

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

Thomas Fober, Eyke Hüllermeier, Marco Mernberger

FB Mathematik und Informatik

Philipps-Universität Marburg

Hans-Meerwein-Straße

D-35032 Marburg

Tel.: +49(0)6421 28 - 21586

Fax: +49(0)6421 28 - 21573

E-Mail: {thomas, eyke, mernberger}@mathematik.uni-marburg.de

Abstract

The concept of multiple graph alignment has recently been introduced as a novel method for the structural analysis of biomolecules. Using inexact, approximate graph-matching techniques, this method enables the robust identification of approximately conserved patterns in biologically related structures. In particular, using multiple graph alignments, it is possible to characterize 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 greedy strategy.

1 Introduction

In the field of bioinformatics, multiple sequence alignment is an established approach for the identification of residues that are conserved across many members of a gene or protein family [13, 4, 12]. However, as this approach relies on evolutionary conserved sequences in DNA or protein chains to detect similarities between different molecules, it is only capable of detecting functional similarities based on heredity. Consequently, sequence analysis is not optimally suited for the identification of *functional* similarities between molecules, as functionality is more closely associated with structural than with sequential features. In fact, it is well-known that there is no strong correspondence between sequence similarity and structural similarity.

Focusing on the the identification of *structural* similarities of biomolecules, this paper presents the concept of *multiple graph alignment* (MGA) as a structural counterpart to sequence alignment. As opposed to homology-based methods, this approach allows one to capture non-homologous molecules with similar functions as well as evolutionary conserved functional domains. Accordingly, MGA offers a much wider field of application, as the method is generally applicable to different types of biological or biochemical objects (and, in principle, even beyond the bioinformatics domain). In this regard, our special interest concerns the analysis of protein structures or, more specifically, protein binding sites. However, graph alignments can also be used for analyzing other types of biomolecules, e.g., to detect similarities between chemical molecules such as protein ligands (see Section 4.2).

Our previous approach to the MGA problem makes use of a greedy algorithm [14]. In this paper, we present an alternative method using evolutionary strategies. As will be shown experimentally, the latter significantly outperforms the former with regard to the quality of alignments, albeit at the cost of an increased runtime.

The paper is organized as follows: In Section 2, we introduce the concept of a multiple graph alignment. The problem of computing an MGA is then addressed in Section 3, where an evolutionary algorithm is proposed for this purpose. Section 4 is devoted to the experimental validation of the approach, and Section 5 concludes the paper.

2 Graph-Based Modeling and Multiple Graph Alignment

2.1 Graph-Based Modeling of Biomolecules

Graph models are often used to represent and analyze three-dimensional biomolecules, i.e., single biomolecules are represented in terms of a graph G consisting of a set of (labeled) nodes V and (weighted) edges E . As mentioned previously, our special interest concerns protein binding pockets and chemical compounds.

2.1.1 Modeling Chemical Compounds

In the case of chemical compounds, atoms are represented as nodes labeled with their corresponding atom type, using the SYBYL atom type notation. The edges between nodes are labeled by the Euclidean distance between the respective atoms. This way, important geometric properties of the compound are captured. Alternatively, it is possible to use a representation in which edges correspond to molecular bonds and are weighted by the bond order.

2.1.2 Modeling Protein Binding Sites

Regarding the modeling of protein structures, the present work builds upon Cavbase [11, 9], a database system for the fully-automated detection and extraction of protein binding pockets from experimentally determined protein structures (available through the database PDB [6]). 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 [10].

To model a binding pocket as a graph, the geometrical arrangement of the pocket and its physicochemical properties are first represented by pre-defined *pseudocenters* – spatial points that represent the center of a particular property. The type and the spatial position of the centers depend on the amino acids that border the binding pocket and expose their functional groups. They are derived from the protein structure using a set of pre-defined rules [11]. As possible types for pseudocenters, hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic and aromatic properties are considered. Pseudocenters can be regarded as a compressed representation of areas on the cavity surface where certain protein-ligand interactions are experienced.

The assigned pseudocenters form the nodes $v \in V$ of the graph representation, and their properties are modeled in terms of node labels $l(v) \in \{P1, P2 \dots P5\}$, where P1 stands for donor, P2 for acceptor, etc. Two centers are connected by an edge in the graph representation if their Euclidean distance is below 11.0 \AA and each edge $e \in E$ is labeled with the respective distance $w(e) \in \mathbb{R}$.¹ The edges of the graph thus represent geometrical constraints among points on the protein surface.

In Cavbase, a graph representation of a binding pocket has around 85 nodes on average; however, also graphs with several hundred nodes are frequently detected and extremes with thousands of nodes exist. The graphs are rather dense as approximately 20 percent of all pairs of nodes are connected by an edge.

2.2 Multiple Graph Alignment

In the following, we assume a set $\mathcal{G} = \{G_1(V_1, E_1) \dots G_n(V_n, E_n)\}$ of connected, node-labeled and edge-weighted graphs to be given, each of which represents a biomolecule. To make the discussion more concrete, we subsequently consider the case of protein binding sites.

When comparing homologs from different species in protein cavity space, one has to deal with the same mutations that are also given in sequence space. 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. Thus, one cannot expect that the graph descriptors for two related binding pockets match exactly. In our approach, the following types of edit operations are allowed to account for differences between a graph $G_1(V_1, E_1)$ and another graph $G_2(V_2, E_2)$:

1. Insertion or deletion of a node $v_1 \in V_1$ (“InDel”). A pseudocenter can be deleted or introduced due to a mutation in sequence space. Alternatively, a conformational difference can affect the exposure of a functional group toward the binding pocket, accordingly an insertion or deletion in the graph descriptor could result.
2. Change of the label $l(v_1)$ of a node $v_1 \in V_1$ (“Node Mismatch”). The assigned physicochemical property (“type”) of a pseudocenter can change if a mutation replaces a certain functional group by another type of group at the same position.
3. Change of the weight $w(e_1)$ of an edge $e_1 \in E_1$ (“Edge Mismatch”). The distance between two pseudocenters can change due to conformational differences.

By assigning a cost 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. Figure 1).

¹An interaction distance of 11.0 \AA is typically enough to capture the geometry of a binding site, and ignoring larger distances strongly simplifies the graph representation.

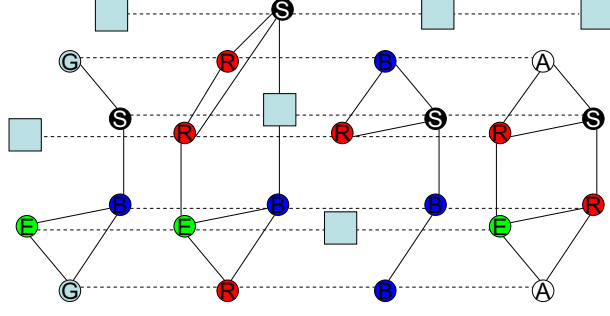


Figure 1: An alignment of four similar, but not identical graphs. The node labels are indicated by the letters assigned to the nodes (shown as circles). The edge labels are omitted for simplification. The assignments among the different graph nodes are indicated by the dashed lines. Large boxes in gray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

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

1. for all $i = 1 \dots n$ and for each $v \in V_i$ there exists exactly one $a = (a_1 \dots a_n) \in \mathcal{A}$ such that $v = a_i$ (i.e., each node of each graph occurs exactly once in the alignment).
2. for each $a = (a_1 \dots a_n) \in \mathcal{A}$ there exists at least one $1 \leq i \leq n$ such that $a_i \neq \perp$ (i.e., each tuple of the alignment contains at least one non-dummy node).

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

To assess the quality of a given alignment, a scoring function is needed. This scoring function corresponds to the above-mentioned edit distance, as each graph alignment defines a set of edit operations that have to be performed to transform one of the aligned graphs into another entry of the alignment. Here, a scoring function that follows a sum-of-pairs scheme is proposed, i.e., the score s of a multiple alignment $\mathcal{A} = (a^1 \dots a^m)$ is defined by the sum of scores of all induced pairwise alignments:

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

where the *node score* (ns) is given by

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

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

Parameter	Value
ns_m	1.0
ns_{mm}	-5.0
ns_{dummy}	-2.5
es_m	0.2
es_{mm}	-0.1

Table 1: Parameter setting used in the experiments.

match if their respective lengths, a and b , differ by at most a given threshold t_{max} , and as a mismatch otherwise. The *edge score* (es) is then given by

$$es \left(\left(\begin{array}{c} a_1^i \\ \vdots \\ a_n^i \end{array} \right), \left(\begin{array}{c} a_1^j \\ \vdots \\ a_n^j \end{array} \right) \right) = \sum_{1 \leq k < l \leq n} \begin{cases} es_{mm} & (a_k^i, a_k^j) \in E_k, (a_l^i, a_l^j) \notin E_l \\ es_{mm} & (a_k^i, a_k^j) \notin E_k, (a_l^i, a_l^j) \in E_l \\ es_m & d_{kl}^{ij} \leq t_{max} \\ es_{mm} & d_{kl}^{ij} > t_{max} \end{cases}$$

where $d_{kl}^{ij} = \|w(a_k^i, a_k^j) - w(a_l^i, a_l^j)\|$. The scores for the different cases (i.e., ns_m , ns_{mm} , ns_{dummy} , es_m , es_{mm}) have to be defined by the user. Table 1 shows the parameter setting that was used for the experiments in Section 4.

2.3 Computing Multiple Graph Alignments

The problem of calculating an optimal MGA, that is, an alignment with maximal score for a given set of graphs is computationally very complex. The subgraph isomorphism problem (which is known to be NP-complete [1]) can be seen as a special case of the graph alignment problem where the cost for mismatches is set prohibitively high. Thus, one cannot expect to find an efficient algorithm that is guaranteed to find an optimal alignment for a given set of graphs. In [14], simple and effective heuristics for the MGA problem have been described 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 the problem of pairwise alignment (i.e., calculating an optimal graph alignment for only two graphs) in a first step; these pairwise alignments are subsequently merged into a multiple alignment in a second step.

Both steps, finding optimal pairwise alignments and merging them into a multiple alignment, are only heuristics that do not guarantee to find a good solution on every problem instance. Even worse, as there is no information about the distance to the optimum, is hardly possible to get an idea about the quality a solution produced by the above strategy.

Motivated by these problems, we investigated the use of evolutionary algorithms as an alternative approach. As will be detailed in Section 3, we resorted to an evolution strategy with a non-standard representation and examined two different approaches: While the first approach seeks to optimize a complete multiple alignment directly, the second approach sticks to the decomposition strategy that is also used by the greedy algorithm and, hence, uses evolutionary optimization only for the pairwise case.

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

- Initially, a population consisting of μ random graph alignments is generated; μ is the *population size* per generation.
- 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 two parent individuals at random and submits them to the *recombination* operator. This operator generates an offspring by exchanging the information of both individuals. The new individual is further modified by the *mutation* operator.
- The offsprings are evaluated and added to the parent population. Among the individuals in this temporary population T , the plus-selection chooses the best μ candidates, which then form the population of the next generation.
- The whole procedure is repeated until no further improvements are achieved during several generations.

As mentioned above, the recombination and mutation operators must be adapted to the representation, which is in our case a matrix scheme. Our recombination operator chooses a row r of the matrix scheme at random and cuts both individuals at this position, thereby producing four sub-matrices. An offspring is then created by combining the upper block of the first individual (first sub-matrix), consisting of r rows, with the lower block of the second individual (fourth sub-matrix), consisting of $m - r$ rows. Simply switching the complete blocks is not possible, however, since the nodes are not ordered in a uniform way. Instead, to assign the columns of the first sub-matrix to columns of the fourth sub-matrix in a correct way, we use the elements of the selected row as pivot elements: If the k -th node of the r -th graph appears in column i in the first individual and in column j in the second one, the i -th row of the first sub-matrix is combined with the j -th row of the fourth sub-matrix. General experience has shown that recombination increases the speed of convergence, and this was also confirmed by our experiments.

The mutation operator selects one row and two columns at random and swaps the entries in the corresponding cells. To enable large mutation steps, this procedure is repeated 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 [5]; see Section 4.1.

3.1 Combining Evolutionary Optimization and Pairwise Decomposition

As mentioned above, the search space of an MGA problem grows exponentially with the number of graphs. Moreover, the evaluation time of the fitness function grows quadratically with the number of graphs. Therefore, the application of our EA algorithm becomes extremely expensive for large problems.

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 in our greedy approach, where the merging step has been realized by means of the star-alignment algorithm [14]. 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. 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 EA algorithm. 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.

4 Experimental Results

This section presents two experimental studies. The purpose of the first study was to optimize our evolutionary algorithms by means of a proper adjustment of its exogenous parameters, which have a strong influence on performance and runtime. In our case, the following parameters are concerned:

- the population size μ and the selective pressure ν ;
- selfadaption and recombination, which can assume values $\{\text{on}, \text{off}\}$, and allow the automatic step size control and the recombination operator to be enabled or disabled, respectively;
- 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.

In the second study, we used our algorithms to analyze two data sets with different characteristics, the first consisting of a set of small molecules (benzamidine) and the second of a set of protein binding sites (thermolisin). For comparison purpose, we also included a simple hill-climbing strategy, namely a $(1 + 1)$ -EA.

4.1 Speeding up the Evolutionary Algorithm

The main goal of this study was to reduce the runtime of the EA without sacrificing solution quality. To this end, we have experimented with the *sequential parameter optimization toolbox* (SPOT) [2] that enables a quasi-automatic adjustment of exogenous parameters. The standard parametrization by Bäck [3] gives a first idea of a useful range. We have chosen a range between 1 and 50 for the population size μ and a range between 1 and 20 for the selective pressure ν . For initial step size, we allowed values between 1 and $\frac{1}{2} \times \text{columnlength}$.

Test problems were produced by generating a random graph first, and replicating this graph (with renumbered nodes) $m - 1$ times afterward. Obviously, the optimal solution of a problem of that kind is a perfect match of these m graphs; we modified the scoring system (by a linear transformation) such that the score of this solution is 0. SPOT generates a sequence of design points in the parameter space with the goal to minimize the number of required function evaluations for reaching a fitness of zero. The experiments were conducted for alignment problems of size $m \in \{2, 4, 8, 16\}$ which, however, all yielded similar results. According to these results, the parameter configuration shown in Table 2 seems to be well-suited for the problem.

Parameter	μ	ν	selfadaption	recombination	initial step size
Settings	4	20	off	on	21

Table 2: Optimal settings of the exogenous parameters of the EA.

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.

4.2 Mining Molecular Fragment Data

As a first proof-of-concept for the algorithms presented in the previous section, we analyzed a data set consisting of 87 compounds that belong to a series of selective thrombin inhibitors and were taken from a 3D-QSAR study [7]. 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 and 16 compounds have been selected at random, and for each subset, an MGA has been calculated using the greedy heuristic (Greedy), our evolutionary algorithm with standard parametrization (EA_{std}), optimized parametrization (EA_{opt}), and in combination with a star-alignment procedure (EA^*).

Before performing these experiments, we have compared the optimized EA with a simple hill-climbing strategy, namely a $(1 + 1)$ -EA. To ensure a fair comparison, each search strategy was allowed a fixed number of fitness function evaluations. The results are summarized in Table 3. As can be seen, the $(1 + 1)$ -EA performs very poorly in comparison with EA_{opt} .

The results of the main experiment, comparing the three EA-variants with the greedy strategy, are shown in Table 4. As a measure of comparison, we derived the relative

	2 Graphs		4 Graphs	
	EA _{opt}	(1 + 1)-EA	EA _{opt}	(1 + 1)-EA
best	46.1	-143.8	95.5	-1069.1
median	-41.1	-213.4	-93.7	-1129.8
worst	-132.6	-257.9	-304.3	-1307.0
mean	-22.2	-191.425	-98.4125	-1152.4
std	74.7106	47.6079	144.6097	87.4351

	8 Graphs		16 Graphs	
	EA _{opt}	(1 + 1)-EA	EA _{opt}	(1 + 1)-EA
best	-128.8	-5300.4	-3854.3	-29597
median	-776.2	-6157.9	-6043.2	-35780
worst	-971.8	-8361.6	-8355.8	-36577
mean	-539.25	-6484.2	-5888.0	-33004
std	352.1576	1156.4	1626.4	3249.4

Table 3: Comparison between a simple hill-climbing strategy and an optimized EA.

improvement of the fitness value, defined as

$$\frac{\text{fitness(EA)} - \text{fitness(Greedy)}}{|\text{fitness(Greedy)}|}. \quad (2)$$

The results confirm our expectations: Even in its standard setting, the EA significantly outperforms the greedy algorithm, and the optimized version even improves upon these results. In general, the full EA-variants also perform better than EA*, even though there is a notable exception for one of the 4-graph problems.

	2 Graphs	4 Graphs	8 Graphs	16 Graphs
best	2.6522	2.4931	0.9819	0.9061
median	0.5798	0.7641	0.8960	0.8551
worst	0.3183	0.5536	0.8517	0.8298
mean	1.0286	1.0245	0.9198	0.8692
std	0.7279	0.6150	0.0437	0.0289
best	3.2271	2.1952	0.9668	0.9060
median	0.5798	0.8798	0.9136	0.8616
worst	0.5260	0.5536	0.8721	0.8350
mean	1.3853	1.2043	0.9174	0.8721
std	1.1326	0.5673	0.0313	0.0264
best	3.2271	11.9850	1.1423	0.6881
median	0.5798	1.0864	0.7441	0.5423
worst	0.5260	0.7967	0.7074	0.5350
mean	1.3853	3.7989	0.8761	0.6042
std	1.1326	4.8849	0.1758	0.0678

Table 4: Relative improvement for EA_{std} (top), EA_{opt} (middle), and EA* (bottom) in comparison with Greedy.

The runtimes, shown in Table 5, confirm that EA_{opt} does not only produce better results but is also more efficient than EA_{std}, reducing runtime by a factor of 10. Of course, the smallest runtime among all alternatives is still produced by the greedy strategy. A good compromise between solution quality and efficiency is achieved by EA*.

	2 graphs				4 graphs			
	Greedy	EA _{std}	EA _{opt}	EA*	Greedy	EA _{std}	EA _{opt}	EA*
best	0.219	9.766	7.703	6.934	0.953	86.255	78.692	57.383
median	0.250	14.094	11.953	12.540	1.141	129.740	106.624	80.394
worst	0.296	24.735	25.015	27.548	1.406	197.837	230.401	105.536
mean	0.252	14.804	12.651	12.874	1.1601	137.33	126.10	79.938
std	0.0241	4.177	4.7445	5.151	0.165	30.955	44.896	13.103
	8 graphs				16 graphs			
	Greedy	EA _{std}	EA _{opt}	EA*	Greedy	EA _{std}	EA _{opt}	EA*
best	4.687	1042.92	768.299	320.456	18.609	6568.8	3630.5	693.83
median	5.141	1978.23	1048.41	375.024	20.563	8643.8	5249.2	775.74
worst	6.766	2957.48	4009.34	474.978	28.407	21411.8	16008.5	1731.6
mean	5.553	1945.35	1305.6	383.86	22.296	11275	7102.9	999.69
std	0.790	533.08	784.74	42.494	3.7992	4904.3	3870.9	407.61

Table 5: Comparison of runtimes of the search strategies.

Regarding the aforementioned benzamidine core fragment, which consists of 25 atoms (11 hydrogens), it turned out that it was fully conserved, throughout all experiments, in all the solutions produced by our EA. For the greedy strategy, the corresponding degree of conservation was significantly smaller.

4.3 Mining Protein Binding Pockets

As our main interest concerns the characterization of protein binding pockets, we subsequently examined the performance of our algorithms on a second data set consisting of 74 structures derived from the Cavbase database. Each structure represents a protein binding pocket belonging to the protein family of thermolysine, bacterial proteases frequently used in structural protein analysis and annotated with the E.C. number 3.4.24.27 in the ENZYME classification database. The data set is suited for our purpose, as all binding pockets belong to the same enzyme family and, therefore, should share evolutionary related, highly conserved substructures. On the other hand, with binding pockets ranging from about 30 to 90 pseudocenters, the data set is also diverse enough to present a real challenge for graph matching techniques.

Again, 100 graph alignments of size 2, 4, 8, and 16 were produced, respectively, for randomly chosen structures, and the results are summarized in Table 6. The improvements achieved by the EA are not as high as in the previous experiment, but still significant. The most important difference is observed for the star-alignment approach, which now performs rather poorly. This is probably due to the much higher diversity of the structures in this experiment, which makes an aggregation of sub-alignments more difficult, and a heuristic like star-alignment more likely to fail.

5 Conclusions

Multiple graph alignment (MGA) has recently been introduced as a novel method for analyzing biomolecules on a structural level. Using robust, noise-tolerant graph matching

	2 graphs		4 graphs		8 graphs		16 graphs	
	EA _{opt}	EA*	EA _{opt}	EA*	EA _{opt}	EA*	EA _{opt}	EA*
best	0.753	0.749	0.739	0.1391	0.786	0.226	0.7887	0.1713
median	0.576	0.576	0.732	0.0898	0.725	0.070	0.7523	0.1013
worst	0.434	0.431	0.713	0.0375	0.721	0.047	0.7474	0.0812
mean	0.595	0.594	0.728	0.0869	0.755	0.145	0.7613	0.1228
std	0.117	0.115	0.01	0.0344	0.032	0.077	0.0171	0.0342
best	14.1	13.01	895.3	517.9	8971.7	2147.1	16005.0	2269.0
median	65.3	71.05	2109.1	713.6	21878.9	2861.0	51198.7	3039.7
worst	267.9	259.18	3112.8	970.7	39681.1	5207.7	93706.2	3493.1
mean	95.4	93.48	2041.5	736.83	21929.0	3438.3	50665.5	2972.1
std	85.1	78.17	679.46	103.15	9765.7	1097.6	25488	383.98

Table 6: Relative improvement (upper block) and runtime (lower block) of EA_{opt} and EA*.

techniques, MGA is able to discover approximately conserved patterns in a set of graph-descriptors representing a family of evolutionary related biological structures. As the computation of optimal alignments is a computationally complex problem, this paper has proposed an evolutionary algorithm (EA) as an alternative to a hitherto existing greedy strategy.

Our experiments, carried out on several data sets with different characteristics, have shown the high potential of this approach and give rise to the following conclusions: The EA significantly outperforms the greedy strategy and is able to produce alignments of much higher quality, albeit at the cost of a considerable increase in runtime. In fact, as runtime seems to increase exponentially with the number of graphs to be aligned, we also considered a combination of evolutionary optimization with decomposition techniques, namely the star-alignment method. This approach appears to be a reasonable compromise as it achieves a good trade-off between solution quality and runtime.

As part of ongoing work, we currently elaborate on decomposition techniques in more detail. In particular, we investigate alternative decomposition schemes and aggregation procedures. An especially interesting idea in this regard is to replace a heuristic aggregation strategy again by an evolutionary optimization step. This line of research is complemented by more standard means to improve evolutionary optimization, such as alternative representations and more sophisticated genetic operators.

References

- [1] Atallah, M. J. (editor): *Algorithms and Theory of Computation Handbook*. CRC Press LLC. 1999.
- [2] Bartz-Beielstein, T., Lasarczyk, C. and Preuß, M.: *Sequential Parameter Optimization Toolbox. Technical Report, Universität Dortmund, Germany*. 2006.
- [3] Bäck, T.: *Evolutionary Algorithms in Theory and Practise. Dissertation, Universität Dortmund, Germany*. 1994.
- [4] Bateman, A., Coin, L., Durbin, R., Finn, R. D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E. L. L., Studholme, D. J.,

- Yeats, C. and Eddy, S. R.: The Pfam protein families database. *Nucl. Acids. Res.*, 32(90001):D138–141. 2004.
- [5] Beyer, H.-G. and Schwefel, H.-P.: Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1):3–52. 2002.
- [6] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. and Bourne, P.E.: The protein data bank. *Nucleic Acids Research*, 28:235–242. 2000.
- [7] Böhm, M., Stürzebecher, J. and Klebe, G.: 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.
- [8] Bron, C. and Kerbosch, J.: Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 16(9):575–577. 1973.
- [9] Hendlich, M., Bergner, A., Günther, J. and Klebe, G.: Relibase: Design and Development of a Database for Comprehensive Analysis of Protein-Ligand Interactions. *Journal of Molecular Biology*, 326:607–620. 2003.
- [10] Hendlich, M., Rippmann, F. and Barnickel, G.: LIGSITE: Automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15:359–363. 1997.
- [11] Schmitt, S., Kuhn, D. and Klebe, G.: A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology. *J. Mol. Biol.*, 323(2):387–406. 2002.
- [12] Servant, F., Bru, C., Carrère, S., Courcelle, E., Gouzy, J., Peyruc, D. and Kahn, D.: Prodom: Automated clustering of homologous domains. *Briefings in Bioinformatics*, 3(3):246–251. 2002.
- [13] Thompson, J. D., Higgins, D. G. and Gibson T. J.: 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.
- [14] Weskamp, N., Hüllermeier, E., Kuhn, D. and Klebe, G.: Graph Alignments: A New Concept to Detect Conserved Regions in Protein Active Sites. *German Conference on Bioinformatics 2004*, 131–140. 2004.