^{L_0}a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}}_{t}}^{k_{3,2}}\overbrace{\bot\bot\bot\bot\bot}^{k_{3,3}}$$

$$\begin{array}{ll}
L_0 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{L_0}a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}a_6^{L_0}\bot\ \bot\ \bot\ \bot\ \bot\ \ldots \\
V_1 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{V_1}a_2^{V_1}a_3^{V_1}a_4^{V_1}a_5^{V_1}\bot\ \bot\ \bot\ \bot\ \bot\ \ldots \\
V_2 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{V_2}a_2^{V_2}a_3^{V_2}a_4^{V_2}a_5^{V_2}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \ldots \\
V_3 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{V_3}a_2^{V_3}a_3^{V_3}a_4^{V_3}a_5^{V_3}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \ldots \\
K_1 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{K_1}a_2^{K_1}a_3^{K_1}a_4^{K_1}a_5^{K_1}a_6^{K_1}a_7^{K_1}a_8^{K_1}a_9^{K_1}a_{10}^{K_1}\ldots \\
K_2 & a_1^{K_2}a_2^{K_2}a_3^{K_2}a_4^{K_2}a_5^{K_2}a_6^{K_2}a_7^{K_2}a_8^{K_2}a_9^{K_2}a_{10}^{K_2}\bot\ \bot\ \bot\ \bot\ \bot\ a_{11}^{K_2}a_{12}^{K_2}a_{13}^{K_2}a_{14}^{K_2}a_{15}^{K_2}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \ldots \\
K_3 & \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_1^{K_3}a_2^{K_3}a_3^{K_3}a_4^{K_3}a_5^{K_3}a_6^{K_3}a_7^{K_3}a_8^{K_3}a_9^{K_3}a_{10}^{K_3}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \ldots
\end{array}$$

$$\begin{array}{ll}
L_0 & \ldots\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_7^{L_0}a_8^{L_0}a_9^{L_0}a_{10}^{L_0}a_{11}^{L_0}a_{12}^{L_0}a_{13}^{L_0}a_{14}^{L_0}a_{15}^{L_0}a_{16}^{L_0}a_{17}^{L_0}a_{18}^{L_0}a_{19}^{L_0}a_{20}^{L_0}a_{21}^{L_0} \\
V_1 & \ldots\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_6^{V_1}a_7^{V_1}a_8^{V_1}a_9^{V_1}a_{10}^{V_1}a_{11}^{V_1}a_{12}^{V_1}a_{13}^{V_1}a_{14}^{V_1}a_{15}^{V_1}a_{16}^{V_1}a_{17}^{V_1}a_{18}^{V_1}a_{19}^{V_1}a_{20}^{V_1} \\
V_2 & \ldots\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_6^{V_2}a_7^{V_2}a_8^{V_2}a_9^{V_2}a_{10}^{V_2}a_{11}^{V_2}a_{12}^{V_2}a_{13}^{V_2}a_{14}^{V_2}a_{15}^{V_2}a_{16}^{V_2}a_{17}^{V_2}a_{18}^{V_2}a_{19}^{V_2}a_{20}^{V_2} \\
V_3 & \ldots\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ a_6^{V_3}a_7^{V_3}a_8^{V_3}a_9^{V_3}a_{10}^{V_3}a_{11}^{V_3}a_{12}^{V_3}a_{13}^{V_3}a_{14}^{V_3}a_{15}^{V_3}a_{16}^{V_3}a_{17}^{V_3}a_{18}^{V_3}a_{19}^{V_3}a_{20}^{V_3} \\
K_1 & \ldots\ a_{11}^{K_1}a_{12}^{K_1}a_{13}^{K_1}a_{14}^{K_1}a_{15}^{K_1}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot \\
K_2 & \ldots\ \bot \\
K_3 & \ldots\ \bot\ \bot\ \bot\ \bot\ \bot\ a_{11}^{K_3}a_{12}^{K_3}a_{13}^{K_3}a_{14}^{K_3}a_{15}^{K_3}\bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot\ \bot
\end{array}$$

Fig. 6: Multiple alignment of structures corresponding to the solution of an instance of **3SAT** in example 2.

As can be seen in this example, an alignment corresponding to a **3SAT** solution contains an assignment of value to each variable (φ_i^T or φ_i^F for every i), a selection of a variable used to satisfy each clause (φ_{ij}^K for all i and $j = 1,2,3$), and selection of one of three literals used to satisfy a clause (φ_{ij}^L for all i and $j = 1,2,3$). One can easily see that for each clause only one of φ_{ij}^L can be used. Same applies to φ_{ij}^K if all variables have values assigned. Otherwise each of residues $a_2^{L_0}a_3^{L_0}a_4^{L_0}a_5^{L_0}$ would be aligned to more than one residue in K_i. If an alignment contains local similarities

encoding assignment of value for each variable and this assignment satisfies all clauses, it can easily be extended with respective local similarities encoding choices of variables and literals for each clause. Elements of Φ_L guarantee that a variable can be used to satisfy a clause only if it is assigned **1** and occurs in positive literal, or is assigned **0** and occurs in negative literal. Such an alignment has a required size.

It remains to be proven, that every alignment of size $\frac{2(20k+10l)}{(k+l)(k+l+1)}$ contains either φ_i^T or φ_i^F for each variable. We begin with an observation that local similarities related to different variables are independent in a way that they cannot cause contradiction in an alignment. This means that when searching for an optimal alignment one can independently deal with sets $\left\{\varphi_i^T, \varphi_i^F, \varphi_{ip}^K, \varphi_{iq}^K, \varphi_{ir}^K, \varphi_{ps}^L, \varphi_{qt}^L, \varphi_{ru}^L\right\}$, where p,q,r are the numbers of clauses containing variable i, and s,t,u are positions of variable i in these clauses. If an alignment does not contain neither φ_i^T nor φ_i^F it may contain all of $\varphi_{ip}^K, \varphi_{iq}^K, \varphi_{ir}^K$ (Only one similarity from Φ_L may be picked for each clause.). Similarities $\varphi_{ip}^K, \varphi_{iq}^K, \varphi_{ir}^K$ contribute $\frac{30}{(k+l)(k+l+1)}$ to the alignment size, while each of φ_i^T, φ_i^F contributes $\frac{40}{(k+l)(k+l+1)}$. Therefore any alignment which does not contain assignments of value for all variables is suboptimal. This concludes the proof and explains the introduction of seemingly unnecessary segments v_i' and l'. The remaining technical details have been left to the reader.

With the theorem above we have established, that (provided P is not equal to NP) all algorithms performing multiple alignment of structures have exponential computational complexity with respect to the number of structures. This is due to the fact that although every multiple alignment can be described with a set of pairwise alignments, not every set of pairwise alignments induces a proper multiple alignment. Conflicts that prevent inducing a multiple alignment may involve alignments of any number of structures and thus cannot be efficiently resolved by performing clean-ups on subsets of structures. The last theorem in this section formalizes this observation. Let us assume that $\mathscr{S}$ is divided into two subsets, $\mathscr{S}_1$ and $\mathscr{S}_2$, and we have already calculated optimal multi-alignments of these sets. We will consider computing a multiple alignment of $\mathscr{S}$ which contains alignments of $\mathscr{S}_1$ and $\mathscr{S}_2$.

Definition 11. For a given set of structures $\mathscr{S}$, multiple alignments of its disjoint subsets $\mathscr{S}_1$ and $\mathscr{S}_2$ ($\mathscr{S}_1 \cup \mathscr{S}_2 = \mathscr{S}$), set Φ and number k, the *Optimal Alignment of Multiple Aligments problem* (**OAMA**) is to determine whether there exists a multiple alignment in $\mathscr{S}$ with support in Φ of size no less than k containing given alignment of $\mathscr{S}_1$ and $\mathscr{S}_2$.

OAMA is a natural simplification of **OSMA**. Very often problems requiring computations on a given set are solved with a "divide and conquer" paradigm in which data is divided into two or more parts, which are solved independently and their solutions are merged to achieve a solution for the whole dataset. This step is repeated recursively. **OAMA** would occur in the merging stage of such an approach.

Theorem 8. ***OAMA*** *is NP-complete.*

This theorem does not require a proof nor a comment, since **OAMA** contains **OSA**.

Theorem 9. ***OAMA*** *is NP-complete even if its input is extended with optimal alignments of all pairs of structures from* $\mathscr{S}_1 \times \mathscr{S}_2$.

Due to lack of space we will leave this theorem without proof, since it is similar to the proof of Theorem 7.

In this section we have shown that computing optimal multiple alignments adds one more level of intractability to an already difficult problem of computing optimal alignments of two structures. Conflicts that may occur between alignments of pairs prevent them from being merged into a multiple alignment. Resolving these conflicts is intractable by itself.

5 Practical aspects of computing structural alignments

In this section we describe certain issues which may arise in comparison of protein structures. The content of this section pertains mostly to application of local descriptors to computing alignments, but may be treated as a collection of tips and tricks to be used elsewhere. We will begin with a fundamental problem of setting correct thresholds.

5.1 Dealing with uncertainty

Setting thresholds is a common practice in all kinds of parameter dependent models and theories. Threshold values are usually carefully estimated or derived from theoretical principles. Nevertheless, whenever we are dealing with physical objects, which are always measured with some uncertainty, values are rounded, and cannot be treated as indisputable truth. Therefore, it is in principle impossible to set a threshold which could be used to classify data into discrete categories. Whenever a value is close to a threshold, classification cannot be trusted. This issue is usually dismissed, because a mistake in classifying of values close to the threshold is rather insignificant, once we realize that all thresholds are somewhat arbitrary.

However, the situation changes whenever a significant result is dependent on an apparently insignificant difference. This does happen when two local descriptors are compared. As described in section 2.2.2, there are two thresholds to be satisfied – alignment quality (RMSD) and alignment size (number of aligned residues, contacts, and segments). RMSD does not cause any doubts, since it is a real number and any uncertainty may be blamed on inaccuracies of coordinates in experimentally obtained protein structures. Alignment size, on the other hand, is discrete. It may happen that descriptors are considered dissimilar because one of them lacks an element the other one may have. This would depend on the thresholds used to define contacts. It is quite probable that dissimilar descriptors could be made similar by adding a contact to one of them which just slightly exceeds the threshold.

This issue can be solved using a formalism of rough sets [38]. The idea is to introduce a third logical value – "maybe". If distances between residues fall within contact thresholds by a very little margin, such contact is flagged as optional. When descriptors are aligned, optional contacts which have their counterparts in the other descriptor are treated normally, while non-aligned optional contacts are disregarded (i.e. not counted in computation of the descriptor size). This is equivalent to giving such contacts the benefit of the doubt – they might have a counterpart in the other protein with a distance slightly too large to qualify as a contact.

It should be noted that equivalent results cannot be achieved by adjusting thresholds. No matter how conservative or liberal the definition of a contact is, pairs of residues with distances near to the threshold will always exist and descriptors containing them will be susceptible to the described issue.

5.2 Similarity measure

5.2.1 Segment swaps

In chapter 3 we considered methods of computing alignments with or without segment swaps. However, we did not focus on the nature of segment swaps. Instead we assumed that any mapping (with a support in a given set) is a valid alignment. In some cases it may be useful to restrict the number of swaps to a certain value (e.g. so-called circular permutations contain exactly one swap).

We say that a pair of aligned residues $\left\langle a^{(k)}, b^{(m)} \right\rangle$ is a *swap site*, if residues $a^{(l)}$ and $b^{(n)}$, which are the lowest numbered residues in the alignment following $a^{(k)}$ and $b^{(m)}$ in respective structures, are not aligned together (see Fig. 7).

It is easy to compute the number of swap sites in a given alignment. If it exceeds the preset limit it is possible to find the largest subalignment with a desired number of swaps.

5.2.2 Alignment quality

Usually, finding the best structural alignment is a problem of optimization of two variables – alignment size, and its quality. Root Mean Square distance (RMSD) [24, 25] is the most popular measure due to its simplicity – it has a compact mathematical solution. Other methods (e.g. MaxSub[47]) exist, but did not manage to achieve popularity. All these methods, however, rely on superimposing aligned residues and assessing the quality of the superposition bases on distances between aligned residues. Structures are therefore treated as rigid objects. Nevertheless, it is commonly accepted that proteins are flexible to some extent. In section 1 we have suggested that methods of aligning protein structures should take such flexibility into account, and thus quality measures overcoming the rigid-body limitation

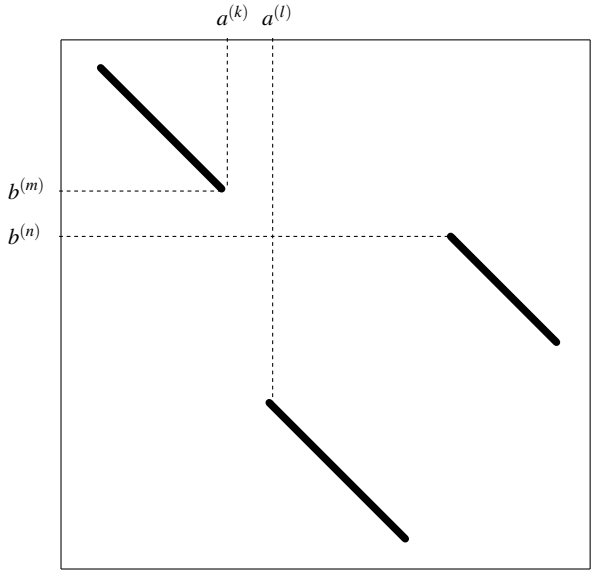


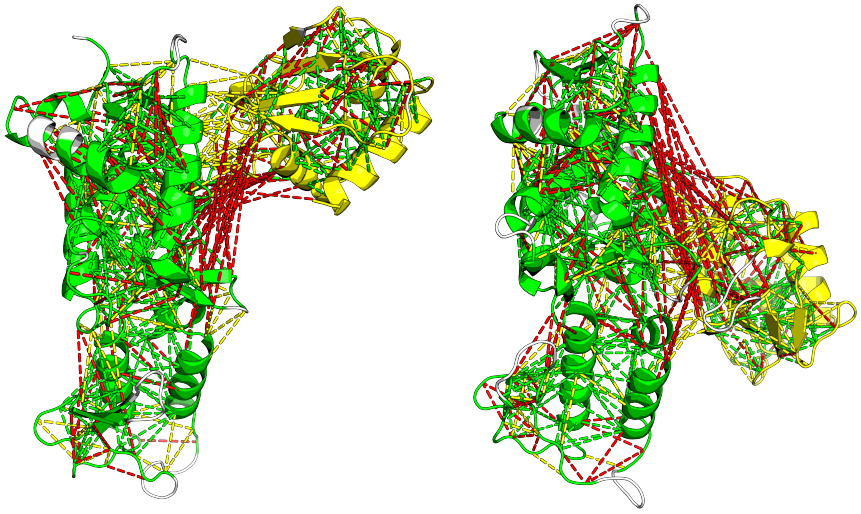
Fig. 7: Example of a swap site. Axes correspond to residue sequences. Aligned segments are plotted with diagonal lines.

should be used. The simplest way is to introduce explicit "hinges" which connect rigid fragments [52].

In this section we will present a solution used in the DEDAL method, which was designed to allow for flexible rearrangements of loosely coupled substructures (e.g. domains) and small local distortions, while penalizing deformations significantly changing the arrangement of interactions which stabilize structures. As in the case of local descriptors, where inter-residue contacts are used to define the structural neighborhood of a chosen residue, contacts may be used to detect a network of interactions responsible for the rigidity of a protein structure. One may imagine that residues in contact are connected with springs, which have to be somewhat extended or compressed, if these residues were to be superimposed onto their counterparts in the other structure. Degree of deformations of such springs may be treated as an indicator of the structural similarity.

Definition 12. An *aligned contact* is a pair $\left\langle \left\langle a^{(k)}, b^{(m)} \right\rangle, \left\langle a^{(l)}, b^{(n)} \right\rangle \right\rangle$, such that $a^{(k)}$ is aligned to $b^{(m)}$, and $a^{(l)}$ is aligned to $b^{(n)}$ and at least one of $\left\langle a^{(k)}, a^{(l)} \right\rangle$, $\left\langle b^{(m)}, b^{(n)} \right\rangle$ are in contact. We call an aligned contact *proper* if it exists in both structures.

A distortion of a single aligned contact is called *local tension* and is expressed as an RMS distance between descriptor elements:



(a) (b)

Fig. 8: Similar structures (ASTRAL domains (a) `d1d5fa_` i (b) `d1nd7a_`) comprise two differently arranged subdomains. Properly aligned contacts are marked with green lines. Yellow lines denote aligned contacts which are not preserved in the other structure. Red lines mark residue pairs not in contact, which are aligned with residues in contact. In order to superimpose these structures it is necessary to extend springs corresponding to yellow contacts to lengths of respective red lines.

$$tens(\left\langle \left\langle a^{(k)}, b^{(m)} \right\rangle, \left\langle a^{(l)}, b^{(n)} \right\rangle \right\rangle) = RMSD((El(a^{(k)}) \cup El(a^{(l)})), (El(b^{(m)}) \cup El(b^{(n)})))$$

A *tension* of the alignment ξ is a square mean of its local tensions computed for each residue, and then for the whole structure:

$$tens(\xi) = \sqrt{\sum_{a^{(i)} \in Dom(\xi)} \frac{\sum_{a^{(j)} \in T_{a^{(i)}}} \frac{\left[tens(\left\langle \left\langle a^{(i)}, \xi(a^{(i)}) \right\rangle, \left\langle a^{(j)}, \xi(a^{(j)}) \right\rangle \right\rangle) \right]^2}{|T_{a^{(j)}}|}}{|Dom(\xi)|}}$$

where $a^{(j)} \in T_{a^{(i)}}$, if $\left\langle \left\langle a^{(i)}, \xi(a^{(i)}) \right\rangle, \left\langle a^{(j)}, \xi(a^{(j)}) \right\rangle \right\rangle$ is an aligned contact in ξ.

5.3 Case studies

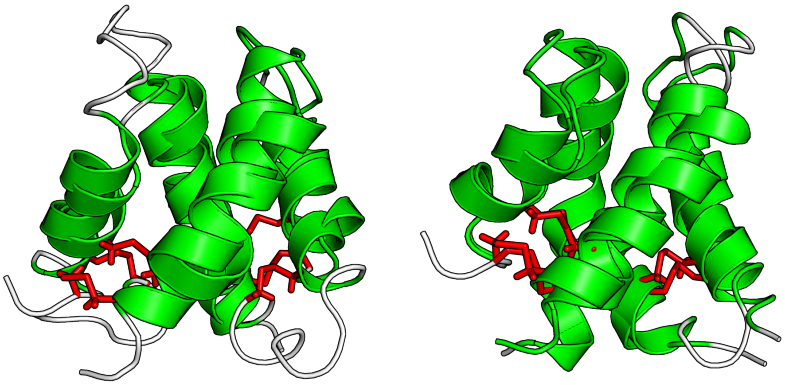
To illustrate the issues arising in computing alignments of protein structures we present three cases of difficult structure alignments not handled effectively by methods limited by the rigid-body or sequence-dependency constraints.

Saposins

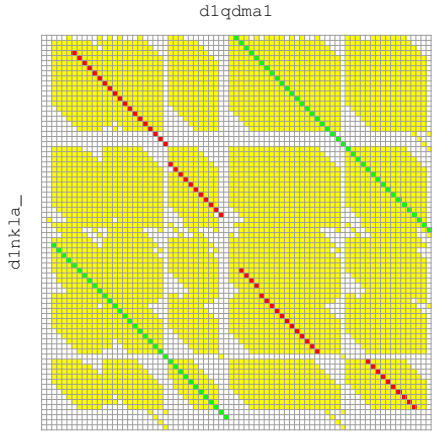
Similarity between saposin and saposin-like "swaposin" domains is one of the first circular permutations discovered. It was first indicated by sequence analysis[40], and verified when the crystal structures became available. NK-lysin (SCOP domain `d1nkla_`) is composed of five α-helices arranged in the "folded leaf" architecture[29]. The "swaposin" domain (`d1qdma1`) of aspartic proteinase prophytepsin has the same architecture, but the helices are in a different order[27] (Fig. 9). Nevertheless, despite the obvious similarity most of the structure comparison methods align the helices in agreement with their order along the sequence, which results in a visually poor superposition (Fig. 9a). The similarity of continuous segments commonly used does not provide enough information concerning the arrangement of helices and at the same time supports an alignment without swaps (Fig. 9c). It should be emphasized that, apart from the worse RMS distance, alignments without swaps incorrectly match cysteine residues forming the disulfide bonds. FlexSnap[42] and DEDAL[10] identify the similarity correctly (Fig. 9b).

GTPases

Guanine nucleotide-binding proteins (G proteins) are important cellular regulators. They act as *binary switches*, and use the GTP-GDP-GTP cycle to flip between the *on* and *off* states. GTPase domains they contain are responsible for the GTP/GDP binding. The GTPase activity depends on the set of five conserved sequence motifs[36]. An alternative circularly permuted GTPase structure (cpGTPase)[45] which contains all five motifs in a different order also exists(Fig. 10a and 10b). Despite a different topology the cpGTPase domains retain the GTP binding activity, and have the same architecture as GTPases. Although the crucial motifs are highly conserved and identifiable by sequence analysis[3], many structure comparison methods are unable to correctly align residues which form the GTP/GDP binding site. CE[46] and DALI[21] yield 36% accuracy, while FlexSnap[42] and C_α-match[4] have 90% accuracy (reference alignment contains residues responsible for GTP binding). In contrast, DEDAL[10] yields an entirely accurate superposition in this region (Fig. 10c and 10d).



(a) (b)



(c)

Fig. 9: The Saposin domain of NK-lysin (SCOP domain `d1nkla_`) and the "swaposin" domain of prophytepsin (`d1qdma1`). Despite differing topologies these two domains have the same architecture and identical disulfide bonds. (a) Methods incapable of handling segment swaps wrongly align cysteine residues (figure shows alignment computed by DALI). (b) DEDAL correctly identifies the best superposition and the disulfide bond network. (c) Alignments shown in (a) (red) and (b) (green) are plotted against local similarity of single segments of length 5 (yellow). It can be observed that similarity of continuous segments is insufficient to discover the correct alignment.

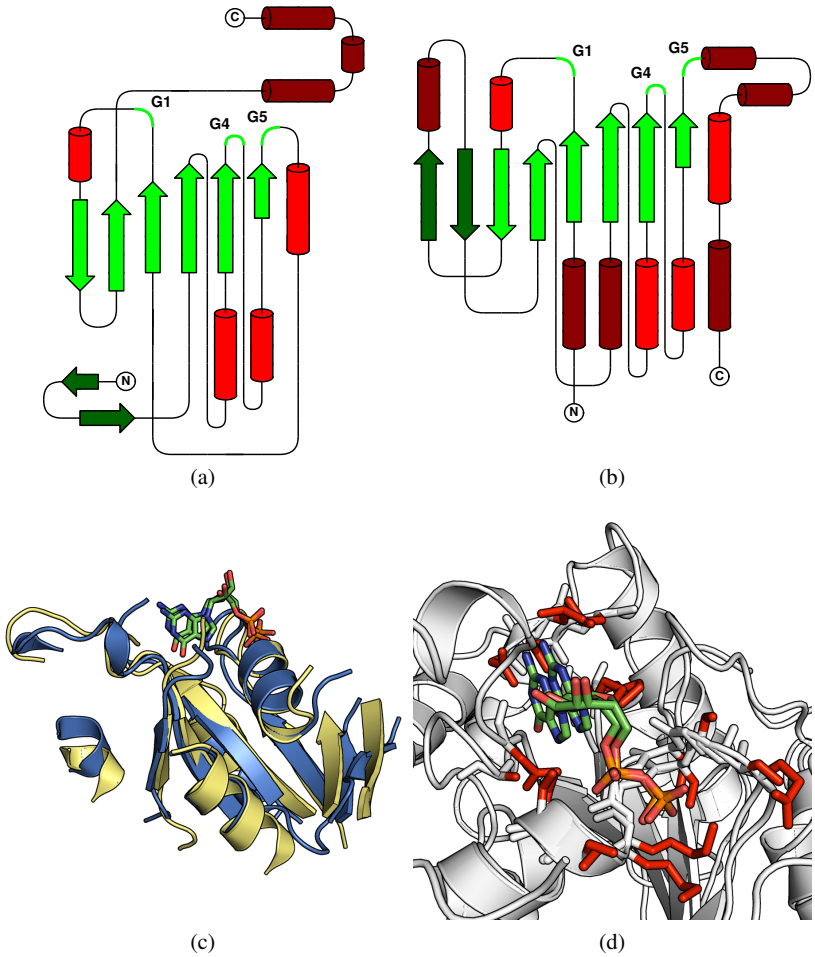


Fig. 10: Topologies of (a) the Dynamin A GTPase (SCOP domain `d1jwyb_`) and (b) cpGTPase domain from the YjeQ protein (`d1u0la2`). Aligned SSEs are indicated by lighter colors. (c) DEDAL superposition of the GTPase and the cpGTPase domains (yellow and blue, respectively). For clarity, only the aligned parts of the structures are shown. (d) View of the binding site in the same superposition showing residues participating in the GDP/GTP binding (red) and the GDP molecule. Despite significant topological differences, DEDAL effectively handles all alignable SSEs and correctly superimposes the active sites. The sequence identity of the superimposed regions is 24.2%.

Cyanovirin-N

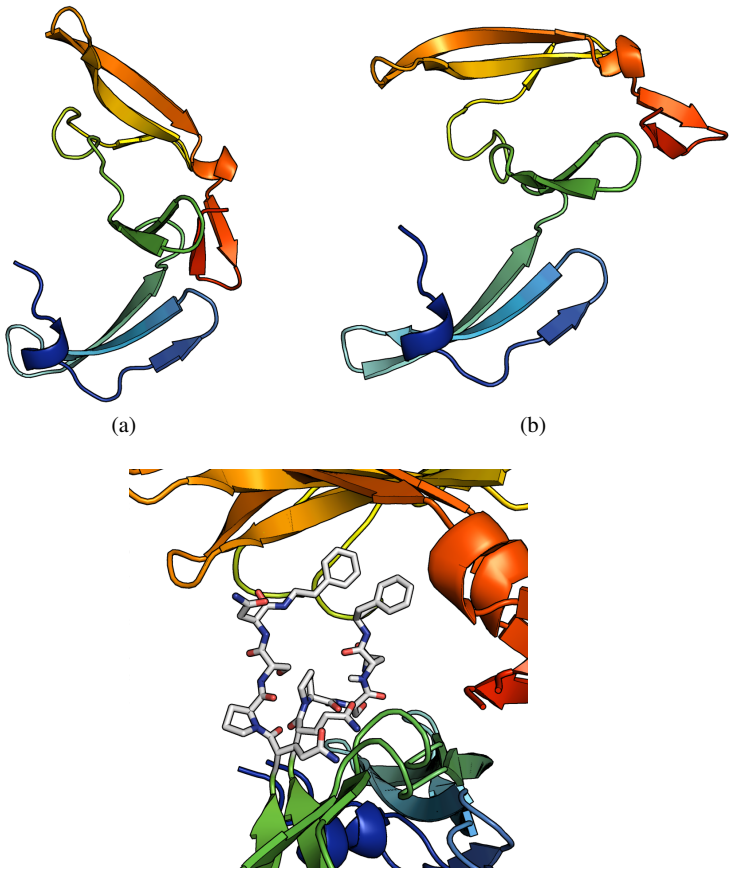


Fig. 11: Conformation of the Cyanovirin-N dimeric form depends on the molecular environment. (a) X-ray (`d1l5ba_`) and (b) NMR (`d1l5ea_`) structures have different conformations of the "hinge" region (PRO51-ASN53) (c). To fully analyze the similarity of the two structures it is necessary to abandon the rigid-body approach. The regions on both sides of the "hinge" have to be superimposed separately. DEDAL accomplishes this by extending local similarities in both regions and effectively defining the "hinge" as the boundary between them.

Cyanovirin-N is a potent HIV-inactivating protein, which exists in both monomeric and domain-swapped dimeric forms. Although the monomeric form is predominant in solution, and was determined first[7], the metastable dimeric form is also present.

The dimeric form is stabilized in the crystalline state[51] and eventually its structure was also obtained by NMR[5]. For the dimeric form, it can be observed that the X-ray (SCOP domain `d1l5ba_`) and NMR (`d1l5ea_`) structures exhibit a slightly different arrangement of subdomains (Fig. 11a and 11b), and that the local conformations of all residues except for the hinge region (PRO51-ASN53, Fig. 11c) are identical. Nevertheless, the similarity between the two structures cannot be easily determined by the rigid-body techniques, which are capable of aligning only one subdomain. Surprisingly FlexSnap[42], although in principle capable of handling conformational variability, gives only 50% accuracy with the reference alignment.

6 Conclusions

Computing protein structure alignments is one of the most commonly performed tasks in computational structural biology. Most knowledge about proteins and their functions which is not gathered experimentally is inferred from properties of sequentially or structurally analogous proteins. Therefore, despite the maturity of this field, similarity measures and efficient methods of computing alignments are subjects of ongoing research.

In this study we have concentrated on structure alignment methods based on so-called local fragments. This is probably the most prominent approach. Being based on local similarities, it also has the capability to build alignments containing segment swaps and deal with spatial distortions. These features are crucial in detecting "difficult" similarities, some of which were presented.

One of the reasons for the lack of a "golden standard" structure alignment method is the fact that under most circumstances the problem of computing the optimal alignment is intractable. Therefore, all computationally feasible methods have either to be heuristic or be based on a simplified similarity measure. In particular, we have shown that the problem of computing an optimal alignment based on similarities comprising three disjoint segments is NP-complete even if segment swaps are forbidden. If segment swaps are allowed, the problem is also intractable for two segmented similarities. We have described algorithms that could be used for computing such alignments, and presented a particularly effective formalism of local similarities – Local Descriptors of Protein Structure. This method has been implemented, tested, and is publicly available (DEDAL[10]).

Computing multiple alignments is even harder. It is NP-complete since it contains an intractable problem of pairwise alignment, but just like in the case of sequence alignments, even if computing pairwise alignments was effectively solved, computing an optimal multiple alignment would require time exponential with respect to the number of structures.

Nevertheless, despite all theoretical difficulties there exists a plethora of successful methods. Most of them overcome inherent intractability by disallowing segment swaps, using a rigid-body quality measure, and using simple local similarities (e.g. single segments). One of the reasons for their success is that reference databases

of structural alignments which could be used to benchmark them have not yet been developed. At the same time, they generally performed well enough to allow for a proper similarity analysis by a human expert. Traditionally, alignment methods were ranked by the number of residues they aligned under certain quality thresholds. Although this seems to be a valid and fully objective method, it by no means takes into account the biological meaning of the computed similarities. This is akin to the concern frequently raised in the protein structure prediction community, that models with correct prediction of an overall topology (fold) may be less relevant than models correctly predicting conformation of an active site.

In recent years, studies evaluating alignment algorithms on human curated biologically significant alignments have emerged[32, 6] giving a basis for more relevant benchmarks. DEDAL[10] – a method based on the principles presented in this review and employing local descriptors – outperforms established methods despite its simplicity, especially when tested on the most difficult cases. DAMA[9] – its prototype extension carrying out multiple structural alignments, has already been announced.

One of the underexplored directions of research (in authors' opinion) are heuristics based on relaxation. Computing an alignment is by its nature a discrete combinatorial problem. Nevertheless, there are successful applications of techniques converting a discrete problem into an easier continuous one with the aim of obtaining an approximate solution. This technique might be of use in efficiently computing multiple alignments (private communications, unpublished work).

As the number of known protein structures and high quality models increases, computing biologically relevant alignments is becoming a serious option in the area traditionally reserved to genome-wide sequence searches. It is generally accepted that a sequence of residues implies a spatial structure, which in turn determines atomic functional motions and other properties of a molecule. Therefore conclusions inferred from the structure comparison are in general more reliable than ones based on sequence alignments.

One should note that although causal relations between sequence, structure, atomic motions and function are often discussed in biological literature, until now such relations do not have any formal, consistent mathematical framework. Nevertheless, during the past few years, based on methodologies developed for complex systems in economy and neurophysiology, a prototype of causal analysis for biomolecular systems has been proposed[12]. In particular, applying the presented methodology to trajectories obtained from molecular dynamic simulations can help to elucidate the actual logic of its functioning. The development of such formalism for causal relations is one of the challenging tasks in structural biology and bioinformatics.

[12] For an overview and references related to this topic see[11].

Acknowledgements

This study was supported by the Biocentrum-Ochota Project (POIG.02.03.00-00-003/09), the research grant (DEC-2011/03/D/NZ2/02004) of the National Science Centre, and partially by BST/BF funds of the University of Warsaw. Figures 1, 10 and 11 are reproduced from an earlier study by the same authors[10].

References

[1] Alexandrov N (1996) SARFing the PDB. Protein Engineering 9(9):727

[2] Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25(17):3389–402

[3] Anand B, Verma SK, Prakash B (2006) Structural stabilization of GTP-binding domains in circularly permuted GTPases: implications for RNA binding. Nucleic Acids Res 34(8):2196–205

[4] Bachar O, Fischer D, Nussinov R, Wolfson H (1993) A computer vision based technique for 3-D sequence-independent structural comparison of proteins. Protein Eng 6(3):279–88

[5] Barrientos LG, Louis JM, Botos I, Mori T, Han Z, O'Keefe BR, Boyd MR, Wlodawer A, Gronenborn AM (2002) The domain-swapped dimer of cyanovirin-N is in a metastable folded state: reconciliation of X-ray and NMR structures. Structure 10(5):673–86

[6] Berbalk C, Schwaiger CS, Lackner P (2009) Accuracy analysis of multiple structure alignments. Protein Sci 18(10):2027–35

[7] Bewley CA, Gustafson KR, Boyd MR, Covell DG, Bax A, Clore GM, Gronenborn AM (1998) Solution structure of cyanovirin-N, a potent HIV-inactivating protein. Nat Struct Biol 5(7):571–8

[8] Bystroff C, Baker D (1998) Prediction of local structure in proteins using a library of sequence-structure motifs. J Mol Biol 281(3):565–77, DOI 10.1006/jmbi.1998.1943

[9] Daniluk P, Lesyng B (2011) DAMA: a novel method for aligning multiple protein structures. In: Multi-Pole Approach to Structural Biology Conference, Warsaw, Poland

[10] Daniluk P, Lesyng B (2011) A novel method to compare protein structures using local descriptors. BMC Bioinformatics 12(1):344, DOI 10.1186/1471-2105-12-344

[11] Daniluk P, Dziubiński M, Hallay-Suszek M, Rakowski F, Walewski L, Lesyng B (2012) From experimental structural probability distributions to the theoretical causality analysis of molecular changes. CAMES (In press)

[12] Dobbins S, Lesk V, Sternberg M (2008) Insights into protein flexibility: The relationship between normal modes and conformational change upon

protein–protein docking. Proceedings of the National Academy of Sciences 105(30):10,390

[13] Dror O, Benyamini H, Nussinov R, Wolfson H (2003) MASS: multiple structural alignment by secondary structures. Bioinformatics 19 Suppl 1:i95–104

[14] Elias I (2006) Settling the intractability of multiple alignment. J Comput Biol 13(7):1323–39, DOI 10.1089/cmb.2006.13.1323

[15] Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. A Series of books in the mathematical sciences, W. H. Freeman, San Francisco

[16] Gerstein M, Echols N (2004) Exploring the range of protein flexibility, from a structural proteomics perspective. Current opinion in chemical biology 8(1):14–19

[17] Gibrat JF, Madej T, Bryant SH (1996) Surprising similarities in structure comparison. Curr Opin Struct Biol 6(3):377–85

[18] Grishin NV (2001) Fold change in evolution of protein structures. J Struct Biol 134(2-3):167–85

[19] Guerler A, Knapp EW (2008) Novel protein folds and their nonsequential structural analogs. Protein Sci 17(8):1374–82

[20] Holm L, Park J (2000) DaliLite workbench for protein structure comparison. Bioinformatics 16(6):566–7

[21] Holm L, Sander C (1993) Protein structure comparison by alignment of distance matrices. J Mol Biol 233(1):123–38

[22] Ilyin VA, Abyzov A, Leslin CM (2004) Structural alignment of proteins by a novel TOPOFIT method, as a superimposition of common volumes at a topomax point. Protein Sci 13(7):1865–74

[23] Jung J, Lee B (2000) Protein structure alignment using environmental profiles. Protein Eng 13(8):535–43

[24] Kabsch W (1976) A solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A 32(5):922–923

[25] Kabsch W (1978) A discussion of the solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A 34(5):827–828

[26] Kawabata T, Nishikawa K (2000) Protein structure comparison using the markov transition model of evolution. Proteins 41(1):108–22

[27] Kervinen J, Tobin GJ, Costa J, Waugh DS, Wlodawer A, Zdanov A (1999) Crystal structure of plant aspartic proteinase prophytepsin: inactivation and vacuolar targeting. EMBO J 18(14):3947–55

[28] Konagurthu AS, Whisstock JC, Stuckey PJ, Lesk AM (2006) MUSTANG: a multiple structural alignment algorithm. Proteins 64(3):559–74, DOI 10.1002/prot.20921

[29] Liepinsh E, Andersson M, Ruysschaert JM, Otting G (1997) Saposin fold revealed by the NMR structure of NK-lysin. Nat Struct Biol 4(10):793–5

[30] Lindqvist Y, Schneider G (1997) Circular permutations of natural protein sequences: structural evidence. Curr Opin Struct Biol 7(3):422–7

[31] Mavridis L, Ritchie D (2010) 3D-blast: 3D protein structure alignment, comparison, and classification using spherical polar fourier correlations. In: Pacific Symposium on Biocomputing, vol 2010, pp 281–292
[32] Mayr G, Domingues FS, Lackner P (2007) Comparative analysis of protein structure alignments. BMC Struct Biol 7:50
[33] Menke M, Berger B, Cowen L (2008) Matt: local flexibility aids protein multiple structure alignment. PLoS Comput Biol 4(1):e10, DOI 10.1371/journal.pcbi.0040010
[34] Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21(6):1087
[35] Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–53
[36] Niemann HH, Knetsch ML, Scherer A, Manstein DJ, Kull FJ (2001) Crystal structure of a dynamin GTPase domain in both nucleotide-free and GDP-bound forms. EMBO J 20(21):5813–21
[37] Orengo CA, Taylor WR (1996) SSAP: sequential structure alignment program for protein structure comparison. Methods Enzymol 266:617–35
[38] Pawlak Z (1991) Rough Sets: Theoretical Aspects of Reasoning About Data. Theory and decision library: System theory, knowledge engineering, and problem solving, Kluwer Academic Publishers
[39] Pearson W, Lipman D (1988) Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences 85(8):2444
[40] Ponting CP, Russell RB (1995) Swaposins: circular permutations within genes encoding saposin homologues. Trends Biochem Sci 20(5):179–80
[41] Rocha J, Segura J, Wilson RC, Dasgupta S (2009) Flexible structural protein alignment by a sequence of local transformations. Bioinformatics 25(13):1625–31
[42] Salem S, Zaki M, Bystroff C (2010) FlexSnap: Flexible Non-sequential Protein Structure Alignment. Algorithms for Molecular Biology 5(1):12
[43] Shatsky M, Nussinov R, Wolfson HJ (2004) FlexProt: alignment of flexible protein structures without a predefinition of hinge regions. J Comput Biol 11(1):83–106
[44] Shatsky M, Nussinov R, Wolfson HJ (2004) A method for simultaneous alignment of multiple protein structures. Proteins 56(1):143–56, DOI 10.1002/prot.10628
[45] Shin DH, Lou Y, Jancarik J, Yokota H, Kim R, Kim SH (2004) Crystal structure of YjeQ from Thermotoga maritima contains a circularly permuted GTPase domain. Proc Natl Acad Sci U S A 101(36):13,198–203
[46] Shindyalov IN, Bourne PE (1998) Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. Protein Eng 11(9):739–47
[47] Siew N, Elofsson A, Rychlewski L, Fischer D (2000) MaxSub: an automated measure for the assessment of protein structure prediction quality. Bioinformatics 16(9):776–785

[48] Swendsen RH, Wang JS (1986) Replica Monte Carlo simulation of spin glasses. Phys Rev Lett 57(21):2607–2609
[49] Vogel C, Morea V (2006) Duplication, divergence and formation of novel protein topologies. Bioessays 28(10):973–8, DOI 10.1002/bies.20474
[50] Wohlers I, Domingues FS, Klau GW (2010) Towards optimal alignment of protein structure distance matrices. Bioinformatics 26(18):2273–80
[51] Yang F, Bewley CA, Louis JM, Gustafson KR, Boyd MR, Gronenborn AM, Clore GM, Wlodawer A (1999) Crystal structure of cyanovirin-N, a potent HIV-inactivating protein, shows unexpected domain swapping. J Mol Biol 288(3):403–12
[52] Ye Y, Godzik A (2003) Flexible structure alignment by chaining aligned fragment pairs allowing twists. Bioinformatics 19 Suppl 2:ii246–55
[53] Ye Y, Godzik A (2005) Multiple flexible structure alignment using partial order graphs. Bioinformatics 21(10):2362–9, DOI 10.1093/bioinformatics/bti353