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

Thomas Fober, Eyke Hüllermeier, Marco Mernberger
FB Mathematik/Informatik, Philipps-Universität Marburg

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, multiple graph alignments enable 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 greedy strategy. Empirically, it is shown that the former yields significantly better results than the latter, albeit at the cost of an increased runtime.

1 Introduction

Focusing on 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. 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.

The problem of comparing graphs occurs in many applications and, correspondingly, has been studied in different research fields, including pattern recognition [5], network analysis [2] and kernel-based machine learning [6, 4]. These approaches, however, almost exclusively focus on the comparison of two graphs, while our method, in analogy to multiple sequence alignment, seeks to analyze multiple graphs simultaneously. Moreover, most existing approaches target on exact matches between graphs or parts thereof, often resorting to the concept of subgraph isomorphism [8].

This work draws on [10], in which the concept of MGA was first introduced. That paper proposed an 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, albeit at the cost of an increased runtime.

The paper is organized as follows: Subsequent to a brief introduction to graph-based mo-

deling of protein binding sites in Section 2, we introduce the concept of a multiple graph alignment in Section 3. 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 validation of the approach, and Section 6 concludes the paper.

2 Graph-Based Modeling of Protein Binding Sites

In bio- and chemoinformatics, single biomolecules 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, our special interest concerns the modeling of protein binding pockets. More specifically, our work builds upon Cavbase [9], a database system for the automated detection, extraction, and storing 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 [7].

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 [9]. As possible types for pseudocenters, hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic, metal ion, pi (accounts for the ability to form π–π interactions) 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 `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 a certain threshold 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.

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

[1] 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 fitness calculation.

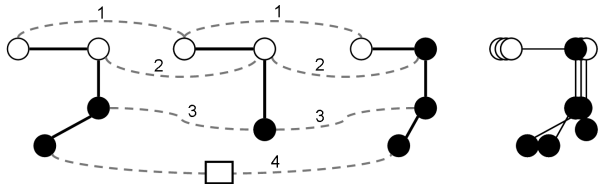


Figure 1: Simple illustration of MGA by an approximate match of three graphs with two types of labels (black and white). Mutual assignments of nodes are indicated by dashed lines. Note that the second assignment involves a mismatch, since the node in the third graph is black. Likewise, the fourth assignment 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.

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 $G_1(V_1, E_1)$ and another graph $G_2(V_2, E_2)$. **Insertion or deletion** of a node $v_1 \in V_1$: A pseudocenter can be deleted or introduced 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. **Label mismatch**, i.e., a change of the label $\ell(v_1)$ of a node $v_1 \in V_1$: 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. **Node mismatch**, i.e., a change of the weight $w(e_1)$ of an edge $e_1 \in E_1$: 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. Fig. 1).

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

Each $a \in \mathcal{A}$ corresponds to a vector of mutually assigned nodes from the graphs $G_1 \dots G_n$.

Note that, by matching nodes, a mutual assignment of edges is determined in an implicit way. To assess the quality of a given alignment, a scoring function is is used that 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. Our scoring function follows a sum-of-pairs scheme, 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:

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

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

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

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 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

$$\mathrm{es}\left(\begin{pmatrix} a_1^i \\ \vdots \\ a_m^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_m^j \end{pmatrix}\right) = \sum_{1 \le k < l \le m} \begin{cases} \mathrm{es}_{mm} & (a_k^i, a_k^j) \in E_k,\ (a_l^i, a_l^j) \notin E_l \\ \mathrm{es}_{mm} & (a_k^i, a_k^j) \notin E_k,\ (a_l^i, a_l^j) \in E_l \\ \mathrm{es}_m & d_{kl}^{ij} \le \epsilon \\ \mathrm{es}_{mm} & d_{kl}^{ij} > \epsilon \end{cases}$$

where $d_{kl}^{ij} = \|w(a_k^i, a_k^j) - w(a_l^i, a_l^j)\|$. 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 in [10]: $\mathrm{ns}_m = 1$, $\mathrm{ns}_{mm} = -5$, $\mathrm{ns}_d = -2.5$, $\mathrm{es}_m = 0.2$, $\mathrm{es}_{mm} = -0.1$, $\epsilon = 0.2$.

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 [10], 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 the problem of pairwise alignment (i.e., calculating an optimal graph alignment for only two graphs) in a first step. Resorting to the idea of star-alignment, which is well-known in sequence analysis, these pairwise alignments are subsequently 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

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 [3]: (i) Initially, a population consisting of μ individuals, each representing a candidate solution, is generated at random; μ specifies the *population size* per generation. (ii) 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. (iii) 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 then form the population of the next generation. (iv) The whole procedure is repeated until a stopping criterion is met.

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 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 the number 0) of these graphs.

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 $n_1 + \ldots + n_m$, where $n_i = |V_i|$. 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 $n + 1$ columns, where $n = \max\{n_1, n_2 \ldots n_m\}$. For each graph G_i, a permutation of its nodes is then inserted, with dummies replacing the index positions $j > |V_i|$. As an aside, 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. 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, the ordering of the first sub-matrix is used as a reference, i.e., the elements of the first row serve as pivot elements. General experience has shown that recombination increases the speed of convergence, and this was also confirmed by our experiments (see Section 5).

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 [3]. However, our experiments indicated that a simple mutation operator performing only single swaps solves the problem most effectively.

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 in the greedy approach, where the merging step has been realized by means of the star-alignment algorithm [10]. 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 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. 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

In a first step, we adjusted the following exogenous parameters of our EA using the *sequential parameter optimization toolbox* (SPOT) [1] in combination with suitable synthetic data: 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.

After optimizing the parameters on diverse datasets, the following parameter configuration turned out 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 Protein Binding Pockets

We examined the performance of our algorithms 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 classification 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.

We produced 100 graph alignments of size 2, 4, 8, 16 and 32, respectively, for randomly chosen structures, using the greedy heuristic (Greedy), our evolutionary algorithm with optimized parametrization (EA), and in combination with a star-alignment procedure (EA*). 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 EA (or EA*) and Greedy. This measure is positive if the EA (EA*) 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).

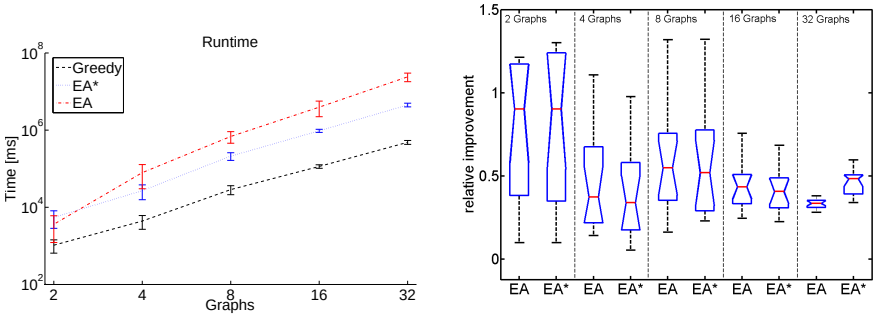
The results are summarized in Fig. 2. As can be seen, the EA solutions are never worse and often significantly better than the Greedy solutions. In terms of runtime,[2] it is clear that Greedy is still more efficient. Yet, a good compromise between solution quality and efficiency is achieved by EA*, as the runtime is much better than for EA, especially for a larger number of graphs.

5.2 Influence on Similarity Retrieval

Pairwise similarity scores are often used to rank the objects stored in a database with respect to a given query object. For this purpose, the *absolute* similarity degrees are less important than the *relative* ones. Consequently, one may ask whether our EA, in addition to finding alignments with higher score, does actually yield rankings that differ from those produced by the Greedy algorithm. This is not self-evident since, for example, a constant improvement by a factor c, the same for each pairwise alignment, would not have any influence on a ranking.

Therefore, we compared 26 protein cavities belonging to the ClpP proteasome complex of *E. coli* with a set of 964 other cavities using EA and Greedy, respectively. Thus, we generated 2 sets of 25064 pairwise alignments and ranked the alignments according to

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



(a) Runtimes

(b) Relative improvements (boxplot)

Figure 2: Results of the first experiment: (a) Runtimes in milliseconds (mean and standard deviation) of greedy heuristic, EA using star alignment decomposition (EA*) and pure EA. (b) Relative improvements as defined in (2).

their score. We subsequently compared the generated rankings by computing the overlap of top-k ranks for both algorithms. This is done by calculating the intersection I of the top-k lists from the EA and the Greedy ranking. The results in terms of $k \mapsto f(k) = \frac{1}{k}|I|$ mappings are shown in Fig. 3. As one can see, the rankings produced by both algorithms significantly differ with respect to their top ranks. As indicated by a value $f(k) = 0$, most rankings (one ClpP cavity compared to 964 others) do not share any cavity in their top positions. In fact, there are only three rankings that share a few cavities in their top-10 lists. Although some curves appear to start increasing rather soon, one has to keep in mind that the real interest is most often focused on the top-positions only.

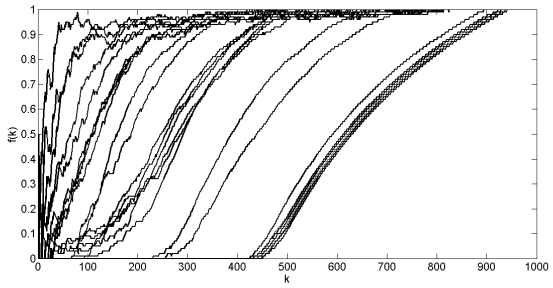


Figure 3: Top$_k$-Cuts showing the overlap in the top k ranks ($k = 1 \ldots 964$) for 26 protein cavities each compared to 964 other cavities.

6 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 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 an existing greedy strategy.[3]

Our experiments have shown the high potential of this approach and give rise to the following conclusions: The EA is computationally more complex but significantly outperforms the greedy strategy in terms of MGA scores. The alignments produced by the EA are better in the sense that conserved substructures are discovered more reliably. Besides, the improved similarity computation also leads to better performance in similarity retrieval. Finally, a reasonable compromise between solution quality and runtime is achieved by a combination of evolutionary optimization with binary decomposition techniques.

Literatur

[1] Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation—The New Experimentalism. Springer-Verlag. 2006.

[2] J. Berg and M. Lassig.: Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences*, 101(41):14689, 2004.

[3] Beyer, H.-G. and Schwefel, H.-P.: Evolution strategies – A comprehensive introduction. *Natural Computing*, 1(1):3–52. 2002.

[4] Borgwardt, K. M., and Kriegel, H.-P.: Shortest-path kernels on graphs. *Proc. Intl. Conf. Data Mining*, 74 - 81. 2005.

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

[6] Gärtner, T.: A survey of kernels for structured data. *SIGKKD Explorations*, 5(1): 49 - 58. 2003.

[7] 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.

[8] Raymond, J., and Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16: 521–533. 2002.

[9] 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.

[10] Weskamp, N., Hüllermeier, E., Kuhn, D. and Klebe, G.: Multiple Graph Alignment for the Structural Analysis of Protein Active Sites. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):310–320, 2007.

[3] An implementation of this algorithm along with a user's guide can be downloaded at `http://www.uni-marburg.de/fb12/kebi/research/software`.