Multiple Geometrical Alignments for the Analysis of Protein Active Sites

Thomas Fober and Eyke Hüllermeier

Department of Mathematics and Computer Science
University of Marburg, Germany
{thomas,eyke}@informatik.uni-marburg.de

Abstract

Geometric objects are often represented approximately in terms of a finite set of points in three-dimensional Euclidean space. In this paper, we extend this representation to what we call labeled point clouds. A labeled point cloud is a finite set of points, where each point is not only associated with a position in three-dimensional space, but also with a discrete class label that represents a specific property. This type of model is especially suitable for modeling biomolecules such as proteins and protein binding sites, where a label may represent an atom type or a physico-chemical property. Proceeding from this representation, we address the question of how to compare two labeled points clouds in terms of similarity. Using fuzzy modeling techniques, we develop a suitable similarity measure as well as an efficient evolutionary algorithm to compute it. Having calculated the optimal superposition it is easy to establish an *alignment* in the sense of a one-to-one correspondence between the basic units of two or more protein structures. From a biological point of view, alignments of this kind are of great interest, as they offer important information about evolution, heredity, and the mutual correspondence between molecular constituents. In this paper, we therefore additionally developed a method for computing pairwise or multiple alignments of protein structures on the basis of labeled point cloud superpositions.

1 Introduction

Geometric objects are often represented in terms of a set of points in three-dimensional Euclidean space. This type of representation is finite and hence approximate (even though the number of points can become very large, as for example in laser range scanning), focusing on the most important characteristics of the object while ignoring less important details. A well-known example of a representation of this kind is the *Molfile* format [10], where molecules are described in terms of the spatial coordinates of all atoms. However, since not only the position but also the type of an atom is of interest, this representation is not a simple point cloud. Likewise, other biomolecular structures, such as proteins and protein binding sites, are not only characterized by their geometry but also by additional features, such as physico-chemical properties. In this paper, we therefore introduce the concept of a *labeled point cloud.* A labeled point cloud is a finite set of points, where each point is not only associated with a position in three-dimensional space, but also with a discrete class label that represents a specific property. Formally, a labeled point cloud P is a set of points $\{p_1, \ldots, p_n\}$ with two associated functions: $c : P \rightarrow \mathbb{R}^3$ maps points to coordinates in the Euclidean space, and $\ell : P \rightarrow \mathcal{L}$ assigns a label to each point.

Since theory formation in the biological sciences is largely founded on similarity-based and analogical reasoning principles, the comparison of two (or more) objects with each other is a fundamental problem in bioinformatics. To compare two point clouds, the authors in [14] make use of a measure based on the *Gromov-Hausdorff distance* of sets. This approach is limited to unlabeled point clouds, however. Another possibility is to transform a labeled point cloud into a (labeled) graph first, capturing, in one way or the other, geometrical information in terms of edges, and to apply graph matching techniques afterward. This strategy was recently proposed in [1], where the use of *graph kernels* as similarity measures [5, 6, 9] has been especially advocated. At first sight, this idea looks appealing, especially since methods for comparing graphs abound in the literature. Nevertheless, it also comes with a number of disadvantages. For example, many techniques for matching and comparing graphs capture aspects of similarity which are reasonable for graphs but not necessarily for geometric objects. Besides, graph matching techniques are typically quite complex from a computational point of view.

Perhaps most importantly, however, a graph representation captures the geometrical information only in an *implicit* way, namely through the presence, absence, and possibly the labeling of edges. Moreover, the transformation is often not even lossless. Matching objects while obeying geometrical constraints can then become troublesome, since the geometrical information is not explicitly available. Instead, it must be reconstructed from the graph representation whenever needed.

As an alternative to an indirect approach of that kind, we therefore propose the method of *labeled point cloud superposition* (LPCS), which operates on labeled point clouds directly. Thus, it preserves as much geometrical information as possible and facilitates the exploitation thereof. Related to the concept of an LPCS, we introduce a similarity measure which makes use of modeling techniques from fuzzy set theory. This measure proceeds from the idea of equivalence (inclusion) of point clouds in a set-theoretic sense, but is tolerant toward exceptions (on the level of label information) and geometric deformations.

Yet, in contrast to methods for *multiple graph alignment* as recently introduced in [17], LPCS does not establish a one-to-one correspondence between the basic units of two or more protein structures. From a biological point of view, alignments of this kind are of great interest, as they offer important information about evolution, heredity, and the mutual correspondence between molecular constituents. Additionally, we therefore develop a method for computing pairwise or multiple alignments of protein structures on the basis of labeled point cloud superpositions.

The remainder of the paper if organized as follows. Subsequent to a brief introduction to protein binding sites and their representation in Section 2, we introduce the concept of LPCS in Section 3. The problem of computing an LPCS is then addressed in Section 4, where an evolution strategy is proposed for this purpose. Section 5 introduces the concept of multiple geometrical alignments. Section 6 is devoted to the experimental validation of the approach, and Section 7 concludes the paper.

2 Modeling Protein Binding Sites

In this paper, our special interest concerns the modeling of protein binding sites. More specifically, our work builds upon CavBase [15], a database for the automated detection, extraction, and storing of protein cavities (hypothetical binding sites) from experimentally

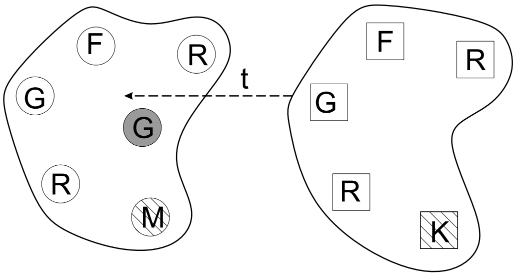


Figure 1: Two point clouds A (left, points as circle) and B (right, points as squares): The intra-point distances are the same in both point clouds, except for the additional gray point in A. Labels are depicted as letters within the circles and boxes, respectively.

determined protein structures (available through the PDB). In CavBase, a set of points is used as a first approximation to describe a binding pocket. The database currently contains 113,718 hypothetical binding sites that have been extracted from 23,780 publicly available protein structures using the LIGSITE algorithm [11].

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 [15]. 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. Obviously, just like in the case of Molfile, this representation is already in the form of a labeled point cloud: pseudocenters are given with their coordinates and labels, so that no further transformation is needed.

3 Labeled Point Cloud Superposition

Intuitively, two labeled point clouds are similar if they can be spatially superimposed. That is, by fixing the first and "moving" the second one (as a whole, i.e., without changing the internal arrangement of points) in a proper way, an approximate superposition of the two structures is obtained. More specifically, we will say that two point clouds are well superimposed if, for each point in one of the structures, there exists a point in the other cloud which is spatially close and has the same label. As an illustration, the example in Fig. 1 shows two point clouds A and B, for simplicity only in two dimensions. By moving B to the left (or A to the right), a superposition can be found so that, except for the hatched and gray nodes, all points in A spatially coincide with a corresponding point in B having the same label, and vice versa. So, A and B can be considered as being similar, at least to some extent.

More formally, let

$$A = \{(x_1, \ell(x_1)), \ldots, (x_m, \ell(x_m))\}$$

be a point cloud consisting of m points $x_i = (x_{i1}, x_{i2}, x_{i3}) \in \mathbb{R}^3$ with associated label $\ell(x_i) \in \mathcal{L}$, where $\mathcal{L}$ is a discrete set of labels (in the context of modeling protein binding sites, as discussed in the previous section, $\mathcal{L}$ is given by the seven types of pseudocenters). Moreover, let

$$B = \{(y_1, \ell(y_1)), \ldots, (y_n, \ell(y_n))\}$$

be a second point cloud to be compared with A. In the following, we define a function $\mathrm{SIM}(\cdot, \cdot)$ that returns a degree of similarity between two such structures A and B.

Roughly speaking, we consider similarity as a generalized (fuzzy) equivalence, which we in turn reduce to two inclusion relations, namely the inclusion of A in B and, vice versa, of B in A. Thus, we are first of all interested in whether each point $y \in B$ is also present in A (and each point $x \in A$ also present in B). For a fixed $y \in B$, we define the membership degree of this point in A by

$$\mu_A(y) \;=\; \exp\left(-\gamma \cdot d(y, A)\right) \;\;, \tag{1}$$

where

$$d(y, A) = \min_{\substack{x \in A \\ \ell(x)=\ell(y)}} \|y - x\|_1$$

is the distance between a point $y \in B$ and the closest point $x \in A$ having the same label ($d(y, A) = \infty$ and hence $\mu_A(y) = 0$ if no such point exists); for $x \in A$, $\mu_B(x)$ and $d(x, B)$ are defined analogously.

In its proper sense, the inclusion of a set B in a set A means that *each* point $y \in B$ is also contained in A or, stated differently, if a point y is in B, then it is also present in A. If membership is a matter of degree, i.e., if A and B are fuzzy sets, this condition is often formalized in terms of a fuzzy implication [16]:

$$\min_{y \in B} \left(\mu_B(y) \rightarrow \mu_A(y)\right) \;\; .$$

Here, the minimum operator plays the role of a generalization of the universal quantifier. In our case, $\mu_B(y) \equiv 1$, so that the above expression can be simplified as follows:

$$inc(B, A) \;=\; \min_{y \in B} \mu_A(y) \;\; . \tag{2}$$

However, a universal quantification (modeled by the $\min$ operator) is too strict in our biological context, where data is typically inexact and noisy. To relax this definition of fuzzy inclusion, we replace the minimum by a fuzzy quantifier Q, which is specified in the form of a non-decreasing $[0, 1] \longrightarrow [0, 1]$ mapping [19, 8]. This leads to

$$inc(B, A) \;=\; \min_{i=1\ldots|B|} \;\; \max\{Q(i/|B|), m_i\} \;\;,$$

where m_i is the i-th largest membership degree in the fuzzy set $\{\mu_A(y) \,|\, y \in B\}$. (Note that we recover (2) for Q defined by $Q(1) = 1$ and $Q(t) = 0$ for $0 \leq t < 1$.) Here, we simply take Q as the identical mapping $t \mapsto t$. Roughly speaking, $inc(B, A)$ thus defined can be interpreted as the generalized truth degree of the proposition that A is *almost* contained in B. The degree of inclusion of A in B, $inc(A, B)$, is defined analogously.

As mentioned above, the idea of our approach is to define the similarity between two labeled point clouds in terms of the best superposition of these two clouds. Therefore, let

$\mathrm{TF}(\cdot, t)$ be a function that moves a point cloud via rotation and translation, as specified by the six-dimensional vector $t = (\theta_1, \theta_2, \theta_3, \delta_1, \delta_2, \delta_3) \in [0, 2\pi]^3 \times \mathbb{R}^3$. Thus,

$$B^* = \mathrm{TF}(B, t) = \{(y_1^*, \ell(y_1^*)), \ldots, (y_n^*, \ell(y_n^*))\}$$

is the point cloud obtained by translating the point cloud B by $\delta = (\delta_1, \delta_2, \delta_3)$ (which means adding δ to each point $y \in B$) and rotating the result thus obtained by the angles θ_1, θ_2, and θ_3. Note that this operation leaves the label information unchanged (i.e., $\ell(y_i) = \ell(y_i^*)$). The position-invariant degree of inclusion of B in A is then given by

$$\mathrm{INC}(B, A) \; = \max_{t \in [0,2\pi]^3 \times \mathbb{R}^3} inc(\mathrm{TF}(B, t), A) \;\; , \tag{3}$$

and $\mathrm{INC}(A, B)$ is defined analogously.

Based on these degrees, the similarity between A and B, in the sense of a generalized equivalence, can be defined as

$$\mathrm{SIM}(A, B) \; = \; \min\{\, \mathrm{INC}(A, B), \; \mathrm{INC}(B, A) \,\} \;\; . \tag{4}$$

It is worth mentioning, however, that (4) is not always appropriate, especially if A and B greatly differ in size. In some applications, it makes sense to have a high similarity degree even if A is only a substructure of B, for example if A is a subpocket of B containing the most important catalytic residues (while the rest of the binding site B is functionally less important). Obviously, this is not guaranteed by (4). An interesting generalization, therefore, is to let

$$\begin{aligned} \mathrm{SIM}(A, B) =& \alpha \cdot \min\{\mathrm{INC}(A, B), \mathrm{INC}(B, A)\} \\ &+ (1 - \alpha) \cdot \max\{\mathrm{INC}(A, B), \mathrm{INC}(B, A)\} \;\; . \end{aligned} \tag{5}$$

Formally, this similarity measure can be motivated from a fuzzy logical point of view as follows. Considering the $\min$ ($\max$) operator as a generalized conjunction (disjunction), the first (second) combination of the two inclusion degrees is the truth degree of the proposition that A is contained in B AND (OR) B is contained in A. A conjunctive combination of the two degrees of inclusion is obviously more demanding than a disjunctive one, as the former requires equality between A and B while the latter only requires inclusion of A in B or B in A. The measure (5), which formally corresponds to an OWA (ordered weighted average) combination of the two degrees of inclusion [18], achieves a trade-off between these two extreme aggregation modes, which is controlled by the parameter $\alpha \in [0, 1]$: The closer α is to 0, the closer the aggregation is to the maximum, i.e., the less demanding it becomes. The optimal α is application-specific and depends on the purpose of the similarity measure.

4 Solving the LPCS Problem

The computation of the similarity (5) involves the solution of a real-valued optimization problem, namely the problem of finding an optimal vector t in (3) and, thus, an optimal point cloud superposition. The objective function to be maximized here is highly nonlinear and multimodal. As an illustration, Fig. 2 shows the objective function obtained for the superposition of a randomly generated two-dimensional point cloud A (in which

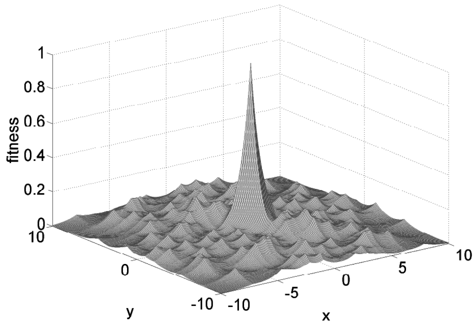


Figure 2: Example of an LPCS objective function.

all points have the same label) with itself. This function maps each two-dimensional translation vector $t = (x, y)$ to the corresponding similarity degree between $\mathrm{TF}(A)$ and A (where we used $\alpha = 0.5$ in (5) and did not consider rotation). As can be seen, there is a sharp peak at $t = (0, 0)$, which corresponds to the optimal superposition. Surrounding this solution, however, there are also many local optima.

The problem of local optima also becomes clear from the small example in Fig. 1. Moving the point cloud A from left to right, into the direction of B, has the following effect: First, a good superposition of two sub-clouds will be found, namely the right part of cloud A and the left part of cloud B. This results in a local maximum. Moving A further to the right leads to a larger local maximum (sub-clouds are growing), until the global maximum will eventually be reached.

4.1 Evolution Strategies

To solve the LPCS problem, we resort to *evolution strategies* (ES), a population-based, stochastic optimization method inspired by biological evolution and specifically developed for real-valued optimization problems [3]. An evolution strategy is based on a population, a set of μ (sub-optimal) candidate solutions that are initially spread randomly over the search space. In each generation, new solutions are generated by applying the genetic operators *recombination* and *mutation*. Recombination randomly selects ρ individuals from the current population and combines them to a new solution. Mutation takes this solution and shifts it randomly in the search space. An ES produces $\lambda = \lceil \mu \cdot \nu \rceil$ offsprings per iteration, so that this procedure has to be repeated λ times. A selection operator implements the "survival of the fittest" principle by picking the best individuals for the new population. There are two kinds of selection: The *plus*-selection chooses the best μ individuals among the offsprings plus the parents, while the *comma*-selection ignores the parent generation (this requires $\nu > 1$). A main advantage of the ES is its self-adaptation mechanism that controls the step sizes used in the mutation operator. One property of this mechanism (the advantage during optimization is obvious) is that step sizes decrease dramatically if the optimization reached a maximum. This property can be used as a qualitative termination criterion (stop when the largest step size falls below a given threshold).

Population-based optimization methods are especially advantageous for highly multimodal problems. Using a large population leads to an increased probability to generate a candidate solution in a region where the direction of descent points to the global maximum. Choosing the membership function (1) as a strictly monotone decreasing function which converges to zero ensures to have this direction in each point $t \in [0, 2\pi]^3 \times \mathbb{R}^3$ and thus greatly simplifies the maximization problem. However, our experiments indicated that the solution we found was most often only a local maximum. Therefore, we propose to use *fast restarts* of the ES. This means that the ES is started n times using comma-selection and weak termination criteria to achieve a large and quick but inexact exploration of our search space. We thus obtain n results in total. In a last step, we use the ES with plus-selection and strong termination criterion. Additionally, we include the best solution so far in the start population. The last run of the ES usually yields a globally optimal degree of similarity.

4.2 Complexity

Even though evolution strategies are generally known to be quite efficient solvers, the concrete complexity does of course depend on the application at hand. The application-specific part is the fitness function, i.e., the objective function to be optimized. This function has to be evaluated frequently and, therefore, is an important factor for the runtime. In our case, this function is given by the similarity measure (5), and its evaluation is strongly dominated by the nearest neighbor search which has to be conducted for each single point in both structures (recall that, according to (1), membership degrees are determined by the distance to closest points with the same label).

There exist a lot of data structures for supporting nearest neighbor search; see e.g. [7]. The most efficient among them need time $\mathcal{O}(n \log^2 n)$ for construction and $\mathcal{O}(\log^3 n)$ for answering a query. Unfortunately, we are not aware of an approach that allows for updating a data structure in an efficient and dynamic way. This would be desirable for our problem, in which the point clouds permanently change (the point cloud associated with an individual changes in each iteration). Instead, conventional approaches necessitate a construction from scratch in every iteration.

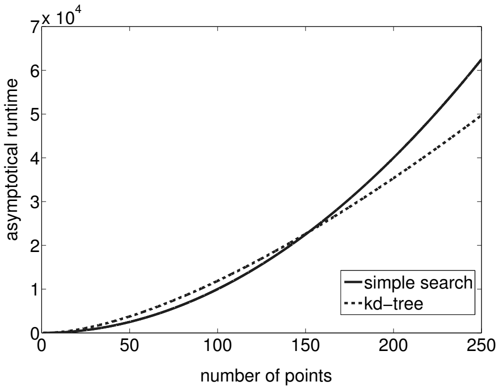


Figure 3: Runtime of a simple procedure and a more complex data structure as a function of the number of points.

Fig. 3 compares the runtimes, as a function of the number of points, for two approaches: (1) The use of a kd-tree data structure, which is reconstructed in each iteration and then used for query processing. (2) The use of a simple linear data structure, in which the points are stored in a fixed order. It needs linear instead of logarithmic time to answer a query but, on the other hand, does not cause additional costs for reconstruction. As can be seen, the use of a more complex approach pays off only for sufficiently large point clouds: The kd-tree reaches a break-even point at approximately 150 points.

In our application, we are mainly concerned with protein binding sites, which are characterized by around 180 points on average (even though much larger structures do of course exist). The use of a complex data structure did therefore not pay off. Nevertheless, we increased efficiency by hashing the points x_i of a point cloud, using the label $\ell(x_i) \in \mathcal{L}$ as a key. Since nearest neighbors are only searched among points having the same label, this obviously reduces runtime by a factor of approximately $|\mathcal{L}|$.

5 Multiple Geometrical Alignment

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 point cloud descriptor. For example, a pseudocenter can be deleted or introduced due to a mutation in sequence space. Likewise, if a mutation replaces a certain functional group by another type of group at the same position, the physicochemical property of a pseudocenter can change. Finally, the distance between two pseudocenters can change due to conformational differences.

Due to the above reasons, one cannot expect that point clouds of two related binding pockets match exactly. When looking for an alignment of two structures in the form of a one-to-one correspondence between pseudocenters, it is therefore necessary to allow for mismatches as well as pseudocenters for which no matching partner is defined. This situation is quite similar to sequence alignment, where mismatches between symbols and the insertion of blanks (to compensate for non-existing matching partners) is also allowed.

In this paper, we derive alignments from labeled point cloud superpositions and, therefore, refer to the latter as *geometric alignments*.

Definition 1 (Multiple Geometrical Alignment) *Let $\mathcal{P}$ be a set of m point clouds $P_i = \{p^i_1, \ldots, p^i_{n_i}\}$, $i = 1, \ldots, m$. A multiple geometrical alignment of these point clouds is a subset $\mathcal{A} \subseteq (P_1 \cup \{\perp\}) \times \cdots \times (P_m \cup \{\perp\})$ with the following properties:*

1. *for all $i = 1 \ldots m$ and for each $p \in P_i$ there exists exactly one $a = (a_1 \ldots a_m) \in \mathcal{A}$ such that $p = a_i$;*
2. *for each $a = (a_1 \ldots a_m) \in \mathcal{A}$ there exists at least one $1 \leq i \leq n$ such that $a_i \neq \perp$.*

Here, the symbol $\perp$ denotes a "dummy point" which is needed to compensate for non-existing matching partners.

Each tuple in the alignment represents a mutual assignment of m points, one from each point cloud P_i (possibly a dummy). Thus, the second property in the above definition requires that each tuple of the alignment contains at least one non-dummy point, and the first property means that each point of each point cloud occurs exactly once in the alignment. While these properties can be satisfied by a large number of alignments, we are of course looking for an alignment in which mutually assigned points have the same label and nearby spatial positions.

5.1 Construction of pairwise alignments

To construct a pairwise alignment of two point clouds P_1 and P_2, we reduce the alignment problem to a problem of optimal assignment. To this end, we need a square matrix $M = (m_{i,j})$, where $m_{i,j} \in \mathbb{R}$ defines the costs for assigning point $p_i \in P_1$ to point $p_j \in P_2$. According to definition 1, the maximal length of a pairwise alignment is $n = n_1 + n_2 = |P_1| + |P_2|$. Therefore, to consider all possible alignments, the matrix M has size $n \times n$.

The entries $m_{i,j}$ are derived from the optimal superposition of point clouds P_1 and P_2 as produced by our LPCS method. Since this approach calculates in sum two independent t-vectors (one for each INC function) we had to modify this approach slightly. Instead of using eq. (4) we define the similarity as

$$\mathrm{SIM}_{\mathrm{3DA}}(A, B) \;=\; \max_{t \in [0,2\pi]^3 \times \mathbb{R}^3} \frac{1}{2} inc(\mathrm{TF}(B,t), A) + \frac{1}{2} inc(A, \mathrm{TF}(B,t)) \tag{6}$$

and search now for exactly one t-vector.

Given such an optimal spatial superposition, it makes sense to define $m_{i,j}$ by the distance between point $p_i \in P_1$ and $p_j \in P_2$ in the superimposed point clouds. To account for point-to-dummy mappings, the distance between a point and a dummy is specified by a parameter k. Finally, dummy-dummy assignments are scored by zero, so that these mappings will not influence the construction of the alignment. As an illustration, Table 1 shows a matrix M for two point clouds $P_1 = \{a, b, c, d\}$ and $P_2 = \{a', b', c'\}$.

Table 1: Matrix representation of the optimal assignment problem.

	a'	b'	c'	$\perp$	$\perp$	$\perp$	$\perp$
a	$d(a,a')$	$d(a,b')$	$d(a,c')$	k	k	k	k
b	$d(b,a')$	$d(b,b')$	$d(b,c')$	k	k	k	k
c	$d(c,a')$	$d(c,b')$	$d(c,c')$	k	k	k	k
d	$d(d,a')$	$d(d,b')$	$d(d,c')$	k	k	k	k
$\perp$	k	k	k	0	0	0	0
$\perp$	k	k	k	0	0	0	0
$\perp$	k	k	k	0	0	0	0

Formally, an assignment (weighted bipartite matching) problem is specified by a graph $G = (V, E)$ with $V = V_1 \cup V_2$ ($V_1 \cap V_2 = \emptyset$) and $E = \{\{u, v\} \,|\, u \in V_1,\, v \in V_2\}$. The problem is to find a subset of edges $M \subseteq E$ such that $e \cap e' = \emptyset$ for all $e, e' \in M$ (i.e., one point has exactly one mapping partner),

$$\bigcup_{(v_1,v_2)\in M} \{v_1\} = V_1, \qquad \bigcup_{(v_1,v_2)\in M} \{v_2\} = V_2,$$

and

$$\sum_{e \in M} c(e) \quad \rightarrow \quad \min,$$

where $c(e)$ is the cost associated with edge e. In our case, the sets V_1 and V_2 represent, respectively, the points in point cloud P_1 with additional $|P_2|$ dummy points and the points in cloud P_2 with additional $|P_1|$ dummy points. Moreover, the costs $c(e)$ are given by the corresponding matrix entries $m_{i,j}$. See Figure 4 for an illustration.

To solve the weighted bipartite matching problem, we use the Hungarian algorithm [13] that needs time $\mathcal{O}(n^3)$. Once a cost-minimal assignment has been found, the geometric alignment is defined by the corresponding node-to-node and node-to-dummy assignments, while dummy-to-dummy assignments are ignored.

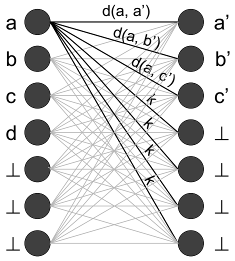


Figure 4: Illustration of the weighted bipartite graph matching problem.

5.2 Construction of Multiple Alignments

Pairwise alignments can be used, for example, to derive a measure of similarity between two objects. From a biological point of view, however, it is even more interesting to look for a *multiple* alignment, that is, the simultaneous alignment of a set of $m > 2$ structures. Alignments of this type are of interest, for example, to discover conserved patterns in a family of evolutionary related proteins.

To derive a multiple geometrical alignment (3DA) of m point clouds, we resort to the star alignment approach [17]: One of the point clouds, say, P_1, is selected and aligned in a pairwise way with all other clouds P_i, $i = 2, \ldots, m$. The pairwise alignments are then "merged" by using P_1 as a pivot structure. Thus, if $p_{ij} \in P_i$ denotes the point (possibly a dummy) aligned with $p_j \in P_1$ in the alignment of P_1 and P_i, then a single assignment in the multiple alignment is of the form

$$(p_j, p_{2j}, p_{3j}, \ldots, p_{mj}).$$

Since the quality of a multiple alignment is strongly influenced by the choice of the pivot structure, we try each point cloud as a pivot and adopt the best result. Thus, $m(m-1)/2$ pairwise alignments have to be computed in total.

5.3 Conserved Patterns

As already mentioned each $a \in \mathcal{A}$ corresponds to a vector of mutually assigned points from the point clouds $P_1, \ldots, P_m$. Note that, by matching points, a mutual assignment of distances is determined in an implicit way. Once a 3DA has been established, it can be used to derive approximately conserved patterns. This can be done in different ways, we propose to use fuzzy consensus graphs [17] originally introduced for *Multiple Graph Alignments (MGA)*. In a first step we generate a *fuzzy consensus graph* $\tilde{G} = (V, E)$. V contains a node for each tupel $a \in \mathcal{A}$ and all pairs of nodes are connected by an edge $e \in E = V \times V$. Each node is labeled with the distribution of the mutually assigned points. Additionally, a degree of conservation $cons(v)$ is calculated, which is defined by the relative number of point clouds in which this point is present. The edges of the consensus graph are defined accordingly; see [17] for details. For given thresholds $\omega, \xi \in (0, 1]$, a conserved pattern can then be defined in terms of the subgraph of G consisting of all nodes v with $cons(v) \geq \omega$ and $maj(v) \geq \xi$, where $maj(v)$ is the relative frequency of the most frequent label in a.

6 Experimental Results

In our experimental study, we perform two types of experiments. Both have in common that we compare the introduced geometrical approach with graph-based approaches. In the first study we will only consider the similarity scores and use them for classification. In the second study we will consider alignments and again compare the geometrical and a graph-based approach [17].

6.1 Data

For the experimental study different data sets are needed. The first type of experiment require a data set consisting of at least two classes so that a classification can be performed. For the second type of experiment we need a data set that consists of many structures that share a common fragment for that we can search using the multiple alignment approaches.

6.1.1 NADH/ATP

One important problem in pharmaceutical chemistry is the identification of protein binding sites that bind a certain ligand. We selected two classes of binding sites that bind, respectively, to NADH or ATP. This gives rise to a binary classification problem: Given a protein binding site, predict whether it binds NADH or ATP.

More concretely, we compiled a set of 355 protein binding pockets representing two classes of proteins that share, respectively, ATP and NADH as a cofactor. To this end, 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. We thus had to ensure the selection of binding pockets with ligands bound in similar conformation. To achieve this, we used the Kabsch algorithm [12] to calculate the root mean square deviation (RMSD) between pairs of ligand structures. Subsequently, we combined all proteins whose ligands yielded a RMSD value below a threshold of 0.2, thereby ensuring a certain degree of similarity. This value was chosen as a trade-off between data set size and similarity. Eventually, we thus obtained a two-class data set comprising 214 NADH-binding proteins and 141 ATP-binding proteins.

6.1.2 Benzamidine

For a first proof-of-concept of the 3DA approach, 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 [4]. The data set is suitable for conducting experiments in a systematic way, as it is quite homogeneous and relatively small (the descriptors contain 47 – 100 points, where each point 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.

6.1.3 Thermolysin

Additionally, we used 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 well-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 pseudo-centers and not all of them being real binding pockets, the data set is also diverse enough to present a real challenge for matching techniques.

6.2 Classification

In our experiments, first we compared our novel method (LPCS) with existing graph-based approaches, namely the random walk (RW) kernel [9], the shortest path (SP) kernel [6], and the method of multiple graph alignment (MGA) recently introduced in [17]. Given two labeled points clouds as input, all these methods produce a degree of similarity as an output. Yet, for the graph-based approaches, it is of course necessary to transform a point cloud into a graph representation in a preprocessing step. This was done as as proposed in [17]:

1. each point is transformed into a node with corresponding node label
2. for each pair of nodes:
   (a) the Euclidean distance between both nodes is calculated
   (b) if the distance is below a certain threshold (here 11 Å to ensure connected graphs), an edge with weight equal to this distance is added

Our ES was restarted $n = 5$ times. The parameterization was optimized with the *sequential parameter optimization toolbox* [2] and was chosen as follows:

- inexact ES: $\mu = 30, \nu = 4, \rho = 2$, comma-selection, termination criteria: largest step size < 0.05, discrete recombination for strategy- and object-component.
- exact ES: $\mu = 30, \nu = 4, \rho = 6$, plus-selection, termination criteria: largest step size < 0.00001, intermediate recombination for object and discrete recombination for strategy-component.

A comprehensive explanation of the different ES parameters and operators can be found in [3].

For both variants we initialized the object-component in $[-150, 150]^3$ for translation and $[0, 2\pi]^3$ for rotation: The step sizes were initialized in $[5, 15]^3$ and $[1, \pi]^3$, respectively. The SP-kernel is parameter-free, the RW-kernel expects a parameter λ that is set to the largest degree of a node in the data set to ensure a geometric series during calculation, which results in a simpler evaluation [5]. Since the geometric information of real-world data is noisy, we also need a tolerance parameter ϵ to decide whether two edges have equal length (difference $\leq \epsilon$) or not; in our experiments, we used $\epsilon = 0.2$. For MGA, we chose the parameterization proposed in [17].

The assessment of a similarity measure for biomolecular structures, such as protein binding sites, is clearly a non-trivial problem. In particular, since the concept of similarity by itself is rather vague and subjective, it is difficult to evaluate corresponding measures in an objective way. To circumvent this problem, we propose to evaluate similarity measures in an indirect way, namely by means of their performance in the context of nearest neighbor (NN) classification. The underlying idea is that, the better a similarity measure is, the better should be the predictive performance of an NN classifier using this measure for determining similar cases.

6.2.1 Results

The results of a leave-one-out cross validation, using the simple 1-NN classifier for prediction, are summarized in Table 2. As can be seen, the kernel-based methods (SP and RW) perform very poorly and are hardly better than random guessing. In terms of accuracy, MGA is much better, though still significantly worse than LPCS. In fact, LPCS performs clearly best on this problem.

Table 2: Accuracy and runtimes (in seconds with standard deviation, referring to a single comparison) of LPCS ($\alpha = 0.5$, with restarts like described above), MGA, RW, and SP on the NADH/APT data set.

Method	Accuracy	Runtime
MGA	0.7662	121.74 ± 418.02
SP	0.6056	$\mathbf{9.75 \pm 97.77}$
RW	0.5972	65.51 ± 89.07
LPCS	$\mathbf{0.9352}$	20.04 ± 24.65

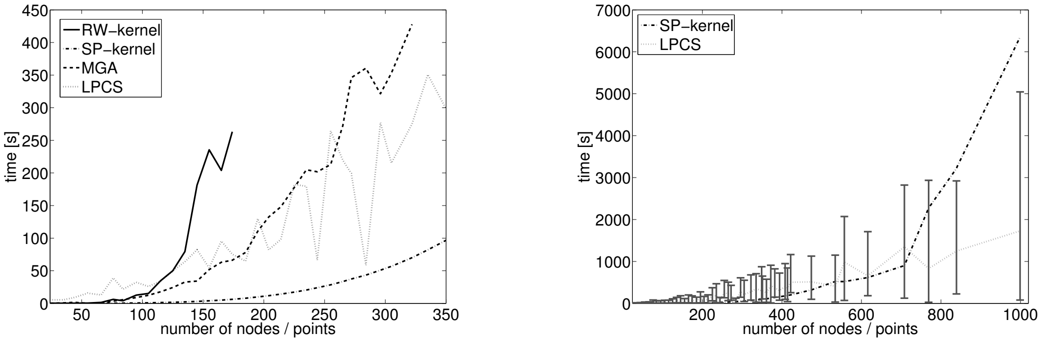
Table 3 furthermore shows how the performance of LPCS depends on the choice of the trade-off parameter α in (5). As can be seen, this parameter does indeed have an influence, even though the differences are not extreme. For this data set, α-values around 0.5 yield better results than extreme values close to 0 or 1; the optimal choice would be $\alpha = 0.7$. In practice, α can be considered as a tuning parameter to be adapted to the problem at hand (e.g., by means of a cross-validation on the training data).

Table 3: Accuracy of LPCS for different values of α in (5).

α	accuracy	α	accuracy
0	0.9042	0.6	0.9352
0.1	0.9183	0.7	0.9380
0.2	0.9126	0.8	0.9239
0.3	0.9154	0.9	0.9267
0.4	0.9267	1	0.9183
0.5	0.9352		

6.2.2 Runtime

To investigate the behavior regarding runtime of the approaches applied in this paper we used again the NADH/ATP data set and chose protein binding sites of size approximately $25, 35, \ldots, 985, 995$. For a size s this was done by selecting the largest binding site that is smaller than s and a smallest binding site that is larger s. Doing this has the advantages that first the size of the problem to solve is in mean s, and second that both selected protein binding sites are different, so that side effects due to equivalence of both binding sites can be avoid. For MGA, SP- and RW-kernel the runtime for each size s was evaluated once since these methods are deterministic and have always same runtime. Since LPCS is based on a stochastic optimizer we repeated this experiment for each s 10 times using the same point clouds. The results are summarized in figure 5. As can be seen, from a certain problem size for MGA and RW-kernel a calculation is not possible since these methods works with the product graph that is growing quadratically with the size of the



(a) runtimes of all methods in the range $[25, 350]$, for LPCS only median was ploted

(b) runtimes of SP-kernel and LPCS (min, median, max) in the range $[25; 1000]$

Figure 5: Runtimes of LPCS, MGA, SP-, and RW-kernel w.r.t. problem size; for RW-kernel and MGA a calculation was posible to a certain size of the problem since the memory requirement was becoming too high

input graphs so that even modern computers (2 GB RAM) cannot offer sufficient memory. The LPCS approach has for very small problems the highest runtime though the runtime is growing very slow w.r.t. problem size, so that LPCS is already for point clouds of size 200 faster than MGA and RW-kernel, approached that are appropriate only for small structures. It sticks out that the LPCS runtime fluctuate strongly. The reason is quite simple. Since we use a real world data set the distribution of point labels vary. As already mentioned for the nearest neighbor search we hash points with equal label. So, if the labels are distributed uniformly the search is more efficient than if there exists a label that dominates the point cloud. The SP-kernel has cubic runtime, so that this method is for $s < 600$ the most efficient of all alternatives. However, it completely fails in terms of predictive accuracy. That LPCS is becoming the most efficient approach for $s > 600$ is hardly surprising, since the dimensionality of the LPCS optimization problem is constant (six parameters have to be optimized) and does not depend on the number of data points. It is true that the size of the point clouds does have an influence on the evaluation of the objective function, which involves a nearest neighbor search for each point. The increase in runtime is at most quadratic, however.

6.3 Alignment Quality

In the second study, we compared the quality of the alignments calculated, respectively, by 3DA and MGA. To this end, 100 alignments of size 2 were calculated for randomly chosen structures. Restricting to pairwise alignments is justified since both 3DA and MGA use the star alignment procedure to derive multiple alignments. The quality of a pairwise alignment $\mathcal{A}$ is evaluated in terms of two criteria. The first criterion is the fraction of assignments of pseudocenters preserving the label information:

$$s_1 = \frac{1}{|\mathcal{A}|} \sum_{(a_1,a_2)\in\mathcal{A}} \left\{ \begin{array}{ll} 1, & \ell(a_1) = \ell(a_2)) \\ 0, & \ell(a_1) \neq \ell(a_2)) \end{array} \right. ,$$

where $\ell(a_1)$ is the label of the pseudocenter a_1. Similarly, the second criterion evaluates to what extent the geometry of the structures is preserved. Since an MGA does not include information about the position of single psedocenters, this has to be done by looking at distances between pairs of pseudocenters in each structure:

$$s_2 = \frac{1}{N} \sum_{(a_1,a_2),(b_1,b_2)\in\mathcal{A}} \left\{ \begin{array}{ll} 1, & |d(a_1,b_1) - d(a_2,b_2)| \leq \epsilon \\ 0, & |d(a_1,b_1) - d(a_2,b_2)| > \epsilon \end{array} \right. ,$$

where $d(a_1, b_1) = |c(a_1) - c(b_1)|$ and $N = |\mathcal{A}|(|\mathcal{A}| - 1)/2$. We summarize the evaluation by the vector

$$\boldsymbol{s} = (s_1, s_2) \in [0,1] \times [0,1] \, .$$

To measure the improvement of our method, we calculate the relative improvement

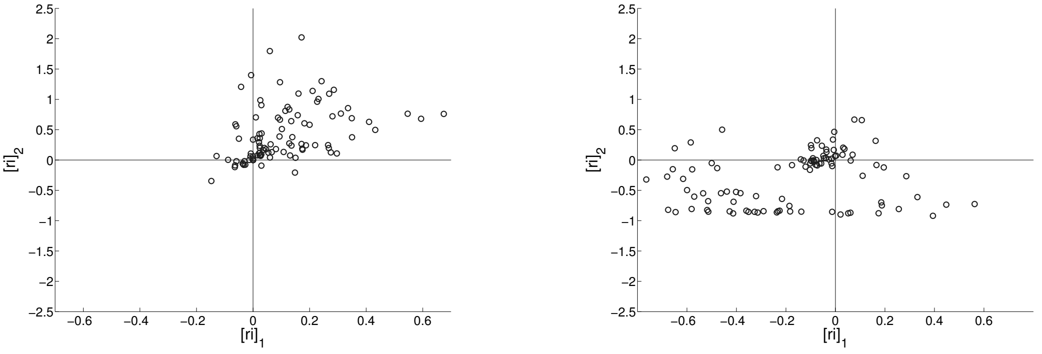
$$\boldsymbol{ri} = \left(\begin{array}{c} \dfrac{[\boldsymbol{s}_{3DA}]_1 - [\boldsymbol{s}_{MGA}]_1}{[\boldsymbol{s}_{MGA}]_1} \\ \dfrac{[\boldsymbol{s}_{3DA}]_2 - [\boldsymbol{s}_{MGA}]_2}{[\boldsymbol{s}_{MGA}]_2} \end{array} \right) \tag{7}$$

where $\boldsymbol{s}_{3DA}$ and $\boldsymbol{s}_{MGA}$ denote, respectively, the evaluations of 3DA and MGA and where $[s]_i$ gives the i-th element of a vector $\boldsymbol{s}$.

6.3.1 Results

For our calculations we parameterized MGA as proposed in [17], for 3DA we set $k = 6$ and performed experiments like described above. The results for the benzamidine data set are shown in Figure 6, where the relative improvement vectors are plotted. As one can see, most of the ri vectors are lying in the first quadrant, indicating a positive improvement for both criteria.

The corresponding results for the thermolysin data set are depicted in Figure 6. Here, the picture is not as clear, and the number of negative improvements is even slightly higher than the number of positive ones. Apparently, 3DA performs especially good on highly similar structures while not improving on structures that are more diverse. This is hardly surprising, since 3DA strongly exploits information about the geometry of the structures.



(a) ri on the benzamidine dataset

(b) ri on the thermolysin dataset

Figure 6: Relative improvements (ri) obtained by substituting the MGA approach in 3DA

6.3.2 Parametrization

As an important advantage of 3DA, it deserves mentioning that it only has a single parameter, while MGA has six parameters. In spite of this, we found that if often produces better results, even when trying to parameterize MGA in an optimal way. For example, Figure 7 shows a set of solutions for the benzamidine data that we found by varying the parameters in 3DA and MGA. For ease of exposition, we only plotted the solutions that are Pareto optimal in the two respective sets of solutions; in total, 7776 result vectors $\boldsymbol{s}$ were computed for MGA by variation its 5 parameters in a systematic way. This was done by varying penalties from -5 to 0 and awards form 0 to 5 and considering all possible combinations (see [17] for an explanation of these parameters). For 3DA there was only one parameter (threshold k) to vary, so that here only 12 results were calculated by considering $k = 0, \ldots, 11$. To have a readable plot we removed results that are not Pareto optimal[1] and plot only the remaining Pareto optimal points. The resulting plot is illustrated in figure 7. As one can see the 3DA solutions were independent of parameterization always better than the MGA results, so that we can claim that our novel method is easy to adjust and will lead to results that are better, even for an optimal adjusted MGA approach.

[1] Given a set of results S only such results $\boldsymbol{s} \in S$ are called Pareto optimal that are not dominated by other solutions. A vector $\boldsymbol{x}$ dominates another vector $\boldsymbol{y}$ if $\boldsymbol{x}[i] \geq \boldsymbol{y}[i]$ for all i and $\boldsymbol{x}[i] > \boldsymbol{y}[i]$ for some i.

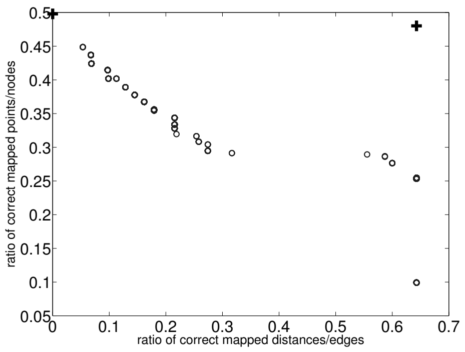


Figure 7: Pareto optimal solutions found by MGA (circles) and 3DA (crosses)

6.4 Structure Retrieval

The focus of the third study is on the ability to detect common substructures in a set of biochemical structures. We randomly selected 100 subsets of c compounds from the benzamidine data set and used 3DA and MGA to calculate an alignment. Then, we checked whether the aforementioned benzamidine core fragment, an amide derivative of benzol which consists of 25 atoms (11 hydrogens), was fully conserved in the alignment, which means that all pseudocenters belonging to the core were mutually assigned in a correct way. The results, shown in Table 4 for different numbers c, clearly show that 3DA is able to retrieve the core fragment much more reliably than MGA.

Table 4: Percent of alignments in which the benzamidine core fragment was fully conserved in the alignment of $c = \{2, 4, 8, 16\}$ structures.

c	2	4	8	16
MGA	0.85	0.38	0.14	0.04
3DA	0.96	0.92	0.80	0.76

For detecting the core fragment we searched for conserved patterns in the alignment and used the parameter $\omega = 1$ and $\xi = 0.9$.

7 Conclusions

In this paper, we have introduced labeled point cloud superposition (LPCS) as a novel tool for structural bioinformatics, namely as a method for comparing biomolecules on a structural level. Besides, using fuzzy modeling techniques, we have defined a related similarity measure. The concept of a labeled point cloud appears to be a quite natural representation for biological structures, especially since it is closely leaned on existing database formats. In comparison to other approaches, such as the prevalent graph-based methods, the modeling is hence simplified and does not involve any complex transformations. More importantly, a labeled point cloud preserves the full geometric information and makes it easily accessible to computational procedures.

A labeled point cloud superposition is a spatial "alignment" of two point clouds which is optimal in the sense of a given scoring (similarity) function. As for related problems in bioinformatics, such as sequence alignment, the computation of the similarity between two objects hence involves the solution of an optimization problem. To this end, we have proposed the use of an evolution strategy, an approach from the family of evolutionary algorithms, which appears to be especially suitable for this problem.

First experimental results with classification data are quite promising and suggest that our approach is able to compare protein binding sites in a reasonable way. In terms of classification accuracy, LPCS turned out to be significantly better than existing (graph-based) methods used for comparison. Moreover, even though it is computationally more complex than these methods for small data sets, it scales much better and becomes more efficient for larger data sets. This is due to the fact that, in contrast to graph-based methods, the search space does not depend on the size of the point clouds and remains low-dimensional.

In this paper, we proposed an extension of the method of labeled point cloud superposition (LPCS), too. Motivated by applications in structural bioinformatics, we extended LPCS for the calculation of multiple geometric alignment which, based on a given superposition, computes an one-to-one correspondence between the points. First experiments carried out in the context of protein structure comparison are quite promising and show that our method is competitive, if not even superior, to state-of-the-art graph-based methods for multiple structure alignment. All things considered, multiple geometric alignment is therefore a viable option for protein structure comparison and might even be of interest beyond the field of structural bioinformatics.

References

[1] Francis R. Bach. Graph kernels between point clouds. In *International Conference on Machine Learning*, pages 25–32, Helsinki, Finland, 2008.

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

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

[4] 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.

[5] K. M. Borgwardt. *Graph Kernels*. PhD thesis, Ludwig-Maximilians-Universität München, Germany, 2007.

[6] K. M. Borgwardt and H. P. Kriegel. Shortest-path kernels on graphs. In *International Conference on Data Mining*, pages 74–81, Houston, Texas, 2005.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer, New York, 2000.

[8] J. Fodor and R.R. Yager. Fuzzy set-theoretic operators and quantifiers. In D. Dubois and H. Prade, editors, *Fundamentals of Fuzzy Sets*, pages 125–194. Kluwer Academic Publishers, Boston/London/Dordrecht, 2002.

[9] Thomas Gärtner. A survey of kernels for structured data. *SIGKKD Explorations*, 5(1):49 – 58, 2003.

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

[11] 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.

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

[13] H.W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics*, 52(1):7–21, 2005.

[14] F. Mémoli and G. Sapiro. Comparing point clouds. In *Eurographics / ACM SIGGRAPH symposium on Geometry processing*, pages 32–40, Nice, France, 2004.

[15] 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.

[16] D. Sinha and E.R. Dougherty. Fuzzification of set inclusion: theory and applications. *Fuzzy Sets and Systems*, 55(1):15–42, 1993.

[17] 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.

[18] R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. 18(1):183–190, 1988.

[19] L.A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Comput. Math. Appl.*, 9:149–184, 1983.