Evolutionary Construction of Multiple Graph Alignments for the Structural Analysis of Biomolecules

Thomas Fober [1], Marco Mernberger [1,2], Gerhard Klebe [2], Eyke Hüllermeier [1]

[1]Department of Mathematics and Computer Science
[2]Department of Pharmaceutical Chemistry
Philipps-Universität Marburg, Germany

The concept of multiple graph alignment has recently been introduced as a novel method for the structural analysis of biomolecules. Using approximate graph matching techniques, this method enables the robust identification of approximately conserved patterns in biologically related structures. In particular, multiple graph alignment enables the characterization of functional protein families independent of sequence or fold homology. This paper first recalls the concept of multiple graph alignment and then addresses the problem of computing optimal alignments from an algorithmic point of view. In this regard, a method from the field of evolutionary algorithms is proposed and empirically compared to a hitherto existing heuristic approach. Empirically, it is shown that the former yields significantly better results than the latter, albeit at the cost of an increased runtime.

1 Introduction

Multiple sequence alignment is an established method for similarity analysis and the identification of residues that are conserved across many members of a gene or protein family [23, 2, 18]. However, as this approach relies on evolutionary conserved sequences in DNA or protein chains, it is only capable of detecting similarities between different molecules that are based on heredity. Consequently, sequence analysis is not optimally suited for the identification of *functional* similarities, as functionality is more closely associated with structural than with sequential features. In fact, it is well-known that structural similarity does not necessarily come along with sequence similarity [8].

1

Focusing on the identification of *structural* similarities, the comparison of protein structures and the identification of common three-dimensional patterns and substructures has received considerable attention in the recent years, and a number of different approaches has been proposed for this purpose [12, 22, 19, 25, 14, 26, 27, 28]. In principle, approaches of that kind allow one to capture non-homologous molecules with similar functions as well as evolutionary conserved functional domains. Restrictively, however, it needs to be mentioned that many of the existing methods use exact matching techniques, such as subgraph isomorphism, which makes it difficult to discover biologically meaningful patterns that are only approximately conserved. Moreover, most methods are restricted to the comparison of pairs of graphs, with only a few notable exceptions that are able to compare multiple graphs simultaneously [6, 20, 15, 21]. A detailed review of related work is given in the supplementary material.

This work draws on the concept of *multiple graph alignment* (MGA), which was first introduced in [24] as a structural counterpart to multiple sequence alignment. Our special interest concerns the analysis of protein structures or, more specifically, protein binding sites, even though graph alignments can also be used for analyzing other types of biomolecules.[1] Weskamp et al. [24] proposed a heuristic algorithm which employs a simple greedy strategy to construct MGAs in an incremental way. Here, we present an alternative method using evolutionary algorithms. As will be shown experimentally, significant improvements in terms of the quality of alignments can thus be achieved.

The paper is organized as follows: We start with a short introduction to graph-based modeling of biomolecules in Section 2. In Section 3, we introduce the concept of a multiple graph alignment. The problem of computing an MGA is then addressed in Section 4, where an evolutionary algorithm is proposed for this purpose. Section 5 is devoted to the experimental evaluation of the approach, and Section 6 concludes the paper.

2 Modeling Biomolecules as Graphs

Single biomolecules in bio- and chemoinformatics are often modeled at an abstract level in terms of a graph G consisting of a set of (labeled) nodes V and (weighted) edges E. In this paper, we are especially interested in the graph-based modeling of protein binding sites. Yet, our experiments later on will also include a data set of chemical compounds.

2.1 Modeling Chemical Compounds

In case of chemical compounds, atoms are represented as nodes labeled with their corresponding atom type, e.g., using the SYBYL atom type notation [7]. Edges between atoms can represent chemical bonds or Euclidean distances. In the former case, edges are weighted by the bond order. In the latter case, edge weights represent Euclidean distances, thereby capturing the inherent geometry of the compound.

---

[1]The same term is also used by Berg and Lässig [3]. However, the problem they consider is actually more related to motif search and frequent pattern mining, which is quite different from an algorithmic point of view.

2.2 Modeling Protein Binding Sites

The graph-based modeling of protein binding pockets is less obvious. Our work builds upon Cavbase [16, 17], a database system for the automated detection, extraction, and storage of protein cavities (hypothetical binding pockets) from experimentally determined protein structures (available through the PDB). In Cavbase, graphs are used as a first approximation to describe binding pockets. The database currently contains $113,718$ hypothetical binding pockets that have been extracted from $23,780$ publicly available protein structures using the Ligsite algorithm of Hendlich et al. [9].

To model a binding pocket as a graph, the geometrical arrangement of the pocket and its physicochemical properties are first represented by predefined *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 predefined rules [17]. As possible types for pseudocenters, hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic, metal ion, pi (accounts for the hydrophobic character present above/below amide, guanidinium and carboxylate planes) 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. Consequently, a set of pseudocenters is an approximate representation of a spatial distribution of physicochemical properties.

The assigned pseudocenters form the nodes $v \in V$ of the graph representation, and their properties are modeled in terms of node labels $\ell(v) \in \{\texttt{P1}, \texttt{P2} \ldots \texttt{P7}\}$, where $\texttt{P1}$ stands for donor, $\texttt{P2}$ for acceptor, etc. Two centers are connected by an edge in the graph representation if their Euclidean distance is below a certain threshold[2] and each edge $e \in E$ is labeled with the respective distance $w(e) \in \mathbb{R}$. The edges of the graph thus represent geometrical constraints among points on the protein surface.

3 Multiple Graph Alignment

When comparing protein cavities on a structural level, one has to deal with the same mutations that also occur on the sequence level. Corresponding mutations, in conjunction with conformational variability, strongly affect the spatial structure of a binding site as well as its physicochemical properties and, therefore, its graph descriptor. This is even more an issue when it comes to the comparison of proteins that might share a common function but lack a close hereditary relationship. Thus, one cannot expect that the graph descriptors for two functionally related binding pockets match exactly. Our approach therefore includes the following types of edit operations to account for differences between a graph $G(V, E)$ and another graph.

1. Insertion or deletion of a node $v \in V$: A pseudocenter can be deleted or inserted

---

[2]An interaction distance of 11.0 Å is typically enough to capture the geometry of a binding site, and ignoring larger distances strongly simplifies the graph representation and hence accelerates the subsequent calculations.
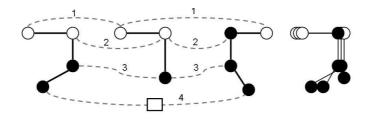
Figure 1: Simple illustration of MGA by an approximate match of three graphs with two types of labels (black and white). Mutual assignments of nodes are indicated by dashed lines. Note that the second assignment involves a mismatch, since the node in the third graph is black. Likewise, the fourth assignment involves a dummy (indicated by a box), since a node is missing in the second graph. The rightmost picture is a graphical overlay of the three structures.

due to a mutation in sequence space. Alternatively, an insertion or deletion in the graph descriptor can result from a conformational difference that affects the exposure of a functional group toward the binding pocket.

2. Change of the label $\ell(v)$ of a node $v \in V$: The assigned physicochemical property of a pseudocenter can change if a mutation replaces a certain functional group by another type of group at the same position.

3. Change of the weight $w(e)$ of an edge $e \in E$: The distance between two pseudo-centers can change due to conformational differences.

By assigning a cost value to each of these edit operations, it becomes possible to define an edit distance for 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 transforms graph G_1 into G_2. As in sequence analysis, this allows for defining the concept of an alignment of two (or more) graphs. The latter, however, also requires the possibility to use dummy nodes $\perp$ that serve as placeholders for deleted nodes. They correspond to the gaps in sequence alignment (cf. Fig. 1).

Definition 1 (*Multiple Graph Alignment*) *Let $\mathcal{G} = \{G_1(V_1, E_1) \dots G_m(V_m, E_m)\}$ be a set of node-labeled and edge-weighted graphs. Then $\mathcal{A} \subseteq (V_1 \cup \{\perp\}) \times \cdots \times (V_m \cup \{\perp\})$ is an alignment of the graphs in $\mathcal{G}$ if and only if the following two properties hold: (i) Each node of each graph occurs exactly once in the alignment, i.e., for all $i = 1 \dots m$ and for each $v \in V_i$ there exists exactly one $a = (a_1 \dots a_m) \in \mathcal{A}$ such that $v = a_i$. (ii) Each tuple of the alignment contains at least one non-dummy node, i.e., for each $a = (a_1 \dots a_m) \in \mathcal{A}$ there exists at least one $1 \leq i \leq m$ such that $a_i \neq \perp$.*

Each $a \in \mathcal{A}$ corresponds to a vector of mutually assigned nodes from the graphs $G_1 \dots G_m$. Note that, by matching nodes, a mutual assignment of edges is determined in an implicit way. Once an MGA has been established, it can be used to derive approximately conserved patterns. This can be done in different ways, for example by

4

extracting highly conserved subgraphs from a "fuzzy" consensus graph $\mathcal{G}$. This graph contains one node $v = v(a)$ for each $a \in \mathcal{A}$ that summarizes the label distribution of the mutually assigned nodes from the original graphs. Additionally, a degree of conservation $cons(v)$ is calculated, which is defined by the relative number of graphs in which this node is present. The edges of the consensus graph are defined accordingly; see [24] for details. For given thresholds $\alpha, \beta \in (0, 1]$, a conserved pattern can then be defined in terms of the subgraph of $\mathcal{G}$ consisting of all nodes v with $cons(v) \geq \alpha$ and $maj(v) \geq \beta$, where $maj(v)$ is the relative frequency of the most frequent label in a.

To assess the quality of a given alignment, a scoring function is used that corresponds to the above-mentioned edit distance (each graph alignment defines a set of edit operations that have to be performed to transform one of the aligned graphs into another graph of the alignment). Our scoring function follows a sum-of-pairs scheme, i.e., the score s of a multiple alignment $\mathcal{A} = (a^1 \ldots a^n)$ is defined by the sum of scores of all induced pairwise alignments. Moreover, it consists of two parts, a node score and an edge score:

$$s(\mathcal{A}) = \sum_{i=1}^{n} \text{ns}(a^i) + \sum_{1 \leq i < j \leq n} \text{es}(a^i, a^j) \ . \tag{1}$$

The node score (ns) is defined by the sum of the evaluations of the individual columns of the alignment, where such an evaluation is defined as follows:

$$\text{ns} \begin{pmatrix} a_1^i \\ \vdots \\ a_m^i \end{pmatrix} = \sum_{1 \leq j < k \leq m} \begin{cases} \text{ns}_m & \ell(a_j^i) = \ell(a_k^i) \\ \text{ns}_{mm} & \ell(a_j^i) \neq \ell(a_k^i) \\ \text{ns}_{dummy} & a_j^i = \perp, a_k^i \neq \perp \\ \text{ns}_{dummy} & a_j^i \neq \perp, a_k^i = \perp \end{cases}$$

Thus, to evaluate a single vector of mutually assigned nodes, these nodes are considered in a pairwise manner, and three cases are distinguished: (i) the labels of two nodes are equal (match), (ii) the labels differ (mismatch), and (iii) one of the nodes is a dummy.

Comparing two edges is somewhat more difficult than comparing two nodes, as one cannot expect to observe edges of exactly the same lengths. We consider two edges as a match if their respective lengths, a and b, differ by at most a given threshold ϵ, and as a mismatch otherwise. The edge score (es) is then given by

$$\text{es} \left(\begin{pmatrix} a_1^i \\ \vdots \\ a_m^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_m^j \end{pmatrix} \right) =$$

$$\sum_{1 \leq k < l \leq m} \begin{cases} \text{es}_{mm} & (a_k^i, a_k^j) \in E_k, (a_l^i, a_l^j) \notin E_l \\ \text{es}_{mm} & (a_k^i, a_k^j) \notin E_k, (a_l^i, a_l^j) \in E_l \\ \text{es}_m & d_{kl}^{ij} \leq \epsilon \\ \text{es}_{mm} & d_{kl}^{ij} > \epsilon \end{cases}$$

5

where $d_{kl}^{ij} = \left| w(a_k^i, a_k^j) - w(a_l^i, a_l^j) \right|$. The parameters (i.e., ns_m, ns_{mm}, ns_{dummy}, es_m, es_{mm}) are constants used to reward or penalize matches, mismatches and dummies, respectively. Throughout our experiments in Section 5, we used the parameters recommended by Weskamp et al. [24]: $\mathrm{ns}_m = 1$, $\mathrm{ns}_{mm} = -5$, $\mathrm{ns}_d = -2.5$, $\mathrm{es}_m = 0.2$, $\mathrm{es}_{mm} = -0.1$, $\epsilon = 0.2$.

The problem of calculating an optimal MGA, that is, an alignment with maximal score for a given set of graphs, is provably NP-complete. In [24], simple and effective heuristics for the MGA problem have been devised that were found to be useful for the problem instances that were examined. The main idea of these methods is to reduce the multiple alignment problem to several pairwise problems (i.e., calculating an optimal graph alignment for only two graphs) in a first step. To align a pair of graphs, an exact seed solution in the form of a subgraph isomorphism is computed and then extended to a complete alignment in a greedy manner. Resorting to the idea of star-alignment, which is well-known in sequence analysis, all pairwise alignments are finally merged into a multiple alignment.

In this paper, we elaborate on the use of evolutionary algorithms as an alternative approach. On the one hand, evolutionary optimization is of course more expensive from a computational point of view. On the other hand, the hope is that this approach will be able to improve the solution quality, i.e., to produce alignments that are better than those obtained by the simple greedy strategy.

4 An Evolutionary Algorithm for Multiple Graph Alignment

In this section, we introduce a new algorithm for multiple graph alignment called GAVEO (which is short for Graph Alignment Via Evolutionary Optimization). To this end, we resort to the class of *evolution strategies*. An evolution strategy is a special type of evolutionary algorithm (EA) that seeks to optimize a *fitness function*, which in our case is given by the sum-of-pairs score (1). To this end, it simulates the evolution process by repeatedly executing the following loop [4]:

1. Initially, a population consisting of μ individuals, each representing a candidate solution, is generated at random; μ specifies the *population size* per generation.

2. In each generation, $\lambda = \nu \cdot \mu$ offspring individuals are created; the parameter ν is called *selective pressure*. To generate a single offspring, the mating-selection operator chooses ρ parent individuals at random and submits them to the *recombination* operator. This operator generates an offspring by exchanging the genetic information of these individuals. The new individual is further modified by the *mutation* operator.

3. The offsprings are evaluated and added to the parent population. Among the individuals in this temporary population T, the *selection* operator chooses the best μ candidates, which form the population of the next generation.

4. Steps (2) and (3) are repeated until a stopping criterion is met.

```
A: 1 2 . 4 3 5 .
B: 6 1 2 3 4 5 .
C: 4 3 2 . 1 . .
D: . 3 4 . 2 1 .
```

Figure 2: Matrix representation of an MGA. Dummies are represented by a dot. Note that the order of the columns is arbitrary.

4.1 Representation of Individuals

Regarding the representation of individuals, note that in our case candidate solutions correspond to MGAs. Given a fixed numbering of the nodes of graph G_i from 1 to $|V_i|$ (not to be confused with the node labeling), an MGA can be represented in a unique way by a two-dimensional matrix, where the rows correspond to the graphs and the columns to the aligned nodes (possibly a dummy, indicated by a dot) of these graphs. Fig. 2 shows an example of such a matrix. The first column indicates a mutual assignment of the first node of graph A, the sixth node of graph B, and the fourth node of graph C, while there is no matching partner other than a dummy in graph D.

In the course of optimizing an MGA, the graphs can become larger due to the insertion of dummy nodes. For the matrix representation, this means that the number of columns is in principle not known and can only be upper-bounded by $|V_1| + \ldots + |V_m|$. This, however, will usually be too large a number and may come along with an excessive increase of the search space. From an optimization point of view, a small number of columns is hence preferable. On the other hand, by fixing a too small length of the alignment, flexibility is lost and the optimal solution is possibly excluded.

To avoid these problems, we make use of an *adaptive* representation: Starting with a single extra column filled with dummies, more such columns can be added if required or, when becoming obsolete, again be removed (see below). Thus, our matrix scheme is initialized with m rows and $n_{max} + 1$ columns, where $n_{max} = \max\{|V_1|, |V_2|, \ldots, |V_m|\}$. For each graph G_i, a permutation of its nodes is then inserted, with dummies replacing the index positions $j > |V_i|$. In passing, we note that dummy columns are of course excluded from scoring, i.e., the insertion or deletion of dummy columns has no influence on the fitness.

4.2 Evolutionary Operators

Among the proper selection operators for evolution strategies, the deterministic plus-selection, which selects the μ best individuals from the union of the μ parents and the λ offsprings, is most convenient for our purpose. In fact, since the search space of an MGA problem is extremely large, it would be very unfortunate to loose a current best solution. This excludes other selection techniques such as fitness-proportional or simulated annealing selection.

As we use a non-standard representation of individuals, namely a matrix scheme, the commonly used recombination and mutation operators are not applicable and have to

be adapted correspondingly.

4.2.1 Recombination

Our recombination operator randomly selects ρ parent individuals from the current population (according to a uniform distribution). Then, $\rho - 1$ random numbers r_i, $i = 1 \ldots \rho - 1$, are generated, where $1 \leq r_1 < r_2 < \ldots < r_{\rho-1} < m$, and an offspring individual is constructed by combining the sub-matrices consisting, respectively, of the rows $\{r_{i-1} + 1 \ldots r_i\}$ from the i-th parent individual (where $r_0 = 0$ and $r_\rho = m$ by definition). Simply stitching together complete sub-matrices is not possible, however, since the nodes are not ordered in a uniform way. Therefore, in merging step i, the ordering of the r_i-th row is used as a reference. An example illustrating the recombination operator is given in the supplementary material. General experience has shown that recombination increases the speed of convergence, and this was also confirmed by our experiments (see Section 5).

4.2.2 Mutation

The mutation operator selects one row and two columns at random and swaps the entries in the corresponding cells. To enable large mutation steps, we have tried to repeat this procedure multiple times for each individual. As the optimal number of repetitions was unknown in the design phase of the algorithm, it was specified as a strategy component adjusted by a self-adaptation mechanism [4]. However, our experiments indicated that a simple mutation operator performing only single swaps solves the problem most effectively (see Section 5).

4.2.3 Adaptation of Alignment Length

To adapt the length of an MGA (number of columns in the matrix scheme), it is checked in randomly chosen intervals whether further dummy columns are needed or existing ones have become unnecessary. Three cases can occur: (i) There exists exactly one dummy column, which means that the current length is still optimal. (ii) There is more than one dummy column: Apparently, a number of dummy columns are obsolete and can be removed, retaining only a single one. (iii) There is no dummy column left: The dummy column has been "consumed" by mapping dummies to real nodes. Therefore, a new dummy column has to be inserted.

4.3 Combining Evolutionary Optimization and Pairwise Decomposition

The search space of an MGA problem grows exponentially with the number of graphs, which is of course problematic from an optimization point of view. One established strategy to reduce complexity is to decompose a multiple alignment problem into several pairwise problems and to merge the solutions of these presumably more simple problems into a complete solution. This strategy has already been exploited by Weskamp [24], who realized the decomposition and merging steps by means of the star-alignment algorithm.

In star-alignment, a center structure is first determined, and this structure is aligned with each of the other $m - 1$ structures. The $m - 1$ pairwise alignments thus obtained are then merged by using the nodes of the center as pivot elements. As the quality of an MGA derived in this way critically depends on the choice of a suitable center structure, one often tries every structure as a center and takes the best result. In this case, all possible pairwise alignments are needed, which means that our evolutionary algorithm must be called $\frac{1}{2}(m^2 - m)$ times.

As star-alignment is again a purely heuristic aggregation procedure, the gain in efficiency is likely to come along with a decrease in solution quality, compared with the original evolutionary algorithm (GAVEO). This is not necessarily the case, however. In fact, a decomposition essentially produces two opposite effects, a positive one due to a simplification of the problem and, thereby, a reduction of the search space, and a negative one due to a potentially suboptimal aggregation of the partial solutions. For a concrete problem, it is not clear in advance which among these two effects will prevail. Roughly speaking, it is well possible that constructing good pairwise alignments and aggregating them in an ad-hoc way is better than getting astray in a huge search space of multiple alignments.

5 Experimental Results

GAVEO as outlined in the previous section has a number of exogenous parameters: μ, the population size; ν, the selective pressure; ρ, the recombination parameter; τ, the probability to check for dummy columns; `selfadaption`, which can assume values $\{\texttt{on}, \texttt{off}\}$, and enables or disables the automatic step size control; `initial step size`, which defines the initial step size for the mutation. If the automatic step size control is disabled, this parameter is ignored and a constant step size of 1 is used for the mutation.

To optimize these parameters, we have applied the *sequential parameter optimization toolbox* (SPOT) [1] in combination with suitable synthetic data sets. Based on the results, the following parameter configuration appears to be well-suited for our problem class: $\mu = 4$, $\nu = 15$, `selfadaption = off`, $\rho = 4$, $\tau = 0.35$. As can be seen, a small value for the population size (only large enough to enable recombination) is enough, probably due to the fact that local optima do not cause a severe problem. On the other hand, as the search space is extremely large, a high selective pressure is necessary to create offsprings with improved fitness. The self-adaptation mechanism is disabled and, hence, the mutation rate is set to one (only two cells are swapped by mutation). This appears reasonable, as most swaps do not yield an improvement and instead may even produce a deterioration, especially during the final phase of the optimization. Thus, an improvement obtained by swapping two cells is likely to be annulled by a second swap in the same individual. Finally, our experiments suggest that a recombination is very useful and should therefore be enabled. The probability τ is set to a relatively high value due to avoiding long times of stagnation because of an insufficient alignment length.

5.1 Mining Molecular Fragment Data

As a first proof-of-concept, we analyzed our algorithms on a data set consisting of 87 compounds that belong to a series of benzamidine derivatives, selective thrombin inhibitors, that were taken from a 3D-QSAR study [5]. The data set is suitable for conducting experiments in a systematic way, as it is quite homogeneous and relatively small (the graph descriptors contain 47–100 nodes, where each node corresponds to an atom). Moreover, as the 87 compounds all share a common core fragment (which is distributed over two different regions with a variety of substituents), the data set contains a clear and unambiguous target pattern. From this data set, 100 subsets of 2, 4, 8, 16 and 32 compounds have been selected at random, and for each subset, an MGA has been calculated using the greedy heuristic (Greedy), our evolutionary algorithm with optimized parametrization (GAVEO) and, as introduced in Section 4.3, in combination with a pairwise reduction and star-alignment procedure (GAVEO*).

In a first experiment, we investigated whether or not the expense (large population, complex mutation and recombination operators) of an EA is justified at all. To this end, we compared GAVEO with a simple hill-climbing search, namely a $(1 + 1)$-EA. To guarantee a fair comparison, we fixed the number of function evaluations for both algorithms. The results, summarized in the supplementary material, were clearly in favor of GAVEO.

The results of the main experiment, comparing the two EA-variants with the greedy strategy, are shown in Fig. 3. As a measure of comparison, we derived the relative improvement of the score (1),
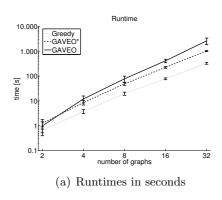
$$\frac{s(\mathcal{A}') - s(\mathcal{A})}{\min\{|s(\mathcal{A}')|, |s(\mathcal{A})|\}} \, , \tag{2}$$
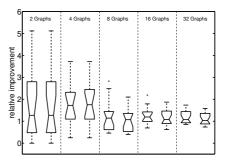
where $\mathcal{A}'$ and $\mathcal{A}$ denote, respectively, the alignment produced by GAVEO (or GAVEO*) and Greedy. This measure is positive if the GAVEO solution yields a higher score than the Greedy solution; e.g., a relative improvement of 1 would mean an increase in score by a factor of 2 (note that $s(\mathcal{A}) < 0$ is possible).

The results, shown in Fig. 3, confirm our expectations: Both EA variants significantly outperform Greedy with respect to alignment quality. The runtimes,[3] on the other hand, confirm that the greedy strategy is still the most efficient among all alternatives. A good compromise between solution quality and efficiency is achieved by GAVEO*: Its runtime is much better than the one of GAVEO, especially for a larger number of graphs, while the alignment quality is only slightly worse, as can also be seen in Fig. 3. In fact, in terms of average relative improvement, it loses in the range of only 4-6% (with a tendency to increase with the number of structures).

Regarding the identification of conserved patterns among the derivatives, we were mainly interested in the retrieval of the aforementioned benzamidine core fragment, an amide derivative of benzol, which consists of 25 atoms (11 hydrogens). Upon inspection of the subgraphs that are highly conserved in the sense of satisfying $\alpha = 1$ and $\beta \geq 0.9$

---

[3]Intel Core 2 Duo 2.4 GHz, 2 GB memory, Windows XP SP 2 operating system.

(a) Runtimes in seconds

(b) Relative improvements (boxplot) of GAVEO (each left box) and GAVEO* (each right box) in comparison to Greedy; positive values indicate an improvement

Figure 3: Results of the experiment on the benzamidine data set: (a) Runtimes in seconds (mean and standard deviation) of Greedy, GAVEO, and GAVEO*. (b) Relative improvements of GAVEO and GAVEO* as defined in (2).

Table 1: Percent of alignments in which the benzamidine core fragment was fully conserved.

# graphs	2	4	8	16	32
Greedy	58%	38%	14%	4%	2%
GAVEO	99%	96%	95%	98%	98%
GAVEO*	99%	96%	95%	95%	98%

(cf. Section 3), it turned out that the core fragment was retrieved by GAVEO and GAVEO* in nearly all experiments; see Table 1 for a summary of the results. Since the greedy approach is prone to local optima, it is probable to miss at least parts of this fragment, and this probability increases with the number of structures.

5.2 Mining Protein Binding Pockets

We examined the performance of our algorithms also on a data set consisting of 74 structures derived from the Cavbase database. Each structure represents a protein cavity belonging to the protein family of thermolysin, bacterial proteases frequently used in structural protein analysis and annotated with the E.C. number 3.4.24.27 in the ENZYME database. The data set is suited for our purpose, as all cavities belong to the same enzyme family and, therefore, evolutionary related, highly conserved substructures ought to be present. On the other hand, with cavities (hypothetical binding pockets) ranging from about 30 to 90 pseudocenters and not all of them being real binding pockets, the data set is also diverse enough to present a real challenge for graph matching techniques. Again, we produced 100 graph alignments of size 2, 4, 8, 16 and 32, respectively, for randomly chosen structures, and derived the relative improvement (2)
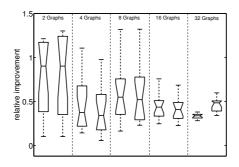
Figure 4: Relative improvements as defined in (2) of GAVEO (each left box) and GAVEO* (each right box) on the thermolysin data set; positive values indicate an improvement.

as a measure of comparison. (A visual representation of an alignment is shown in the supplementary material.)

The results, summarized in Fig. 4, are in agreement with the previous experiment. In terms of alignment scores, the GAVEO solutions are never worse and often significantly better than the Greedy solutions. In terms of runtime, Greedy is still more efficient. Again, a good compromise between solution quality and efficiency is achieved by GAVEO* (this time losing even less in terms of relative improvement, namely around 2-3%).

From the relative improvement (2) alone, it is hard to say to what extent two graph alignments are indeed different. Therefore, we have developed a visualization tool that aims at conveying the essential information about an MGA in a compact form. An example is shown in the supplementary material. The visual inspection of various solutions produced, respectively, by GAVEO and Greedy for the thermolysin structures clearly confirms that an evolutionary optimization is able to improve the quality of an MGA.

Regarding the discovery of conserved patterns, the highly conserved part ($\alpha = 1$, $\beta = 0.9$) includes a metal pseudocenter surrounded by several acceptor and donor/acceptor centers (a graphical illustration is shown in the supplementary material). As thermolysin is a bacterial metalloprotease, we obviously captured the subpart of the cavity hosting the zinc ion of thermolysin. The surrounding acceptor pseudocenters probably correspond to residues interacting with the ion. Upon reconciliation with the corresponding protein structures, we could verify that these acceptor centers resembled histidine and glutamate residues that form a coordinate bond with the ion. The zinc ion is essential for the enzyme activity, as removal of the zinc abolishes enzyme function [11]. Additionally, we discovered a highly conserved mapping of acceptor centers in close proximity that resembled the residue GLU143, a key residue of the active site which is supposed to act as a "proton shuttle", transferring a proton from an attacking water molecule to a nitrogen of the cleaved substrate. Some other conserved donor and acceptor pseudocenters might be required to form hydrogen bonds with the substrate. In fact, the underlying

Table 2: Accuracy of Greedy and GAVEO on the NADH/APT data set, determined by means of a leave-one-out cross-validation.

k	1	3	5	7	9
Greedy	76.62	71.83	72.39	71.83	71.27
GAVEO	78.87	76.62	78.03	78.59	76.62

residues have been shown to interact with an inhibitor of the enzyme [10]. This example shows that our approach is able to retrieve relevant groups for the enzyme function and, therefore, nicely demonstrates its usefulness for the analysis of protein function.

5.3 Similarity-Based Classification of Proteins

In a final study, we investigated the influence of the improved similarity scores on the classification accuracy on a two-class data set. To assess the predictive power of our approach on a real-world data set, we compiled a set of 355 protein binding pockets representing two classes of proteins that each share a certain prominent cofactor, adenosine-5'-triphosphate (ATP) and nicotine amide dinucleotide (NADH), respectively. Here we used Cavbase to retrieve all known ATP and NADH binding pockets that were co-crystallized with the respective ligand. Subsequently, we reduced the set to one cavity per protein, thus representing the enzymes by a single binding pocket. As protein ligands adopt different conformations due to their structural flexibility, it is likely that the ligands in our data set are bound in completely different ways, hence the corresponding binding pocket does not necessarily share much structural similarity. Thus, we had to ensure that we chose binding pockets with ligands bound in similar conformation. To achieve this, we used the Kabsch algorithm [13] to calculate the root mean square deviation (RMSD) between pairs of ligand structures. Subsequently, we used this measure to sample a sufficiently large data set while enforcing similar conformations on the ligands. To this end, we used an RMSD threshold value of 0.4, which yields a good trade-off between data set size and similarity. The data set thus obtained contains 214 NADH-binding proteins and 141 ATP-binding proteins and gives rise to a two-class classification problem.

To test the discriminative power of our approach, we calculated all possible pairwise alignments in our data set and thus derived two similarity matrices based on the obtained similarity scores, one for the greedy approach and one for GAVEO. We then used a standard k-nearest-neighbor algorithm to make a class prediction for each protein binding pocket. This approach simply predicts the class which is prevalent among the k most similar structures. The results of several leave-one-out cross-validation studies, performed for different values of k, are summarized in Table 2.

As can be seen, GAVEO again outperforms the greedy strategy, this time in terms of classification accuracy. Note that the classification accuracies can also be taken as an indicator of how well the pairwise similarity scores are in agreement with the class structure. In this regard, it is noticeable that, in the case of GAVEO, the accuracy degrees are all comparable and do not strongly depend on k. For the greedy algorithm,

the results are less stable and actually deteriorate for larger values of k. This suggests that GAVEO does indeed achieve a better class separation.

To demonstrate that our structure-based approach is able to extract useful information even in cases of low sequence similarity, we repeated the experiment with an alternative ATP/NADH data set. This data set includes only proteins with a sequence identity below 30% (ligand similarity was neglected) and comprises 48 ATP and 48 NADH structures. For $k = 1$ (leave-one-out cross-validation), GAVEO still achieved a respectable accuracy of 67.69%, compared to 61.54 % of the greedy strategy. As to be expected, a classification based on sequence alignment completely fails and yields results not better than random guessing (49.23%).

6 Summary and Conclusions

Multiple Graph Alignment (MGA) has recently been introduced as a novel method for analyzing structured data, especially biomolecules. Using robust, noise-tolerant graph matching techniques, this method is able to discover approximately conserved patterns in a set of graph descriptors representing a family of evolutionary related biological structures. As the calculation of optimal alignments is a difficult and computationally complex problem, this paper has proposed GAVEO, an evolutionary algorithm for multiple graph alignment, as an alternative to an existing greedy strategy. An implementation of this algorithm along with a user's guide can be downloaded at `www.uni-marburg.de/fb12/kebi/research/`.

Our experiments have shown the high potential of this approach and give rise to the following main conclusions: GAVEO significantly outperforms the greedy strategy and produces alignments that are better both in terms of their scores and in the sense of revealing conserved substructures in a more reliable way. The improved alignments also produce better (pairwise) similarity degrees, which in turn leads to better results in related performance tasks such as (nearest neighbor) classification.

On the other hand, the evolutionary optimization of graph alignments is computationally more complex than the greedy approach. Fortunately, however, a good compromise between solution quality and runtime can be achieved by a combination of evolutionary optimization with a star-alignment decomposition. Finally, it is worth mentioning that GAVEO has advantages regarding space complexity. In fact, its memory requirements are significantly smaller than those of the greedy algorithm, which needs to construct and store the product of the input graphs to generate an initial solution.

The methods introduced so far compare graph structures by taking the whole molecule, respectively the whole binding pocket, into account. While this approach is reasonable for structures that strongly resemble each other, it may fail if this is not the case. In fact, ligands do not necessarily occupy the whole protein cavity in which it is bound. Thus, two cavities both hosting the same ligand may still have larger parts that are not similar to each other and, therefore, can become difficult to align. In future work, we aim at extending our methods to cope with such cases. Roughly speaking, this comes down to developing graph-based counterparts to *local* or *semi-global* sequence alignment.

References

[1] Thomas Bartz-Beielstein. *Experimental research in evolutionary computation: The new experimentalism.* Springer, 2006.

[2] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The Pfam protein families database. *Nucleic Acids Research*, 32:138–141, 2004.

[3] J. Berg and M. Lässig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(41):14689–14694, 2004.

[4] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Journal Natural Computing*, 1(1):3–52, 2002.

[5] M. Böhm, J. Stürzebecher, and G. Klebe. Three-dimensional quantitative structure-activity relationship analyses using comparative molecular field analysis and comparative molecular similarity indices analysis to elucidate selectivity differences of inhibitors binding to trypsin, thrombin, and factor xa. *Journal of Medicinal Chemistry*, 42(3):458–477, 1999.

[6] O. Dror, H. Benyamini, R. Nussinov, and H. Wolfson. MASS: Multiple Structural Alignment by Secondary Structures. *Bioinformatics*, 19(1):i95–104, 2003.

[7] Johann Gasteiger and Thomas Engel. *Chemoinformatics.* Wiley-Vch, Weinheim, 2003.

[8] J. F. Gibrat, T. Madej, and S. H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6(3):377–385, 1996.

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

[10] H.M. Holden and B.W. Matthews. The binding of L-valyl-L-tryptophan to crystalline thermolysin illustrates the mode of interaction of a product of peptide hydrolysis. *Journal of Biological Chemistry*, 263(7):3256–3260, 1988.

[11] B. Holmquist and B.L. Vallee. Esterase activity of zinc neutral proteases. *Biochemistry*, 15(1):101–107, 1976.

[12] M. Jambon, A. Imberty, G. Deleage, and C. Geourjon. A New Bioinformatic Approach to Detect Common 3 D Sites in Protein Structures. *Proteins Structure Function and Genetics*, 52(2):137–145, 2003.

[13] Wolfgang Kabsch. A solution of the best rotation to relate two sets of vectors. *Acta Crystallographica*, 32:922–923, 1976.

[14] K. Kinoshita and H. Nakamura. Identification of the Ligand Binding Sites on the Molecular Surface of Proteins. *Protein Science*, 14(3):711–718, 2005.

[15] N. Leibowitz, R. Nussinov, and H.J. Wolfson. MUSTA-A General, Efficient, Automated Method for Multiple Structure Alignment and Detection of Common Motifs: Application to Proteins. *Journal of Computational Biology*, 8(2):93–121, 2001.

[16] S. Schmitt, M. Hendlich, and G. Klebe. From structure to function: A new approach to detect functional similarity among proteins independent from sequence and fold homology. *Angewandte Chemie International Edition*, 40(17):3141 – 3144, 2001.

[17] S. Schmitt, D. Kuhn, and G. Klebe. A new method to detect related function among proteins independent of sequence and fold homology. *Journal of Molecular Biology*, 323(2):387–406, 2002.

[18] F. Servant, C. Bru, S. Carrère, E. Courcelle, J. Gouzy, D. Peyruc, and D. Kahn. Prodom: Automated clustering of homologous domains. *Briefings in Bioinformatics*, 3(3):246–25, 2002.

[19] D. Shasha, J.T.L. Wang, and R. Giugno. Algorithmics and Applications of Tree and Graph Searching. In *Proc. 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 39–52. ACM Press New York, USA, 2002.

[20] M. Shatsky, R. Nussinov, and H.J. Wolfson. A Method for Simultaneous Alignment of Multiple Protein Structures. *Proteins Structure Function and Bioinformatics*, 56(1):143–156, 2004.

[21] M. Shatsky, A. Shulman-Peleg, R. Nussinov, and H. J. Wolfson. The multiple common point set problem and its application to molecule binding pattern detection. *Journal of Computational Biology*, 13(2):407–428, 2006.

[22] R.V. Spriggs, P.J. Artymiuk, and P. Willett. Searching for Patterns of Amino Acids in 3D Protein Structures. *J. of Chem. Inform. and Comp. Sciences*, 43(2):412–421, 2003.

[23] 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.

[24] N. Weskamp, E. Hüllermeier, D. Kuhn, and G. Klebe. Multiple graph alignment for the structural analysis of protein active sites. *IEEE Transactions on Computational Biology and Bioinformatics*, 4(2):310–320, 2007.

[25] X. Yan, P.S. Yu, and J. Han. Graph Indexing: A Frequent Structure-based Approach. In *ACM SIGMOD International Conference on Management of Data*, pages 335–346. ACM New York, NY, USA, 2004.

[26] X. Yan, P.S. Yu, and J. Han. Substructure Similarity Search in Graph Databases. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 766–777, New York, 2005.

[27] X. Yan, F. Zhu, J. Han, and P.S. Yu. Searching Substructures with Superimposed Distance. In *International Conference on Data Engineering*, volume 88, 2006.

[28] S. Zhang, M. Hu, and J. Yang. Treepi: A novel graph indexing method. In *23th International Conference on Data Engineering*, pages 966–975, 2007.