Efficient Similarity Search in Protein Structure Databases: Improving Clique-Detection through Clique Hashing

Nils Weskamp[*], Daniel Kuhn[#], Eyke Hüllermeier[*§] and Gerhard Klebe[#]
[*]Department of Mathematics and Computer Science
[#]Institute of Pharmaceutical Chemistry
University of Marburg
35032 Marburg, Germany

[§]Corresponding author: eyke@informatik.uni-marburg.de

In order to make the structural comparison of protein binding sites more efficient, we propose a two-step method that combines advantages from both graph-based clique-detection and geometric hashing. The search for protein similarity is completely independent of sequence and fold information. Instead, it is based on a recent approach for the automatic extraction of binding sites from protein structures and the representation of their geometric and physicochemical properties. We also present some empirical results for similarity search in a medium-sized dataset.

Keywords: protein structure comparison, binding pockets, clique-detection, geometric hashing, index structures, drug design, convergent evolution

1 Introduction

Most ambitious among the competing techniques for the structural comparison of proteins are those which operate completely independent of sequence and fold information. These techniques are capable to reveal functional relations among proteins that are not due to phylogenetic dependency.

Unfortunately, such approaches are computationally very complex and require a reduction of the problem size, which is usually achieved by considering only the secondary structure elements of the protein. Our approach is instead based on a recent method for the automatic extraction and description of protein binding sites [Schmitt et al., 2002]. Following the idea that the function of a protein is determined by the shape and the physicochemical properties of the binding pocket, we assign pseudocenters – as 3D-descriptors – to the cavity-flanking amino acids to represent their interaction properties. Two binding sites are regarded as similar, if they share a common spatial arrangement of assigned pseudocenters and expose similar physicochemical properties into the binding site. The binding site-composing amino acids, the attributed pseudocenters and a cavity surface are stored in the database Cavbase [Schmitt et al., 2002].

The original approach by Schmitt et al. uses a clique-detection method that interprets the pseudocenters as nodes of a graph. Typically, this results in graphs of the size 50-150 (yet graphs with more than 1000 nodes do exist). The physicochemical properties of the centers are modeled as a coloring of the nodes. An edge is inserted between two nodes, weighted with the geometric distance between the adjacent nodes. To reduce the complexity, an edge is inserted if its length does not exceed 12.0 Å. A standard algorithm [Bron-Kerbosch, 1973] is used to detect connected maximal common substructures in the graph representations, applying the modifications suggested in the study of [Koch et al., 1996]. The size of the common substructures is our measure of similarity. Additionally, the common substructures may be used to calculate a geometric superimposition of the structures [Kabsch, 1976] and to perform more complex similarity measurements based on mutual surface-surface matches of common cavity patches. Although both of these measures formally do not fulfill the properties of a metric, they were subjected to empirical validation and usually revealed good results. The implications for clustering are not in the scope of this paper and will be discussed elsewhere.

Due to the comparatively fine-grained and therefore large graph representations, the runtime complexity of the method is relatively high. To allow efficient similarity searches in large datasets and all-against-all comparisons as a basis for clustering, we developed an improved hybrid method for the detection of common substructures in binding sites. This method combines advantages from both, graph-based clique-detection and geometric hashing, as will be detailed in Section 3. Before, Section 2 recalls the standard clique-detection approach and points out some of its drawbacks. Finally, we shall present some empirical results in Section 4.

2 Standard clique-detection techniques

Clique-detection is a well-known strategy for the structural comparison of proteins or drug-sized organic molecules. Given two labeled input graphs ***G1*** and ***G2***, the first step is the construction of a so-called product graph ***P*** (cf. Figure 1). Each node of ***P*** consists of a pair of nodes with identical labels from the input graphs, thus representing an "isomorphism" (matching) of size ***1*** between the two graphs. Two nodes ***(u,v)*** and ***(u',v')*** of ***P*** are connected by an edge if the respective matchings are compatible, i.e. the edge connecting ***u*** and ***u'*** in ***G1*** (if any) has the same label as the edge connecting ***v*** and ***v'*** in ***G2***. As shown by Levi [Levi, 1972], a maximal complete subgraph (clique) of ***P*** corresponds bijectively to a maximal subgraph isomorphism between ***G1*** and ***G2***. Thus, it is possible to detect a common substructure of ***G1*** and ***G2*** using the clique-detection algorithm of Bron and Kerbosch. In the example of Figure 1 (left), this yields the mapping ***A-a***, ***B-b***, and so on.

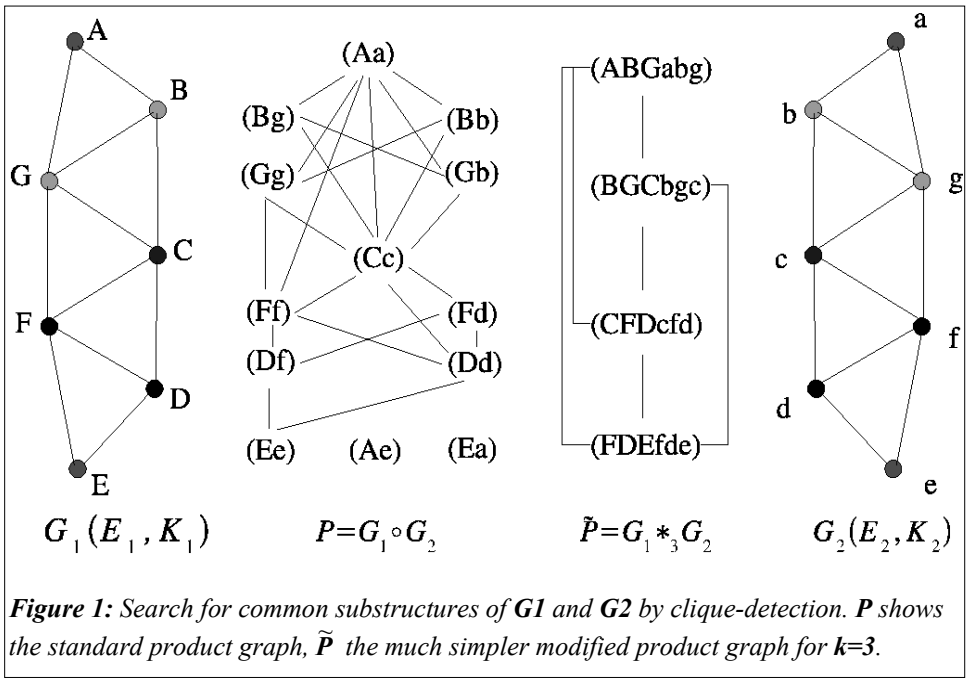

Figure 1: *Search for common substructures of* ***G1*** *and* ***G2*** *by clique-detection.* ***P*** *shows the standard product graph,* $\tilde{P}$ *the much simpler modified product graph for* ***k=3****.*

Even though clique-detection methods provide a powerful tool for the structural comparison of proteins, they possess some major drawbacks: First, depending on the size of the input graphs and the distribution of the different label types, the product graph soon becomes relatively large and densely connected, which in turn leads to huge running times of the applied algorithms. This problem is particularly severe in our case, as we are indeed interested in the comparison of large graphs. Second, it is difficult to estimate the result of a comparison in advance. Hence, it is necessary to perform many superfluous calculations, as it is impossible to detect cases in which structures are obviously unrelated. In the next section, we propose a new hybrid approach which overcomes these limitations at least to some extent.

3 Clique-detection based on clique hashing

One reason for the large size of the product graph ***P*** is the fact that false-positive matches are very likely to occur for single nodes of the input graphs: ***P*** will usually contain a large number of nodes

having the same label just by chance. Take for instance the node ***A*** of graph ***G1***, which is isomorphic to node ***a*** of graph ***G2*** (Figure 1). Apart from this isomorphism, the mapping ***A-e*** has to be considered as well, as ***e*** has the same color as ***A***. But since ***A*** and ***e*** have a completely different environment (nodes ***G*** and ***B*** may not be mapped to ***d*** and ***f***), this is actually not necessary. The clique-detection algorithm spends most of its running time on the elimination of such false-positive matches and the assembling of real matches into larger matchings.

Intuitively, assembling individual pieces becomes simpler when starting from larger pieces. Our method therefore starts from a modified product graph, whose nodes represent larger local matches of size ***k***. This substantially reduces the probability of false-positive matches and leads to smaller product graphs (Figure 1 right). At the same time, the information content of the local matches increases. Thus, it can be hoped that the number of local matches allows for an estimation of the similarity of the complete structures under consideration.

How can we generate the local matches? Our approach is motivated by the geometric-hashing technique [Nussinov-Wolfson, 1991]. The input graphs are split into a large number of ***k-***cliques, complete subgraphs of a fixed size ***k***. As complete graphs of a fixed size are always isomorphic, each substructure is defined uniquely by its node and edge labels. Hence, to search for similar substructures, it is sufficient to consider only those labels. The labels are then mapped onto points in Euclidian space by using a normalization scheme, whereas the node coloring is mapped onto a discrete attribute and the edge weights are represented directly. These points are then stored in an R*-tree [Guttman, 1984; Beckmann et al., 1990], a standard spatial index structure for external memory. As opposed to the commonly used hash tables, this structure adapts itself dynamically to the distribution of the data points and therefore guarantees a better worst-case behavior at the expense of a slightly worse average-case behavior. Additionally, boundary effects related to the binning of the data points are automatically avoided, and the dynamic insertion and deletion of entries is simplified since no global re-organization of the index is necessary.

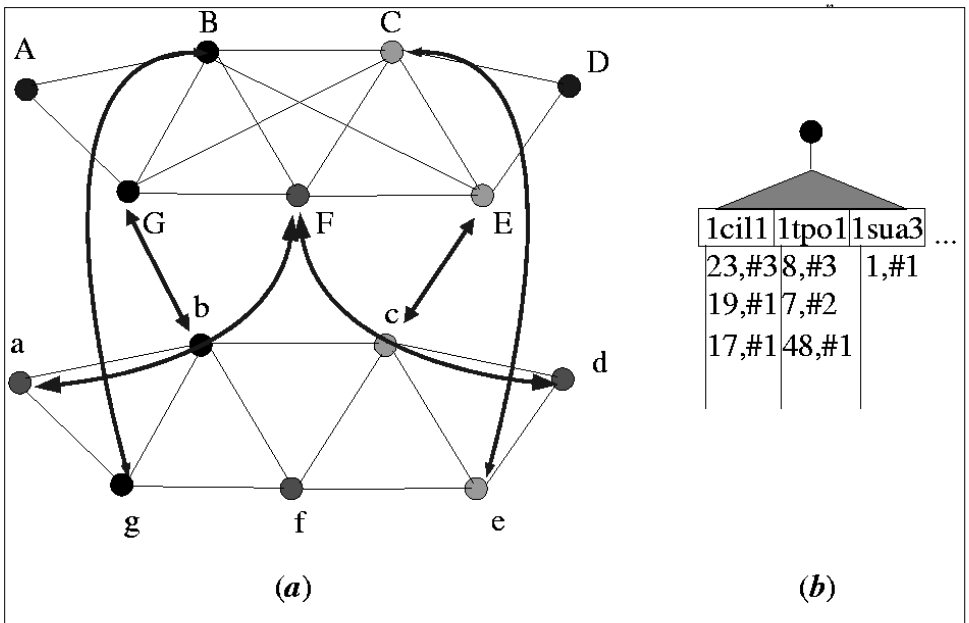

Figure 2: (a) *shows a false-positive match, indicating an ambiguous mapping of* ***F*** *to* ***a*** *and* ***d (b)*** *shows the hit list for the comparison of one pseudocenter against a dataset of* ***3*** *binding pockets.*

Once this index structure is built for all structures of the dataset in a preprocessing stage, it is possible to perform an efficient similarity search. Given a query structure, for all ***k***-cliques in its graph representation ***Q***, a window query in the index is performed, resulting in a number of ***k***-cliques with the same node coloring and a similar (up to a parameter ε) edge weighting. Each such hit indicates a local match (“isomorphism” of size ***k***) between query structure ***Q*** and a hit structure

H from the indexed dataset. The number of local matches found for each hit structure allows for a rough estimation of the similarity between ***Q*** and ***H***. Moreover, the local matches may be used to build the modified product graph mentioned earlier. Note that the decomposition of the graph structures is highly redundant, hence it is neither necessary to use all possible ***k***-cliques for the index generation, nor is it required to launch all possible queries associated to the ***k***-cliques of the query pocket. Instead, random sampling strategies may be used to reduce the index size and to accelerate the querying process.

A critical point concerns the selection of the parameter ***k***, which determines the size of the substructures under consideration. On the one hand, it would be preferable to choose a high value of ***k*** to reduce the probability for false-positive matches. On the other hand, the number of ***k***-cliques in a graph depends exponentially on ***k***, thus large values of ***k*** lead to huge index structures and immense numbers of queries that have to be performed. To overcome this problem, we developed a filtering step for the identification of obviously false-positive chance matches. Using this technique, which is detailed below, it is possible to use relatively small values for ***k*** without being overwhelmed by a huge amount of insignificant hits. For our experiments, we used ***k=3***.

Query Pocket	Number of entries from same SCOP-family in dataset	Original Approach	Optimized Approach - 100% (25%)	Optimized Approach - 80 % (20%)	Optimized Approach - 60 % (15%)	Optimized Approach - 40 % (10%)	Optimized Approach - 20 % (5%)	Optimized Approach - 10 % (2.5%)
1AHX.5	78	78	73	73	75	73	73	63
		1823 s	416 s (4.4)	346 s (5.3)	284 s (6.4)	209 s (8.7)	122 s (15.0)	83 s (22.0)
1COM.1	20	20	20	20	20	19	13	1
		823 s	150 s (5.5)	112 s (7.3)	94 s (8.8)	73 s (11.2)	54 s (15.2)	33 s (25.0)
1CYD.1	86	85	73	72	68	64	59	47
		2437 s	767 s (3.2)	710 s (3.4)	561 s (4.3)	434 s (5.6)	241 s (10.1)	137 s (17.8)
1DD7.2	16	10	10	10	10	10	10	10
		1897 s	511 s (3.7)	449 s (4.2)	338 s (5.6)	239 s (8.0)	134 s (14.2)	94 s (20.2)
1LDN.15	31	29	24	23	23	23	19	16
		2099 s	625 s (3.4)	513 s (4.1)	405 s (5.2)	299 s (7.0)	172 s (12.2)	103 s (20.4)
1QQ8.3	5	5	5	5	5	5	5	2
		1640 s	373 s (4.4)	301 s (5.4)	231 s (7.1)	167 s (9.8)	104 s (15.8)	74 s (22.2)
2KCE.6	77	77	74	75	73	72	71	56
		2344 s	531 s (4.4)	509 s (4.6)	392 s (6.0)	281 s (8.3)	159 s (14.7)	104 s (22.5)
4COX.2	41	31	25	24	24	24	24	21
		3178 s	706 s (4.5)	622 s (5.1)	477 s (6.7)	339 s (9.4)	180 s (17.7)	114 s (27.9)
4VGC.1	44	42	42	42	42	42	42	35
		1208 s	371 s (3.3)	276 s (4.4)	225 s (5.4)	170 s (7.1)	101 s (12.0)	72 s (16.8)
7TIM.2	33	32	32	32	31	30	20	1
		737 s	99 s (7.4)	81 s (7.4)	69 s (10.7)	57 s (12.9)	44 s (16.8)	23 s (32.0)

Table 1: *Results of some similarity searches in a representative dataset of 2138 binding pockets. The SCOP-classification [Murzin et al., 1995] of the query structures is used as a reference, because similar proteins should possess similar binding sites. For the original approach, we present the runtime (in seconds) and the number of entries from the respective SCOP-family yielding a similarity score above 5.0. (On average, 49.27 (2.3 %) entries score above this threshold.) For the optimized approach, we show the associated results (incl. relative speed-up factor) for different sampling rates. The rates are given as percentage of queries made (percentage of comparisons made), as we use a 25% sampling when building the index.*

The clique-detection technique for the modified product graph intuitively tries to assemble the local matches into larger matchings and is thus somewhat related to the problem of DNA fragment assembly. As in fragment assembly, the assembling is guided by overlaps. We define a neighborhood relation for the substructures of a structure as follows: two ***k***-cliques of a graph descriptor are overlapping (neighbored) if and only if they share at least one common node. If two substructures of the query structure are overlapping, only those hits for these structures which are also overlapping are of interest. Other hits can be discarded: they cannot be merged into larger matchings since this would lead to an ambiguous mapping of graph nodes. Figure 2 (a) shows an example of two matches which do not maintain the neighborhood relation: the ***3***-cliques ***BFG*** and ***CEF*** are neighbors, as they share the common node ***F***. The mapping of ***BFG*** to ***abg*** and of ***CEF*** to

cde as indicated by the arrows is discarded, as ***abg*** and ***cde*** are not overlapping and thus a merging of those local matchings would lead to an ambiguous mapping of ***F***.

This filtering step is carried out using a novel construct called *hit list* (Figure 2 (b)). Each node of the query graph is endowed with such a list. The latter collects all possible matching partners for the respective query node during the index querying process. Each time a hit is found for a ***k***-clique containing the respective node, this hit "votes" for the mapping of the respective node to the associated node of the an indexed structure. This "voting" is stored in the hit list. At the end of the whole querying process, each hit list contains a number of possible mappings of the respective node to different nodes of structures from the indexed dataset. Mappings with a high number of "votes" are supported by a large number of overlapping hit substructures, because these hits all share a common node (i.e. the one involved in the respective mapping). Such high-scoring hit list entries are therefore likely to participate in a larger local matching. In other words, there is a low probability that these entries are false-positive matches. In the example of Figure 2 (b), there are ***3*** votes for the mapping of the respective node to node ***23*** of structure ***1cil.1***, but only ***1*** vote for the mapping to node ***19***. Thus, a mapping to node ***23*** is more likely. Note that the whole filtering process depends only on the query structure and is independent of the hit structure. Thus, queries may be processed using solely information from the query and the index structures. This is a major advantage when working with large datasets, as it is not necessary to keep large portions of the dataset in main memory. At the same time, the filtering step is linear in the (usually large) number of hits and thus more efficient than an explicit check of the overlap, which would require an enumeration of all pairs of hits from the same dataset entry.

Whether an entry of a hit list qualifies for further consideration depends on a user-defined threshold parameter MIN_VOTES. The choice of this parameter has a considerable influence not only on the runtime of the method but also on the quality of the query results, as it is responsible for the separation of significant and insignificant matches. We are currently building a probabilistic model for the method in order to derive statistical distributions for the number of votes. These distributions will provide the basis for an optimal choice of the MIN_VOTES parameter. If a sampling strategy is applied, the parameter has to be adjusted, as fewer hits will lead to a smaller number of "votes".

The list of qualified entries for each pair of binding pockets is finally used to build the product graph for the clique-detection process. As each entry of this list is part of a larger local matching, the product graph will be relatively small, containing only few false-positive matches. The size of the matching generated by clique-detection is then used as a measure of similarity. As noted earlier, it is also possible to calculate a superimposition from the mapping and to perform a more complex grid-based analysis of the resemblance of the surfaces.

4 Results

The implementation of our approach has been carried out in C++ and is based on the Cavbase extension of the Relibase system [Hendlich et al., 2003]. We use Cavbase [Schmitt et al., 2002] for the data handling and exploit some of its infrastructure. A dataset of 2138 binding pockets originating from 1560 PDB entries has been derived and used to assess the performance and to validate our approach. The dataset is intended to be a representative subset of all enzyme binding sites in the PDB. The main selection criteria have been: (i) availability of a SCOP-classification [Murzin et al., 1995] for the respective protein structure, (ii) availability of an E.C.-classification for the respective protein structure from the ENZYME-classification database [Bairoch, 2000] and (iii) whether the binding pocket contains a complete ligand structure. The dataset is highly diverse and covers more than 200 different E.C.-numbers. Typically, not only the catalytic pockets of a structure are included, but also binding sites of co-factors etc.

The preprocessing step – which has to be performed only once for each dataset – required a few hours on five standard Linux computers. It produced an index of approximately 772 MB, where we used a random sampling strategy with a sampling rate of 25% (i.e. only 1 of 4 ***k***-cliques has been included in the index, the full index has a size of approximately 2.9 GB).

After the index is built, similarity searches may be performed efficiently in the dataset. Depending on the configuration of the applied hardware and on the size of the query structure, a single (one against all) search may be performed in a few minutes. Again, it is possible to reduce this runtime by making use of a random sampling strategy: Table 1 summarizes some query results for different types of binding pockets and compares them to the SCOP-classification of the respective protein structures. The runtimes are given for the original approach as well as for the optimized approach combined with different (query) sampling rates. Remind that a fixed sample rate of 25 % has been used while building the index. To allow a fair comparison of the results, both implementations used the grid-based surface-overlap measure of the original approach for the scoring. All entries of the dataset with a similarity score above ***5.0*** where considered as hits. Typically, only 2-3 % of all entries score above this threshold. For each query, the number of entries from the same SCOP-family among the hits is shown.

Obviously, it is possible to achieve a significant speed-up (approximately 6-fold, in some cases even one order of magnitude) relative to the original approach without missing too many hits. Note that even a sampling rate of 20 % – which means that only 5 % of the implicit comparisons of the ***k***-cliques of two binding pockets are actually made (as 20 % of the ***k***-cliques from the query pocket are compared to 25 % of the ***k***-cliques from the entries of the dataset) – usually yields sufficiently good results.

As a matter of fact, a comparison against SCOP should be seen only as an initial validation, as SCOP is based on comparisons in sequence and fold space. Therefore, we expect to find structural similarities not automatically covered by SCOP or comparable classification methods. We also performed comparisons with the E.C.-classification of the respective query structures and obtained similar overall results (results not shown). A detailed validation of the approach is clearly beyond the scope of this paper. The interested reader is referred to [Schmitt et al., 2002].

5 Conclusions

We presented an efficient two-step approach for the search of functionally related proteins in large datasets. The first step makes use of an index structure in order to detect common substructures of a fixed size ***k*** in an efficient way. The (small) local hits are then merged into larger matchings in the second step. The two steps are connected by a novel filtering step for the elimination of false-positive matches. The filtering step does not rely on structure superimposition. It is therefore independent of rigid-body assumptions and able to handle conformational flexibility. Additionally, it depends only on information which is available from the query structure and from the index. This is a major advantage when dealing with large datasets that may not be kept in main memory. We applied our method to a medium-sized dataset and presented some of the obtained results. In comparison with the original approach, we achieved a significant speed-up and retained most of the desired hits.

Clearly, the presented approach is intended mainly for larger datasets, which will be accessed by a high number of queries– i.e. for the development of a public web-server, which allows the comparison of a query pocket against a representative subset of the PDB. For searching smaller datasets, the index generation might be too costly and a direct clique-detection implementation is more appropriate.

Existing approaches for the detection of common side chain patterns based on efficient indexing of triplets or triangles (e.g. [Hamelryck, 2003]) have to be very selective to avoid a high amount of false-positive matches. Therefore, they usually detect only highly conserved resemblances. Using our method for post-processing, their criteria could eventually be weakened and thus more distant relationships among structures detected.

As clique-detection techniques are widely used for the structural comparison of proteins, small organic molecules and also for docking applications, our approach might also be applicable to related problems.

References

[Bairoch, 2000] Bairoch, A., “The ENZYME database in 2000”, *Nucl. Acid. Res.,* 2000, 28, 304-305

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B., “The R*-tree: an efficient and robust access method for points and rectangles”, *Proc. ACM SIGMOD 1990*, ACM Press, 322-331

[Bron-Kerbosch, 1973] Bron, C., Kerbosch, J., “Algorithm 457: Finding All Cliques of an Undirected Graph”, *Comm. of the ACM*, 1973, 16 (9), 575-577

[Guttmann, 1984] Guttman, A., “R-Trees: A Dynamic Index Structure for Spatial Searching”, *Proc. ACM SIGMOD 1984*, ACM Press, 47-57

[Hamelryck, 2003] Hamelryck, T., “Efficient identification of side-chain patterns using a multidimensional index tree”, *Proteins*, 2003, 51(1), 96-108

[Kabsch, 1976] Kabsch, W., “A solution for the best rotation to relate two sets of vectors“, *Acta Crystallographica*, 1976, A32, 922-923

[Koch et al., 1996] Koch, I., Lengauer, T., Wanke, E., „An Algorithm for Finding Maximal Common Subtopologies in a Set of Protein Structures“, *J. Comp. Biol*., 1996, 3 (2), 289-306

[Levi, 1972] Levi, G., “A note on the derivation of maximal common subgraphs of two directed or undirected graphs”, *Calcolo*, 1972, 9, 341-352

[Murzin et al., 1995] Murzin, A. G., Brenner, S. E., Hubbard, T., Chothia, C., „SCOP: a structural classification of proteins database for the investigation of sequences and structures“, *J. Mol. Biol.*, 1995, 247, 536-540

[Nussinov-Wolfson, 1991] Nussinov, R.,Wolfson, H.J., “Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques”, *Proc. Natl. Acad. Sci. USA*, 1991, 88, 10495-10499

[Schmitt et al., 2002] Schmitt, S., Kuhn, D., Klebe, G., “A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology*”, J. Mol. Biol*., 2002, 323 (2), 387-406

[Hendlich et al., 2003] Hendlich, M., Bergner, A., Günther, J., Klebe, G., “Relibase: Design and Development of a Database for Comprehensive Analysis of Protein-Ligand Interactions”, *J. Mol. Biol*., 2003, 326 (2), 607-620