Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

DANIEL BAUM

Multiple Semi-flexible 3D Superposition of Drug-sized Molecules

ZIB-Report 04-52 (December 2004)

Multiple Semi-flexible 3D Superposition of Drug-sized Molecules

Daniel Baum

Abstract

In this paper we describe a new algorithm for multiple semi-flexible superpositioning of drug-sized molecules. The algorithm identifies structural similarities of two or more molecules. When comparing a set of molecules on the basis of their three-dimensional structures, one is faced with two main problems. (1) Molecular structures are not fixed but flexible, i.e., a molecule adopts different forms. To address this problem, we consider a set of conformers per molecule. As conformers we use representatives of conformational ensembles, generated by the program ZIBMol. (2) The degree of similarity may vary considerably among the molecules. This problem is addressed by searching for similar substructures present in arbitrary subsets of the given set of molecules.

The algorithm requires to preselect a reference molecule. All molecules are compared to this reference molecule. For this pairwise comparison we use a two-step approach. Clique detection on the correspondence graph of the molecular structures is used to generate start transformations, which are then iteratively improved to compute large common substructures. The results of the pairwise comparisons are efficiently merged using binary matching trees.

All common substructures that were found, whether they are common to all or only a few molecules, are ranked according to different criteria, such as number of molecules containing the substructure, size of substructure, and geometric fit. For evaluating the geometric fit, we extend a known scoring function by introducing weights which allow to favor potential pharmacophore points.

Despite considering the full atomic information for identifying multiple structural similarities, our algorithm is quite fast. Thus it is well suited as an interactive tool for the exploration of structural similarities of drug-sized molecules.

Keywords: pharmaceutical drug design, multiple superposition, semi-flexible alignment, clique detection, iterative closest point, matching tree.

Mathematics subject classification: 68W25, 92-08.

1 Introduction

In pharmaceutical drug design, one is often faced with the question, which properties a drug (ligand) must have to bind to a specific receptor. In absence of the receptor structure, the only given information is a set of ligands for which we know or assume, that they all bind to the same receptor using similar binding modes. In this case, one can only guess which properties a ligand needs to have to bind to the receptor. To put this guesswork on solid ground, the ligands need to be compared with each other, and similarities as well as differences need to be revealed. Beside the physico-chemical properties of the ligand, its form plays a major role in the binding process. Hence, it is not enough to consider the two-dimensional structure of the ligands, but one also needs to look at their three-dimensional forms. In general, the active forms of the ligands are not known. Therefore, the ligand's flexibility needs to be taken into account.

There exist two classes of algorithms that account for the flexibility of the ligands. The first class of algorithms keeps the ligands flexible during the comparison stage. The advantage of this approach is that the search space is not limited to a precomputed number of (generally) low-energy conformers. And indeed, active conformers often have a slightly higher energy. The disadvantages of this approach are two-fold. On the one hand, the search space needs to be sampled for each comparison. On the other hand, one might end up with statistically very unlikely conformers. The second class of algorithms uses precomputed conformers to consider the flexibility of the ligands, hence they are called semi-flexible algorithms. Their advantage is that the conformers of each molecule need to be computed only once, independent of the comparison to be accomplished, and thus a more exhaustive conformational analysis can be carried out. Also, the development of comparison algorithms using multiple conformers is uncoupled from sampling the conformational space. The disadvantage of this approach is that the conformational space of a molecule might be very large and one can easily generate thousands of different conformers, posing the question, which conformers should be used for the comparison. This problem arises, since the conformers are generated before and not during the comparison.

In the remainder of the introduction, we briefly want to describe related work and compare our approach to previously published ones. Since the number of publications in this field is very large, we will not cover all developments in molecular superposition. Good coverage of the work predating 1990 can be found in [3, 24]. Publications predating 2000 are excellently reviewed in [22].

We will first review publications belonging to the class of algorithms considering true flexibility. Sheridan et al. [32] use distance geometry to search for conformations containing a pharmacophore present in all molecules. Dammköhler et al. [6] use different systematic search methods to sample the conformational space. For comparing patterns in the conformers of different molecules they use distance maps. Both approaches require a predefined pharmacophore to be present in each molecule. McMartin et al. [27] and Lemmen et al. [21] propose methods for pairwise superposition, keeping one of the molecules rigid and the other one flexible. However, while McMartin et al. use a combination of a Monte Carlo perturbation and an energy minimization, Lemmen et al. decompose the molecule in relatively rigid parts and incrementally fit these to the rigid molecule. Finally, Jones et al. [16] and Handschuh et al. [14] apply genetic algorithms to flexibly superimpose sets of molecules.

Since our algorithm makes use of rigid-body alignment, in the following we will mention publications dealing with both rigid-body and semi-flexible superposition. Many rigid-body alignment methods try to maximize some kind of volume overlap. The volume is generally given through Gaussian functions, approximating different properties, such as van der Waals overlap [11], electron density overlap [29], or electrostatic potential overlap [10, 26]. The overlap optimization methods range from simplex optimization [10], gradient optimization [26], Fourier space methods [29, 20] to Monte Carlo optimization [30]. The methods of Cosgrove et al. [5] and Hofbauer [15] use molecular surface similarity to superimpose molecules. Both methods apply clique detection to find matchings for a reduced set of surface points. Feuilleaubois et al. [7] use neural networks to search for optimal sets of inter-atomic matches. Other approaches include, e.g., the work of Masek et al. [25], Klebe et al. [19], Finn et al. [8], and Miller et al. [28]. The work most closely related to our approach is the multiple semi-flexible superposition algorithm by Martin et al. [23]. They specify a reference molecule to which all other molecules are pairwise aligned. Multiple conformers are considered separately. For the pairwise alignment they require pharmacophore points to be specified which are matched using a clique detection method. The results of the pairwise comparisons are finally merged to get multiple matchings. Since the number of pharmacophore points in each molecule is considerably smaller than the overall number of atoms, clique detection can easily be applied and merging the results is relatively easy. In parts, our work is based on the work of Kirchner [18], who uses a greedy matching strategy for finding the optimal matching of any two conformations of two molecules. Since he is only interested in the optimal matching between two molecules, he can apply a branch-and-bound method to prune large parts of the search tree.

Our work is motivated by the need for a multiple semi-flexible superposition algorithm which quickly generates a number of different good matchings, yet uses the full atomic information. To our knowledge, no such algorithm was as yet available. Thus, we try to bridge the gap between the work of Martin et al. [23], which is fast, but uses only parts of the full structural information, and very detailed, expensive algorithms which explore the full structural information, but either are not able to handle flexibility or cannot do multiple superpositioning.

2 Materials and Methods

In this chapter, we describe all methods used in our algorithm. A graphical overview of the algorithm is shown in Fig. 1. Since the term *matching* will be used extensively in this chapter, we want to give its definition already at this early position within the paper. Informally, a matching is a one-to-one mapping of atoms of two molecules. Formally, it can be defined as follows.

Definition 2.1 *Let* $m, n \in \mathbb{N} \setminus \{0\}$*. A function* M *is called* pairwise matching*, or simply* matching*, if it meets two properties:*

1. $M : \{1, \ldots, m\} \longrightarrow \{0, \ldots, n\}$
2. $\forall i, j \in \{1, \ldots, m\} : M(i) = M(j) \Rightarrow i = j \vee M(i) = 0$

The set $M^* := \{i | M(i) \neq 0\}$ *contains all elements that are not mapped to 0.*

In the core of our algorithm we compute pairwise matchings (Sec. 2.1) between one conformation of the reference molecule, also referred to as reference conformation, and one conformation of a query molecule, also referred to as query conformation. In order not to overlook interesting matchings, we need to

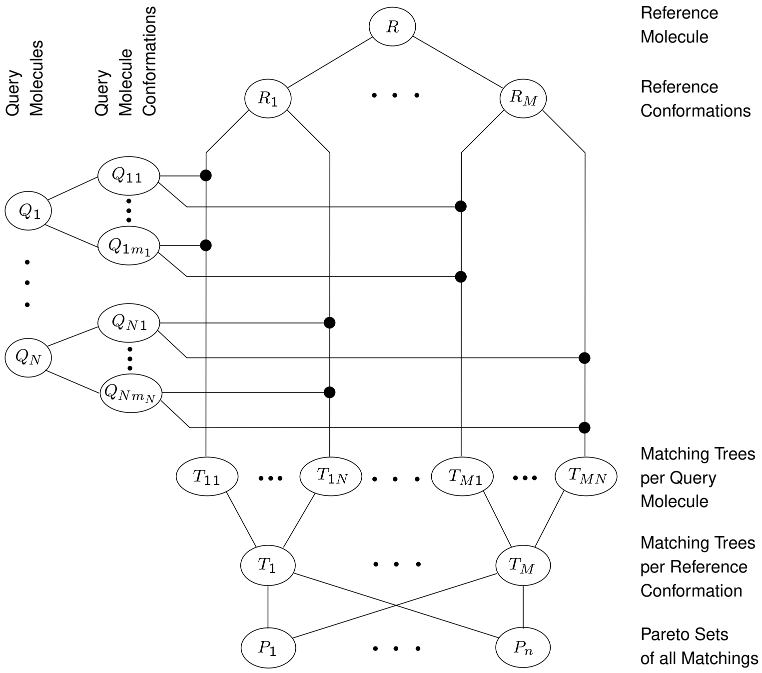

Figure 1: Overview of the algorithm. The reference molecule is denoted by R, its conformations by R_j. The query molecules are denoted by Q_i, their conformations by Q_{ij}. The $\bullet$ symbol denotes the computation of pairwise matchings. T_{ij} denotes the matching tree containing all pairwise matchings between molecule Q_i and reference conformation R_j. T_i is a final matching tree comprising all multiple matchings (matching clusters) with respect to R_i. Finally, P_i is a Pareto set containing multiple matchings of different reference conformations.

compute a considerable number of matchings. For each matching we compute a score. If the score is small, we reject the matching, otherwise it is accepted. Note, that for our purpose it is not sufficient to compute only the best matching, since we are mainly interested in substructures present in more than one query molecule and the substructure corresponding to the best matching might not be present in other query molecules. All matchings are stored in so-called *matching trees* (Sect. 2.2). They allow to quickly identify matchings with equal target substructure. Matching trees are well suited for storing a large number of matchings but also for computing intersections of matchings, i.e., substructures present in more than one query molecule. All matching trees corresponding to the same reference conformation are merged into a single *final matching tree* containing all *matching clusters* for this conformation. Finally, all matching clusters contained in all final matching trees are scored using competing score values (Sect. 2.3). To account for all competing score values we use *Pareto sets*. The complete algorithm is described in Sect. 2.4 and the algorithm's parameters are commented on in Sect. 2.5.

2.1 Pairwise Matchings

Since the computation of pairwise matchings lies at the core of the algorithm, it needs to be efficient. Except for very small point sets it is not possible to compute all potential matchings. Hence, we need to preselect a feasible subset of matchings (Sect. 2.1.2). These starting matchings are iteratively improved, i.e., we search for the local maximum near a given start transformation (Sect. 2.1.3). Instead of using an exact algorithm, which is computationally very expensive, we use a greedy method, which in most cases finds the optimum or gets very close to it [18]. We favor large matchings with a small distance between the matched points. Since those two qualities compete with each other, we need a scoring function that balances them (Sect. 2.1.1). This scoring function can also be used to bound the iterative search (Sect. 2.1.3).

2.1.1 Scoring Function

Let $A_1, \ldots, A_m$, and $B_1, \ldots, B_n$ be the coordinates of all non-hydrogen atoms of molecules A and B, let M be a matching and T a transformation. Then we can define the *root mean square (rms)* distance and the *score* [33] of two molecules A and B under matching M and transformation T as

$$rms(A,B;M;T) := \sqrt{\frac{\sum_{i\in M^*} \|A_i - TB_{M(i)}\|^2}{|M^*|}} \quad \text{, and} \tag{1}$$

$$score(A,B;M;T) := \frac{|M^*|}{\min(m,n)} \cdot e^{-rms(A,B;M;T)} \quad . \tag{2}$$

If we define w as $w : \{1, \ldots, m\} \times \{1, \ldots, n\} \longrightarrow \mathbb{R}^+$, we can further define the *weighted rms* distance and the *weighted score* as

$$rms^*(A,B;M;T) := \sqrt{\frac{\sum_{i\in M^*} w(i, M(i)) \cdot \|A_i - TB_{M(i)}\|^2}{|M^*|}} \quad \text{, and} \tag{3}$$

$$score^*(A,B;M;T) := \frac{|M^*|}{\min(m,n)} \cdot e^{-rms^*(A,B;M;T)} \quad . \tag{4}$$

The functions *score*, which is a pure geometric score, and *score**, which allows to take atom types into account, will be used to evaluate pairwise matchings.

2.1.2 Start Transformation

There exist several ways to generate start transformations for the matching search. One strategy is to select triples of atoms in both molecules [18]. If the triples are within some distance bounds to each other in terms of interatomic distances, i.e., if they span similar triangles, they will be used to compute a least squares fit [17, 34] between the three matching pairs. The resulting transformation can be used as a start transformation.

We use clique detection [4] to identify small subsets of atoms in both molecules with similar structure [3]. These subsets are then used in a similar fashion as the triples above to compute start transformations by computing the least squares fit. The clique detection method has several advantages over the triples-of-atoms method. First, apart from reducing the number of start transformations by decreasing the distance threshold, we can also reduce the number of start transformations by increasing the minimum clique size, which is often more suitable for larger molecules. Second, the number of start transformations will in general be much less using the same distance constraints, since a clique with more than three nodes contains several triples which would lead to very similar if not equal start transformations. A third advantage is, that we can also integrate further constraints (see below) into the generation of the correspondence graph and thus, we do not have to handle all constraints explicitly.

Correspondence Graph. The graph in which we search for cliques is called *correspondence graph* (or *product graph*). It consists of nodes and undirected edges. Each node represents a pair of atoms - one atom from each molecule - that are allowed to be matched. E.g., we might only want atoms to be matched if they are of similar type, thus, only atom pairs of similar type will be represented as nodes in the correspondence graph. An edge connecting two nodes is found in the correspondence graph, if the two pairs of atoms fulfill all desired constraints.

Constraints. The main constraint is usually concerned with *distance*. If node n_i represents atom pair (u, x) and node n_j represents atom pair (v, y), then n_i and n_j will be connected by an edge of the correspondence graph, if $|d(A_u, A_v) - d(B_x, B_y)| < \delta_{graph}$, where $d(\cdot, \cdot)$ is the Euclidean distance. A second constraint is usually $u \neq v$ and $x \neq y$, i.e., no atom of one molecule should be matched to two or more atoms of the other molecule. To reduce the size and the number of cliques, a third constraint could be used to ensure that atoms u and v, and atoms x and y, respectively, should be a minimum number of bonds, δ_{bond}, apart, i.e., $d_{bond}(u, v) \geq \delta_{bond}$ and $d_{bond}(x, y) \geq \delta_{bond}$, where $d_{bond}(a, b)$ denotes the bond distance between atoms a and b. This constraint is motivated by the fact that the matchings we are looking for should contain many atoms, and hence atoms very close to each other are not that interesting for generating start transformations. Note that if δ_{bond} is 1, the second and third constraints are equal.

Cliques. A clique in a graph is a subgraph in which all nodes are connected to each other, and which is maximal in the sense that no node can be added to the subgraph while preserving the first property. Hence, a clique in the correspondence graph contains a maximal number of nodes, such that all pairs of nodes mutually fulfill all constraints. Since the first constraint ensures similar distances, the distance matrices of the two point sets corresponding to the clique are equal, and hence, except for space reflections, the point sets will have similar structures. Since each node of the clique corresponds to one atom pair, the clique also gives us an explicit atom-to-atom-matching, which allows us to compute a start transformation.

2.1.3 Matching Search

Given two molecules A and B and a transformation T (e.g., a start transformation computed with clique detection), the aim is to compute the matching M maximizing Eq. (2) (or Eq. (4)). There exists an exact graph theoretical algorithm to compute the optimal matching [18]. However, this algorithm has a run time of $\mathcal{O}(n^3)$, where $n = \max\{|A|, |B|\}$.

Greedy matching using *score*. It was shown in [18], that there exists a greedy algorithm which in most cases computes the optimal matching or gets very close to it. Let $E_{\leq\delta} := \{(i, j) | i \in \{1, \ldots, m\} \wedge j \in \{1, \ldots, n\} \wedge \|A_i - TB_j\|^2 \leq \delta\}$ be the set of edges between atoms of molecule A and atoms of

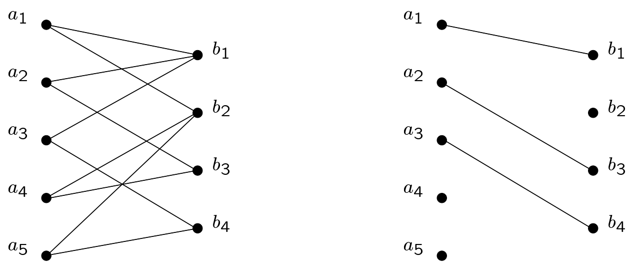

Figure 2: Two bipartite graphs depicting the set $E_{\leq\delta}$ *(left)* and the best matching *(right)*. The nodes a_i and b_j represent atoms of molecules A and B, respectively. The best matching consists of only three matching pairs, since adding one out of two possible further matching pairs reduces the *score* of the matching.

molecule B transformed by T whose squared distances are bound by δ. The algorithm now works as follows.

1. Compute the set $E_{\leq\delta}$ and assign a weight to each edge given by the squared distance of the transformed atoms, the edge represents.
2. Sort the edges according to their weights, starting with the edge having the smallest weight.
3. Compute matching M_1 which contains the edge with the smallest weight.
4. Compute matching M_{i+1} from M_i by adding the edge e having the smallest weight with neither end node coinciding with an end node of any edge already in M_i.

The algorithm terminates if no further matching can be computed. For each matching we compute its *score*. The best matching is the one with the highest score. Note that, due to the scoring function, the best matching is not necessarily the one with the largest number of matching pairs. An example is shown in Fig. 2.

Greedy matching using *score.** The only modification we need to accomplish to compute the best matching according to *score** instead of *score*, is to scale the squared lengths of the edges by the weights of their corresponding atoms. I.e., $E_{\leq\delta}$ is now defined as $E_{\leq\delta} := \{(i,j) | i \in \{1,\dots,m\} \wedge j \in \{1,\dots,n\} \wedge w(i,j) \cdot \|A_i - TB_j\|^2 \leq \delta\}$, and the edges are sorted by their weighted squared lengths. This strategy allows to favor matchings with atom pairs that have similar properties, such as, e.g., atom type. In contrast to *score*, *score** favors atom pairs with a small weight, even though the atoms might be further away from each other. As result, matching pairs with slightly larger distance but preferable atomic properties will be added to the matching.

Iterative improvement of matchings. Let M^1 be the best matching found by applying greedy matching to a given start transformation. From matching M^i we can compute matching M^{i+1} by applying the greedy matching method to the new start transformation given by least squares fitting the molecules using matching M^i. We continue in this manner until the score does not increase from M^i to M^{i+1}. In general, this process converges very quickly and we only need about 5 iterations. The last matching is considered the best matching for the original start transformation. This strategy was also described in [18]. We use the iterative algorithm to compute a good matching for each clique in the correspondence graph.

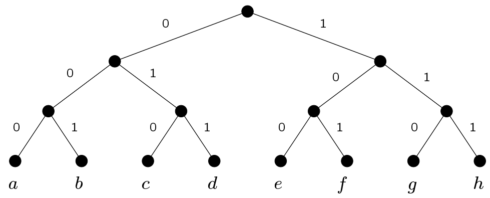

Figure 3: Complete matching tree for a reference molecule with three atoms. For example, leaf a represents the empty substructure, whereas leaf d represents the substructure with atoms 2 and 3.

2.2 Matching Tree

A *matching tree* is a binary tree used for storing matchings corresponding to a single reference conformation. A matching tree allows to find all matchings corresponding to the same substructure in the reference conformation in time $\mathcal{O}(m)$, where m is the number of atoms in the reference molecule. A matching tree has depth m. Level 0 of the matching tree represents the root node, level $i > 0$ represents the i'th atom in the reference molecule. Each leaf corresponds to a unique substructure of the reference molecule (see Fig. 3 for an example). The maximum number of leaves is 2^m. The path $p = [p_0, \dots, p_m]$ from the root to a leaf uniquely defines the substructure associated with the leaf: If atom i of the reference molecule is contained in the leaf's substructure, p_i will be the right child of p_{i-1}, otherwise it will be the left child. Each matching will be inserted into a matching tree according to the matched substructure of the reference conformation. Thus, two matchings M_x and M_y will be stored in the same leaf if $\forall i : M_x(i) \neq 0 \Leftrightarrow M_y(i) \neq 0$. For each leaf we maintain a sorted list of matchings. The substructure corresponding to a leaf can also be thought of as a bit field, where the i'th bit is set to 1 if the i'th atom of the reference molecule is in the substructure.

For each query molecule (and each reference conformation) we maintain a separate matching tree in which all matchings are stored. Once we have computed all matching trees corresponding to the same reference conformation, we need to merge these matching trees to find substructures common in more than two molecules.

Merging of matching trees. Two matching trees are merged in two steps. In the first step, we compute the intersections of all substructures corresponding to the leaves of the first tree and all substructures corresponding to the leaves of the second tree and insert these intersections into the first tree. If we take the bit field representation of substructures, the intersection of two substructures is the result of the bitwise AND operation of the two bit fields. If S is the substructure resulting from the intersection of two substructures, and M is a matching of either matching list of the two leaves associated with the intersected substructures, we define the restricted matching M_S as

$$M_S(i) := \begin{cases} M(i) & \text{if} \quad i \in S, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

All restricted matchings resulting from intersecting two substructures are inserted in the first matching tree. Technically, this is done by just inserting the pointers to the original matchings. The restricted matching can easily be generated from the substructure associated with the leaf and the original matching.

In the second step, we insert all matchings of the second tree into the first tree, if they are not already there.

Merging of the matching trees is done iteratively, i.e., we merge two trees into one of them which is then merged with another one and so forth, until all trees are merged into a single tree, the *final matching tree*.

The merging step drastically reduces the amount of work to be done when bringing together matchings of different query molecules. Since in each merging step the intersected substructures are inserted into the merged tree, substructures contained in multiple query molecules only need to be intersected once during a single merging step.

2.3 Evaluation of Matching Clusters

A final matching tree resulting from iteratively merging N matching trees contains leaves with matchings from 1 up to N query molecules, whereby we might have several matchings for each query molecule. Leaves representing larger substructures will in general contain matchings from fewer query molecules than leaves representing smaller substructures.

We call the matchings gathered at one leaf of the final matching tree together with its reference substructure *matching cluster*. A matching cluster contains possibly more than one matching per query molecule. Note that these matchings will in general belong to different conformations, but some matchings might also belong to the same conformation.

2.3.1 Sorting the Matchings of a Matching Cluster

To evaluate a matching cluster, we need to find the best matching per query molecule, i.e., the matching which best fits the reference substructure. To evaluate a matching of the matching cluster, it needs to be restricted by the reference substructure S, as described in the previous section. This restriction is due to the intersection of leaves in the merging step.

Let T be the transformation minimizing $rms(A, B; M_S; T)$ (see Eq. (1)), given by least-squares-fitting A and B according to M_S. We can then sort all matchings of a matching cluster belonging to the same query molecule according to $rms(A, B; M_S; T)$ or $rms^*(A, B; M_S; T)$ (see Eqs. (1) and (3)), respectively. Note that we do not need to compute $score(A, B; M_S; T)$ or $score^*(A, B; M_S; T)$, since all restricted matchings are of the same size and we only compare matchings of the same query molecule with each other. The $score(^*)$ values only need to be computed for the best matching per query molecule within each cluster. Only the values of the best matchings are used to compute the matching cluster value.

2.3.2 Sorting of Matching Clusters

After sorting all matchings within a matching cluster belonging to one query molecule, we can compute all cluster values and sort the matching clusters according to their values. Since we have competing matching cluster values, such as the number of query molecules in the cluster, the size of the cluster substructure, and the averaged score of all remaining cluster matchings, we consider the competing values of each matching cluster as a vector and define a relation on these vectors. Using this relation we generate so-called *Pareto sets* [35], where each set contains matching clusters that are considered equally good. There exists a total order on these sets, but not on the elements of each set.

Definition 2.2 *A vector* $u = (u_1, \dots, u_k)$ *is said to dominate vector* $v = (v_1, \dots, v_k)$ *(denoted by* $u \succ v$*) if and only if* u *is partially larger than* v*, i.e.,* $\forall i \in \{1, \dots, k\}$, $u_i \geq v_i \wedge \exists i \in \{1, \dots, k\} : u_i > v_i$.

Definition 2.3 *Let* $P \subset \mathbb{R}^k$ *be a set of vectors of dimension* k*. Then* P *is called* Pareto set *if and only if* $\nexists u, v \in P : u \succ v$.

Definition 2.4 *Let* P *and* Q *be two Pareto sets. Then* P *is said to dominate* Q *(similarly denoted by* $P \succ Q$*) if and only if* $\forall q \in Q\ \exists p \in P : p \succ q$.

Definition 2.5 *Let* $V \subset \mathbb{R}^k$ *be a set of vectors of dimension* k*, and let* $\mathcal{P} = \{P_1, \dots, P_n\}$ *be a family of Pareto sets. Then* $\mathcal{P}$ *is called a* decomposition of V*, if* $\bigcup_{i=1}^{n} P_i = V$ *and* $\forall i, j \in \{1, \dots, n\}, i \neq j$*, either* $P_i \succ P_j$ *or* $P_j \succ P_i$*. If* $\forall i \in \{1, \dots, n\}, i \neq k : P_k \succ P_i$*, then* P_k *is said to be the* Pareto optimal set *of* $\mathcal{P}$.

For sorting all matching clusters, we compute the Pareto sets $P_1, \dots, P_n$ representing the decomposition of V, where V is the set of score vectors of all matching clusters. Thus, P_1 is the Pareto optimal set. Furthermore, $\forall i, j, i < j \wedge \forall v \in P_j \Rightarrow \exists u \in P_i$, such that u dominates v.

2.4 Algorithm

After introducing all methods used in our algorithm, we can now describe the complete algorithm. It consists of three steps, whereby the first two steps are executed separately for each reference conformation, and only the last step is done for all reference conformations.

1. For each reference conformation we take all query molecules in turn and compute pairwise matchings (Sect. 2.1) between the query molecule's conformations and the specified reference conformation. All pairwise matchings belonging to the same query molecule and reference conformation are stored in a separate matching tree (Sect. 2.2).
2. All matching trees belonging to the same reference conformation are merged iteratively (Sect. 2.2), thereby producing all matching clusters, which are then evaluated in the last step. At the end of this step we have as many *final matching trees* as we have reference conformations.
3. We consider all matching clusters of all final matching trees. For each matching cluster we compute its score values. Interpreting these score values as vectors, we can compute the Pareto sets (or only the Pareto optimal set) for all matching clusters. Note that a Pareto set will, in general, contain matching clusters corresponding to different reference conformations.

A graphical overview of the algorithm is given in Fig. 1.

2.5 Parameters

For our algorithm a couple of parameters need to be set. These shall be shortly described next.

2.5.1 Atom Types

We allow to use three different kinds of atom types for the clique detection method as well as for the greedy matching-search. These are *none*, *atomic number*, and *pharmacophore*. For clique detection and greedy matching different atom types can be used.

The type *none* does not consider any atom types and hence, any atoms can be matched. Thus, we can use the score of Eq. (2).

The *atomic number* type does not allow atoms with different atomic number to be matched. Thus, atom pairs with different atomic numbers will not be represented by a node in the correspondence graph, and they will also not be considered in the greedy matching-search by setting their distance to infinity. In this case we can use the score of Eq. (2).

The *pharmacophore* type is based on the atom type classification used in the Merck Molecular Force Field (MMFF) [12]. According to this type declaration, all atoms can be sorted into one or more of the following groups: donors, acceptors, aromatics, and hydrophobics. Each atom pair is rated according to the group membership of the two atoms. The given rate values reflect the importance of a matching pair.

- Donors and acceptors are to be favored most. Therefore, if two atoms belong either both to the donor or both to the acceptor group, they get a rate value of 2.0.
- If two atoms are both aromatic or both hydrophobic, this pair is slightly favored by assigning a rate value of 1.1.
- If one atom is a donor (or acceptor) and the other one is hydrophobic or aromatic but not a donor (or acceptor), we penalize this atom pair by assigning a rate value of 0.5.
- Finally, if two atoms do not belong to either of the above groups, they are assigned a rate value of 1.0.

The weights of the atom pairs as used in Eqs. (3) and (4) are the reciprocal values of their rate values, thus we have weights of 0.5, 0.91, 2.0, and 1.0, respectively.

2.5.2 Clique Detection Parameters

We use three parameters for determining start transformations with clique detection. The first parameter is the *minimum clique size*, min_c. This parameter fits easily into the clique detection algorithm [4], since this is a branch-and-bound algorithm into which the minimum clique size can be integrated as a further bound condition.

The other two parameters are no direct parameters of the clique detection algorithm, but need to be considered during the construction of the correspondence graph. The distance threshold parameter δ_{graph} ensures that every two corresponding atom pairs of a matching have similar distances, and thus, that a clique represents similar substructures (except for inversions at the chirality

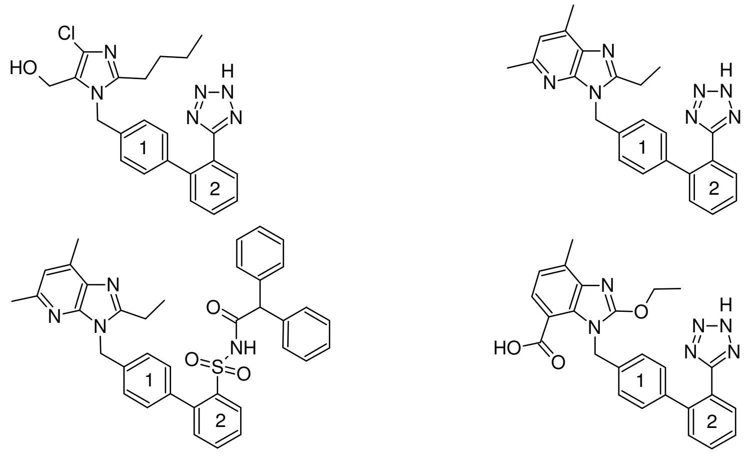

Figure 4: Structural formulas of the angiotensin II antagonists: Losartan, L-158,809, L-159,894, and CV-11974 (from left to right).

center). If δ_{graph} is small, we generally get smaller and fewer cliques. For the size of molecules we are interested in, we usually set $\delta_{graph} = 0.2$.

The bond distance parameter δ_{bond} ensures that atom pairs connected by an edge in the correspondence graph are at least δ_{bond} bonds apart in both molecules. We introduced δ_{bond} to compute start transformations based on substructures containing atoms that are spread over the molecule instead of being too closely packed. E.g., a start transformation based on a substructure of three consecutive atoms (in terms of binding structure) might not be a good starting point. However, δ_{bond} needs to be carefully chosen, since a large δ_{bond} might easily lead to overlooking good matchings. In our experiments we often set δ_{bond} to 2 or 3.

2.5.3 Greedy Matching Parameters

The parameter δ was already introduced in Sect. 2.1.3. In all our experiments we set δ to 2.0. This is in accordance with the observation made in [18]. Due to our scoring function it is unlikely for the size of molecules we consider that increasing δ will lead to different results.

The second parameter is the *minimum matching size*, min_m, which is used to reject matchings that are too small. This parameter is also used in the merging step to reject intersections smaller than min_m.

3 Results

We tested the described algorithm on several groups of molecules. Unfortunately, there are no benchmarks for superimposing molecules. Therefore, we chose two groups that had previously been used for the assessment of superposition algorithms and thus seemed well suited [13, 15]. The first group consisted of a set of four angiotensin II antagonists. For this group the active conformations were not available. We therefore computed the metastable conformations [9] for each of these molecules and used representatives of these conformations as

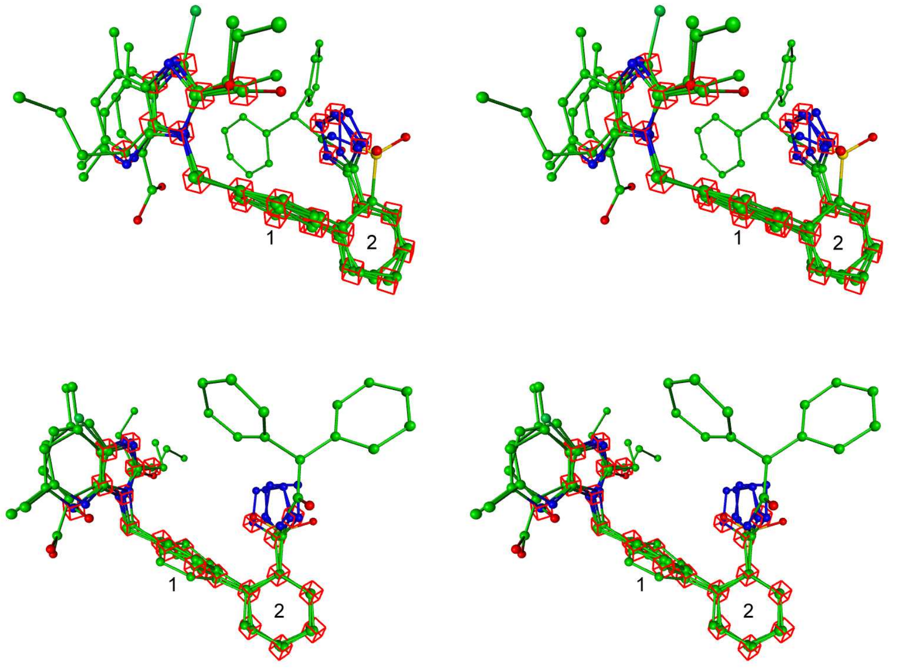

Figure 5: Stereo views of two matchings of four angiotensin II antagonists from the first (*top*) and the second (*bottom*) Pareto set, respectively. The red wire-frame cubes denote the common substructure. The numbers denote the same rings as in Fig. 4.

input to our algorithm. The second group consisted of seven thermolysin inhibitors. For this set of molecules the active conformations were available from the PDB (Protein Data Bank) [1], and thus we used these conformations as input to the algorithm. Further tests were done, e.g., with a group of about 40 odor molecules. The results of these investigations will be described elsewhere.

All computations were performed on a PC workstation with 3GHz processor. The implementations were done in C++, and the source code was integrated into the visualization software AmiraMol [2]. All images in this chapter were also made with AmiraMol.

3.1 Angiotensin II Antagonists

The four angiotensin II antagonists considered here are Losartan, L-158,809, L-159,894, and CV-11974. The two-dimensional structures as well as the nomenclature of these molecules were taken from [36]. The three-dimensional structures were generated using CORINA [31]. These structures were used as input to the program ZIBMol [9], which generated 20, 13, 13, and 25 metastable conformations, respectively. From these metastable conformations we chose the structures with minimum energy as representatives. The representatives were then used as input to our algorithm.

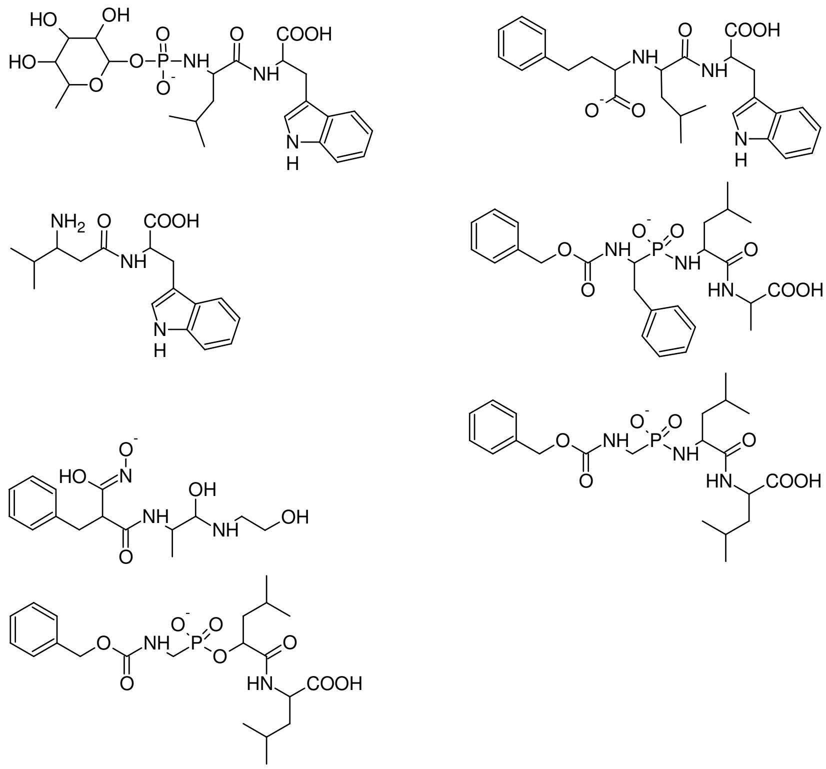

Figure 6: Structural formulas of the thermolysin inhibitors: 1TLP, 1TMN, 3TMN, 4TMN, 5TLN, 5TMN, and 6TMN (from left to right and top to bottom).

3.1.1 Computational Results

For the computation of the multiple structure superposition of the angiotensin II antagonists the following settings were used. The atom type to be considered was *pharmacophore* for both clique detection and greedy matching. The minimum clique size, min_c, was set to 5. The distance threshold, δ_{graph}, was set to 0.2, the minimum matching size, min_m, was set to 15, and the minimum bond distance, δ_{bond}, was set to 1. We chose Losartan as reference molecule.

325196 matching clusters were computed in 116 seconds. The large number of matching clusters is due to the large similarity between the molecules which leads to many slightly different matchings. Two matching were selected, one from the first and one from the second Pareto set, which are shown in Fig. 5. The two matchings differ in the relative orientation of the imidazole ring of Losartan to the double-ring of the other three molecules. In the first matching the second nitrogen atom is not matched to the nitrogen atoms of the other molecules, in the second matching the nitrogen atoms are matched. In both matchings the imidazole ring as well as the aromatic rings are matched between all molecules. Furthermore, the tetrazole ring of Losartan, L-158,809, and CV-11974 is matched to the sulfonamide group of L-159,894. These groups are probably responsible for forming hydrogen bonds to the receptor.

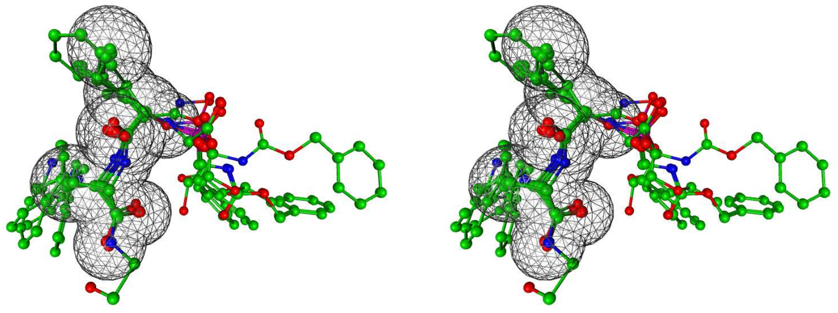

Figure 7: Stereo view of matching of all 7 thermolysin inhibitors. The matching has size 12. The wire-frame surface denotes the common substructure.

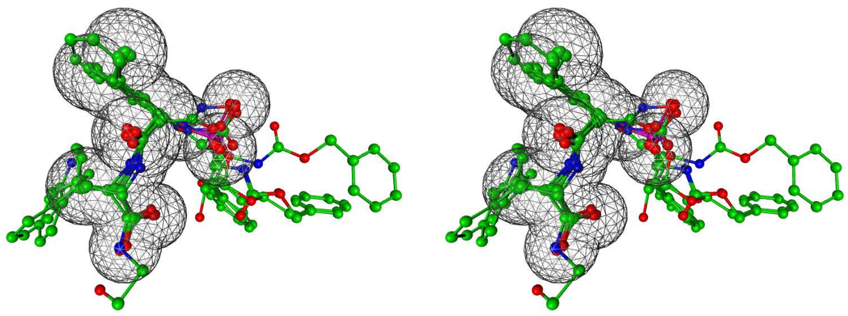

Figure 8: Stereo view of matching of 6 thermolysin inhibitors (without 3TMN). The matching has size 16. The wire-frame surface denotes the common substructure.

3.2 Thermolysin Inhibitors

The active conformations of seven inhibitors of thermolysin (TLN, EC-number 3.4.24.27) were compared with each other. These inhibitors were extracted from the enzyme-inhibitor complexes found in the PDB: 1TLP, 1TMN, 3TMN, 4TMN, 5TLN, 5TMN, and 6TMN. The separated inhibitors were then parametrized using the Merck Molecular Force Field (MMFF) implemented in the program ZIBMol. The parametrization was needed for assigning atom types (donor, acceptor, hydrophobic, aromatic).

3.2.1 Computational Results

For the computation of the multiple structure superposition of the seven thermolysin inhibitors the following settings were used. The atom type to be considered was *pharmacophore* for both clique detection and greedy matching. The minimum clique size, min_c, was set to 5. The distance threshold, δ_{graph}, was set to 0.2, the minimum matching size, min_m, was set to 8, and the minimum bond distance, δ_{bond}, was set to 1. We chose the 4TMN inhibitor as reference molecule. However, this choice was somewhat arbitrary since we got similar results for the other inhibitors except for that of 3TMN. This is due to the size of the 3TMN inhibitor, which is considerably smaller and in particular misses the functional group responsible for binding to the zinc ion in the active site.

The computation of the superposition took less than a second. 308 matchings were computed which were sorted into 70 Pareto sets. Two superpositions from the Pareto optimal set, namely the one with all inhibitors and the one with 6 inhibitors are shown in Figs. 7 and 8, respectively. Due to the size of the 3TMN inhibitor and the absence of the functional group responsible for binding to the zinc ion, the first matching size is considerably smaller with 12 atoms in contrast to 16 atoms.

Of special interest in the second matching is the zinc ion binding group. For the 1TLP, 4TMN, 5TMN, and 6TMN inhibitors this region is identical, consisting of a phosphor atom which two oxygen atoms are bound to. The 1TMN and 5TLN inhibitors, however, differ in this region from the rest and between each other. At the position of the phosphor atom, in the 1TMN inhibitor we find a carbon which a COO- is bound to. That is, apart from the absence of the phosphor atom, there is an additional carbon which slightly dislocates the oxygen atoms. This is similar to the 5TLN inhibitor, here, however, we have an additional nitrogen atom inserted between the carbon and the oxygen atoms, further dislocating one of the oxygen atoms. Despite this, the oxygen atoms were correctly matched.

4 Discussion

We presented a new algorithm for multiple semi-flexible superposition of drug-sized molecules. Our algorithm is similar to DISCO [23] in two points. (1) A reference molecule needs to be specified to which all other molecules are compared. (2) The results of these pairwise comparisons are then merged to get substructures present in all or many molecules. DISCO uses clique detection to identify common substructures. Clique detection, however, is limited to rather small sets of points. DISCO circumvents this problem by prespecifying a few pharmacophore points which are used instead of all atoms. Our approach, in constrast, uses the full atomic information. Using clique detection we would only be able to identify rather small common substructures. We therefore use a two-step approach for the comparison of two molecular structures. In the first step, clique detection is used to generate good start transformations. The second step refines each start transformation and computes the matching. The matchings found using this two-step approach are, in general, considerably larger than those found by clique detection alone. However, the large matchings that we get impose a new problem. Merging of the pairwise results becomes much more expensive. We solve this problem by introducing matching trees to the merging step.

We tested our algorithm on several groups of molecules, the results of two of which are presented in Sect. 3. These results suggest, that the new approach is very well capable of identifying common substructures in a set of molecules. The algorithm depends on the computation of resonable molecular conformations. However, the algorithm is fault-tolerant to the absence of conformations in that it also computes substructures present in only a few molecules.

The results of the algorithm present only suggestions to the user. The classification of the matchings in Pareto sets allows the user to view the results starting with the best matchings. In some cases, matchings most interesting to the user might not be found in the Pareto optimal set (see Sect. 3.1.1). The large number of computed matchings, however, allows the user to look at matchings determined to be not-so-good by the algorithm.

The parameters of the algorithm allow the user to decide how thoroughly the search for common substructures should be performed. A trade-off has to be found between runtime and search detail.

In the future, the main focus will be on considering even more conformations per molecule while keeping short runtimes. The runtime of our algorithm, however, is mainly determined by the number of conformations per molecule. It grows quadratically with the number of conformations. Currently, although some conformations might be similar in large parts of the structure, pairwise comparisons are executed separately for all conformations. Exploiting these similarities might enable us to considerably reduce the algorithm's runtime, and, thus, it might allow for even more flexibility.

Acknowledgments. I would like to thank Johannes Schmidt-Ehrenberg for many helpful discussions and his comments on the paper, and Frank Cordes for his help in generating conformational ensembles and his valuable hints concerning the paper.

References

[1] Protein Data Bank (PDB). URL http://www.rcsb.org/pdb.

[2] *AmiraMol – User's Guide and Reference Manual.* Zuse Institute Berlin (ZIB) and Indeed - Visual Concepts GmbH, Berlin, http://www.amiravis.com, 2002.

[3] Andrew T. Brint and Peter Willett. Algorithms for the identification of three-dimensional maximal common substructures. *J. Chem. Inf. Comput. Sci.*, 27:152–158, 1987.

[4] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[5] D. Cosgrove, D. M. Bayada, and A. P. Johnson. A novel method of aligning molecules by local surface shape similarity. *Journal of Computer-Aided Molecular Design*, 14:573–591, 2000.

[6] R.A. Dammköler, S.F. Karasek, E.F. Shands, and G.R. Marshall. Sampling conformational hyperspace: techniques for improving completeness. *J. Comput. Aided Mol. Des.*, 9(6):491–499, 1995.

[7] E. Feuilleaubois, V. Fabart, and J. P. Doucet. Implementation of the three-dimensional-pattern search problem on hopfield-like neural networks. *SAR QSAR Environ Res.*, 1(2-3):97–114, 1993.

[8] Paul W. Finn, Lydia E. Kavraki, Jean-Claude Latombe, , Rajeev Motwani, Christian R. Shelton, Suresh Venkatasubramanian, and A. Yao. RAPID: randomized pharmacophore identification for drug design. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 324–333. ACM Press, 1997.

[9] Alexander Fischer, Christof Schütte, Peter Deuflhard, and Frank Cordes. Hierarchical uncoupling-coupling of metastable conformations. In Tamar Schlick and Hin Hark Gan, editors, *Computational Methods for Macromolecules: Challenges and Applications, Proceedings of the 3rd International Workshop on Algorithms for Macromolecular Modeling, New York, Oct. 12–14, 2000*, volume 24 of *Lecture Notes in Computational Science and Engineering*, pages 235–259, Berlin, 2002. Springer.

[10] Andrew C. Good and W. Graham Richards E. E. Hodgkin. Utilization of Gaussian functions for the rapid evaluation of molecular similarity. *Journal of Chemical Information and Computer Sciences*, 32(3):188–191, 1992.

[11] J. A. Grant, M. A. Gallardo, and B. T. Pickup. A fast method of molecular shape comparison: A simple application of a Gaussian description of molecular shape. *Journal of Computational Chemistry*, 17(14):1653–1666, 1996.

[12] T. A. Halgren. Merck molecular force field. I-V. *J. Comp. Chem.*, 17(5&6):490–641, 1996.

[13] Sandra Handschuh. *Entwicklung und Einsatz computergestützter Methoden zur Ermittlung struktureller Ähnlichkeiten: Analyse biologisch relevanter Ligand-Rezeptor Wechselwirkungen.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999.

[14] Sandra Handschuh, Markus Wagener, and Johann Gasteiger. Superposition of three-dimensional chemical structures allowing for conformational flexibility by a hybrid method. *J. Chem. Inf. Comput. Sci.*, 38:220–232, 1998.

[15] Christian Hofbauer. *Molecular Surface Comparison. A Versatile Drug Discovery Tool.* PhD thesis, Technische Universität Wien, 2004.

[16] Gareth Jones, Peter Willett, and Robert C. Glen. A genetic algorithm for flexible molecular overlay and pharmacophore elucidation. *J. Comput. Aided Mol. Des.*, 9(6):532–549, 1995.

[17] Wolfgang Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica A*, 34:827–828, 1978.

[18] Stefan Kirchner. Ein Approximationsalgorithmus zur Berechnung der Ähnlichkeit dreidimensionaler Punktmengen. Diploma Thesis, Department of Computer Science, Humboldt University Berlin, 2003.

[19] Gerhard Klebe, Thomas Mietzner, and Frank Weber. Different approaches toward an automatic structural alignment of drug moleculars: Applications to sterol mimics, thrombin and thermolysin inhibitors. *Journal of Computer-Aided Molecular Design*, 8(6):751–778, 1994.

[20] Christian Lemmen, Claus Hiller, and Thomas Lengauer. RigFit: A new approach to superimposing ligand molecules. *Journal of Computer-Aided Molecular Design*, 12(5):491–502, 1998.

[21] Christian Lemmen and Thomas Lengauer. FLEXS: a method for fast flexible ligand superposition. *J. Med. Chem.*, 41(23):4502–4520, 1998.

[22] Christian Lemmen and Thomas Lengauer. Computational methods for the structural alignment of molecules. *Journal of Computer-Aided Molecular Design*, 14:215–232, 2000.

[23] Yvonne C. Martin, Mark G. Bures, Elisabeth Danaher, Jerry DeLazzer, and Isabella Lico. A fast new approach to pharmacophore mapping and its application to dopaminergic and benzodiazepine agonists. *Journal of Computer-Aided Molecular Design*, 7:83–102, 1993.

[24] Yvonne C. Martin, Mark G. Bures, and Peter Willett. Searching databases of three-dimensional structures. In Kenny B. Lipkowitz, editor, *Reviews in Computational Chemistry*, number 1, pages 213–263. Elsevier Science Publishers B.V., 1990.

[25] B. B. Masek, A. Merchant, and J. B. Matthew. Molecular shape comparison of angiotensin II receptor antagonists. *J. Med. Chem.*, 36(9):1230–1238, 1993.

[26] Alan J. McMahon and Paul M. King. Optimization of Carbó molecular similarity index using gradient methods. *Journal of Computational Chemistry*, 18(2):151–158, 1997.

[27] C. McMartin and R.S. Bohacek. Flexible matching of test ligands to a 3d pharmacophore using a molecular superposition force field: comparison of predicted and experimental conformations of inhibitors of three enzymes. *J. Comput. Aided Mol. Des.*, 9(3):237–250, 1995.

[28] M. D. Miller, R. P. Sheridan, and S. K. Kearsley. Sq: a program for rapidly producing pharmacophorically relevent molecular superpositions. *Journal Med. Chem.*, 42(9):1505–1514, 1999.

[29] J. W. M. Nissink, M. L. Verdonk, J. Kroon, and G. Klebe T. Mietzner. Superposition of molecules: Electron density fitting by application of Fourier transforms. *Journal of Computational Chemistry*, 18(5):638–645, 1997.

[30] Martin F. Parretti, Romano T. Kroemer, Jeffrey H. Rothman, and W. Graham Richards. Alignment of molecules by the Monte Carlo optimization of molecular similarity indices. *Journal of Computational Chemistry*, 18(11):1344–1353, 1997.

[31] Jens Sadowski and Johann Gasteiger. From Atoms and Bonds to Three-Dimensional Atomic Coordinates: Automatic Model Builders. In *Chemical Reviews*, pages 2567–2581. 1993.

[32] Robert P. Sheridan, Ramaswamy Nilakantan, J.S. Dixon, and R. Venkataraghavan. The ensemble approach to distance geometry: application to the nicotinic pharmacophore. *J. Med. Chem.*, 29(6):899–906, 1986.

[33] Martin Thimm, Andrean Goede, Stefan Hougardy, and Robert Preissner. Comparison of 2d similarity and 3d superposition. Application to searching a conformational drug database. *J. Chem. Inf. Comput. Sci.*, 44:1816–1822, 2004.

[34] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:676–681, 1991.

[35] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classification, Analyses, and New Innovations.* PhD thesis, Faculty of the Graduate School of Engineering of the Air Force Institute of Technology, Air University, 1999.

[36] Ruth R. Wexler, William J. Greenlee, John D. Irvin, Michael R. Goldberg, Kristine Prendergast, Ronald D. Smith, and Pieter B. M. W. M. Timmermans. Nonpeptide Angiotensin II Receptor Antagonists: The Next Generation in Antihypertensive Therapy. *J. Med. Chem.*, 39(3):625–656, 1996.