Multiple Graph Alignment for the Structural Analysis of Protein Active Sites

Nils Weskamp[*], Eyke Hüllermeier[†], Daniel Kuhn[‡], and Gerhard Klebe[§¶]

9th November 2005

Abstract

Graphs are frequently used to describe the geometry and also the physicochemical composition of protein active sites. Here, the concept of graph alignment as a novel method for the structural analysis of protein binding pockets is presented. Using inexact graph-matching techniques, one is able to identify both conserved areas and regions of difference among different binding pockets. Thus, using multiple graph alignments, it is possible to characterize functional protein families and to examine differences among related protein families independent of sequence or fold homology. Optimized algorithms are described for the efficient calculation of multiple graph alignments for the analysis of physicochemical descriptors representing protein binding pockets. Additionally, it is shown how the calculated graph alignments can be analyzed to identify structural features that are characteristic for a given protein family and also features that are discriminative among related

[*]N. Weskamp is with the Department of Mathematics and Computer Science and with the Institute of Pharmaceutical Chemistry, Philipps-University, Hans-Meerwein-Strasse, 35032 Marburg, Germany, E-mail: weskamp@mathematik.uni-marburg.de

[†]E. Hüllermeier is with the Faculty of Computer Science, Institute of Technical and Business Information Systems, Otto-von-Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany, E-mail: eyke.huellermeier@iti.cs.uni-magdeburg.de

[‡]D. Kuhn is with the Institute of Pharmaceutical Chemistry, Philipps-University, Marbacher Weg 6, 35032 Marburg, Germany, E-mail: kuhnd@staff.uni-marburg.de

[§]G. Klebe is with the Institute of Pharmaceutical Chemistry, Philipps-University, Marbacher Weg 6, 35032 Marburg, Germany, E-mail: klebe@staff.uni-marburg.de

[¶]Corresponding author.

families. The methods are applied to a substantial high-quality subset of the PDB database and their ability to successfully characterize and classify 10 highly populated functional protein families is shown. Additionally, two related protein families from the group of serine proteases are examined and important structural differences are detected automatically and efficiently.

Keywords: Knowledge Discovery in Databases, Structural Pattern Discovery, Fuzzy Patterns, Graph Mining, Drug Design

1 Introduction

Bioinformatics often focuses on the study of large and complex biological molecules such as DNA or proteins. A typical goal is to identify common features present in different molecules and thus to reveal evolutionary relationships among the examined molecules. Frequently, one is interested in the identification of homologies, i.e. to extract pairs of molecules that are likely to have a common ancestor. A plethora of methods based on sequence alignment has been developed to find such evolutionary dependencies in biological sequence data. In particular, the multiple sequence alignment is an established approach for the identification of residues that are conserved across many members of a gene or protein family [36, 6, 32]. Position specific scoring matrices are derived from multiple sequence alignments and allow for the assignment of novel sequences to known families [1].

Here, we present the development of a comparable concept for the identification of *functional* relationships among proteins that is solely based on structural information and independent of sequence or fold homology. Proteins interact specifically with other molecules, such as substrates, agonists, antagonists or allosteric modulators. These functional binding partners are recognized by the protein in binding sites that are complementary in shape and physicochemical properties to the bound molecule. The shape of such a binding pocket is therefore intimately related to the function of the respective protein. In particular, the characterization of binding pockets is of interest for pharmaceutical

research as strong binding to the active site of a protein is frequently the mechanism of action of drugs.

A number of authors have proposed the use of graph models to represent and analyse three-dimensional protein structures [15, 26, 25, 18, 21, 37]. The present work builds upon Cavbase [31, 19], a database system for the fully-automated detection and extraction of protein binding pockets from experimentally determined protein structures (available through the database PDB [8]). Graphs are used as a first approximation to describe binding pockets in Cavbase.

A large number of approaches has been described in the literature for the structural analysis of protein active sites. These can be roughly divided into two main categories: approaches that derive templates from a family of related structures [2, 5, 35, 37, 16, 24] and those that rely on the comparison of individual structures [29, 34, 21, 31]. The work presented here is closely related to the template-based approaches. Although some of the referenced methods are based on graph models, none of them is using inexact graph matching techniques to derive templates or to classify structures. The major advantage of the method presented here is the fact that the consensus graphs used to characterize a functional protein family are not of a fixed small size. Instead, the size of the consensus graph depends on the size of the generated graph alignment. It does not only contain a rather small number of highly (or even perfectly) conserved residues, but it contains instead a node for each center that is found in at least one of the aligned binding pockets (cf. Section 4.1). Thus, also weakly conserved patterns that are not necessarily found in each member of a family can be identified. As a degree of conservation is calculated and stored for each node in a consensus graph, it is also possible to reduce a consensus graph in such a way that it contains only highly conserved (e.g., catalytic) residues.

Inexact graph matching techniques have been studied intensively in the field of pattern recognition [13]. But there, with very few exceptions [38], only the pairwise graph matching case is considered. Additionally, it is not the goal of pattern recognition applications to derive descriptive and interpretable models for certain classes of objects. Recently, graph alignment gained increasing attention for the analysis of biological networks in the

field of systems biology [7, 23]. Yet, the methods developed for systems biology applications are not applicable to the problem examined here: Biological networks are huge, but sparse graphs (i.e., tree-like graphs) and usually, a nearly unique correspondence among the different nodes of different networks exists. Thus, the problem of aligning a small number of networks is algorithmically different from the problem of aligning a large number of very dense protein binding pocket descriptors that is studied here.

This paper is organized as follows: In Section 2, we introduce some basic concepts and state the problems examined in this paper. Additionally, the section contains a formal description of the concept of graph alignment. In Section 3, we present an algorithm for the efficient calculation of graph alignments and in Section 4, methods for the analysis of graph alignments are discussed. Section 5 describes the experimental validation of the approach analyzing the number of known functional families of enzymes.

2 Problem Statement

Throughout this paper, it is assumed that a set $\mathcal{G} = \{G_1(V_1, E_1), \ldots, G_n(V_n, E_n)\}$ of connected, node-labeled and edge-weighted graphs is given, each of them representing a protein binding pocket. The geometrical arrangement and the physicochemical properties of a binding pocket are represented by pre-defined pseudocenters — spatial points that represent the center of a particular property. The type and the spatial position of the centers depend on the amino acids that border the binding pocket and expose their functional groups. They are derived from the protein structure using a set of pre-defined rules [31]. As possible types for pseudocenters hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic and aromatic properties are considered. Pseudocenters can be regarded as a compressed representation of areas on the cavity surface where certain protein-ligand interactions are experienced.

The assigned pseudocenters form the nodes $v \in V_i$ of the graph representation, and their properties are modeled in terms of node labels $l(v) \in \{1, \ldots, 5\}$, whereas 1 stands for donor, 2 for acceptor, etc. Two centers are connected by an edge in the graph represen-

tation if their Euclidean distance is below 12.0 Å and each edge $e \in E_n$ is labeled with the respective distance $w(e) \in \mathbb{R}$[1]. The edges of the graph thus represent geometrical constraints among certain properties. Cavbase currently contains 113,718 hypothetical binding pockets that have been extracted from 23,780 publicly available protein structures using the LIGSITE-algorithm [20]. In the crystal structures used as input, not all of the extracted cavities actually host a bound ligand. Some of them represent clefts or depressions on the protein surface to which a straight-forward assignment of a biochemical function is difficult. On average, a graph representation of a binding pocket has approximately 85 nodes; however, also graphs with several hundred nodes are frequently detected and extremes with thousands of nodes are generated. The graphs are rather dense as approximately 20 percent of all pairs of nodes are connected by an edge. Some PDB entries have been discarded due to their low resolution or due to other qualitative deficiencies[2].

The goal of this paper is to characterize functional protein families by detecting conserved structural patterns in a given family along with discriminative patterns between different, but related families. We thus further assume that we are given a class membership function $c : \mathcal{G} \mapsto \mathbb{N}$ that assigns each graph descriptor to a certain functional class (or category). Here, the functional classification of the ENZYME database [4] is used and all enzyme entries that are annotated by the same E.C.-number [28] are defined as one functional class.

When comparing homologs from different species in protein cavity space, one has to deal with the same mutations that are also given in sequence space. In the cavity space, such mutations result in the replacement of certain functional groups by other functional groups while the overall fold of the protein virtually is preserved. Additionally, proteins exhibit conformational flexibility, thus even structures of the same protein can differ significantly in space. This is particularly the case when structures of protein-ligand complexes are compared to the uncomplexed structures. Such mutations and conformational

[1] An interaction distance of 12.0 Å is chosen as cutoff value to concentrate on local features in the binding sites. Furthermore, the resulting graphs are more sparsely populated.

[2] The figures in this paragraph correspond to the database release as of June 2005.

differences heavily affect also the spatial structure of a binding site, and in consequence the graph descriptors. Thus, one cannot expect that the graph descriptors for two related binding pockets match exactly. In this contribution, the following types of edit operations that distinguish a graph $G_1(V_1, E_1)$ from an other graph $G_2(V_2, E_2)$ are considered:

1. Insertion or deletion of a node $v_1 \in V_1$ ("InDel"). A pseudocenter can be deleted or introduced due to a mutation in sequence space. Alternatively, a conformational difference can affect the exposure of a functional group towards the binding pocket, accordingly an insertion or deletion in the graph descriptor could result.

2. Change of the label $l(v_1)$ of a node $v_1 \in V_1$ ("Node Mismatch"). The assigned physicochemical property ("type") of a pseudocenter can change if a mutation replaces a certain functional group by another type of group at the same position in space.

3. Change of the weight $w(e_1)$ of an edge $e_1 \in E_1$ ("Edge Mismatch"). The distance between two pseudocenters can change due to conformational differences.

By assigning a cost function to each of these edit operations, it is possible to define an edit distance to a pair of graph descriptors. The edit distance of two graphs G_1,G_2 is defined as the cost of a cost-minimal sequence of edit operations that is necessary to transform graph G_1 into G_2. As in sequence analysis, this allows for the introduction of the concept of an alignment of two (or more) graphs. This requires the introduction of dummy nodes $\perp$ that serve as placeholders for deleted nodes. They correspond to the gaps in sequence alignment (cf. Figure 1).

Let $\mathcal{G} = \{G_1(V_1, E_1), \ldots, G_n(V_n, E_n)\}$ be a set of graphs. Then $\mathcal{A} \subseteq (V_1 \cup \{\perp\}) \times \cdots \times (V_n \cup \{\perp\})$ is an alignment of the graphs in $\mathcal{G}$ if and only if

1. for all $i = 1, \ldots, n$ and for each $v \in V_i$ exists exactly one $a = (a_1, \ldots, a_n) \in \mathcal{A}$ such that $v = a_i$ (i.e., each node of each graph occurs exactly once in the alignment).

2. for each $a = (a_1, \ldots, a_n) \in \mathcal{A}$ exists at least one $1 \leq i \leq n$ such that $a_i \neq \perp$ (i.e., each tuple of the alignment contains at least one non-dummy node).

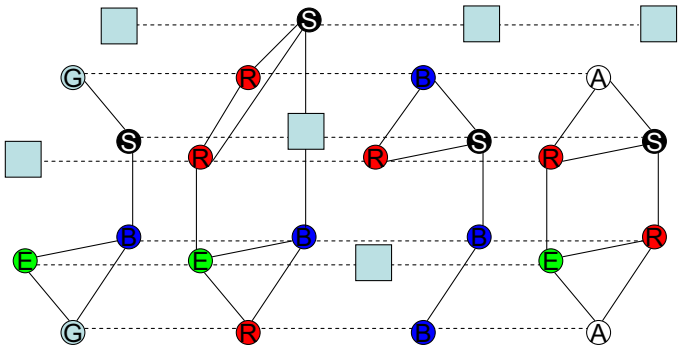


Figure 1: An alignment of four similar, but not identical graphs. The node labels are indicated by the letters assigned to the nodes (shown as circles). The edge labels are omitted for simplification. The assignments among the different graph nodes are indicated by the dashed lines. Large boxes in grey represent dummy nodes that have been introduced in the alignment to represent "missing" nodes.

```
.GSBGE.
SR.BRER
.BSBB.R
.ASRAER
```

Figure 2: Textual representation of the graph alignment in Figure 1. Each row contains the labels of the nodes of one of the graphs. The first row represents the leftmost graph, the last row the rightmost graph. The columns represent nodes that are assigned to each other. Dummy nodes $\perp$ are represented as dots. A major difference compared to sequence alignments is that the order of the columns is arbitrary (i.e., it is possible to swap complete columns without affecting the meaning of the alignment). From this representation, it is obvious that the columns $3, 4, 6, 7$ contain centers which are highly conserved and in which graphs conserved nodes are missing or labelled differently.

Each tuple in the alignment contains a certain number of nodes from the different graphs that are matched upon each other. If a node has no matching partner in a certain graph, it has to be mapped onto a dummy node $\perp$.

To assess the quality of a given alignment, a scoring function is necessary. This scoring system corresponds to the above-mentioned edit distance, as each graph alignment defines a set of edit operations that have to be performed to transform one of the aligned graphs into another entry of the alignment. Here, a scoring function that follows a sum-of-pairs scheme (i.e. a multiple graph alignment is reduced to a set of pairwise graph alignments that are scored individually; the obtained scores are summed subsequently) is proposed:

Parameter	Value
$\underline{nScore}_{match}$	1.0
$\underline{nScore}_{mismatch}$	-5.0
$\underline{nScore}_{dummy}$	-2.5
$\underline{eScore}_{match}$	$0.2 \cdot (2.0 - \lvert w(a_k^i, a_k^j) - w(a_l^i, a_l^j)\rvert)$
$\underline{eScore}_{mismatch}$	-0.1
$\underline{eScore}_{dummy}$	-0.2
ϵ	2.0

Table 1: Default parameter settings used during the experiments presented in this article

The score s for a given alignment $\mathcal{A} = (a^1, \ldots, a^m)$ is defined as

$$s(\mathcal{A}) = \sum_{i=1}^{m} \underline{nodeScore}(a^i) + \sum_{i,j=1,i<j}^{m} \underline{edgeScore}(a^i, a^j) \tag{1}$$

where

$$\underline{nodeScore}\begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix} = \sum_{j,k=1,i<j}^{n} \begin{cases} \underline{nScore}_{match} & l(a_j^i) = l(a_k^i) \\ \underline{nScore}_{mismatch} & l(a_j^i) \neq l(a_k^i) \\ \underline{nScore}_{dummy} & a_x^i = \perp, a_y^i \neq \perp;\ x, y = j, k \end{cases} \tag{2}$$

and

$$\underline{edgeScore}\left(\begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_n^j \end{pmatrix}\right) = \sum_{k,l=1,k<l}^{n} \begin{cases} \underline{eScore}_{match} & \lvert w(a_k^i, a_k^j) - w(a_l^i, a_l^j)\rvert \leq \epsilon \\ \underline{eScore}_{mismatch} & \lvert w(a_k^i, a_k^j) - w(a_l^i, a_l^j)\rvert > \epsilon \\ \underline{eScore}_{dummy} & (a_x^i, a_x^j) \in E_x,\ (a_y^i, a_y^j) \notin E_y; \\ & x, y = k, l \end{cases} \tag{3}$$

The scores for the different cases (i.e., $\underline{nScore}_{match}$, $n\underline{Score}_{mismatch}$, $\underline{nScore}_{dummy}$, $\underline{eScore}_{match}$, $\underline{eScore}_{mismatch}$, $\underline{eScore}_{dummy}$) have to be defined by the user. Table 1 shows the parameter settings that were used for the examples given in section 5.

3 Algorithms

The problem of calculating an optimal graph alignment for a given set of graphs is computationally very complex. The subgraph isomorphism problem (which is known to be NP-complete [3]) can be seen as a special case of the graph alignment problem where the cost for mismatches is set prohibitively high. Thus, one cannot expect to find an efficient algorithm that is guaranteed to find an optimal alignment for a given set of graphs. In this section, simple and efficient heuristics for the graph alignment problem are therefore described that were found to be useful for the problem instances that we examined. First, only the pairwise case (i.e. calculating an optimal graph alignment for only two graphs) is examined and later, it is shown how the multiple case can be reduced to the pairwise case.

3.1 Pairwise Graph Alignments

Our heuristic for the pairwise graph alignment problem is based on the assumption that two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with a relatively small edit distance (and those are of particular interest in our application as described in Section 2) share a common subgraph that is contained in both graphs without any modifications. A similar assumption is often made in string matching, where it can be motivated by the $q-$gram lemma from Jokinen and Ukkonen [22]. The strategy thus proceeds in two steps: Firstly, common subgraphs of the input graphs are identified using well-established techniques for exact graph matching [11, 27]. These subgraphs are subsequently used as seed solutions and extended greedily until all nodes of both graphs are considered in the alignment.

The problem of searching for maximal common subgraphs among the input graphs ("subgraph isomorphism") can be reduced to a clique search problem by constructing an association graph $A(V, E)$ with $V \subseteq V_1 \times V_2$. A contains a node for each pair of nodes from the input graphs G_1 and G_2 that are compatible with each other as they share the same node label. Each node of A thus represents a trivial common subgraph of size 1. Two nodes in A are connected by an edge if and only if either the corresponding nodes in G_1

and in G_2 are connected by edges with compatible edge labels or if they are connected by edges neither in G_1 nor in G_2.[3] The edges in A thus connect trivial common subgraphs that are compatible with each other. Therefore, a maximal clique in A corresponds to a maximal number of trivial common subgraphs that can be merged into one large non-trivial common subgraph of G_1 and G_2. After construction of the association graph A, one can apply the branch-and-bound algorithm of Bron and Kerbosch [11] to identify maximal cliques in A.[4] Each of the generated subgraph isomorphisms is subsequently subjected to the second step of our strategy, which is detailed below.

This second step expects as input a subgraph isomorphism (i.e. a common subgraph) that serves as a seed for further extension. The goal of the extension step is to find, for each node of G_1 that is not already part of the seed solution, a match partner in G_2 (possibly a dummy node). Vice versa, for each node in G_2, a corresponding node in G_1 has to be detected. As a graph alignment has to be constructed and mismatches are allowed, many different assignments among the remaining nodes in G_1 and G_2 are possible. To reduce the number of possibilities, additional constraints on the mapping are necessary. These result from the edges in G_1 and G_2. Thus, the extension of the initial solution should start with a node of G_1 for which only few alternative match partners in G_2 exist, i.e. for which many constraints are given. Therefore, the algorithm starts with the remaining center v_1 of G_1 that has the highest number of edges to those nodes that are already part of the solution. This center is highly constrained and should thus map to G_2 relatively unambiguously. We now enumerate all remaining nodes v_2 of G_2 and calculate the score for the mapping of v_1 to v_2. If $l(v_1) = l(v_2)$, a bonus for the matching label is added ($\underline{nScore}_{match}$), otherwise a penalty score for a mismatch is added ($\underline{nScore}_{mismatch}$). Similarly, match and mismatch scores are calculated for the edges by comparing all edges starting from v_1 and ending within the initial solution in G_1 to the

[3] To further improve the efficiency of the method, we introduce an edge in A only if the corresponding nodes in G_1 and G_2 are connected. Thus, only complete common subgraphs (i.e., cliques) are identified. Yet, as the graph descriptors in our application are quite dense, this is not a too strong restriction. Additionally, one should keep in mind that the detected common subgraphs are only used as seed solutions and are further extended in the second step.

[4] As the number of cliques can be quite large for our graph descriptors, we determined experimentally that it is sufficient to consider only up to $n = 100$ seed solutions for our graph descriptors.

corresponding edges starting from v_2 and ending within the initial solution in G_2. Then, the best matching partner v_2' is selected and the mapping of v_1 to v_2' is incorporated in the solution. If no favorable mapping is found for v_1 (i.e., the mapping of v_1 to a dummy node yields a higher score than any other alternative), the respective dummy node is inserted in the alignment. The process continues iteratively. Note that, as the solution is extended step by step, the number of constraints on the remaining nodes of G_1 and G_2 can increase and thus lead to a better alignment (cf. Algorithm 1).

The strategy that we described in this section is capable to calculate an alignment for two graphs with $80 - 120$ nodes in less than 5 seconds on a standard Linux machine, whereas an exhaustive enumeration of the search space would be intractable. The obtained alignments achieve a high quality with respect to the experiments detailed below.

3.2 Multiple Graph Alignments

The number of graph descriptors to be aligned usually exceeds two as many different structures are available for most of the relevant protein families. Thus, the approach for the determination of pairwise graph alignments has to be extended to multiple comparisons. As a sum-of-pairs scheme is used for the scoring function that reduces the scoring of the multiple case to a series of pairwise cases, it is reasonable to construct a multiple graph alignment from a set of pairwise alignments. Therefore, the strategy for calculating pairwise graph alignments described in the previous section can be applied without modification and used as a sub-routine by the algorithm described in the following.

The problem of calculating a multiple graph alignment thus reduces to the problem of incrementally merging pre-calculated pairwise alignments. The simplest strategy to perform such an incremental merging is to calculate a star alignment, i.e. one of the graph instances is used as a pivot (which is also called center) and all other instances are aligned to this pivot. A similar strategy is also frequently used in sequence alignment [33]. The different pairwise alignments are incrementally merged into one multiple alignment: If node v_1 of the pivot graph G_1 is mapped onto $v_2 \in G_2$ and independently onto $v_3 \in G_3$,

Input : Graphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$
Output: Alignment $\mathcal{A}$ for the graphs G_1,G_2
//construct association graph $A(V, E)$
foreach $v_1 \in V_1$ **do**
  foreach $v_2 \in V_2$ **do**
    if $l(v_1) = l(v_2)$ **then**
      insert node (v_1, v_2) **into** V
    end
  end
end
foreach $v = (v_1, v_2) \in V$ **do**
  foreach $v' = (v_1', v_2') \in V$ **do**
    if $|w(v_1, v_1') - w(v_2, v_2')| \leq \epsilon$ **then**
      insert edge (v, v') **into** E
    end
  end
end
// find seed matches and extend greedily
for *up to* n *cliques* C *in* A **do**
  insert each $(v_1, v_2) \in C$ **into** $\mathcal{A}$
  foreach $v_1 \in V_1$ *with* $v_1 \notin C$ **do**
    collect in W_1 **edge weights to all nodes contained in** C **adjacent to** v_1
    foreach $v_2 \in V_2$ *with* $v_2 \notin C$ **do**
      collect in W_2 **edge weights to all nodes contained in** C **adjacent to** v_2
      if $l(v_1) = l(v_2)$ **then**
        $score := \underline{nScore}_{match}$
      else
        $score := \underline{nScore}_{mismatch}$
      end
      foreach $w_1 \in W_1$ **do**
        look up corresponding $w_2 \in W_2$
        if $|w_1 - w_2| \leq \epsilon$ **then**
          $score := score + \underline{eScore}_{match}$
        else
          $score := score + \underline{eScore}_{mismatch}$
        end
      end
    end
    select v_2' **with maximal** $score$
    if $score > \underline{nScore}_{dummy}$ **then**
      insert (v_1, v_2') **into** $\mathcal{A}$
    else
      insert $(v_1, \perp)$ **into** $\mathcal{A}$
    end
  end
end
select and return $\mathcal{A}$ with maximum overall score

Algorithm 1: Algorithm for the calculation of pairwise graph alignments using a greedy strategy.

it is assumed that also v_2 can be mapped onto v_3 and the tuple (v_1, v_2, v_3) is considered in the multiple alignment. If one of the considered nodes v_1, v_2, v_3 is a dummy node, it is handled accordingly. Thus, if a dummy node is inserted in one of the pairwise alignments, this decision is never reversed (cf. Algorithm 2).

The quality of the generated alignment depends on the choice of the center graph. To cope with this fact, one typically tests different (if not all possible) graphs as centers and calculates the overall score of all generated multiple alignments. Finally, the alignment with the best score is selected for further consideration.

A major advantage of the star-like alignment scheme is its ability to construct alignments incrementally. As the number of publicly available protein structures is growing very fast in the recent years, it is important to update the alignment models for a certain protein class once new data become available. Fortunately, this is easily done for star alignments: Given a star alignment and a new graph descriptor that should be added to the alignment, one simply calculates an alignment to the known pivot graph and then adds the new graph descriptor to the alignment as described above.

Input : Alignments $\mathcal{A}_1, \mathcal{A}_2$, both containing pivot graph $G_i(V_i, E_i)$
Output: Merged Alignment from $\mathcal{A}_1$ and $\mathcal{A}_2$
foreach $a = (a.1, \ldots, a.n) \in \mathcal{A}_2$ **do**
  if $\exists a' = (a'.1, \ldots, a'.m) \in \mathcal{A}_1$ *s.t.* $a.i = a'.i$ **then**
    replace $a' \in \mathcal{A}_1$ by $a'' = (a'.1, \ldots, a'.m, a.1, \ldots, a.n)$
  else
    insert $a'' = (\perp, \ldots, \perp, a.1, \ldots, a.n) \in \mathcal{A}_1$
  end
end
foreach $a' = (a'.1, \ldots, a'.n) \in \mathcal{A}_1$ **do**
  if $\nexists a = (a.1, \ldots, a.n) \in \mathcal{A}_2$ *s.t.* $a.i = a'.i$ **then**
    insert $a'' = (a'.1, \ldots, a'.n, \perp, \ldots, \perp) \in \mathcal{A}_1$
  end
end
return $\mathcal{A}_1$

Algorithm 2: Algorithm for the star-like merging of graph alignments using a fixed reference graph G_i as center.

4 Analysis of Graph Alignments

Once a set of graph descriptors is arranged in a common graph alignment, it is easy to examine the dependencies among the presence, the absence and the labelling of certain nodes and edges with respect to the class membership of the corresponding graph instances. This is possible because graph alignments induce a unique assignment across the nodes present in the different graphs. This provides the opportunity to find "equivalent" nodes for each given node of one graph in the other graphs (including possible dummy nodes).

4.1 Consensus Graphs

For a given set of graph descriptors assigned to a certain functional class, it is interesting to determine which of the pseudocenters are highly conserved across all aligned binding pockets and thus can be regarded as characteristic for the respective functional class. To achieve this goal, a "fuzzy" consensus graph $G'(V', E')$ is calculated based on the graph alignment $\mathcal{A}$ that contains a node v' for each tuple $a = (a_1, \ldots, a_n) \in \mathcal{A}$ in the alignment. For each node v' of the consensus graph, a degree of conservation $\underline{con}(v')$ is calculated:

$$\underline{con}(v') = \frac{\sum_{i=1}^{n} s(a_i)}{n} \tag{4}$$

where

$$s(a_i) = \begin{cases} 0 & a_i = \perp \\ 1 & a_i \neq \perp \end{cases}, \tag{5}$$

i.e. $\underline{con}$ is simply the relative number of graphs in $\mathcal{A}$ in which the respective node is present. The relative frequencies of the different e node labels considered in the analysis are calculated accordingly. Consequently, for each edge e connecting two nodes of the consensus graph a similar degree of conservation is calculated. For the continuous edge weight $w(e)$, one calculates the average value $\underline{avg}(e)$ and the variance $\underline{var}(e)$ around that value. The thus obtained consensus graph can be seen as a prototype or even as a "mean"

for the class of graph descriptors.

It is therefore possible to use the consensus graphs as nearest neighbor classifiers and additionally as prototypes for the functional class under consideration. As it contains information about the degrees of conservation and the distributions of edge and node labels, it is clearly superior to a single centroid graph. To assess for a given graph G its membership in a certain class, one has to examine whether G is present in the consensus graph G' for the respective class. As a robust search for such occurrences is desirable, again the concept of graph alignment is applied. Given the consensus graph G' and the query graph G, a pairwise graph alignment $\mathcal{A}$ can be calculated using the methods described in Section 3.1. As already noted above, it is always possible to calculate such an alignment — even if G does not contain a single conserved node from G'. It is therefore necessary to score the obtained alignment $\mathcal{A}$ to assess how well G fits into the class represented by G'. This scoring step is performed using a modified version of the scoring scheme applied to score graph alignments as described in Section 2. These modifications affect mainly the values for $\underline{nScore}_{match}, \underline{nScore}_{mismatch}, \underline{nScore}_{dummy}, \underline{eScore}_{match}$ and $\underline{eScore}_{mismatch}$. They are no longer constant, but depend on the variations that are observed in the alignment of the different graph instances. E.g., the difference among the continuous edge weight of an edge e' of the consensus graph G' and the respective weight of the corresponding edge e from G is scored depending on the variance of the former weight that was observed in the alignment that was used to calculate G'. If a certain distance among two pseudocenters is strictly conserved in all binding pockets of a functional family, it is also important that this distance is preserved exactly in G if G should be an additional member of this protein family. Otherwise (i.e. if the observed variance of the weight was high), also a rather high deviation from the observed mean should be tolerated in G. For the continuous edge weights, the score

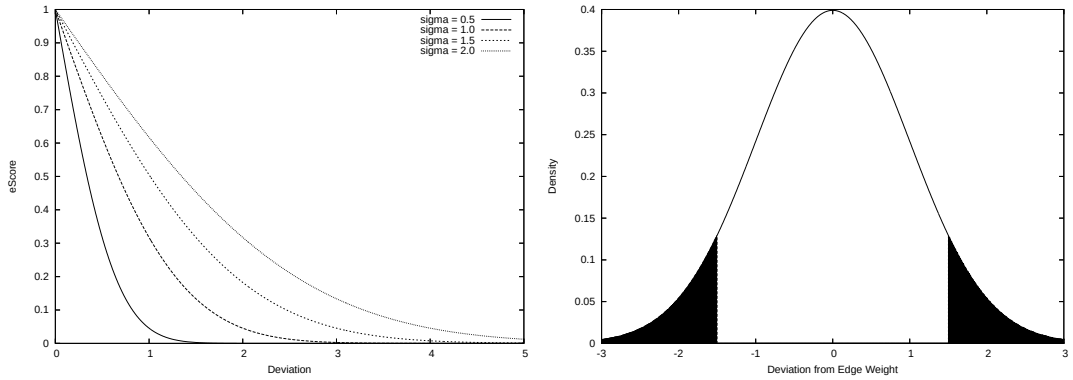
$$\underline{eScore}_{match} = sig(\underline{cons}(e')) \cdot (1 - 2 \cdot (\Phi_{0,\underline{var}(e')}(dev) - \frac{1}{2})), \tag{6}$$

was found to yield good results in our experiments, where $\underline{cons}(e')$ is the degree of con-

servation of the respective edge e', $\underline{var}(e')$ is the observed variance for e' and Φ_{μ,σ^2} is the cumulative distribution function of the normal distribution with mean μ and variance σ^2. dev is the absolute value of the difference between the edge weights $w(e)$ and $w(e')$. $sig(\cdot)$ is the sigmoidal function

$$sig(x) = \frac{1}{1 + \exp((\frac{1}{2} - x) \cdot 10)} \tag{7}$$

Figure 3 (a) shows the score obtained from (6) for different values of $\underline{var}(e')$ and $dev = |w(e) - w(e')|$. The strictness of the scoring system depends on the variance that was observed while calculating the graph alignment $\mathcal{A}$. The score (6) is a measure of how many corresponding edges in the alignment $\mathcal{A}$ have a higher deviation from the mean value $w(e')$ assuming a normal distribution (cf. Figure 3 (b)). The degree of conservation of the edge e' also influences the score, as highly conserved edges should have a higher impact on the overall score than less conserved ones. The sigmoidal function (7) will filter out edges that exhibit a rather low degree of conservation (e.g., below 0.1). Furthermore, it weights edges that show a rather high degree of conservation (e.g., above 0.9) as nearly perfectly conserved. As the practical applications show, one often observes a high number of edges exhibiting a rather small degree of conservation. These are filtered out because it is in question whether they are actually relevant and discriminative for the respective class under consideration. In contrast, if edges are detected corresponding to a very high degree of conservation, one can expect that these edges would only be missing in the remaining graph instances due to experimental uncertainties or inherent deficiencies in our applied heuristics. Accordingly, they are considered as nearly perfectly conserved. In consequence, the score assigned to each aligned edge receives a value in $[0, 1]$ depending on the degree of conservation of this edge and the deviation of its edge weight from the mean value in the consensus graph. For the nodes, a similar score has been be defined that takes the degree of conservation and the relative frequencies of the different node labels into account. We therefore obtain a similarity score s for the graph G and the consensus graph G' that indicates how well the binding pocket represented by G fits into



(a) (b)

Figure 3: (a) The edge score $\underline{eScore}_{match}$ according to (6) for different values of the observed standard deviation σ (i.e., $\sqrt{\underline{var}(e')}$). A conservation degree $\underline{cons}(e')$ of 1.0 is assumed. The x-axis shows the absolute value of the difference dev between $w(e)$ and $w(e')$, the y-axis the obtained score. If the observed variance was low, also the scoring is very strict and even small deviations from the observed mean value are strongly penalized. If the variance was instead comparably high, the scoring is also less strict and higher deviations are tolerated. (b) In the case $\underline{cons}(e') = 1$, the edge score $\underline{eScore}_{match}$ calculates the filled area in the normal distribution (shown as a density function). It therefore measures how many of the edge weights in the dataset support the current edge weight as they had a higher deviation.

the functional class described by G'. It is obtained by summing up all the individual scores for the aligned edges and nodes.

4.2 Calculation of Discriminative Patterns

Most of the methods described in literature [29, 34, 21, 31] for the structural comparison of protein active sites focus on features shared in common or present with high similarity in the examined sites (cf. Section 1). A major advantage of the graph alignment approach is that it also captures dissimilarities using the same framework. E.g., if a certain amino acid is mutated at a given position of a protein, this exchange can replace one physicochemical property by another one. In a graph alignment, this replacement is captured by a mismatch of the assigned pseudocenters while the correspondence would be lost completely in many other approaches. This advantage of graph alignments allows us to identify discriminative features in related, yet distinct functional families of proteins

in terms of their binding pockets.

Let $\mathcal{A}$ be a graph alignment for a set $\mathcal{G} = \{G_1(V_1, E_1), \ldots, G_n(V_n, E_n)\}$ of graphs that belong to two (or more) different classes, i.e.

$$c(G_1) = \cdots = c(G_i) \neq c(G_{i+1}) = \cdots = c(G_n)$$

without loss of generality. Then, it is possible to analyze the influence of the structural features of the graphs in $\mathcal{G}$ on the class membership. This can be done using a plethora of different data analysis methods by deriving a "flat" feature-vector representation for the graphs in $\mathcal{G}$ using the assignment among the nodes that is induced by the graph alignment $\mathcal{A}$. Assuming an arbitrary fixed ordering of the elements $a_1, \ldots, a_n \in A$, one is able to derive a flat feature vector g_i for all $G_i \in \mathcal{G}$:

$$g_i = (\tilde{l}(t_1.i), \ldots, \tilde{l}(t_n.i)), \tag{8}$$

where

$$\tilde{l}(v) = \begin{cases} l(v) & v \neq \perp \\ \perp & v = \perp \end{cases} \tag{9}$$

and $t_j.i$ denotes the $i-$th position of the vector t_j. g_i thus comprises all entries of all tuples in the graph alignment $\mathcal{A}$ that correspond to the graph G_i. It is a "flat" representation of the structured object represented by G_i. Nevertheless it has a direct structural meaning as each entry of g_i corresponds to a particular structural feature (i.e. node). It is therefore straightforward to interpret the results obtained using standard methods for data mining.

E.g., if an appropriate method for data analysis such as decision tree induction or decision rule induction identifies a certain position (i.e. feature, attribute) in the feature vectors as discriminative among the considered classes, it is possible to map this information back to a certain node in each of the aligned graphs by direct lookup in the alignment $\mathcal{A}$ (cf. Figure 4 for an example). We aim at developing a descriptive model that is able to extract and explain discriminative features among related families in an easily

```
0:  ABCDE
0:  ABCDE
1:  ABCDF
1:  ABCDF
```

Figure 4: Textual representation of a simple graph alignment of graphs belonging to two classes. The class labels are indicated in front of the rows. The classes are apparently very similar as four nodes are perfectly conserved over all graphs. Yet, the last column contains nodes with different labels. It is easy to see that this last column is perfectly correlated with the class membership. A rule learning method would derive e.g., the decision rule "IF $col5 = "E"$ THEN $label = 0$ ELSE $label = 1$" to distinguish the two classes. As the last column corresponds to a certain node in each of the considered graphs, it is easy to map this rule back into the respective binding pockets and to identify discriminative features or residues.

interpretable fashion and we do not want to solve a classification problem. Therefore, we only consider data mining methods that generate such interpretable models. Nevertheless, it is also possible to use the obtained models in a predictive way (i.e., for the classification of unlabelled instances). The predictive model is derived from a multiple graph alignment of the graph instances in the training set as described above. To classify new instances, one adds these to the pre-calculated alignment containing the training examples. As already mentioned in section 3.2, graph alignments can be constructed incrementally. It is possible that new tuples have to be added to the alignment while inserting new instances, but these tuples contain only dummy nodes in the positions that correspond to entries of the training dataset as the original alignment already contained all nodes of all instances of the training dataset by definition. Hence, the predictive model is not (or at most trivially) affected through the extension of the alignment. Once the new instances have been added to the graph alignment, feature vectors can be derived from the alignment and the classification proceeds as before.

The ability to analyze automatically and efficiently also differences among related functional (sub-)families is a key advantage of the graph alignment approach. It allows for performing large-scale analyses of large protein families such as the serine and aspartyl proteases or the protein kinases and to identify characteristic and discriminative patterns for these families. This is an important prerequisite, e.g. for the design of novel inhibitors with a high affinity and selectivity for a certain functional protein (sub-)family.

5 Experimental Evaluation

5.1 Consensus Graphs

To examine the performance of our approach, the ENZYME classification database (Release 37) was used as an information source for the functional classification for proteins [4, 28]. A set of ten diverse and highly populated enzyme families was selected (cf. Table 2). For each of the families, graph alignments and consensus graphs were calculated using the algorithms described in Section 3. A single pairwise comparison of two graph descriptors could be performed within $1-2$ seconds on average. An all-against-all comparison of a set of 100 binding pockets can therefore be performed within 3 hours on a standard Linux computer.[5] The subsequent determination of the multiple alignment can be performed within a few minutes.

The generated consensus graphs were finally used as prototypes to classify all $113,718$ cavities of the entire Cavbase in a two-fold cross validation. The set of protein structures belonging to each of the ten families was split randomly into two independent subsets. The cavities corresponding to one subset were used to calculate a consensus graph which was assessed subsequently by classifying the entries of the other subset and the remaining Cavbase. As one protein structure can exhibit more than one putative binding pocket and not all of these pockets must be related to a protein function, the obtained classification results were mapped back on the protein structures. In other words, a protein was considered as a member of a functional class if at least one of its cavities was found to match with one of the respective classes.

Each graph descriptor in the dataset was aligned to each of the 10 consensus graphs and scored as described in Section 4.1. Each comparison of one consensus graph against all Cavbase entries took approximately 20 CPU-hours, depending on the size of the consensus graph. Note that always the complete consensus graphs were used which contained in several cases hundreds of nodes. Restricting the consensus graphs to nodes with a degree of conservation beyond 0.25 or 0.5, a significant size reduction of the consensus graph and

[5] P-IV 2.80 GHz, 1024 MB RAM, Debian Linux

thus of the computational complexity is achieved. The process can be easily parallelized as all calculations are independent. For each consensus graph and each entry of the dataset, a score s between 0 and the maximally possible score $s_{\max}$ for the respective consensus graph is obtained. To decide whether a certain graph descriptor falls into the functional class of the consensus graph, a threshold s_t has to be adjusted. If the obtained score exceeds beyond this threshold, the graph instance is classified as a member of the class, otherwise it is discarded as a non-member.

To demonstrate rigorously that the derived classifiers are indeed able to separate positive from negative graph instances, and an optimal threshold value can be assigned, ROC (Receiver Operating Characteristic) curves [9] are shown for each of the 10 consensus graphs in Figure 5. These curves depict the relative numbers of true and false positives that are found considering different threshold values. In some of the cases, it is possible to retrieve more than 90 percent of the true positives before the first false positive entry is detected. In all cases, the resulting curves indicate convincing separation of graph descriptors that correspond to members and non-members of the respective family. An established figure-of-merit for the discriminative power of a particular classifier is the area under the ROC curve [17]. While an optimal classifier would achieve a value of 1.0, the examined classifiers yielded on average a very large area of 0.91 (cf. Table 2).

In general, using the fully automatically derived consensus graphs, it was possible to identify most of the members of a protein family and related families. It has to be regarded that the relevance of the generated consensus graphs depends on the structures selected to calculate the initial alignment. Thus, it might be possible to improve the results using carefully selected and unbiased subsets of the representative structures. In cases of rather heterogeneous families, it might also be useful to calculate more than just one prototype to better represent the respective families, e.g. in the case of kinase structures corresponding to the active or inactive state.

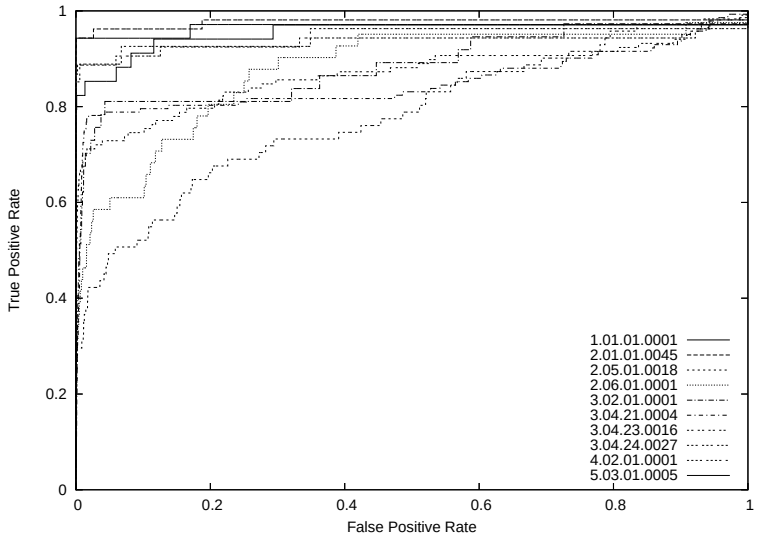


Figure 5: ROC (Receiver Operating Characteristic) curves of the 10 examined classifiers. The numbers in the legend correspond to the EC-numbers of the respective enzyme family (cf. Table 2). On the x-axis, the false positive ratio is plotted while the y-axis shows the true positive ratio, each depending on the selected threshold value. In the ideal case, a true positive ratio of 1.0 would correspond to a false positive ratio of 0.0. Yet in practice, a trade-off exists between both ratios as it is not possible to choose a threshold value such that all positive instances score above and all negative instances score below this threshold. The optimal threshold value is typically defined by the point on the ROC curve that is closest to the upper left corner. The area under the ROC curve is an established measure for the performance of a classifier and the respective values are shown in Table 2.

EC-Number	Family Name	No. of proteins	No. of cavities	AUC Full Dataset	C-V AUC Split 1	C-V AUC Split 2
1.01.01.0001	Alcohol dehydrogenase	80	398	0.946	0.954	0.948
2.01.01.0045	Thymidylate synthase	105	460	0.989	0.977	0.978
2.05.01.0018	Glutathione transferase	148	456	0.764	0.784	0.720
2.06.01.0001	Aspartate transaminase	90	494	0.952	0.880	0.934
3.02.01.0001	Alpha-amylase	87	232	0.842	0.889	0.825
3.04.21.0004	Trypsin	329	442	0.840	0.855	0.937
3.04.23.0016	HIV-1 retropepsin	267	898	0.894	0.871	0.898
3.04.24.0027	Thermolysin	59	70	0.954	0.948	0.901
4.02.02.0001	Carbonate dehydratase	207	363	0.952	0.937	0.901
5.03.01.0005	Xylose isomerase	76	500	0.959	0.967	0.894

Table 2: Results of the experiments with the 10 examined enzyme families. For each of the 10 families, the number of considered PDB entries and the number of hypothetical binding pockets in these protein structures is shown. Additionally, the area under the ROC curve (AUC) is shown as a measure for the performance of the obtained consensus graph classifiers. The AUC was determined for consensus pockets determined from the full set of cavities in a family and also in a two-fold cross validation to examine the influence of overfitting.

5.1.1 Discriminative Patterns

Trypsin and thrombin — two closely related proteins from the family of serine proteases [10] — were selected to examine the power of the graph alignment concept to detect patterns that discriminate both binding pockets. Thrombin is an important target for the development of anticoagulants, whereas trypsin is an enzyme involved in gastrointestinal digestion. To equip putative inhibitors of thrombin with the required selectivity it is important to trace the structural differences between trypsin and thrombin.

A dataset of 111 binding pockets extracted from thrombin structures and 105 trypsin pockets was composed. A multiple graph alignment was calculated for the graph representations of these binding pockets, and 216 "flat" feature vectors were derived for the dataset as described in Section 4.2. Each of the vectors contained 114 categorical attributes (including one class label attribute). Note that each of these attributes corresponds directly to a particular node of the represented graph if it is not flagged as a "dummy" node. The WEKA package [39] served to analyze the dataset.

The C4.5 algorithm [30] for decision tree induction was applied to the data set and resulted in a (pruned) decision tree that contained only one inner node. Obviously, it is sufficient to consider only one of the 113 attributes to classify all but one of the cases correctly. A similar run using the rule learning algorithm RIPPER [12] revealed the same result. Closer analysis of the discriminating feature revealed that the selected attribute corresponds to the aliphatic property of the alanine-190 residue, an amino acid that is part of the S1-subpocket of thrombin and that is replaced by serine in trypsin [14, 10]. The method was thus able to discover automatically a known structural difference among the two classes under consideration.[6] Yet, as there are more relevant structural differences between trypsin and thrombin, it was necessary to switch to another class of methods for data analysis. Simplicity of the generated model is a key objective of most machine learning approaches to reduce the risk of overfitting. Thus, if one attribute is sufficient to achieve a satisfactory classification, any further attributes are not included

[6] To examine also the predictive accuracy of the obtained model — although not a central objective of the method — a 10-fold cross validation experiment was performed and resulted in a classification rate of 99.537 %.

into the model. But, as the aim of our study is to reveal a comprehensive instead of a minimal model, the different "columns" of the alignment were correlated with the class membership vector. This was performed by constructing a vector for each "column" of the alignment that contained an entry for each "row" of the alignment. The entry was set to 1 if the respective entry of the alignment contained a non-dummy entry and to 0 otherwise. Among the 113 attributes, 10 showed a Pearson correlation coefficient above 0.7 with the class membership, three of them showing a correlation above 0.95. As expected, the attribute corresponding to alanine 190, already selected by the data mining algorithms mentioned above, showed the highest correlation ($r = 0.99$). Four of the remaining attributes correspond to residues of the 60-loop, another important structural difference between trypsin and thrombin. In the latter, it partially covers the specificity pocket; however it is entirely missing in trypsin. The method was thus also able to detect automatically two of the known differences among the considered classes [10]. A more detailed discussion of the obtained discriminative patterns is beyond the scope of this paper. In the future, we plan to examine different important and highly populated functional protein families and to study the structural differences among different subfamilies using the techniques described in this contribution.

6 Conclusions

Graph alignments were introduced as a novel concept for the analysis of protein active sites. Using graph alignments, it is possible to detect conserved patterns in binding pockets of proteins exhibiting a similar biochemical function. The method is very robust with respect to noise as typically given evaluating experimental data. Furthermore, conformational flexibility of the examined proteins adds some additional blur to the input data. Thus, it is not necessary that a pattern is fully present in all of the examined structures to be detected. Actually, it is not even required that the entire pattern is contained in full by one single structure as it can be composed by multiple occurrences in different structures. This is a major advantage compared to other existing approaches that attempt to

derive conserved regions in protein active sites.

Additionally, the graph alignment approach, presented in this paper, does not solely focus on conserved regions or similarities in the binding pockets. Instead, it also captures the dissimilarities among the examined structures. Therefore, it is also possible to examine differences across related protein families using our concept. It allows one to gain insights into selectivity determining features among the proteins, which is a prerequisite for the design of selective and safe drug molecules. The power to automatically detect structural differences is another important contribution of the method. To detect such discriminative structural elements in proteins, many established tools from data mining can be applied as "flat" feature vectors for the protein binding pockets under investigation can be derived and analyzed using standard techniques while maintaining the interpretability of the obtained results.

An important feature of the graph alignment approach is the fact that a deduced alignment mirrors the data of a given input set of graph descriptors. Accordingly, if new structures become available, one has to recalculate the graph alignment incorporating novel entries. With respect to general validity of rules derived from empirical data one might argue that this is a major drawback of the approach as the number of available protein structures grows exponentially. However, our algorithms can recalculate graph alignments incrementally in a very efficient way. Thus, the computational cost of adding new protein structures is comparably low. Nevertheless, our analysis is already based on a substantial subset of the PDB and should be representative. It was performed using standard Linux computers in an acceptable amount of time.

Another potential drawback of the graph alignment approach is that it suggests only one unique assignment across the nodes from different graphs. Possibly, other assignments exist that would yield the same or only a slightly smaller score. Accordingly, is our approach too restrictive and possibly neglects many possible assignments? We believe that this is not the case. Firstly, alignments are also computed in sequence analysis, and also there multiple alignments are possible for a given set of sequences. From an application point of view, one can assume that across a family only a unique mapping

of pseudocenters exists. This assumption is supported by the fact that in a binding site a particular function, e.g. an elementary step of a chemical reaction, occurs that requires with very small spatial variations a unique approach direction of the transformed reagents. This pattern has to be conserved across a large variety of binding pockets. Secondly, although typically a huge number of different close-to-optimal alignments exist for a given set of graph descriptors, these alignments usually show a high similarity and differ only by a few pseudocenter correspondences. Key residues of the binding pockets are always assigned to each other and ambiguities usually affect only pseudocenters of minor functional importance. Thus, it is reasonable to focus on one of these alignments that is optimal with respect to the scoring function. This holds in particular as the obtained results do not differ much from those suggested by alternative close-to-optimal alignments. To closer examine the robustness of the best-scored alignment, one could apply techniques similar to those used to calculate suboptimal sequence alignments [40].

[1] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and Lipman David J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, Sep 1997.

[2] Peter J. Artymiuk, Andrew R. Poirette, Helen M. Grindley, David W. Rice, and Peter Willet. A Graph-theoretic Approach to the Identification of Three-dimensional Patterns of Amino Acid Side-chains in Protein Structures. *Journal of Molecular Biology*, 243:327–344, 1994.

[3] Mikhail J. Atallah, editor. *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999.

[4] Amos Bairoch. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1):304–305, 2000.

[5] G. J. Bartlett, C. T. Porter, N. Borkakoti, and J. M. Thornton. Analysis of catalytic residues in enzyme active sites. *Journal of Molecular Biology*, 324:105–121, 2002.

[6] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L. Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The Pfam protein families database. *Nucl. Acids. Res.*, 32(90001):D138–141, 2004.

[7] Johannes Berg and Michael Lässig. Local graph alignment and motif search in biological networks. *Proc. Natl. Acad. Sci. USA*, 101(41):14689–14694, 2004.

[8] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[9] A. P. Bradley. ROC Curves and the Chi^2 Test. *Pattern Recognition Letters*, 17:287–294, 1996.

[10] Carl Branden and John Tooze. *Introduction to Protein Structure*. Garland Publishing, 1999.

[11] Coen Bron and Joep Kerbosch. Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 16(9):575–577, September 1973.

[12] William W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference (ML95)*, pages 115–123. Morgan Kaufmann Publishers, 1995.

[13] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[14] Frank Dullweber, Milton T. Stubbs, Dorde Musil, Jörg Stürzebecher, and Gerhard Klebe. Factorising ligand affinity: A combined thermodynamic and crystallographic study of trypsin and thrombin inhibition. *Journal of Molecular Biology*, 313:593–614, 2001.

[15] Helen M. Grindley, Peter J. Artymiuk, David W. Rice, and Peter Willet. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism Algorithm. *Journal of Molecular Biology*, 229:707–721, 1993.

[16] Thomas Hamelryck. Efficient identification of side-chain patterns using a multidimensional index tree. *Proteins*, 51(1):96–108, 2003.

[17] J. A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36, 1982.

[18] Andrew Harrison, Frances Pearl, Ian Sillitoe, Tim Slidel, Richard Mott, Janet Thornton, and Christine Orengo. Recognizing the fold of a protein structure. *Bioinformatics*, 19(14):1748–1759, 2003.

[19] Manfred Hendlich, Andreas Bergner, Judith Günther, and Gerhard Klebe. Relibase: Design and Development of a Database for Comprehensive Analysis of Protein-Ligand Interactions. *Journal of Molecular Biology*, 326:607–620, 2003.

[20] Manfred Hendlich, Friedrich Rippmann, and Gerhard Barnickel. LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15:359–363, 1997.

[21] Martin Jambon, Anne Imberty, Gilbert Deléage, and Christophe Geourjon. A New Bioinformatic Approach to Detect Common 3D Sites in Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 52:137–145, 2003.

[22] Petteri Jokinen and Esko Ukkonen. Two algorithms for approximate string matching in static texts. In *Proceedings of the 16th Symposium on Mathematical Foundations of Computer Science*, number 520 in LNCS. Springer, 1991.

[23] B. P. Kelley, Yuan B., F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. Pathblast: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32:W83–W88, 2004.

[24] G. J. Kleywegt. Recognition of spatial motifs in protein structures. *Journal of Molecular Biology*, 285:1887–1897, 1999.

[25] Ina Koch. Enumerating all connected common subgraphs in two graphs. *Theoretical Computer Science*, 250:1–30, 2001.

[26] Ina Koch, Thomas Lengauer, and Egon Wanke. An Algorithm for Finding Maximal Common Subtopologies in a Set of Protein Structures. *Journal of Computational Biology*, 3(2):289–206, 1996.

[27] Giorgio Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9:341–352, 1972.

[28] Nomenclature Committee of the IUBMB. *Enzyme Nomenclature.* Academic Press, 1992.

[29] Ruth Nussinov and Haim J. Wolfson. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proc. Natl. Acad. Sci. USA*, 88:10495–10499, December 1991.

[30] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[31] Stefan Schmitt, Daniel Kuhn, and Gerhard Klebe. A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology. *J. Mol. Biol.*, 323(2):387–406, 2002.

[32] Florence Servant, Catherine Bru, Sébastien Carrère, Emmanuel Courcelle, Jérôme Gouzy, David Peyruc, and Daniel Kahn. Prodom: Automated clustering of homologous domains. *Briefings in Bioinformatics*, 3(3):246–251, 2002.

[33] Jaoa Carlos Setubal and Joao Meidanis. *Introduction to Computational Molecular Biology.* International Thomson Publishing, 1997.

[34] Maxim Shatsky, Ruth Nussinov, and Haim J. Wolfson. MultiProt - A Multiple Protein Structural Alignment Algorithm. In R. Guigó and D. Gusfield, editors, *WABI 2002*, number 2452 in LNCS, pages 235–250. Springer-Verlag, 2002.

[35] A. Stark and R. B. Russel. Annotation in three dimensions. PINTS: Patterns in Non-homologous Tertiary Structures. *Nucleic Acids Research*, 31:3341–3344, 2003.

[36] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.