

Graph Alignment: Robust Data Mining on Structured Objects

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

July 15, 2004

Abstract

In bioinformatics and chemoinformatics, one is often concerned with the study of objects that have a complex and variable internal structure, e.g. protein structures or small organic molecules. The structure of such objects is frequently correlated with their relevant properties (e.g., biological function or toxicity), although the connection between structural features and properties is often complex. Graphs offer an appropriate formalism for modeling structured objects, as they allow a representation of structural features free from losses.

Therefore, we propose a method that allows one to examine relationships among the existence of structural patterns in graphs and their membership in certain classes. The method is designed to be very robust against errors and noise as these occur frequently

*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 Department of Mathematics and Computer Science, Philipps-University, Hans-Meerwein-Strasse, 35032 Marburg, Germany, E-mail: eyke@mathematik.uni-marburg.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

¶This is a significantly extended version of a paper that will be presented at the German Conference on Bioinformatics 2004 (GCB 2004).

in datasets derived from experiments. It is able to detect patterns that are characteristic of a given set or class of graphs and to infer patterns that are discriminative among different yet similar classes. The graphs are arranged in a *graph alignment*, a robust generalization of the concept of graph isomorphism. This allows for the simple detection of conserved patterns and also for the application of existing machine-learning techniques. The ease of interpretation of the obtained results is a key advantage of our approach.

Keywords: KDD, Structural Pattern Discovery, Fuzzy Patterns, Graph Mining, Biology, Drug Design

1 Introduction

Many interesting real-world objects have a complex internal structure that considerably influences their properties and behavior. This is particularly the case for those objects that are typically examined in the area of bioinformatics or chemoinformatics (i.e., protein structures or organic molecules).

Researchers in these areas are often interested in finding relationships between the occurrences of certain structural patterns and the presence of a certain property. Traditionally, such insights were obtained conducting carefully designed experiments. Nowadays, the availability of high-throughput methods makes it possible to examine a huge amount of compounds with comparably small effort [6]. Additionally, data on the three-dimensional structure of thousands of proteins and hundred-thousands of organic molecules is available [2, 5, 26].

This development makes the application of data mining-methods necessary and promising. Consequently, the field of graph mining [25] recently received increasing attention, especially since mapping structured objects onto “flat” feature vectors would lose at least part of the structural information. Additionally, “structural descriptors” (i.e., numbers that are calculated for an object to capture some of its internal structure [14]) are usually difficult

to interpret. Mapping results obtained in the data mining process back to the application domain thus becomes difficult.

In this paper, we propose a new method for data mining on structured objects, i.e. objects that can be modeled as a graph. More specifically, our method is able to extract both, characteristic patterns (i.e., subgraphs) for a set of objects that belong to a common class and discriminative patterns for sets of objects belonging to different classes. It does so by arranging the graphs in a common framework that allows for the simple detection of conserved patterns and also the fully automated generation of feature vectors. These feature vectors allow for the application of standard data mining techniques to structured objects and are easily interpretable at the same time. Additionally, the method is very robust against errors and inadequacies frequently present in experimentally determined structure data sets.

This paper is organized as follows: In section 2, we introduce some basic concepts and the problems examined in this paper. Section 3 contains a formal description of our new concept of graph alignment. In section 4, we present an algorithm for the efficient calculation of graph alignments and in section 5, methods for the analysis of graph alignments are discussed. Section 6 describes some experiments on synthetic and real-world datasets.

2 Preliminaries

We assume throughout this paper that we are given a set $\tilde{G} = \{\tilde{G}_1(\tilde{V}_1, \tilde{E}_1), \dots, \tilde{G}_n(\tilde{V}_n, \tilde{E}_n)\}$ of connected, node-labeled and edge-weighted graphs, each of them representing a structured object. Let $l(v) \in \mathbb{N}$ be the label for each node (vertex) $v \in \tilde{V}_i$ and $w(e) \in \mathbb{R}$ be the weight for each edge $e \in \tilde{E}_i$. Many different types of structured objects may be mapped onto such a graph representation as it is flexible and expressive. Typically, the nodes are used to represent certain elements of the structured objects (e.g., atoms or amino acids) while the edges are used to represent bonds or geometrical distance constraints. In the experimental section, we will present some possible mappings in more detail.

We further assume that we are given a class membership function $c : \tilde{G} \mapsto \mathbb{N}$ that assigns each graph to a certain class or group. Such a membership function may either be the result of background knowledge (i.e., one could split a set of molecules into classes that have a certain property and those that do not have that property) or the result of a previously performed cluster analysis. Either way, one expects objects belonging to the same class to be “similar” in some sense. The following interesting questions could be put forward: Which structural elements are characteristic for the members of a particular class or group? And: Which structural elements are discriminative among two different (yet similar) classes or groups (i.e., do subgraphs exist whose occurrence in a graph indicate that this graph belongs to one of the groups and not to the other one)?

First, we will focus on the former question and then see that the obtained approach also provides a mean to answer the latter. The usual way of identifying common patterns (i.e., subgraphs) in two or more graphs is the calculation of a graph isomorphism.

Definition 2.1 *Let $G = \{G_1(V_1, E_1), \dots, G_n(V_n, E_n)\} \subseteq \tilde{G}$ be a set of graphs. A $(n-)$ subgraph isomorphism is a set T of tuples $t = (v_1, v_2, \dots, v_n) \in V_1 \times V_2 \times \dots \times V_n$ such that*

1. *For all $t = (v_1, \dots, v_n), t' = (v'_1, \dots, v'_n) \in T$ with $t \neq t'$: $v_i \neq v'_i$ for $i = 1, \dots, n$, i.e. each node of each graph is contained in at most one tuple to guarantee that the mapping defined by the graph isomorphism is unambiguous.*
2. *For $t = (v_1, \dots, v_n) \in T$: $l(v_1) = \dots = l(v_n)$, i.e. all nodes that are mapped upon each other share the same label.*
3. *For all $t = (v_1, \dots, v_n), t' = (v'_1, \dots, v'_n) \in T$ with $t \neq t'$: If $(v_i, v'_i) \in E_i$ for an $i = 1, \dots, n$, then $(v_j, v'_j) \in E_j$ for all $j = 1, \dots, n$ and $|w(v_i, v'_i) - w(v_j, v'_j)| \leq \epsilon$ for an $\epsilon \in \mathbb{R}$, i.e. if two nodes are connected in one of the graphs, then the associated nodes in the other graphs should also be connected by an edge with a similar weight.*

The nodes v_i contained in a tuple $t = (v_1, \dots, v_n) \in T$ are called isomorphic. ◁

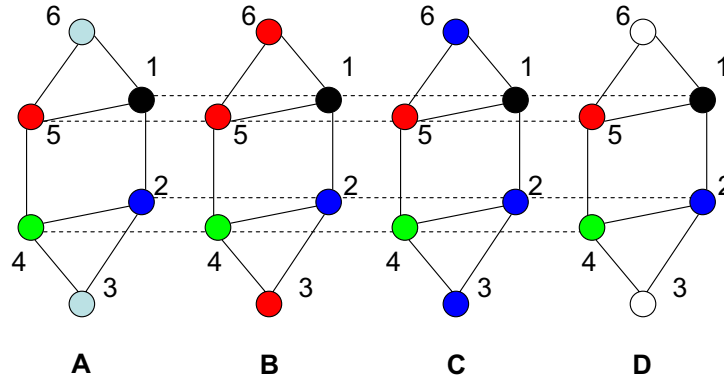


Figure 1: A (multiple) isomorphism among the labeled graphs A, B, C, D is indicated by the dashed lines. All four graphs contain the identical subgraph with the nodes 1, 2, 4, 5 while nodes 3 and 6 have different labels in all graphs.

Different strategies exist to construct such a graph isomorphism for a given set of graphs. Typically, one transforms this problem into a clique search problem by introducing an association graph [20]. This association graph contains as nodes all potential mappings of the nodes of the input graphs (i.e., all elements of $V_1 \times \dots \times V_n$ that satisfy criterion (2) of the definition above) and an edge for each pair of such mappings that are compatible with each other. Thus, a clique in this association graph corresponds directly to a maximal (sub-)graph isomorphism. Such a clique may be detected by standard clique detection algorithms [9].

3 Graph Alignments

In principle, the concept of graph isomorphism is independent of the number of graphs under consideration. Thus, one could simply take a subset $G \subseteq \tilde{G}$ such that $c(g) = c(g')$ for all $g, g' \in G$ and search for common subgraphs to identify conserved and characteristic patterns for the respective family of structured objects. In practice, such a concept is often not feasible, as graph isomorphisms are too strict for experimentally derived real-world datasets:

Example 3.1 *Figure 1 shows a sample of four graphs that contain an identical (i.e., iso-*

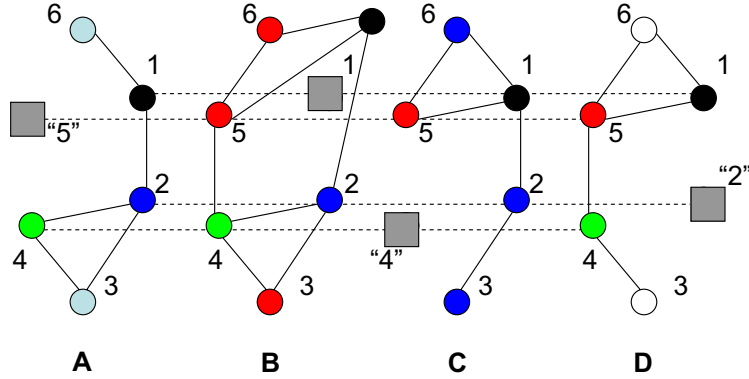


Figure 2: A perturbed version of the graphs in figure 1. No isomorphism exists among these graphs as each of the “conserved” nodes is missing (or is strongly shifted to another position) in one of the graphs. By inserting additional “dummy-nodes” (shown as boxes in dark gray) it is still possible to identify the (partially) conserved pattern.

morphic) subgraph. This “conserved” subgraph may be detected by different well-established standard techniques. Figure 2 shows instead a perturbed version of this graph set as it is more likely to occur in real-life data sets. In this paper, we consider the deletion of nodes or the changing of node-labels and edge-weights as possible perturbations. Standard approaches are highly susceptible to these kinds of changes. For example, in Figure 1, all nodes with the number 5 show mutual correspondence. Yet in Figure 2, the node labeled number 5 is missing in graph A. Therefore, also the correspondence among the remaining three nodes is irretrievable as no matching node exists in graph A. On the other hand, as each of the four nodes 1, 2, 4, 5 is significantly conserved across the series (i.e., in three out of four graphs), one would appreciate to detect the (partially) conserved pattern defined by these nodes. Through introduction of dummy nodes (indicated as squares in dark gray; see below for details), it is possible to perceive a conserved pattern. ◁

Definition 2.1 implies that a node can only be included in the isomorphism T only if it is contained in *each* of the graphs G_1, \dots, G_n . For large values of n , the presence of slight perturbations thus leads to very small or even empty isomorphisms. A simple consideration shows the relevance of this problem: Assume that p is the (small) probability that a particular

conserved node is perturbed in one particular graph, n is the number of conserved nodes and t is the number of graphs. Then — assuming independence of the perturbations — the probability that the complete conserved subgraph will be discovered is $(1 - p)^{n \cdot t}$. For $p = 0.01$ and $n = t = 10$, one obtains a probability of 0.37 (0.13 for $t = 20$, 0.05 for $t = 30$). More generally, the expected size of the actually conserved subgraph is only $(1 - p)^t \cdot 100\%$ of the complete subgraph. In the experimental section, we will examine the practical relevance of the effect using simulations under more realistic assumptions.

Thus, even if the graphs share a significant amount of similarity and contain highly conserved nodes, it is likely that for each conserved node, at least one graph G_i is existing such that the node is missing in this particular graph.

One possible solution for this problem would be to consider only a subset of graphs such that graphs that do not contain important and highly conserved nodes are excluded. This leads to the problem of frequent subgraph mining, for which a number of approaches have been described [17, 19, 25, 28, 29]. These are typically extensions of the apriori-algorithm for frequent subset mining [1] and operate similarly. Yet, such approaches are computationally quite demanding. In particular, for large structured objects (e.g., for protein structures), this can lead to inoperable run times and memory requirements. Additionally, considering only a subset of the respective dataset does not really solve the problem, as it tends to disassemble weakly conserved patterns into a large number of overlapping, yet strictly conserved patterns. E.g., in the case of example 3.1, considering only a subset of the four graphs would allow one to identify a number of conserved patterns with at most two nodes instead of one single, weakly conserved pattern including all four nodes. (See Table 1).

To overcome this problem, we suggest the concept of graph alignment¹: The central idea is to allow the insertion of additional nodes — named “dummy nodes” — into the analyzed graphs. If a conserved node is “missing” in one particular graph, a dummy node is introduced

¹The term “Graph Alignment” was introduced independently by Berg and Lässig [4]. The authors understand by “Graph Alignment” a related, yet different concept that does, e.g., not allow for the introduction of dummy nodes.

Subset of graphs	Conserved pattern
$\{A, B\}$	$\{2, 4\}$
$\{A, C\}$	$\{1, 2\}$
$\{B, D\}$	$\{4, 5\}$
$\{C, D\}$	$\{1, 5\}$

Table 1: “Frequent subgraphs” with a minimum support of 2 that can be detected in the graphs of Figure 2. The actually conserved pattern is “split” into a number of small, overlapping patterns in the presence of errors and noise.

to allow for the construction of a valid mapping among the conserved nodes (see dummy nodes shown as gray squares in Figure 2). Thus, the method (implicitly) edits the input data in cases of likely perturbations.

Definition 3.1 *Let $G = \{G_1(V_1, E_1), \dots, G_n(V_n, E_n)\}$ be a set of graphs. A $(n-)$ graph alignment is a set T of tuples $t = (v_1, v_2, \dots, v_n) \in (V_1 \cup \{\square\}) \times (V_2 \cup \{\square\}) \times \dots \times (V_n \cup \{\square\})$ such that*

1. *For all $t = (v_1, \dots, v_n), t' = (v'_1, \dots, v'_n) \in T$ with $t \neq t'$: If $v_i \neq \square$, then $v_i \neq v'_i$ for $i = 1, \dots, n$, i.e. each non-dummy node of each graph is contained in at most one tuple such that the mapping defined by the graph alignment is unambiguous.*
2. *For $t = (v_1, \dots, v_n) \in T$: $l(v_1) = \dots = l(v_n)$ for all $v_i \neq \square$, i.e. all non-dummy nodes that are mapped onto each other share the same label.*
3. *For all $t = (v_1, \dots, v_n), t' = (v'_1, \dots, v'_n) \in T$ with $t \neq t'$: If $v_i, v'_i \neq \square$ and $(v_i, v'_i) \in E_i$ for an $i = 1, \dots, n$, then $(v_j, v'_j) \in E_j$ for all $j = 1, \dots, n$ with $v_j, v'_j \neq \square$ and $|w(v_i, v'_i) - w(v_j, v'_j)| \leq \epsilon$ for an $\epsilon \in \mathbb{R}$, i.e. if two non-dummy nodes are connected in one of the graphs, then the associated non-dummy nodes in the other graphs should also be connected by an edge with a similar weight.*

We say that nodes v_i contained in a tuple $t = (v_1, \dots, v_n) \in T$ are aligned. ◁

The additional degree of freedom introduced by this extension has a number of consequences. First, it is possible to align two graphs such that all of their nodes are included in

the alignment. Thus, it is not required to distinguish between graph alignments and subgraph alignments. Second, the additional freedom can lead to ambiguities as many different alignments possibly exist for a given set of graphs. Therefore, it is necessary to consider a scoring scheme to distinguish useful from inadequate alignments. Intuitively, dummy nodes should only be inserted if no other valid matching partner can be found. Therefore, the insertion of dummy nodes should be penalized. A very simple (exemplary) scoring system is defined by

$$s(T) = \sum_{t \in T} \sum_{t=(t_1, \dots, t_n)} \begin{cases} 0 & t_i = \square \\ 1 & t_i \neq \square \end{cases} \quad (1)$$

It is of course possible to apply more sophisticated scoring schemes, which may assign different penalties to different dummy nodes and also consider the edges of the different graphs. In principle, one could extend the standard methods for the detection of graph isomorphisms based on clique-detection in association graphs to calculate optimal graph alignments [9, 20]. These require the scoring system to be additive and monotonic in size of T .

Yet, as the maximum clique problem is computationally very complex, such an approach is not feasible for more than a few graphs [11]. This is particularly true as the additional degree of freedom introduced by graph alignments also leads to an increasing number of possibilities and thus to a huge combinatorial explosion. In the next section, we therefore suggest a simple heuristic algorithm for the efficient calculation of graph alignments. These are — as the idea for graph alignments itself — stimulated by concepts from multiple sequence analysis in bioinformatics.

4 Algorithms

In the case of $n = 2$, i.e. for pairwise comparisons, the difference between graph isomorphisms and graph alignments is small. In particular, a generic extension of (maximal) pairwise subgraph isomorphisms to pairwise alignments exists that is optimal with respect to additive

scoring systems such as (1): Let $G = \{G_1(V_1, E_1), G_2(V_2, E_2)\}$ be a set of two graphs and let T be a maximal subgraph isomorphism for G_1 and G_2 . Let further $\bar{V}_1 \subseteq V_1, \bar{V}_2 \subseteq V_2$ be the sets of nodes that are included in T . Then, T can be extended to a graph alignment \tilde{T} as follows

$$\tilde{T} := T \cup \{(v_1, \square) | v_1 \in V_1 \setminus \bar{V}_1\} \cup \{(\square, v_2) | v_2 \in V_2 \setminus \bar{V}_2\} \quad (2)$$

This extension simply assigns all nodes of both graphs that are not part of the isomorphism to a dummy node in the other graph.

Therefore, the calculation of optimal pairwise graph alignments virtually reduces to the calculation of pairwise maximal subgraph isomorphisms. To handle this problem, the standard approaches for the detection of graph isomorphisms are applicable in most cases [9, 20]. Thus, an efficient way has to be developed to calculate multiple graph alignments from pairwise graph alignments. In bioinformatics sequence analysis, star alignments and tree alignments are frequently applied schemes to calculate multiple sequence alignments [24]. We therefore propose to use similar schemes for the calculation of multiple graph alignments.

```

input : Alignments  $T_1, T_2$ , both containing  $G_i(V_i, E_i)$ 
output: Merged Alignment from  $T_1$  and  $T_2$ 
foreach  $t = (t.1, \dots, t.n) \in T_2$  do
  if  $\exists t' = (t'.1, \dots, t'.m) \in T_1$  s.t.  $t.i = t'.i$  then
    replace  $t' \in T_1$  by  $t'' = (t'.1, \dots, t'.m, t.1, \dots, t.n)$ 
  else
    insert  $t'' = (\square, \dots, \square, t.1, \dots, t.n) \in T_1$ 
  end
end
foreach  $t' = (t'.1, \dots, t'.n) \in T_1$  do
  if  $\nexists t = (t.1, \dots, t.n) \in T_2$  s.t.  $t.i = t'.i$  then
    insert  $t'' = (t'.1, \dots, t'.n, \square, \dots, \square) \in T_1$ 
  end
end
return  $T_1$ 

```

Algorithm 1: Algorithm for the star-like merging of graph alignments using a fixed reference graph G_i as center.

Let $G = \{G_1, \dots, G_n\}$ be a set of graphs and let $T_1(G_1, G_2), \dots, T_{n-1}(G_1, G_n)$ be a set of pairwise alignments of G_1 to G_2, \dots, G_n . Then, Algorithm 1 can be used to iteratively merge

two alignments in a star-like manner. The algorithm uses the nodes of the “central” graph G_1 as pivot elements to join two alignments. If a pivot element is not considered in one of the alignments, the respective positions are filled by dummy nodes. The resulting alignment is not necessarily optimal with respect to the scoring scheme, but it can be calculated very efficiently. In the experimental section, we show that alignments calculated by Algorithm 1 are often sufficiently reliable in practice. More sophisticated algorithms do exist, but a detailed analysis of their performance is beyond the scope of this paper and will be discussed elsewhere.

The quality of star alignments strongly depends on the initially selected central graph. Therefore, we calculate n alignments for the graphs G_1, \dots, G_n , each with a different center graph G_i for $i = 1, \dots, n$. Then, we score each of the resulting alignments using (1) and select the alignment with the best score.

5 Analysis of Graph Alignments

5.1 Consensus Graphs

Once a set of graphs is arranged in a graph alignment, it is easily possible to identify patterns that are (weakly) conserved across most of the graphs. As there is a unique assignment among the nodes present in different graphs, it is easy to determine whether and how well a certain node is conserved across the dataset. This allows one to calculate a consensus graph $G'(V', E')$ for a graph alignment T : For each tuple $t = (t_1, \dots, t_n) \in T$, a node v' is introduced into V' and marked by the label $l(v') = l(t_i)$ for a $t_i \neq \square$. This labeling is unique as all nodes in t are assigned to the same label by definition. Additionally, each node $v' \in V'$ is attributed to an additional value $con(v') \in [0, 1]$ that indicates, how well v' is conserved across the graphs in G and thus, to which extent it is representative for G . Many definitions

for $con(\cdot)$ are possible, yet the straight forward definition is

$$con(v') = \frac{\sum_{i=1}^n s(t_i)}{n}, \quad (3)$$

where

$$s(t_i) = \begin{cases} 0 & t_i = \square \\ 1 & t_i \neq \square \end{cases} \quad (4)$$

I.e., $con(v')$ is the relative frequency of the non-dummy nodes in the tuple t_i . To reduce complexity, it is possible to prune nodes with a degree of conservation beyond a predefined threshold². If a threshold of 1.0 is used, the obtained consensus graph has to be completely contained in all of the n graphs under consideration. It therefore corresponds to a graph isomorphism across the graphs in G , which shows that graph alignments are indeed a generalization of graph isomorphisms.

The edges of G' are constructed similarly: If two tuples $t = (t_1, \dots, t_n), t' = (t'_1, \dots, t'_n) \in T$ contain nodes t_i, t'_i such that $(t_i, t'_i) \in V_i$, then the nodes $v, v' \in V'$ corresponding to t, t' are also connected by an edge $e' = (v, v') \in E'$. The edge weight $w(e')$ is defined by

$$w(e') = \frac{1}{k} \cdot \sum_{j=1}^k w(t_{i_j}, t'_{i_j}), \quad (5)$$

where $i_j \in \mathbb{N}, j = 1, \dots, k$ are the indices of the nodes t_{i_j}, t'_{i_j} that are connected by an edge $(t_{i_j}, t'_{i_j}) \in E_{i_j}$ (i.e. $w(e')$ is the average of the weights of the respective edges in the aligned graphs of G). Additionally, it appears useful to store the variance around this average as it supports an accurate scoring while retrieving the occurrences of the consensus pattern. v' is also attributed a number $con(v') \in [0, 1]$ indicating its degree of conservation.

Example 5.1 *In case of Figure 2, a consensus graph of size 4 is obtained when a pruning threshold of 0.25. The consensus contains the nodes 1, 2, 4 and 5, each of them with a 0.75*

²We are typically using a threshold of 0.25.

degree of conservation. The respective edges among these nodes are also contained in the consensus, each of them with a degree of conservation of 0.5. Note that the edges are always less well conserved compared to the corresponding nodes as an edge will generally be lost as soon as only one of its adjacent nodes is missing. \triangleleft

5.2 Classification based on Prototypes

The consensus graph retrieved by the described procedure can be seen as a prototype for the set of graphs under consideration, as it comprises all structural elements shared in common by most of the graphs. Thus, it suggests not only a characterization of the members of a class or cluster that can easily be interpreted by domain experts, but additionally, it serves as a mean for classification. A graph \bar{G} can easily be analyzed for occurrences of the consensus graph G' . Certainly, one should not request \bar{G} to contain the complete consensus graph G' as this would make the classification again highly susceptible to errors or background noise in \bar{G} . Instead, the presence or absence of the conserved structural elements in G' should be scored in terms of their respective degree of conservation. If a highly conserved node or edge is missing in \bar{G} , this should be penalized higher than the absence of a less-conserved element.

To achieve a robust search for occurrences of consensus patterns, we again apply the concept of graph alignment. Given the consensus graph G' and the query graph \bar{G} , it is possible to calculate a pairwise graph alignment T using the methods described in section 4. As already noted above, it is always possible to calculate such an alignment — even if \bar{G} does not contain a single conserved node from G' . It is therefore necessary to score the obtained alignment T to assess how well \bar{G} fits into the class represented by G' . This can be done with an (exemplary) modified version of the scoring system (1):

$$s(T) = \sum_{(t', \bar{t}) \in T, t' \neq \square} \begin{cases} 0 & \bar{t} = \square \\ con(t') & \bar{t} \neq \square \end{cases} \quad (6)$$

Depending on the size of the parameter ϵ that is selected in the calculation of the graph alignment (cf. Definition 3.1), it can also be appropriate to consider the deviation of the edge weights from the average weight stored for each of the edges in G' in the scoring (relative to the observed standard deviation, see the scoring system used for the experiments in section 6.2 as example).

The score defined by (6) matches its maximum when the consensus graph G' is contained completely in \bar{G} at a value of

$$s_{\max} = \sum_{(t', \bar{t}) \in T, t' \neq \square} \text{con}(t')$$

Thus, by scaling the obtained classification scores with a factor of $1/s_{\max}$, normalized scores in the interval $[0, 1]$ are obtained. These can then be interpreted as fuzzy degrees of membership that indicate to which extend the graph \bar{G} belongs to the set of objects represented by G' .

If one is interested in a crisp (i.e., non-fuzzy) classification, it is possible to select a certain membership threshold in a way that all objects scoring beyond this threshold are classified into respective classes. In practice, if a large number of objects has to be classified simultaneously, it is often possible to select this threshold by analyzing the distribution of the obtained classification scores. Alternatively, if the graph \bar{G} is compared against multiple consensus graphs, it is also possible to perform a nearest-neighbor classification by assigning \bar{G} to the class with the best (or, most significant if one is able to estimate the distribution of the scores) classification score.

5.3 Calculation of Discriminative Patterns

So far, we were only concerned with the first question addressed in section 2, i.e. with respect to the detection of patterns that are characteristic of a particular selection of structured objects. In the remainder of this section, we will show that the concept of graph alignment can also give access to the detection of discriminating patterns among different, yet similar

classes of structured objects.

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

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

without loss of generality. Then, it is possible to analyze the influence of the common structural elements of the graphs in G by correlating their presence or absence with the class membership. To do this, one derives a set \bar{T} such that for each $t = (t_1, \dots, t_n) \in T$ exists a $\bar{t} = (\bar{t}_1, \dots, \bar{t}_n) \in \bar{T}$ with

$$\bar{t}_i = \begin{cases} 0 & t_i = \square \\ 1 & t_i \neq \square \end{cases} \quad (7)$$

I.e., each tuple $t \in T$ corresponds to a particular conserved node, whereas an entry of 1 indicates that the node is present in the respective graph and an entry of 0 indicates that the node is a dummy node in the respective graph and thus not present there. For each tuple $\bar{t} \in \bar{T}$, one can calculate the standard Pearson correlation coefficient with the class membership vector $\tilde{t} = (c(G_1), \dots, c(G_n))$. Tuples with a high absolute correlation coefficient represent structural elements (i.e., nodes) of the graphs in G that are highly correlated with class membership and thus discriminate between the classes. Often, such discriminative patterns are of utmost interest for domain experts.

Example 5.2 *In the case of Figure 2, one obtains a graph alignment T of size 4 (i.e., corresponding to the nodes 1, 2, 4 and 5), if entries with a degree of conservation below 0.5*

are pruned. The set \bar{T} has the form

$$\bar{T} = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

Assuming the class membership $c(A) = c(B) = c(C) = 0$ and $c(D) = 1$ for the graphs A, B, C, D and thus a class membership vector of $\tilde{t} = (0, 0, 0, 1)$, one obtains a Pearson correlation of -1 for the vector corresponding to the consensus node 2. For all other nodes, the correlation amounts to 0.33. Therefore, node 2 of the consensus is perfectly anti-correlated with the class membership of the graphs and thus qualifies as a discriminating element. If the node 2 is present in a particular graph, the latter very likely belongs into class 0. If it is absent, one would expect the graph to be a member of class 1. \triangleleft

Unfortunately, this simple correlation-based method does not generalize well to multi-class problems as typically no linear ordering exists on the class labels. For the same reason, it is not possible to distinguish different types of node labels. A possible solution would be the use of a one-versus-all (or all-versus-all) classification scheme.

Yet, it is possible to derive a flat feature vector representation for the graphs in G from T that allows one to apply many standard techniques from machine learning and data mining. Assuming an arbitrary fixed ordering on the elements $t_1, \dots, t_n \in T$, one is able to derive a flat feature vector \bar{g}_i for all $G_i \in G$:

$$\bar{g}_i = (\tilde{l}(t_{1.i}), \dots, \tilde{l}(t_{n.i})), \quad (8)$$

where

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

and $t_j.i$ denotes the i -th position of the vector t_j . \bar{g}_i thus contains all entries of all tuples in the graph alignment T that correspond to the graph G_i . It is a “flat” representation of the structured object represented by G_i , that has a direct structural interpretation as each entry of \bar{g}_i corresponds to a particular structural feature (i.e., node).

Example 5.3 *Again, in the case of Figure 2, one obtains the feature vectors*

$$\begin{aligned} g_A &= (\text{black}, \text{blue}, \text{green}, \square), \\ g_B &= (\square, \text{blue}, \text{green}, \text{red}), \\ g_C &= (\text{black}, \text{blue}, \square, \text{red}), \\ g_D &= (\text{black}, \square, \text{green}, \text{red}) \end{aligned}$$

for the graphs A, B, C and D . Given the class memberships $c(A) = c(B) = c(C) = 0$ and $c(D) = 1$, one could infer the decision rule:

$$g_i \text{ belongs to class } \begin{cases} 0 & g_{i.2} \neq \square \\ 1 & g_{i.2} = \square \end{cases},$$

which indicates again the importance of the consensus node 2 for the class membership. \triangleleft

Although many methods for data analysis can be applied to the described feature vector representation, the use of methods that infer easily interpretable models (e.g., decision trees or decision rules) is reasonable as the interpretability is a key advantage of our approach.

6 Experiments

6.1 Synthetic Data

We performed Monte Carlo simulations with random graphs (using the Erdős-Rényi model assumptions [13]) to examine the different results of methods based on graph isomorphism

and graph alignments under controlled conditions. First, we wanted to examine the robustness of the two approaches against random errors in the input data.

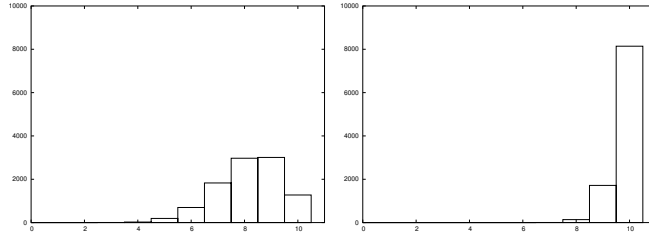
Therefore, we repeatedly generated a random graph with a fixed size of 10 nodes. Among each pair of nodes, an edge was introduced by chance with a fixed probability of 0.25. The nodes were attributed randomly with one of 5 categorial labels and the edge weights were selected uniformly from the interval $[0, 12]$. We used an ϵ -threshold of 2.0 for our calculations³. From each such graph, we generated 10 identical copies (i.e., representing perfect structural conservation) and performed random mutations in the copies to simulate error and noise. For each node of the graph, we changed the node label with a certain low probability P_{node} .

Subsequently, edges were inserted or deleted: A pair of nodes was selected for mutation with a certain small probability P_{edge} . The selected nodes were then connected by an edge with a fixed probability of 0.25, removing previously present edges before (if any). For the newly added edges, a new label was generated.

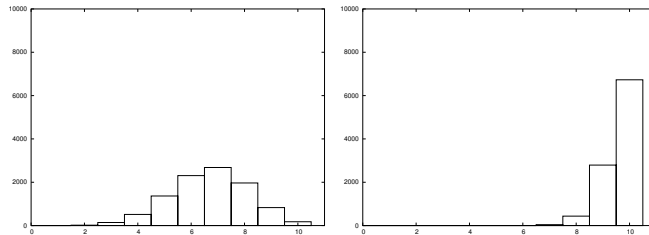
The ten input graphs generated by this procedure were used to calculate a maximal subgraph isomorphism (i.e., to detect the maximal subgraph that was contained in all of the ten graphs) and the size of this isomorphism was stored. Additionally, a graph alignment was calculated for the ten graphs using algorithm 1 and the number of nodes conserved in at least 5 of the 10 graphs is stored. The experiment was repeated 10.000 times.

Figure 3 shows the frequencies of the obtained values for different values of P_{node} and P_{edge} . On the left, the results for graph isomorphisms are shown, the results for the graph alignments are given on the right. Ideally, one would expect that all 10.000 experiments yielded a conserved pattern of size 10 as the considered input graphs are actually identical apart from the introduced errors. As apparent, the graph isomorphisms were highly affected by introduced errors as they are able to recover only 8.1632 ((a), (b): 6.6908, (c): 5.4748) conserved nodes on average. The graph alignments detected instead 9.7994 ((a), (b): 9.6208,

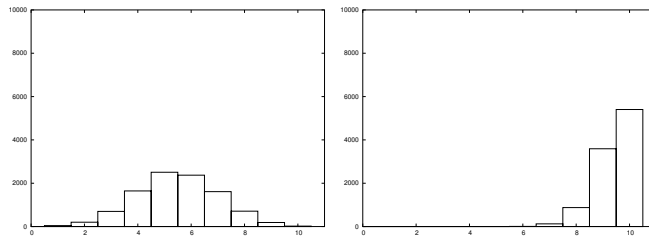
³These parameters have been chosen to match the properties of the dataset examined in the real-world data section. Qualitatively similar results were obtained for varied sets of parameters.



(a) $P_{\text{node}} = P_{\text{edge}} = 0.01$



(b) $P_{\text{node}} = P_{\text{edge}} = 0.02$



(c) $P_{\text{node}} = P_{\text{edge}} = 0.03$

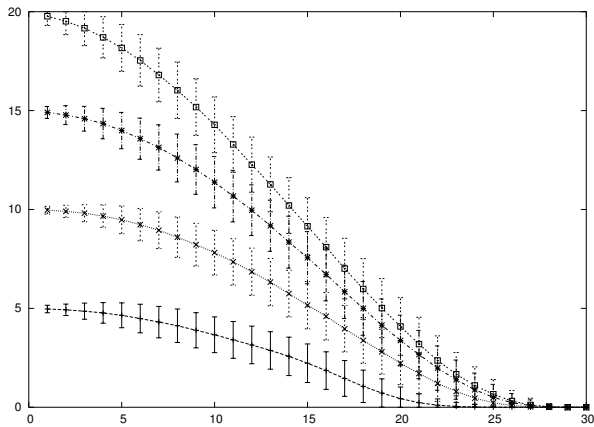
Figure 3: Results of Monte Carlo simulations with random graphs, performed to study the robustness of the approach against random errors. Results for graph isomorphisms are shown on left, while results for graph alignments are shown on the right.

(c): 9.4254) conserved nodes on average. As algorithm 1 uses a very simple heuristic for the calculation of graph alignments, one can expect more sophisticated methods to calculate better graph alignments and thus to recover even more conserved nodes. Instead, the results for the graph isomorphisms appear to be independent from the used algorithm as they are the consequence of fundamental and conceptual limitations.

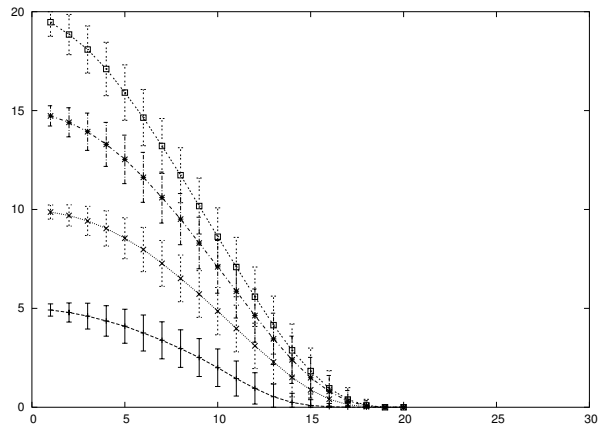
Conversely, graph alignments introduce additional degrees of freedom that could possibly retrieve conserved “patterns” that are obvious artifacts as they occur by chance in some of the examined graphs. I.e., graph alignments reduce the probability of false negatives (β -error), but they increase the probability of false positives (α -error). To examine this effect, we performed additional simulations. Accordingly, ten input graphs were generated independently using the same procedure as described above. Ideally, no conserved patterns can be expected as the generated graphs have no common origin. Figure 4 shows the results of the simulations for different graph sizes and amounts. For each possible minimum support threshold, the average size of the detected conserved pattern and the standard deviation from the average are plotted.

As expected, if a rather small support threshold is used, the detected conserved patterns typically cover most of the input graphs. Thus, one has to carefully interpret “patterns” that are detected at low degrees of conservation. Yet as the support threshold grows, the average pattern size rapidly decreases. Therefore, if a large pattern is detected that has a rather high degree of conservation, it unlikely results by chance correlation. Additionally, the performed simulations allow a quantitative estimation of the significance of a detected pattern. E.g., when comparing a set of 30 graphs of size 20, a pattern of size 15 with a minimum support threshold of 0.5 (i.e., 15 graphs in absolute numbers) achieves a statistical p -value of $2.5 \cdot 10^{-5}$ if a normal distribution of the pattern sizes is assumed⁴. It can thus be considered significant.

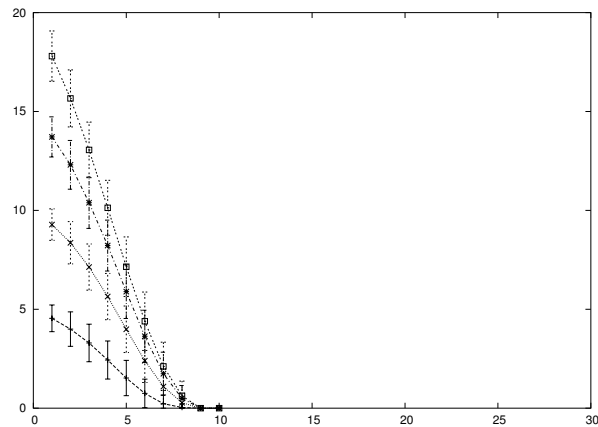
⁴In this particular case, the assumption of a normal distribution is justified as the respective histogram clearly shows a bell-shaped distribution and can be approximated very well by a normal distribution with the observed mean and standard deviation. For other choices of the minimum support threshold, the distributions become more skewed.



(a) 30 graphs of sizes 5, 10, 15 and 20



(b) 20 graphs of sizes 5, 10, 15 and 20



(c) 10 graphs of sizes 5, 10, 15 and 20

Figure 4: Results of Monte Carlo simulations with random graphs, performed to study the susceptibility of the approach against false-positive matches. The curves show the average size of the detected conserved pattern for different minimum support thresholds (in absolute numbers on the x-axis). The error-bars indicate the standard deviation around this average.

6.2 Real-World Data

6.2.1 Motivation

The prime motivation for the present study comes from the area of structural bioinformatics, a branch of bioinformatics that deals with the study of the three-dimensional structure of biomolecules [7]. Proteins are of particular interest as they show a diverse architecture, directly correlated to their biochemical function [8]. They interact specifically with other molecules, such as substrates, agonists, antagonists or allosteric modulators. The latter ones are recognized in binding sites accessible from the protein surface and they are complementary in their geometrical and physicochemical properties. The shape of such a binding pocket is therefore intimately related to the function of the respective protein. Additionally, binding pockets are of interest for pharmaceutical research as the strong binding to the active site of a protein is a frequent mechanism of action for drugs [18].

Cavbase [15, 23] is a database system that allows the fully-automated detection and extraction of protein binding pockets from experimentally determined protein structures available through the database PDB [5].

In Cavbase, protein active sites are described by graphs in a first-order approximation. The geometrical structure and the physicochemical properties of a binding pocket are represented by pre-defined pseudocenters — spatial points that represent the center of a particular property. These centers form the nodes of the graph representation and their properties are modeled in terms of node labels. Two centers are connected by an edge in the graph representation, if their Euclidean distance is below 12.0 \AA and each edge is labeled with the respective distance. The edges of the graph thus represent geometrical constraints among certain properties. Cavbase currently contains 80.617 binding pockets that have been extracted from 22.758 available protein structures. Not all of these entries represent catalytic sites. Some of them are non-catalytic clefts or depressions on the surface of enzymes to which no clear biochemical function can be assigned. Cavbase uses a rather susceptible technique [16] for

the detection of binding pockets. On average, the graph representations have approximately 85 nodes, however also graphs with several hundred nodes are frequently found and extremes with thousands of nodes do exist. The graphs are rather dense as approximately 20 percent of all pairs of nodes are connected by an edge.

Cavbase is thus a promising case for the application of graph mining techniques. Yet, it imposes a number of challenges: First of all, it contains a huge number of comparably large and dense graphs. This impedes the application of many existing approaches for graph mining as they would require impractical amounts of memory and computing time. Secondly, as proteins exhibit a certain amount of flexibility and experimentally determined structures are error-prone, one cannot expect to find identical occurrences of a conserved pattern. Instead, one has to consider also partial matches in the different structures to characterize a particular pattern in protein families.

6.2.2 Characteristic Patterns

To examine the performance of our approach, we used the ENZYME classification database as an existing source for information on the functional classification for proteins [3, 21]. A set of ten diverse and highly populated enzyme families was selected (cf. Table 2). For each of the families, a number of representative pockets was selected and used to calculate a multiple graph alignment (using algorithm 1). As already mentioned in section 4, all entries were tested as centers and the alignment with the best score (according to scoring scheme (1)) was chosen.

From the calculated alignment, a consensus graph was generated for each of the protein families. A single pairwise comparison of two graph representations can usually be performed in less than a second if some heuristic strategies are used [23]. Thus, an all-against-all comparison of a set of 100 binding pockets can be performed within 2 hours on a standard Linux computer. The subsequent calculation of the multiple alignment can be performed within a few minutes.

The generated consensus graphs were then used as prototypes to classify all 80.617 entries of the dataset. We used the scoring function

$$s(T) = \sum_{\substack{\tilde{t}_1 = (t'_1, t_1) \in T, \\ \tilde{t}_2 = (t'_2, t_2) \in T, \\ t'_1, t'_2 \neq \square}} \begin{cases} 0 & \text{otherwise} \\ score(\tilde{t}_1, \tilde{t}_2) & t_1, t_2 \neq \square \end{cases},$$

where

$$score(\tilde{t}_1, \tilde{t}_2) = sig(con(\tilde{t}_1)) \cdot fit(\tilde{t}_1, \tilde{t}_2).$$

$sig(\cdot)$ is the sigmoidal function

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

If T is the graph alignment of a consensus graph with one of the entries of the database, then $s(T)$ sums over all pairs of edges in these two graphs that are mapped upon each other in T . For each such pair, a similarity score from the interval $[0, 1]$ is added that is a product of two factors: The first factor depends on the observed degree of conservation of the edge in the consensus graph that is denoted by $con(\tilde{t}_1)$. A sigmoidal function, $sig(\cdot)$, is applied to this value to increase the weight of highly conserved edges and to reduce the weight of slightly conserved, unreliable edges. It thus has a value of 1.0, if the edge is perfectly conserved in all of the aligned graphs and a value of 0.0, if it is missing in all of the graphs. The second factor, $fit(\cdot, \cdot)$, is a function that determines how well the observed distance among two centers fits into the distribution of distances that was observed when the consensus graph was calculated:

$$fit(\tilde{t}_1, \tilde{t}_2) = 1 - 2 \cdot (\Phi_{0,d(t'_1, t'_2)}(x) - 0.5),$$

where

$$x = |l(t_1, t_2) - a(t'_1, t'_2)|$$

EC-Number	Family Name	Max. Score	Proteins	Cavities	Threshold	Hits	Recovered	Precision	Recall
1.01.01.0001	Alcohol dehydrogenase	2117.15	62	338	55.1	466	54	11.59 %	87.10 %
2.01.01.0045	Thymidylate synthase	1244.90	99	390	45.4	209	95	45.45 %	95.96 %
2.05.01.0018	Glutathione synthase	5771.79	122	388	72.5	110	93	84.54 %	76.22 %
2.06.01.0001	Asp. aminotransferase	1050.52	83	549	41.4	204	76	37.25 %	91.57 %
2.07.01.0112	Protein-tyrosine kinase	609.15	176	486	23.8	306	42	13.73 %	23.86 %
2.07.07.0049	Reverse transcriptase	1500.50	190	846	58.5	234	63	26.92 %	33.16 %
3.04.21.0004	Trypsin	1050.30	263	420	48.4	319	100	31.35 %	38.02 %
3.04.23.0016	HIV protease	1527.54	267	886	59.4	209	151	72.25 %	56.55 %
3.05.02.0006	Beta-lactamase	1308.55	143	413	42.2	193	102	52.85 %	71.33 %
4.02.01.0001	Carbonate dehydratase	1062.15	185	362	45.2	171	169	98.83 %	91.35 %

Table 2: Results of the experiments with the real-world dataset. For each of the considered protein families, the maximum score obtainable for the calculated consensus graph and the score threshold above which an instance is considered a hit are shown. Additionally, we show the numbers of structures and cavities belonging to the family and how many of the protein structures could be recovered correctly. The values for precision (i.e., “Recovered” divided by “Hits”) and recall (i.e., “Recovered” divided by “Proteins”) are calculated with respect to the protein structures as the used classification database annotates only whole proteins.

and $\Phi_{\mu,\sigma}(x)$ is the cumulative distribution function of the normal distribution with mean μ and standard deviation σ at point x . $a(t'_1, t'_2)$ is the average weight of the edge (t'_1, t'_2) and $d(t'_1, t'_2)$ is the standard deviation of this weight. $fit(\cdot, \cdot)$ thus has a value of 1.0, if the weight of the edge (t_1, t_2) matches exactly the average weight for the consensus edge (t'_1, t'_2) . The value then decreases if the difference of the two weights grows, whereas the slope of the decrease depends on the actual data distribution of the observed distances in the graph alignment. The maximum score of $s(T)$ is obtained if the labels of all edges match exactly the observed mean in the consensus graph and yields

$$s_{\max} = \sum_{t \in T} sig(con(t))$$

Each such comparison of one consensus graph against the entire database took approximately 20 CPU-hours. The process can be parallelized easily as all calculations are independent. We are interested in a crisp (i.e., non-fuzzy) classification of the binding pockets in this study, therefore a scoring threshold has to be determined (cf. section 5.2): Entries

of the dataset are classified into a protein family if the score for the comparison with the respective consensus graph lies beyond the threshold. Otherwise, the entry is classified as not belonging to the family.

The scoring scheme used for our experiments yields scores between 0 and s_{\max} . Thus, it would be possible to select a classification threshold arbitrarily from this interval (e.g., $0.25 \cdot s_{\max}$). We decided instead to select the classification threshold based on the distribution of the scores in the different comparisons. For each of the ten protein families under consideration, mean and variance of the scores for the comparison of the consensus graph with the entries of the dataset were calculated. These scores follow approximately a normal distribution. (Note that the members of each protein family form only a very small fraction of the entire dataset. The scores for these entries thus have only a very small influence on the distribution of the scores.) From the calculated values for mean and variance, it was possible to calculate a statistical p -value for each of the obtained scores under the assumption of the normal distribution of the scores. The classification threshold was then selected to match a significance level of 10^{-5} (i.e., entries with a p -value beyond 10^{-5} are classified into the family of the consensus graph).

Table 2 summarizes the results of the classifications. As the ENZYME database contains functional annotations for whole proteins, not for single cavities, the classifications for the binding pockets were mapped back to the corresponding protein structures. A protein structure is considered as correctly classified, if a least one of the respective cavities is correctly classified. This procedure is justified as one protein structure can possess multiple cavities, but not all of these cavities have to be related to the function of the protein. For each of the considered protein families, Table 2 shows the number of protein structures and corresponding cavities annotated with the respective label. Additionally, the number of protein structures with a significant score (“Hits”) and the fraction of these cases that are actually annotated with the respective class label (“Recovered”) are shown. Closer analysis reveals that the “false-positive” hits typically belong to closely related protein families and thus are

correctly retrieved as hits.

Please note that the experimental results presented in this section were not aimed to produce optimal classification rates, but merely to serve as a proof of concept for the methods described in this paper. In practice, one would utilize additional features of the binding pockets (like, e.g., the shape of the solvent-accessible surface in the cavity) and thus obtain significantly improved results. Yet, the respective techniques cannot be formulated completely in a graph-theoretical framework and have therefore been discarded here.

In general, using the automatically derived consensus graphs, it was possible to identify most of the members of a protein family and of related families. Without question, the relevance of the generated consensus graphs strongly 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.

6.2.3 Discriminative Patterns

We selected trypsin and thrombin — two closely related protein families from the family of serine proteases [8] to examine the power of our approach 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.

We constructed a dataset composed by 111 binding pockets extracted from thrombin structures and 105 trypsin pockets. A multiple graph alignment was calculated for the graph representations of these binding pockets, and feature vectors were derived for the dataset as described in section 5.3. We obtained a set of 216 “flat” feature vectors, each of them with 114 categorial 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. We then used the WEKA package [27] to analyze our dataset.

The C4.5 algorithm [22] for decision tree induction was applied to the data set and a classification rate of 99.537 % was obtained in a 10-fold cross validation experiment (i.e., of the 216 cavities, 215 were classified correctly and one thrombin was misclassified as trypsin). Interestingly, the obtained (pruned) decision tree was rather small and contained only one inner node. I.e., 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 [10] 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 replaced by serine in trypsin [8, 12]. The method was thus able to discover automatically a known structural difference among the two classes under consideration.

Yet, as there are more relevant structural differences between trypsin and thrombin, we had to switch to another class of methods for our analysis (i.e., 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, other attributes are not further included into the model.) We therefore correlated the different “columns” of the alignment with the class membership vector as described in section 5.3. Of 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 machine learning algorithms mentioned above, showed the highest correlation (0.99). Four of the remaining attributes correspond to residues of the 60-loop, another important structural difference between trypsin and thrombin that partially covers the specificity pocket of thrombin, but is missing in trypsin. The method was thus also able to detect automatically two major differences among the considered classes [8]. A more detailed discussion of the obtained discriminative patterns is clearly beyond the scope of this paper.

7 Conclusions and Outlook

We introduced graph alignments, a new concept for robust mining on structured objects. The main motivation for graph alignments originates from the field of bioinformatics or chemoinformatics. In these areas, objects with a complex internal structure are examined, whereas this structure strongly influences their relevant properties. It is therefore interesting to study the correlation among the occurrence of certain substructures and the presence or absence of particular features. More precisely, graph alignments allow one to detect both, structural patterns that are characteristic for a certain class of objects, and structural patterns that are discriminative among different, yet related classes. To achieve the latter, a “flat” feature vector representation for the structured objects was derived, that maintains a direct correspondence to certain structural features and can thus easily be interpreted. This is an important advantage of our approach in comparison to existing methods for the mapping of structured objects onto tuples.

Another advantage of graph alignments is their robustness with respect to errors and noise as they frequently occur in experimentally derived datasets. In simulations with synthetic data, we were able to show that graph alignments are actually able to detect conserved patterns more reliable. One direction of future research could involve the development of new algorithms for the calculation of efficient graph alignments. Even though the methods described in this paper lead to satisfactory results in our experiments, they still leave space for further improvements. With the introduction of new algorithms, it might also be possible to further generalize the idea of graph alignments. One such extension could be to allow for mismatches (i.e., to allow the mapping of nodes with different labels onto each other).

Graph alignments introduce additional degrees of freedom in comparison to graph isomorphisms. We therefore examined how significant (i.e., reliable) the detected patterns are as in practice, common patterns will always be obtained for a given set of graphs. Our simulations indicate that one has indeed to carefully interpret retrieved patterns with a low degree of conservation. On the other hand, the simulations also allow one to precisely assess the

statistical significance of a certain pattern in a given set of graphs. As the obtained curves show a rather regular shape, it might be possible to derive closed-form approximations for these curves that allow an efficient estimation of statistical parameters without performing complex simulations for each possible combination of parameters.

We tested our approach also on a large dataset of graph representations of protein binding pockets. Based on an existing classification, consensus graphs were determined for a number of highly populated classes. The obtained prototypes were then used as templates to classify the whole dataset and most of the cavities labeled with the corresponding category were successfully recovered. In the case of rather heterogeneous classes, it might be useful to calculate multiple (alternative) prototypes for one class to better cover the variation within the class and thus to obtain a better overall classification. This is another direction of future research.

Finally, we showed that our approach is actually able to detect relevant discriminating structural patterns among different members of a protein family. Graph Alignments allow one to map graph instances onto “flat” feature vectors such that standard machine learning techniques and methods from statistical analysis can be applied. Yet, an important advantage of this mapping lies in the fact that the obtained features correspond directly to certain structural features of the considered cases. This makes the obtained models easily interpretable for domain experts.

Acknowledgments

The authors would like to thank the Cambridge Crystallographic Data Center (CCDC) for financial support of the work presented here.

References

- [1] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
- [2] Frank H. Allen. The cambridge structural database: a quarter of a million crystal structures and rising. *Acta Crystallographica B*, 58:380–388, 2002.
- [3] Amos Bairoch. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1):304–305, 2000.
- [4] Johannes Berg and Michael Lässig. Local graph alignment and motif search in biological networks. Technical Report COND-MAT/0308251, ArXiv, 2003. Available from <http://xxx.lanl.gov/abs/cond-mat/0205589>.
- [5] 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.
- [6] Konrad H. Bleicher, Hans-Joachim Böhm, Klaus Müller, and Alexander I. Alanine. Hit and lead generation: Beyond high-throughput screening. *Nature Reviews Drug Discovery*, 2(5):369–378, 2003.
- [7] Philip E. Bourne and Helge Weissig, editors. *Structural Bioinformatics*. Wiley-Liss, 2003.
- [8] Carl Branden and John Tooze. *Introduction to Protein Structure*. Garland Publishing, 1999.

- [9] Coen Bron and Joep Kerbosch. Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 16(9):575–577, September 1973.
- [10] William W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference (ML95)*, pages 115–123. Morgan Kaufmann Publishers, 1995.
- [11] Ding-Zhu Du and Panos M. Pardalos, editors. *Handbook of Combinatorial Optimization*, chapter The Maximum Clique Problem. Kluwer Academic Publishers, 1999.
- [12] 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.
- [13] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.
- [14] Johann Gasteiger and Thomas Engel, editors. *Chemoinformatics: A Textbook*. Wiley-VCH, 2003.
- [15] 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.
- [16] 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.
- [17] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. *Machine Learning*, 50:321–354, 2003.

- [18] Povl Krosggaard-Larsen, Tommy Liljefors, and Ulf Madsen, editors. *Textbook of Drug Design and Discovery*. Taylor & Francis, 3. edition, 2002.
- [19] Michihiro Kuramochi and George Karypis. An Efficient Algorithm for Discovering Frequent Subgraphs. Technical Report 02-026, Department of Computer Science/Army HPC Research Center, University of Minnesota, 2002.
- [20] Giorgio Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9:341–352, 1972.
- [21] Nomenclature Committee of the IUBMB. *Enzyme Nomenclature*. Academic Press, 1992.
- [22] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [23] 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.
- [24] Joao Carlos Setubal and Joao Meidanis. *Introduction to Computational Molecular Biology*. International Thomson Publishing, 1997.
- [25] Takashi Washio and Hiroshi Motoda. State of the Art of Graph-based Data Mining. *SIGKDD Explorations*, 5(1), 2003.
- [26] John Westbrook, Zukang Feng, Li Chen, Huanwang Yang, and Helen M. Berman. The Protein Data Bank and structural genomics. *Nucleic Acids Research*, 31(1):489–491, 2003.
- [27] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.

- [28] Xifeng Yan and Jiawei Han. gSpan: Graph-Based Substructure Pattern Mining. In *Proceedings of IEEE International Conference on Data Mining (ICDM'02)*, pages 721–724, 2002.
- [29] Xifeng Yan and Jiawei Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *Proceedings of ACM SIGKDD 2003*, 2003.