Chapter 6

Multiple Surface Alignment

In his book "Algorithms on strings, trees, and sequences" [72], Chapter 14, Dan Gusfield motivates the usefulness of *multiple string comparison* as follows.

> *"For a computer scientist, the multiple string comparison problem may at first seem like a generalization for generalization's sake – 'two strings good, four strings better'. But in the context of molecular biology, multiple string comparison (of DNA, RNA, protein strings) is much more than a technical exercise. It is the most critical cutting-edge tool for* extracting and representing *biologically important, yet faint or widely dispersed, commonalities from a set of strings. These (faint) commonalities may reveal evolutionary history, critical conserved motifs or conserved characters in DNA or protein, common two- and three-dimensional molecular structure, or clues about the common biological function of the strings. These characterizations are then used in database searches to identify other potential members of a family. Because many important commonalities are faint or widely dispersed, they might not be apparent when comparing two strings alone but may become clear, or even obvious, when comparing a set of related strings."*

Clearly, *multiple sequence alignment* and *multiple structure alignment* are very different in many aspects, yet, most of the above citation is also true for multiple structure alignment. For example, the commonalities among a set of molecular structures will, in general, be much smaller than between two molecular structures. But, these *faint* commonalities might in fact be the essential properties a molecular structure needs to have to perform a specific function. Of course, whether the true essentials of some biological function will be found, strongly depends on the diversity of the molecular structures. The essential structural motifs that are responsible for the biological function of the molecules build the pharmacophore, which can be used to search databases *"to identify other potential members of a family"* [72], here, the family of active compounds.

In order to compute a multiple surface alignment from pairwise surface alignments, it is *not* sufficient to take only the best pairwise alignments of all molecules and overlay the molecules according to these best pairwise alignments. The reason for this lies in the fact, that a best pairwise alignment might be the best alignment only due to some "coincidental" similarity of two molecules. This coincidental similarity might not have anything to do with the essential features responsible for some biological function. Instead, there might be other pairwise alignments with slightly worse scores, which, however, might be preferred in the context of a multiple alignment. Nevertheless, it seems to be reasonable that a small set of diverse good pairwise alignments, e.g. 10 or 20, contains the most important alignments for pharmacophore identification. This assumption has to be made in order to be able to compute the multiple surface alignments with the method described in this chapter.

This chapter begins in Section 6.1 with a short overview. This is followed by the main part of this chapter, Section 6.2, in which we explain how to derive multiple alignments from pairwise alignments. Section 6.3 explains, how multiple matchings are scored and sorted using the concept of Pareto dominance. In Section 6.4, we give experimental results, which are followed by a short summary and conclusion in Section 6.5.

6.1 Overview

The computation of multiple alignments from pairwise alignments essentially involves four steps:

1. Computation of pairwise alignments.
2. Reduction of the number of pairwise alignments.
3. Computation of multiple alignments from pairwise alignments.
4. Scoring and sorting of multiple alignments.

Computation of pairwise alignments. The computation of pairwise alignments is a two-step process, which has been thoroughly described in the previous chapter. In the first step of this process, initial transformations are generated (cf. Section 5.2), which are locally optimized in the second step, using an iterative point matching scheme (cf. Section 5.3). The pairwise matchings that were thus computed define pairwise alignments, which can be generated by rms-fitting the matched points.

Reduction of the number of pairwise alignments. Among the pairwise alignments generated by the pairwise surface alignment algorithm, we generally find many alignments that are very similar to other pairwise alignments. In particular, around local maxima of the matching score, the number of similar pairwise alignments is large. These similar pairwise alignments again will lead to many similar multiple alignments being generated in the second step. But not only the large number of similar multiple alignments is a problem, but also the increase in the computational cost is enormous. Due to this observation, the reduction of pairwise alignments is a crucial step. To realize it, we first compute a partitioning of the pairwise alignments w.r.t. the rigid transformations. Then, to allow execution of the second step, we reduce the number of pairwise alignments to a predefined size, say 10 or 20. The details of reducing the pairwise alignments have been given in Section 5.4.

Computation of multiple alignments from pairwise alignments. Once the number of pairwise alignments has been reduced to a practical size, we can perform the actual generation of multiple alignments. Here, we work on the pairwise matchings underlying the pairwise alignments. From the pairwise matchings, we compute multiple matchings by sequentially intersecting the pairwise matchings of different query molecules. That is to say, we start by intersecting the pairwise matchings of two query molecules. The resulting multiple matchings from this first intersection, including the original pairwise matchings, are then intersected with the pairwise matchings of the third query molecule, and so forth, until the pairwise matchings of all query molecules have been handled. This step will be described in detail in Section 6.2.

Scoring and sorting of multiple alignments. The computation of multiple alignments is only finished, if the multiple alignments have been scored and sorted, such that the user has a means of traversing the multiple alignments in a feasible order. This last step, which uses the concept of Pareto dominance, will be explained in Section 6.3.

6.2 From Pairwise to Multiple Alignments

6.2.1 Multiple Matchings

Multiple matchings are the basis for the computation of multiple alignments. They constitute a straightforward extension to pairwise matchings (cf. Definition 5.1.5) introduced in Chapter 5.

Definition 6.2.1 (Multiple Matching)**.** A *multiple matching* f defined on $k+1$ finite sets P, Q^1, …, Q^k is a mapping

$$f : \widetilde{P} \to \widetilde{Q}^1 \times \widetilde{Q}^2 \times \ldots \times \widetilde{Q}^k, \ \widetilde{P} \subseteq P \ , \ \widetilde{Q}^j \subseteq Q^j, \ j = 1, \ldots, k \ ,$$

where each of its *components* $f^j : \widetilde{P} \to \widetilde{Q}^j$ is a pairwise matching. The components of f will also be referred to as the *pairwise matchings induced by* f. We further define

$$\begin{aligned} f^P &:= \{p \in P \mid \forall j \in \{1, \ldots, k\} \ \exists q \in Q^j : f^j(p) = q\} \ , \text{ and} \\ f^{Q^j} &:= \{q \in Q^j \mid \exists p \in P : f^j(p) = q\} \ , \end{aligned}$$

where $f^P = \widetilde{P}$ is the *domain* of f and $f^{Q^1} \times f^{Q^2} \times \ldots \times f^{Q^k}$ its *range.*

We refer to $dim(f) := k$ as the *dimension* of the multiple matching f. The multiple matching f can also be written as a set f^* of (k+1)-tuples from $P \times Q^1 \times Q^2 \times \ldots \times Q^k$ with $(p, q_1, \ldots, q_k) \in f^* \Leftrightarrow f^j(p) = q_j, \ q_j \in Q^j, \ \forall j \in \{1, \ldots, k\}$. The number $|f^*|$ of (k+1)-tuples will be referred to as the *size* of f.

The set of all multiple matchings f with domain $f^P \subseteq P$ and range $f^{Q^1} \times f^{Q^2} \times \ldots \times f^{Q^k} \subseteq Q^1 \times Q^2 \times \ldots \times Q^k$ is denoted by $\mathcal{F}(P, Q^1, Q^2, \ldots, Q^k)$.

Remark 6.2.2 (Reference and Query Sets)**.** While for a pairwise matching $f \in \mathcal{F}(P, Q)$ the sets P and Q can be exchanged, in a multiple matching $\tilde{f} \in \mathcal{F}(P, Q^1, Q^2, \ldots, Q^k)$ the set P has an extraordinary status. We call P the *reference set* and the sets $Q^1, Q^2, \ldots, Q^k$ the *query sets.* Likewise, the underlying molecules will be called *reference molecule* and *query molecules*, respectively.

Remark 6.2.3 (Mapping of Query Sets)**.** In the first place, a multiple matching defines a mapping of the reference set to the Cartesian product of the query sets. However, implicitly it also defines mappings between the query sets, where two points $q_k \in Q^k$ and $q_l \in Q^l$ are mapped to each other, if they belong to the same multiple matching tuple. Hence, we can think of a multiple matching as a mapping that relates the elements of all sets participating in the matching to each other.

The definition of a combined pairwise matching (cf. Definition 5.3.1) w.r.t. decompositions of the involved point sets can also be extended to multiple matchings in a straightforward way. This leads us to the definition of *combined multiple matchings.*

Definition 6.2.4 (Combined Multiple Matching)**.** Let P, Q^1, …, Q^k be finite sets with decompositions $\{P_i\}_{i=1}^N$, $\{Q_i^1\}_{i=1}^N$, …, $\{Q_i^k\}_{i=1}^N$, respectively (cf. Definition 5.3.1). Furthermore, let $f_i \in \mathcal{F}(P_i, Q_i^1, Q_i^2, \ldots, Q_i^k)$, $i = 1, \ldots, N$, be multiple matchings on P_i, Q_i^1, …, Q_i^k. Then we define the *combined multiple matching* $f \in \mathcal{F}(P, Q^1, Q^2, \ldots, Q^k)$ w.r.t. the decompositions $\{P_i\}_{i=1}^N$, $\{Q_i^1\}_{i=1}^N$, …, $\{Q_i^k\}_{i=1}^N$ by

$$f(p) := f_i(p), \text{ if } p \in P_i \ .$$

It directly follows that

$$f^* = \bigcup_{i=1}^{N} f_i^* \ .$$

Even though the notion of a pairwise matching is easily extended to a multiple matching, the score of a multiple matching is not as easily generalized. If we again take a look at multiple sequence alignment, the *sum-of-pairs score*, given by the sum of scores of the pairwise sequence alignments induced by the multiple sequence alignment, has been used by many people [72]. In analogy to multiple sequence alignments, we can define the *sum-of-pairs score* of a multiple matching f as the sum of scores of the pairwise matchings induced by f (cf. Definition 6.2.1).

Definition 6.2.5 (Sum-of-Pairs Score)**.** Let $f : \widetilde{P} \to \widetilde{Q}^1 \times \widetilde{Q}^2 \times \ldots \times \widetilde{Q}^k$ be a multiple matching and let $f^j : \widetilde{P} \to \widetilde{Q}^j$, $j = 1, \ldots, k$, be the pairwise matchings induced by f. Then the *sum-of-pairs score* of the multiple matching f is defined as

$$\text{score}^*_{sp}(P, Q^1, \ldots, Q^k; f) := \sum_{j=1}^{k} \text{score}^*(P, Q^j; f^j) \ .$$

This score is only comparable for multiple matchings with the same dimension k, i.e. for multiple matchings in which the same number of molecules are involved. In order to be able to compare multiple matchings of different dimensions, we need to normalize the sum-of-pairs score, yielding the *normalized sum-of-pairs score.*

Definition 6.2.6 (Normalized Sum-of-Pairs Score)**.** Let $f : \widetilde{P} \to \widetilde{Q}^1 \times \widetilde{Q}^2 \times \ldots \times \widetilde{Q}^k$ be a multiple matching. Then the *normalized sum-of-pairs score* of f is defined as

$$\text{score}^*_{nsp}(P, Q^1, \ldots, Q^k; f) := \frac{\text{score}^*_{sp}(P, Q^1, \ldots, Q^k; f)}{k} \ . \qquad (6.1)$$

6.2.2 Computing Multiple Matchings

In this section we explain how we generate multiple matchings from sets of pairwise matchings. Essentially, this is done by representing each pairwise matching as a bit-string and successively intersecting these bit-strings to generate multiple matchings. After each intersection, we need to check whether the intersected bit-string already exists. To do this efficiently, we use PATRICIA trees. For ease of reading, we only describe PATRICIA trees in Section 6.2.3. Since it is a technical detail not necessary for understanding the computation of multiple matchings, it may be skipped at first reading. For the following subsection it should suffice, that PATRICIA trees enable us to quickly identify equal bit-strings. Each bit-string is represented by a single leaf, which stores the multiple matchings corresponding to the respective bit-string.

Intersection of Matching Bit-Strings

The first step in computing multiple matchings from pairwise matchings is to represent the domain of each pairwise matching as a bit-string of length equal to the number of elements in the reference set (cf. Remark 6.2.2), where each bit is set, if and only if the corresponding element of the reference set is in the domain of the pairwise matching.

Definition 6.2.7 (Pairwise Matching Bit-String)**.** Let $f \in \mathcal{F}(P, Q)$ be a pairwise matching with domain $f^P \subseteq P$. Then $b_f : P \to \{0, 1\}$, defined by

$$b_f(p) := \begin{cases} 1 & \text{, if } p \in f^P \\ 0 & \text{, otherwise} \end{cases} ,$$

denotes the mapping of f to its associated matching bit-string b_f. For a bit-string b we denote by $|b|$ the number of bits equal to 1, i.e. $|b_f| = |f^P|$. The definition of a pairwise matching bit-string can be directly transferred to multiple matchings. Thus, for a multiple matching we will use the same notations as for pairwise matchings.

Since we consider pairwise matchings with the same reference set P only, the bit-strings of these pairwise matchings have equal lengths. For two such strings, the *intersection* can be defined as follows.

Definition 6.2.8 (Intersection of Matching Bit-Strings)**.** Let f' and f'' be two matchings, pairwise or multiple, with matching bit-strings $b_{f'}$ and $b_{f''}$, respectively. Then the intersection of $b_{f'}$ and $b_{f''}$ is denoted by $b_{f'} \wedge b_{f''}$ and defined by

$$(b_{f'} \wedge b_{f''})(p) := \begin{cases} 1 & \text{, if } b_{f'}(p) = 1 \wedge b_{f''}(p) = 1 \\ 0 & \text{, otherwise} \end{cases} .$$

It is not feasible to compute all intersections, in particular those that have a small number of bits set to 1, since the number of intersection bit-strings grows exponentially with the dimension of the multiple matchings. We therefore need to restrict this number to a manageable size. We do this by introducing a *minimum size constraint*, which is a user-specified minimum of the number of bits set to 1 in a multiple matching bit-string.

Definition 6.2.9 (Minimum Size of Multiple Matchings)**.** Let P be the reference point set, and let $Q^1, \dots, Q^M$ be the query point sets. Let $F_i \subset \mathcal{F}(P, Q^i), i = 1, \dots, M$, denote the sets of computed pairwise matchings of point sets P and Q^i. For $F_i, i = 1, \dots, M$, we denote by $\mathit{minSize}(F_i)$ the minimum size a multiple matching containing query set Q^i needs to have.

In our experiments, we specified the $\mathit{minSize}(F_i)$ with the help of a *relative minimum size*, $\mathit{minSize}_{\mathrm{rel}} \in (0.0, 1.0]$, such that

$$\mathit{minSize}(F_i) = \mathit{minSize}_{\mathrm{rel}} \cdot \max_{f_i \in F_i} |f_i| \ . \tag{6.2}$$

Since the maximum size of the pairwise matchings, i.e. the maximum number of bits set to 1 in any pairwise matching bit-string, may vary considerably among the pairwise matching sets F_i, this minimum varies from query point set to query point set. Recall, that we are not only interested in multiple matchings from $\mathcal{F}(P, Q^1, Q^2, \dots, Q^M)$, but in multiple matchings from all possible sets $\mathcal{F}(P, Q^{l_1}, Q^{l_2}, \dots, Q^{l_k})$, $k = 1, \dots, M$, and $l_j \in \{1, \dots, M\}$ pairwise distinct. Included here are pairwise matchings, which, in fact, are multiple matchings of dimension 1 and as such they will be handled.

In order to be able to give the complete algorithm, we need one further definition, namely the definition of a *maximal multiple matching.*

Definition 6.2.10 (Maximal Multiple Matching)**.** Let P be the reference point set, and let $Q^1, \dots, Q^M$ be the query point sets. Furthermore, let $\widetilde{F}$ be a set of multiple matchings from all possible sets $\mathcal{F}(P, Q^{l_1}, Q^{l_2}, \dots, Q^{l_k})$, $k = 1, \dots, M$, and $l_j \in \{1, \dots, M\}$ pairwise distinct. More precisely,

$$\widetilde{F} \subset \bigcup_{\substack{k=1,\dots,M \\ l_j \in \{1,\dots,M\} \\ l_i \neq l_j, \forall i \neq j}} \mathcal{F}(P, Q^{l_1}, Q^{l_2}, \dots, Q^{l_k}) \ .$$

Then a multiple matching $f \in \widetilde{F}$ is a *maximal multiple matching* w.r.t. $\widetilde{F}$ if and only if there does not exist a multiple matching $\tilde{f} \in \widetilde{F}$ satisfying the following two properties (maximality constraints):

1. $f^P = \tilde{f}^P$, and
2. $\mathit{dim}(f) < \mathit{dim}(\tilde{f})$.

The Algorithm

The aim of the algorithm is to compute all maximal multiple matchings (cf. Definition 6.2.10) satisfying the minimum size constraint (cf. Definition 6.2.9) given by the values $\mathit{minSize}(F_i), i = 1, \dots, M$. The complete algorithm for this problem is given in Algorithm 6.1.

We store the maximal multiple matchings, as they are computed, in the PATRICIA tree. For each leaf v representing a multiple matching we maintain a list of matchings, denoted by *v.matchings*. This list contains all pairwise matchings that make up the multiple

Algorithm 6.1 Computation of multiple matchings

Input: Reference set P and query sets $Q^1, \ldots, Q^M$
Input: Family $\mathcal{F} = \{F_i\}_{i=1}^M$ of sets of pairwise matchings of P and Q^i, respectively
Output: PATRICIA tree T, implicitly containing all maximal multiple matchings

```
1: $\widetilde{F} \leftarrow \arg\min_{F \in \mathcal{F}} \mathit{minSize}(F)$
2: for all $f \in \widetilde{F}$ do    ▷ Initialize PATRICIA tree $T$.
3:     $v \leftarrow$ insert $b_f$ into $T$    ▷ Algorithm 6.2
4:     $v.\mathit{matchings} \leftarrow v.\mathit{matchings} \cup f$
5:     $v.\mathit{minSize} \leftarrow \mathit{minSize}(\widetilde{F})$
6: end for
7: $\mathcal{F} \leftarrow \mathcal{F} \setminus \widetilde{F}$
8: while $\mathcal{F} \neq \emptyset$ do
9:     $\widetilde{F} \leftarrow \arg\min_{F \in \mathcal{F}} \mathit{minSize}(F)$
10:    clear PATRICIA tree $\widetilde{T}$
11:    for all $f \in \widetilde{F}$ do
12:        $v \leftarrow$ insert $b_f$ into $\widetilde{T}$    ▷ Algorithm 6.2
13:        $v.\mathit{matchings} \leftarrow v.\mathit{matchings} \cup f$
14:    end for
15:    $L_T \leftarrow$ all leaves $v$ of $T$    ▷ Copy leaves to $L_T$.
16:    for all leaves $\tilde{v}$ of $\widetilde{T}$ do
17:        for all $v \in L_T$ do
18:            $b \leftarrow \tilde{v}.\mathit{string} \wedge v.\mathit{string}$    ▷ Intersection.
19:            if $|b| \geq v.\mathit{minSize}$ then    ▷ Valid multiple matching.
20:                $w \leftarrow$ insert $b$ into $T$
21:                if $w.\mathit{matchings} = \emptyset$ then    ▷ Is $w$ a new leaf?
22:                    $w.\mathit{minSize} \leftarrow v.\mathit{minSize}$
23:                end if
24:                for all $f \in v.\mathit{matchings} \cup \tilde{v}.\mathit{matchings}$ do
25:                    if $f \notin w.\mathit{matchings}$ then    ▷ Ensure that matching is unique.
26:                        $w.\mathit{matchings} \leftarrow w.\mathit{matchings} \cup \{f\}$
27:                    end if
28:                end for
29:            end if
30:        end for
31:        for all $f \in \tilde{v}.\mathit{matchings}$ do    ▷ Insert pairwise matchings.
32:            $w \leftarrow$ insert $\tilde{v}.\mathit{string}$ into $T$
33:            if $w.\mathit{matchings} = \emptyset$ then    ▷ Is $w$ a new leaf?
34:                $w.\mathit{minSize} \leftarrow \mathit{minSize}(\widetilde{F})$
35:            end if
36:            if $f \notin w.\mathit{matchings}$ then    ▷ Ensure that matching is unique.
37:                $w.\mathit{matchings} \leftarrow w.\mathit{matchings} \cup \{f\}$
38:            end if
39:        end for
40:    end for
41:    $\mathcal{F} \leftarrow \mathcal{F} \setminus \widetilde{F}$
42: end while
```

matching. By $v.minSize$ we denote the minimum size a multiple matching containing the matchings stored at v must have.

After Algorithm 6.1 has been run, each leaf of the PATRICIA tree T might in fact contain more than one multiple matching, since more than one pairwise matching of the same set F_i may be contained in the list of matchings. If this happens, we simply generate all possible combinations of pairwise matchings. In practice, however, this is rarely the case, in particular for large point sets.

The minimum size constraint of all multiple matchings is guaranteed by lines 1, 5, 9, 19, 22, and 34. Lines 1 and 9 ensure that we combine the sets of pairwise matchings in a sorted order, starting with the pairwise matching set F, having the smallest $minSize(F)$, i.e., which introduces the most restrictive constraint. In lines 5, 22, and 34 we set the *minSize*-value for new leaves. Since we start with the pairwise matching set having the smallest *minSize*-value, these lines together with line 19 guarantee that each multiple matching obeys the minimum size constraint. Here, line 19 decides whether a new multiple matching, created by intersecting a previously generated multiple matching with a new pairwise matching, is a valid multiple matching. Recall, that $|b|$ denotes the number of bits in b set to 1.

The maximality constraints (cf. Definition 6.2.10) are satisfied by merging all multiple matchings as well as all pairwise matchings into the same PATRICIA tree T (cf. lines 3, 20, and 32). That is to say, this constraint is guaranteed by construction of the algorithm.

The reason for line 15 lies in the fact that we do not want new multiple matchings to be created due to "self-intersection", i.e. due to intersection of a multiple matching already containing a matching from set F with a pairwise matching of the same set F. Pairwise matchings of the same set are only inserted into the same leaf by merging all newly generated multiple matchings into the same PATRICIA tree T (cf. previous paragraph). Therefore, we need to copy all leaves of tree T before we start intersecting the leaves with the pairwise matchings of a new set.

Beside the "exhaustive" computation of multiple matchings as described here, we also implemented a *greedy* method, which was only used in this thesis to compute multiple alignments for the set of odor molecules presented in the introduction, Section 1.2. In this greedy method, we start with the largest pairwise matching as multiple matching and subsequently intersect the current multiple matching with a new pairwise matching, such that in each step the multiple matching is maximized. However, since in each step we compute the maximal multiple matching only w.r.t. the previous maximal multiple matching, it is not guaranteed that we find the true optimum, as is often the case with greedy methods.

6.2.3 PATRICIA Trees

During the generation of multiple matchings, which was described in Section 6.2.2, a large number of matchings is generated which need to be stored efficiently such that matchings with equal domain (cf. Definition 6.2.1) can be identified quickly. This can be achieved by representing the domain of each multiple matching as a bit-string of length equal to the number of elements in the reference set (cf. Remark 6.2.2), where each bit is set, if and only

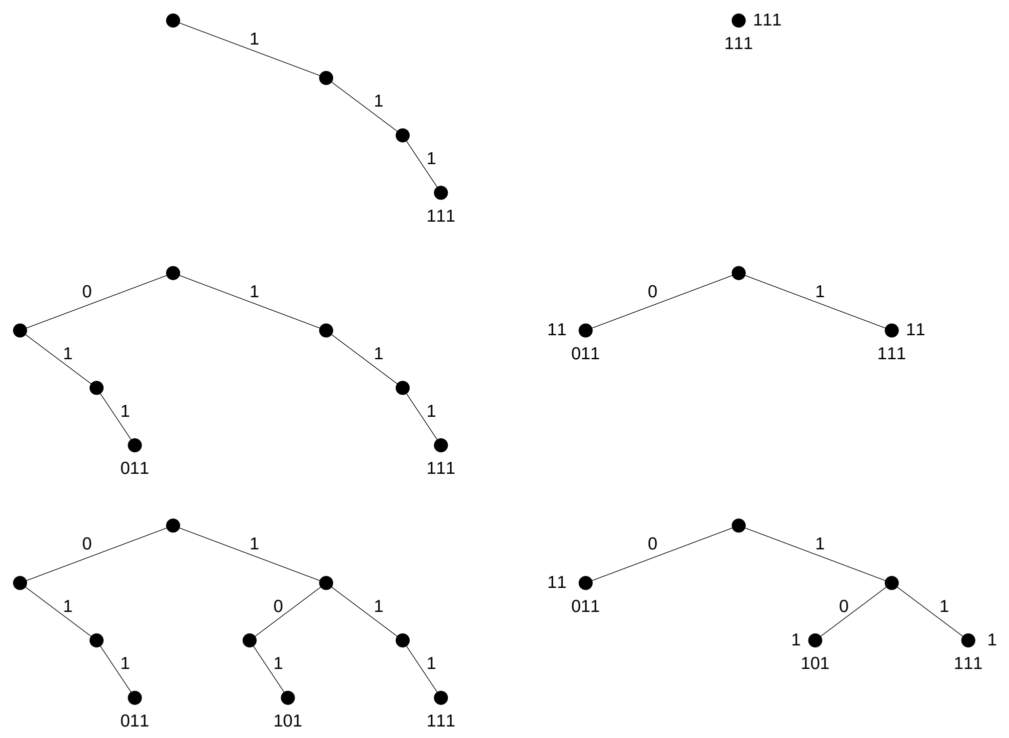

Figure 6.1: Comparison between binary tree (*left*) and PATRICIA tree (*right*), from top to bottom with 1, 2, and 3 stored bit-strings. The numbers along the edges denote a single bit character. The bit-string below each leaf denotes the bit-string stored in that leaf. These bit-strings are equal to the bit-strings built when concatenating the bit characters along the path to the leaf. In the case of the binary tree, these bit characters are simply the bits along the edges. In the case of the PATRICIA tree, the bit-string is made up from the bit characters along the edges plus the substrings next to the nodes, which can have empty substrings.

if the corresponding element of the reference set is in the domain of the multiple matching. With this transformation we can represent the domain of each multiple matching as a bit-string. All bit-strings of multiple matchings with the same reference set have equal lengths. Thus, the domains of two multiple matchings are equal if their bit-strings are equal. So the problem of identifying identical domains reduces to the problem of identifying equal bit-strings.

Kirchner [97] uses a binary tree to store bit-strings. If l is the length of the bit-string, then this binary tree has depth l. Each path from the root of the tree to a leaf represents a different bit-string, where the i'th level represents the i'th bit. Starting from the root, the path is traversed along the right child of the current node, if the corresponding bit is set. Otherwise the path is traversed along the left child. In a binary tree, each path from the root to a leaf has exactly length l (cf. Figure 6.1, left). If the bit-strings are short, binary trees are efficient. However, the number of internal nodes grows exponentially. Thus, for larger bit-string lengths, binary trees need too much storage.

A more efficient data structure, both in terms of run time to identify equal bit-strings and in terms of required storage, was already introduced in 1968 by Donald R. Morrison [128]. This data structure is called PATRICIA tree (PATRICIA – Practical Algorithm To Retrieve Information Code in Alphanumeric). Even though the data structure was originally developed to store alphanumerical strings, it can directly be applied to bit-strings. In a PATRICIA tree, internal nodes are only added if they are necessary for separating distinct bit-strings. As a consequence, the number of internal nodes in a PATRICIA tree is always one less than the number of leaves. So in the simplest case, if only one string is stored in the tree, this string is directly stored in the root node, which in this case is considered to be a leaf (cf. Figure 6.1, right, top). In the binary tree, a bit character is associated with each edge. In the PATRICIA tree, additionally a substring, which may be the empty string, is associated with each node. If we follow a path from the root to a leaf, the bit characters associated with the edges plus the substrings associated with the nodes along the path make up the bit-string represented by that leaf (cf. Figure 6.1, right).

For our purpose we only need a single operation to be performed on the PATRICIA tree, which is *insertion*. When inserting a bit-string into a tree T, we start at the root node. Let $T.r$ denote the root node and for each node v let $v.\mathit{left}$ denote its left child and $v.\mathit{right}$ denote its right child. Furthermore, let $v.\mathit{string}$ denote the substring associated with v, and for a string s let $s.\mathit{length}$ denote the length of s, and let $s[i]$ denote the i'th bit of s. Moreover, by $s.\mathit{substring}(i, j)$ we denote the substring of s from i to j. If $j < i$, $s.\mathit{substring}(i, j)$ is the empty string. Then the insertion operation on a PATRICIA tree is accomplished by Algorithm 6.2.

6.3 Rating of Multiple Matchings

The algorithm presented in Section 6.2 usually generates a large number of multiple matchings. In order to allow the user to step through these multiple matchings and their resulting alignments, we need to bring the alignments into a feasible order. This is not an easy task, not because the number of alignments might be too large, but because the alignments are not easily comparable. The reason for this lies in the fact that the multiple matchings vary not only in *size*, but also in *dimension*, and, e.g., the *normalized sum-of-pairs score* (cf. Definition 6.2.6). Some of these scoring functions are competing, i.e. improving one of them will most certainly worsen another one. For example, if a multiple matching has a larger dimension, it will, in general, have a smaller normalized sum-of-pairs score. Thus, in order to consider all scoring functions equally well, we need a means that is not dependent on a certain choice of weights for the scoring functions. We use the concept of Pareto dominance, which we introduce in the following subsection.

6.3.1 Pareto Sets

What is generally done in the case of multiple scoring functions is to compose an *overall scoring function* from all scoring functions, where a certain weight is assigned to each single scoring function. The quality of such an overall scoring function, however, depends largely on the choice of the weights.

Algorithm 6.2 Insertion operation for a PATRICIA tree

Input: PATRICIA tree T, bit-string s to be inserted

Output: leaf v whose path from the root is equal to s

```
if $T.isEmpty()$ then
    $T.r.string \leftarrow s$
else
    $v \leftarrow T.r$
    $i \leftarrow 1$
    while $i \leq s.length$ do
        $t \leftarrow v.string$
        for $j \leftarrow 1, t.length$ do
            if $s[i] \neq t[j]$ then
                $v' \leftarrow new\ Node$
                $v'' \leftarrow new\ Node$
                $v.string \leftarrow t.substring(1, j-1)$
                $v'.string \leftarrow s.substring(i+1, s.length)$
                $v''.string \leftarrow t.substring(j+1, t.length)$
                if $s[i] = 1$ then
                    $v.right \leftarrow v'$
                    $v.left \leftarrow v''$
                else
                    $v.left \leftarrow v'$
                    $v.right \leftarrow v''$
                end if
                return $v'$
            end if
            $i \leftarrow i + 1$
        end for
        if $i > s.length$ then
            return $v$                ▷ Bit-string $s$ has been inserted previously
        else if $s[i] = 1$ then
            $v \leftarrow v.right$
        else
            $v \leftarrow v.left$
        end if
    end while
end if
```

For sorting alignments according to multiple scoring functions, Handschuh [74] used the concept of *Pareto optimization* developed by Vilfredo Pareto [61] to solve multi-dimensional optimization problems. According to the optimization due to Pareto, an optimal state is reached if no parameter can be improved without worsening another parameter [74]. Instead of combining all scoring functions, i.e. parameters, into a single overall scoring function, we consider the values of all scoring functions of each multiple matching as a vector and define a partial relation on these vectors. Using this relation we generate so-called *Pareto sets* [166], where each set contains multiple matchings that are considered to be equally good. There exists a total order on these sets, but not on the elements within the sets.

Definition 6.3.1 (Vector Dominance)**.** Let $u, v \in \mathbb{R}^k$. Then vector $u = (u_1, \ldots, u_k)$ is said to dominate vector $v = (v_1, \ldots, v_k)$, denoted by $u \succ v$, if and only if u is partially larger than v, i.e.,

$$\begin{aligned} &\forall i \in \{1, \ldots, k\} : u_i \geq v_i \ \ \wedge \\ &\exists j \in \{1, \ldots, k\} : u_j > v_j \ . \end{aligned}$$

By $u \not\succ v$ we denote the fact that u does not dominate v. Note, that $u \not\succ v$ does not necessarily mean $v \succ u$.

With the relation of vector dominance we can now define the notions *Pareto set* and *Pareto set dominance.*

Definition 6.3.2 (Pareto Set)**.** Let $\mathcal{P} \subset \mathbb{R}^k$ be a set of vectors of dimension k. Then $\mathcal{P}$ is called *Pareto set* if and only if $\mathcal{P}$ satisfies

$$\forall u, v \in \mathcal{P} : u \not\succ v \ .$$

Definition 6.3.3 (Pareto Set Dominance)**.** Let $\mathcal{P}$ and $\mathcal{Q}$ be two Pareto sets. Then $\mathcal{P}$ is said to dominate $\mathcal{Q}$, denoted by $\mathcal{P} \succ \mathcal{Q}$, if and only if

$$\begin{aligned} &\forall q \in \mathcal{Q} \ \exists p \in \mathcal{P} : p \succ q \ \wedge \\ &\forall \tilde{p} \in \mathcal{P} \ \forall \tilde{q} \in \mathcal{Q} : \tilde{q} \not\succ \tilde{p} \ . \end{aligned}$$

Given a set of multiple matchings, instead of computing a total order on the elements in this set, we are interested in decomposing the set of all multiple matchings into Pareto sets. This leads us to the definition of *Pareto set decomposition.*

Definition 6.3.4 (Pareto Set Decomposition)**.** Let $\mathcal{P} \subset \mathbb{R}^k$ be a set of vectors of dimension k and let $\{\mathcal{P}_i\}_{i=1}^n$ be a decomposition of $\mathcal{P}$. Then $\{\mathcal{P}_i\}_{i=1}^n$ is called a *Pareto set decomposition* of $\mathcal{P}$, if $\mathcal{P}_i, i = 1, \ldots, n$, are Pareto sets which satisfy

$$(\mathcal{P}_i \succ \mathcal{P}_j) \vee (\mathcal{P}_j \succ \mathcal{P}_i) \ , \ \forall i, j \in \{1, \ldots, n\}, i \neq j \ .$$

It follows directly from Definition 6.3.4 that there exists a total order on the Pareto sets given by the Pareto set decomposition. The Pareto set dominating all other Pareto sets of this decomposition is called *Pareto optimal set.* More formally it can be defined as follows.

Definition 6.3.5 (Pareto Optimal Set)**.** Let $\mathcal{P} \subset \mathbb{R}^k$ and let $\{\mathcal{P}_i\}_{i=1}^n$ be a decomposition of $\mathcal{P}$. Then we call $\mathcal{P}_k$, $k \in \{1, \ldots, n\}$, the *Pareto optimal set* of $\mathcal{P}$, if

$$\mathcal{P}_k \succ \mathcal{P}_i \ , \ \forall i \in \{1, \ldots, n\}, i \neq k \ .$$

6.3.2 Sorting

To use the concept of Pareto dominance for sorting the multiple matchings, for each multiple matching we transform the values of all, say n, scoring functions into a vector of dimension n, where each dimension represents the same scoring function for all multiple matchings. We then compute the *Pareto set decomposition* (cf. Definition 6.3.4) of the set of vectors representing all multiple matchings. There exists a total order on this decomposition which we use to group the multiple matchings. As mentioned previously, the elements of each Pareto set are considered equally good. In general, in each Pareto set there exist multiple matchings of different (potentially all possible) dimensions. Also, e.g. multiple matchings with smaller size but larger score will exist in each Pareto set.

Since the computation of multiple matchings, as proposed in Section 6.2, is based on intersecting pairwise matchings, an additional score might be of interest. This score, the *average matching score*, averages the pairwise matchings from which a multiple matching was computed.

Definition 6.3.6 (Average Matching Score)**.** Let $f \in \mathcal{F}(P, Q^{l_1}, \ldots, Q^{l_k})$ be a multiple matching generated by intersecting k pairwise matchings $f_i \in \mathcal{F}(P, Q^{l_i})$. In contrast to the normalized sum-of-pairs score which averages the score of the induced pairwise matchings, the *average matching score* is defined on the scores of the original pairwise matchings.

$$\text{score}^*_{avg}(P, Q^{l_1}, \ldots, Q^{l_k}; f) := \frac{1}{k} \sum_{j=1}^{k} \text{score}^*(P, Q^{l_j}; f_j) \ . \tag{6.3}$$

This score looks very similar to the one defined in Definition 6.2.6. The crucial difference lies in the pairwise matchings whose score is averaged. In Definition 6.2.6 these were the pairwise matchings induced by the multiple matching. The size of all induced pairwise matchings is the same. In this score, we average over the pairwise matchings from which the multiple matching was computed. The size of these pairwise matchings may differ largely.

6.4 Experimental Results

From the pairwise alignments of the thermolysin and HIV-1 protease inhibitors presented in Section 5.6, we computed multiple alignments with the algorithmic approach described in the preceding sections. This was done both for the experimental conformers and the conformer ensembles. We also computed multiple alignments from pairwise atom alignments and compared them with the multiple alignments computed from pairwise surface alignments. All results are described in the subsequent subsections.

Let us recall, that the pairwise alignments presented in Section 5.6 represent diverse pairwise matchings, and that we only considered the 10 top-ranked pairwise alignments for each query molecule. The same 10 pairwise alignments of each query molecule were now used to compute multiple alignments.

While for the pairwise alignment several parameters had to be specified, for the computation of multiple alignments, we only need one parameter, namely the *relative minimum size* (cf. Equation 6.2). In all experiments we used a *relative minimum size* between 0.3 and 0.4, depending on the number of multiple matchings that were generated. We generally started with a value of 0.4. If the number of multiple matchings generated with this value was too small, we decreased the value in steps of 0.05.

Ranking of the multiple matchings was done using the *normalized sum of pairs score* (cf. Definition 6.2.6), the *average pairwise matching score* (cf. Definition 6.3.6), and the *size* and *dimension* of the multiple matchings (cf. Definition 6.2.1).

After sorting the multiple matchings, which were computed w.r.t. the user-specified relative minimum size, a single multiple matching was selected from the Pareto optimal set (cf. Definition 6.3.5). As criterion for the selection of this multiple matching, we used the minimal rms distance criterion. That is to say, we selected the multiple matching with the minimal rms distance to the experimental conformers over all molecules.

6.4.1 Thermolysin Inhibitors

Experimental Conformers

The computation of multiple alignments, as presented in this paragraph, was done on the basis of the pairwise alignments of experiments 2a and 3 (cf. Section 5.6.2), respectively. The results are summarized in Tables B.1 and B.2.

In all but five cases, the selected multiple surface alignments (cf. Table B.1) positioned the experimental conformers to the experimental position with an rms distance below 1.0 Å. In only one of these cases, the rms distance was above 1.5 Å, yet below 2.0 Å. In this case, the inhibitor 3TMN was involved, which misses the negatively charged group present in the other inhibitors. Multiple alignments without 3TMN gave even better results. A comparison of two multiple alignments, one of which includes 3TMN, is given in Figures 6.2 and 6.3. In these multiple alignments, the inhibitor 5TLN, which is least similar to the rest of the inhibitors, was used as reference structure. A close-up of the negatively charged groups together with their corresponding matched surface points is shown in Figure 6.4.

The results of multiple atom alignment (cf. Table B.2) are much worse than those of multiple surface alignment. In 25 of 56 cases, the rms distance was above 1.0 Å, and in three of these cases even above 2.0 Å. Partly, these results can be explained by the already worse pairwise atom alignments. But in contrast to multiple surface alignment, with multiple atom alignment the rms distances increased compared to the rms distances from

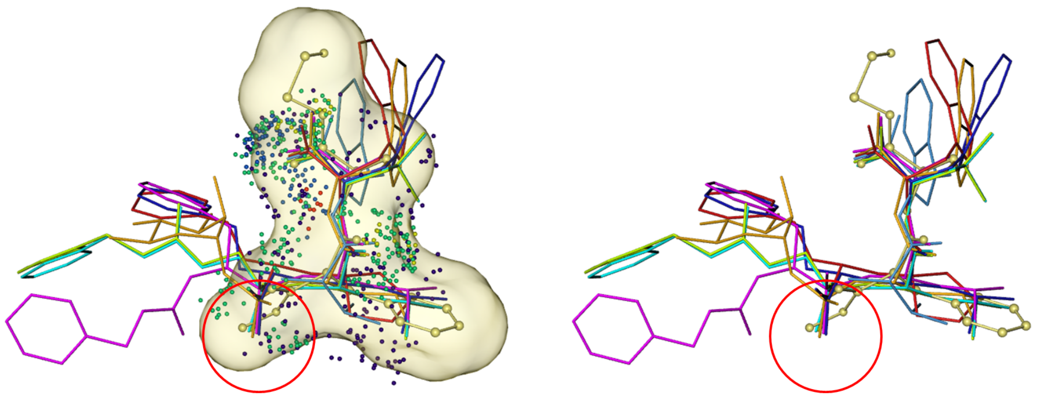

Figure 6.2: Multiple surface alignment of the thermolysin inhibitors with the experimental conformer of 5TLN (yellow, ball-and-stick) as reference structure (cf. Table B.1). The inhibitor 3TMN (light blue) is aligned worst with an rms distance of 1.76 Å. All other inhibitors are aligned with an rms distance below 1.0 Å. In the left image, the surface of 5TLN is shown together with the matched surface points of all molecules. Note, that in the region of the negatively charged group (red circle), there are hardly any matched points. This is due to the fact that inhibitor 3TMN not only misses a negatively charged group in this region, but has no atoms there at all.

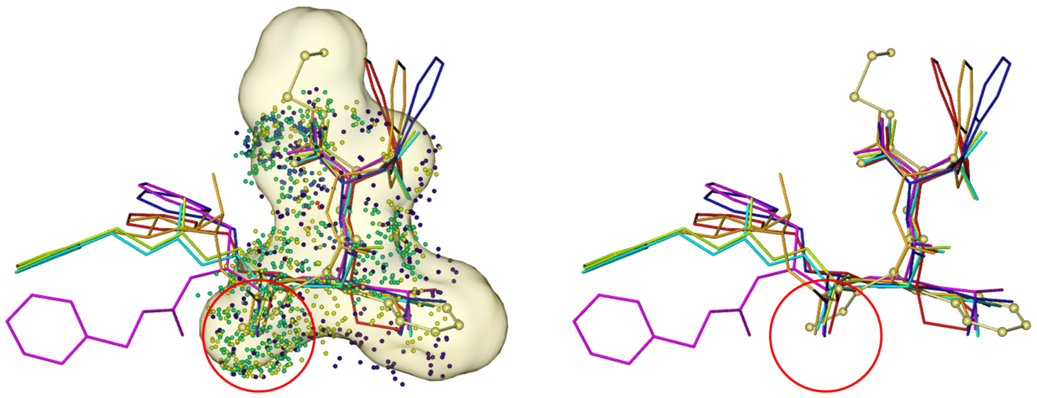

Figure 6.3: Multiple surface alignment of the thermolysin inhibitors except for 3TMN with the experimental conformer of 5TLN (yellow, ball-and-stick) as reference structure. In the left image the surface of 5TLN is shown together with the matched surface points of all molecules. Note the large number of matched surface points in the region of the negatively charged group (red circle). This large number of points is due to a superposition of several properties, namely shape, acceptor and negative MEP, denoted by different colors. In Figure 6.4, this region is shown as close-up view.

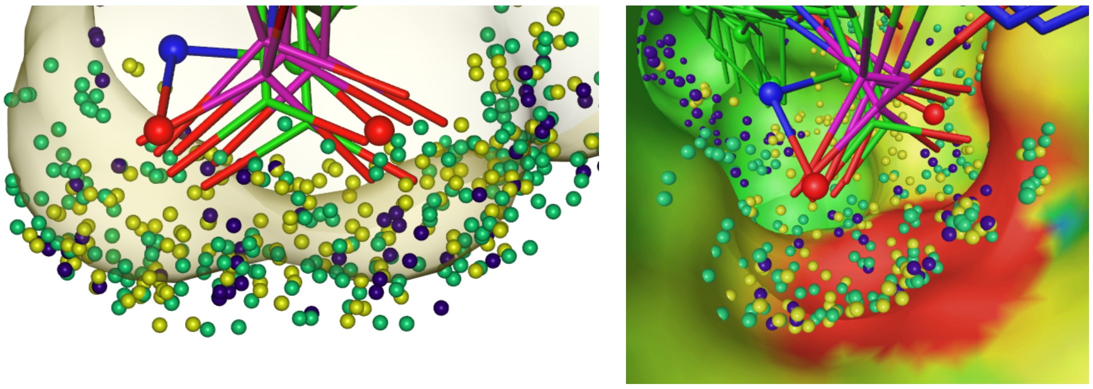

Figure 6.4: Close-up of negatively charged group binding to the zinc ion (cf. Figure 6.3). The small spheres denote the matched surface points of all inhibitors except 3TMN, which misses this group. Different colors represent different properties, i.e., the dark blue spheres represent shape, the light green spheres represent acceptor points, and the yellow spheres represent a negative charge. Note, that even though the atomic structure of the overlaid groups varies, the physico-chemical properties on the surface are very similar. *Left:* The semi-transparent surface depicts the solvent excluded surface of 5TLN. *Right:* Negatively charged groups of the inhibitors in the active site of thermolysin, interacting with the positively charged region (red) around the zinc ion. The SES of thermolysin has been colored according to the electrostatic potential.

pairwise alignment. The most probable reason for this phenomenon lies in the scaffold based alignment. As a result of multiple alignment, the scaffolds become even smaller and thereby the molecules might get aligned worse. For molecules whose scaffolds fit perfectly well, this has no negative effect, but if the scaffolds fit not as good, this effect becomes more dominant.

Ensemble of Conformers

The multiple alignment results presented in this paragraph are based on the pairwise alignments of experiments 4 and 5 (cf. Section 5.6.2), respectively. The results are summarized in Tables B.3 and B.4.

Neither with atom alignment nor with surface alignment did we obtain a satisfying alignment for the inhibitor 3TMN, which is the smallest of the inhibitors and, as previously mentioned, misses the negatively charged group binding to the zinc ion of thermolysin. Consequently, 3TMN is not selective enough to identify active conformers and their "correct" positions via a multiple alignment.

All inhibitors except for 3TMN performed well as reference structures for surface alignment. A look at Table B.3, however, reveals that all inhibitors had some minor problems with 5TLN. In four multiple alignments, the imino-hydroxyl moiety of 5TLN is disoriented (cf. Figure 6.6). This problem can already be observed in the pairwise alignments (cf. Table A.12) and is due to the fact that the surface patch of this wrongly oriented imino-hydroxyl moiety matches very well with the surface of several reference structures (cf. Figure 6.5). Note that in the experimental positions, the imino-hydroxyl moiety of 5TLN

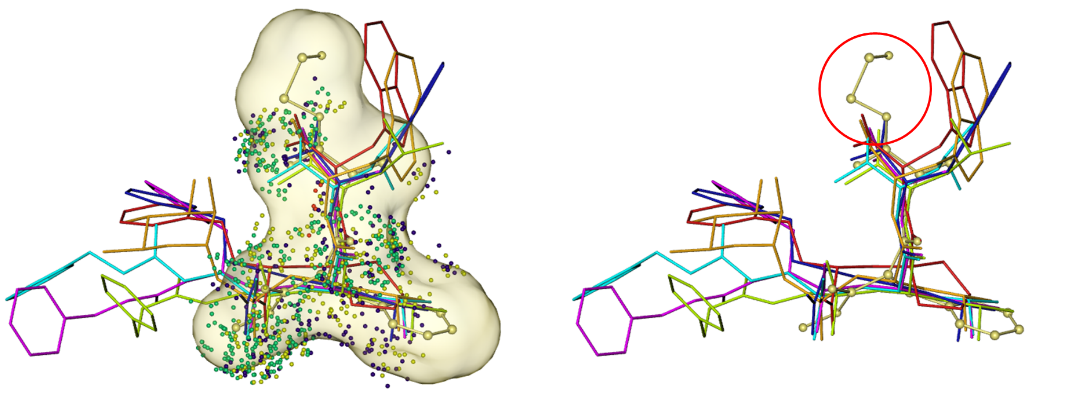

Figure 6.5: Multiple surface alignment of the thermolysin inhibitors except for 3TMN with the experimental conformer of 5TLN (yellow, ball-and-stick) as reference structure and the ensemble conformers of the other inhibitors as query structures (cf. Table B.3). In the left image, the surface of 5TLN is shown together with the matched surface points of all aligned molecules. A comparison with Figure 6.3 reveals a high similarity to the experimental positions w.r.t. the regions inside the solvent excluded surface of 5TLN. Differences are visible outside its surface. Note, that the distribution of matched surface points also shows a high similarity. The red circle denotes the imino-hydroxyl moiety, which is wrongly oriented in the alignment shown in Figure 6.6.

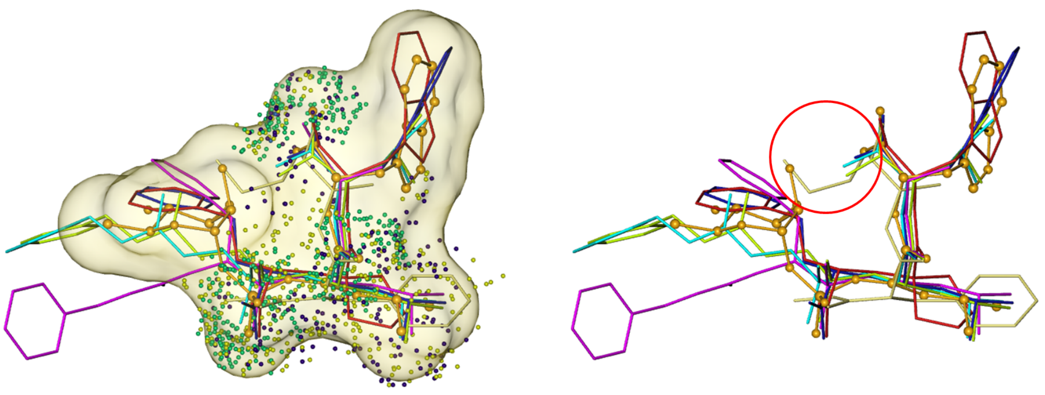

Figure 6.6: Multiple surface alignment of the thermolysin inhibitors except for 3TMN with the experimental conformer of 1TLP (light brown, ball-and-stick) as reference structure and the ensemble conformers of the other inhibitors as query structures (cf. Table B.3). In the left image, the surface of 5TLN is shown together with the matched surface points of all aligned molecules. Again, the distribution of matched surface points is very similar to that of the multiple alignments shown in Figures 6.3 and 6.5. Note, that the imino-hydroxyl moiety (red circle) of 5TLN (yellow), which used to point upwards (cf. Figure 6.5), is wrongly oriented in order to match the surface of 1TLP (left image).

is not overlaid with any other inhibitor. Thus, in this respect the experimental alignment is not favorable over this wrong orientation. In all cases, however, the charged hydroxamic acid group is oriented correctly. Example alignments with 5TLN and 1TLP as reference structures are shown in Figures 6.5 and 6.6, respectively.

The results of atom alignment are similar to those of surface alignment with the exception that with 5TLN as reference structure no satisfactory multiple alignment could be found with atom alignment, but only with surface alignment. Hence, surface alignment seems to be superior to atom alignment also for the task of identifying active conformers and their "correct" positions.

6.4.2 HIV-1 Protease Inhibitors

Experimental Conformers

The multiple alignments of the HIVPI experimental conformers are based on the pairwise alignments of experiments 7a and 8 (cf. Section 5.6.2), respectively. The results are summarized in Tables B.5 and B.6.

The multiple alignment results of the experimental conformers of the HIVPIs are very similar to those of the thermolysin inhibitors. Using surface alignment, for all inhibitors acting as reference structure almost perfect multiple alignments could be found. There are only a few instances with an rms distance slightly larger than 1.0 Å (cf. Table B.5). One example alignment with TPV as reference structure is shown in Figure 6.7. A close-up of TPV in the active site of the HIV-1 protease together with the matched surface points from the multiple alignment is depicted in Figure 6.8.

The results of atom alignment are a lot worse than those of surface alignment. All table entries show an rms distance above 1.0 Å (cf. Table B.6). This is in analogy to the thermolysin inhibitors, where the rms distance also deteriorated from pairwise alignment to multiple alignment. Multiple atom alignment determines small but well fitting scaffolds which take too little of the molecular structures into account to compute good multiple alignments.

Ensemble of Conformers

The computation of multiple alignments of the HIVPI ensemble conformers was done on the basis of the pairwise alignments of experiments 9 and 10 (cf. Section 5.6.2), respectively. The results are summarized in Tables B.7 and B.8.

The results of the surface alignment (cf. Table B.7) are again very satisfying with only two rms distances above 4.0 Å and only one complete miss alignment. Even if tipranavir (TPV) is used as reference structure, which is least similar to the other inhibitors, very good results were obtained. Although the rms distances are above 2.0 Å, except for lopinavir (LPV), a closer look at the aligned structures reveals that the large rms values are mainly due to disoriented terminating groups, which, in this alignment, point towards the flaps of the HIV-1 protease. These groups are also not matched in the alignment of the experimental conformers, since tipranavir does not have atoms in this region. The alignment can be seen in Figure 6.9. A second multiple alignment, here with

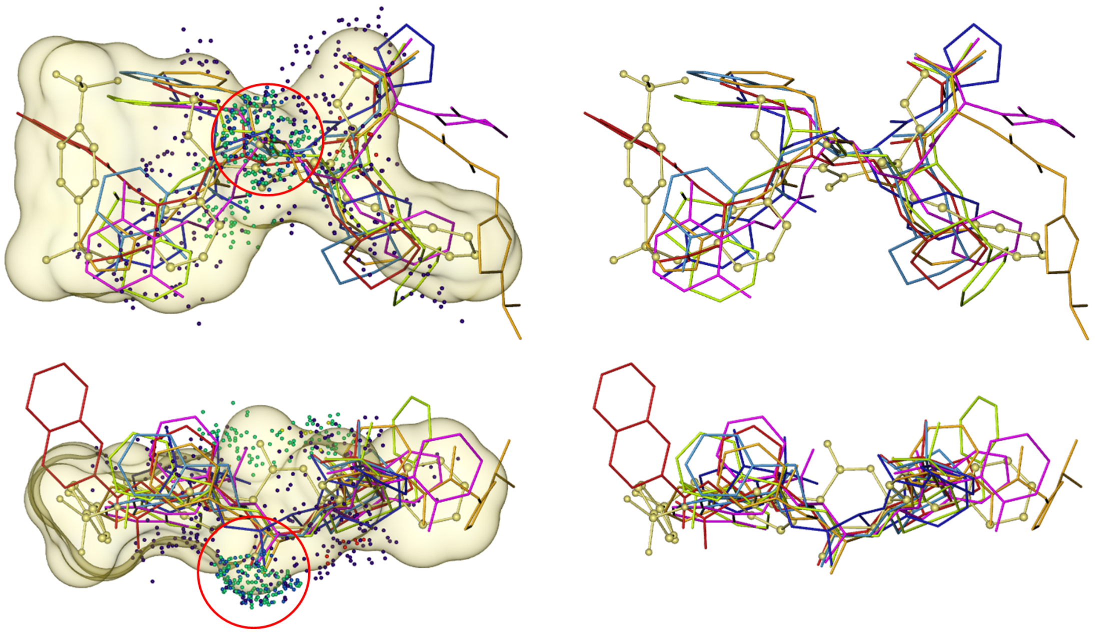

Figure 6.7: Multiple surface alignment of the HIVPI's experimental conformers with the experimental conformer of TPV (yellow, ball-and-stick) as reference structure (cf. Table B.5). *Left column:* Surface of TPV together with the matched surface points of all molecules. One dominant point cluster consisting of shape, donor and acceptor points is clearly visible (red circle). In this region, a hydroxyl group is present in all molecules, allowing to build a hydrogen bond, whereby the oxygen may function as acceptor and the hydrogen as donor. Hence, in this region, both donor and acceptor points are present. *Top row:* Side view. *Bottom row:* Top view.

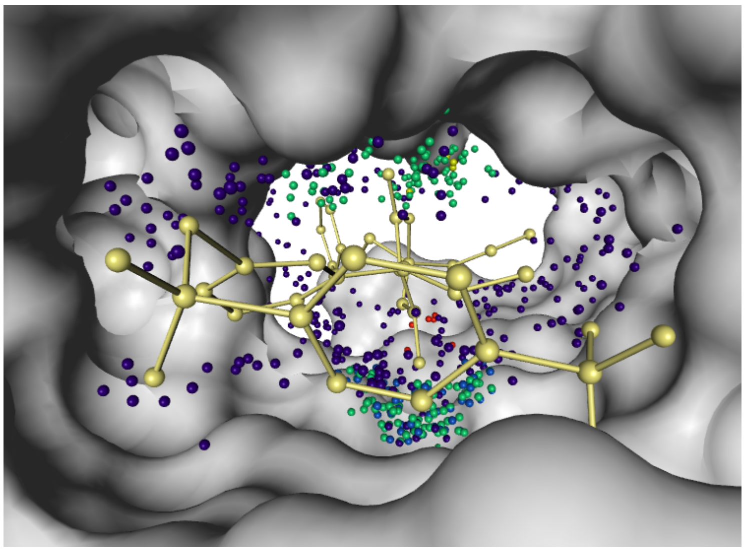

Figure 6.8: Experimental conformer of TPV in active site of HIV-1 protease. In this view, the flaps of the HIV-1 protease, which are not shown, are above the active site (cf. Figure 5.6). The small spheres depict the matched points from the multiple surface alignment shown in Figure 6.7. These points describe the shape of the active site very well. The donor/acceptor region (cf. Figure 6.7) can be seen in the bottom center of the image, depicted by the cluster of green and blue spheres.

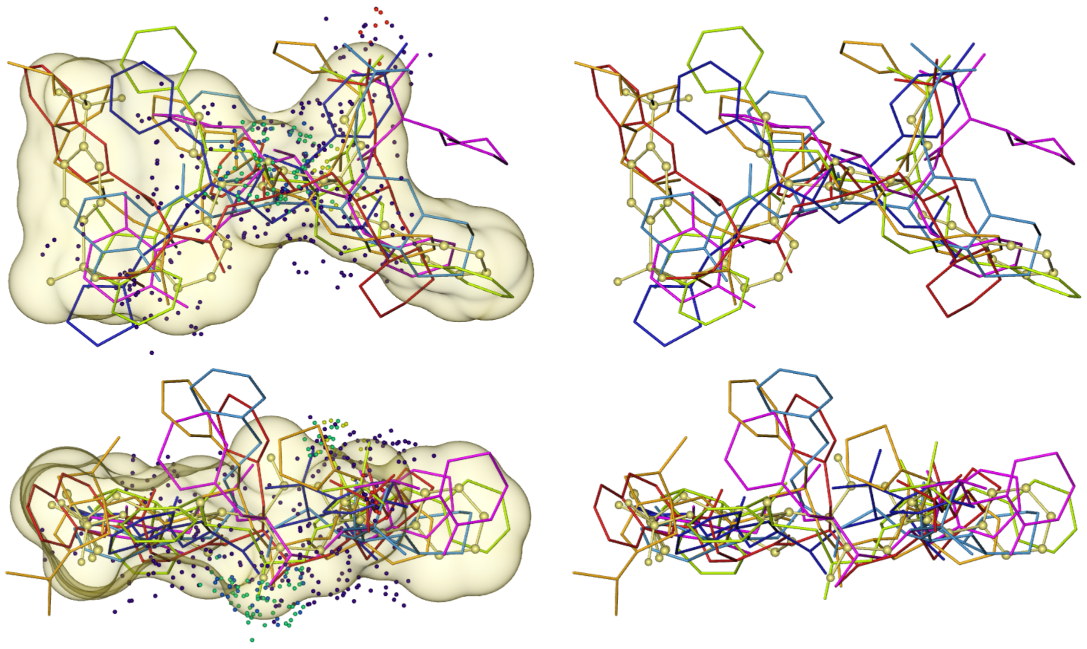

Figure 6.9: Multiple surface alignment of the HIVPIs with the experimental conformer of TPV (yellow, ball-and-stick) as reference structure and ensemble conformers of the other HIVPIs as query structures (cf. Table B.7). *Left column:* Surface of TPV together with the matched surface points of all molecules. The dominant point cluster representing the donor and acceptor region is still visible though less clear (bottom left image, bottom center). The rms distances of all molecules except for that of LPV are above 2 Å. These large rms distances are mainly due to the ring moieties sticking out of the aligned structures (bottom row, top center). *Top row:* Side view. *Bottom row:* Top view.

RTV as reference structure, is shown in Figure 6.10. This alignment correctly overlays the above mentioned terminating groups. The differences regarding these groups are already due to the differences in the computation of pairwise alignments.

6.5 Summary and Conclusion

This chapter described in detail, how multiple alignments are computed from pairwise alignments. To be more precise, multiple matchings, which define multiple alignments, are computed from pairwise matchings defining pairwise alignments.

We computed multiple surface alignments as well as multiple atom alignments from the pairwise alignments presented in the previous chapter (cf. Section 5.6.2). This was done by intersecting pairwise matchings successively. Since the number of possible intersections grows exponentially with the number of molecules, the size of the multiple matchings had to be restricted. In order to efficiently store the multiple matchings, such that identical matching bit-strings can be quickly found, PATRICIA trees were applied. Using the above mentioned size restriction and PATRICIA trees as data structure, multi-

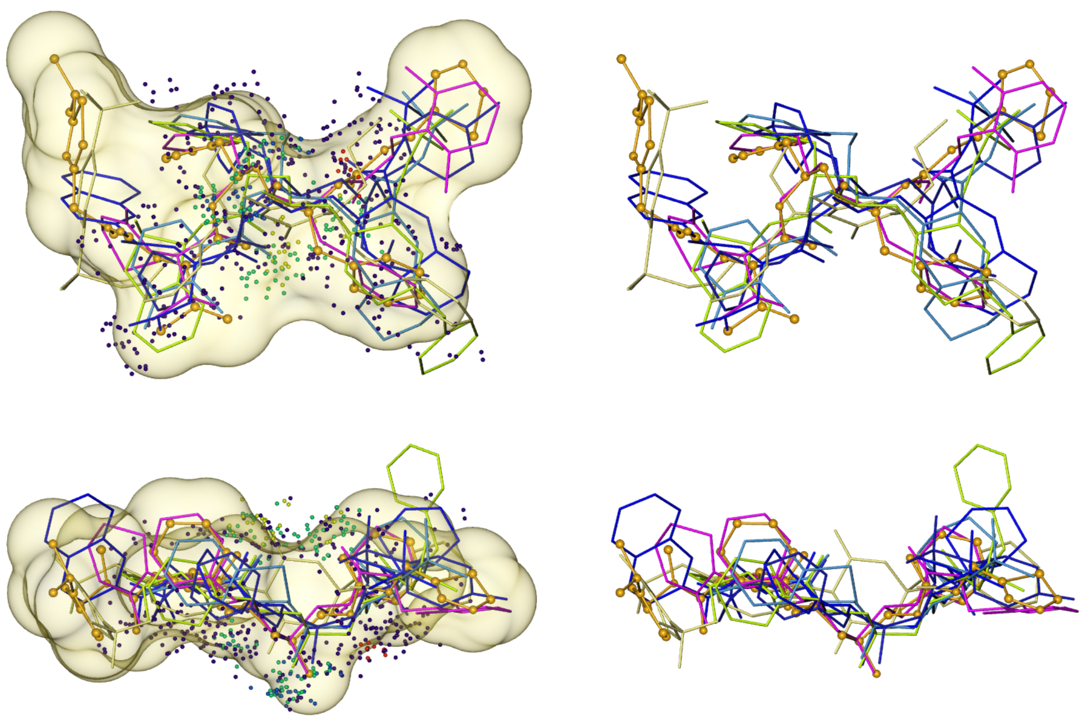

Figure 6.10: Multiple surface alignment of the HIVPIs with the experimental conformer of RTV (light brown, ball-and-stick) as reference structure and ensemble conformers of the other HIVPIs as query structures (cf. Table B.7). *Left column:* Surface of RTV together with the matched surface points of all molecules. Again, the point cluster representing the donor and acceptor region is visible (bottom left image, bottom center). The rms distances of all molecules are below 1.5 Å. *Top row:* Side view. *Bottom row:* Top view.

ple alignments could be computed in interactive times. The multiple matchings were then partially sorted according to multiple scoring functions using the concept of Pareto set decomposition. In order to investigate whether the multiple alignment algorithm generated feasible alignments, the Pareto optimal set was examined whether it contains a multiple alignment close to the experimentally given positions. Note, that all multiple alignments contained in the Pareto optimal set are considered to be equally good. A comparison of the computed multiple surface and atom alignments with the experimental positions yielded the following results.

Alignment of Experimental Conformers. For the thermolysin inhibitors, the multiple alignments obtained from atom alignment and surface alignment gave similar results with a slight advantage of surface alignment over atom alignment, in particular for inhibitor 5TLN as reference structure, which is most dissimilar to the other inhibitors. While multiple surface alignment generally improved the rms distances w.r.t. the pairwise alignments, with multiple atom alignment the opposite is the case. This could also, and even more clearly, be observed for the HIV-1 protease inhibitors. Here, for two inhibitors as reference structure, the multiple atom alignment algorithm could not successfully align tipranavir, which has a distinct scaffold to the other molecules.

The increase of rms distance observed for multiple atom alignments can be explained as follows. The atom alignments are based on atom matchings, i.e. one-to-one correspondences of atoms. Hence, in regions where the atoms cannot get overlaid perfectly, i.e. regions where the molecules' scaffolds differ, the atom alignment algorithm chooses the atom which is closer. However, even though a second atom might only be slightly further away, this atom will be completely ignored. In surface alignment using points distributed on the molecular surface, the surface contribution of each atom is represented by several points. Hence, surface points belonging to one atom of the first molecule can correspond to surface points belonging to distinct atoms of the second molecule. Consequently, although a surface matching represents a one-to-one correspondence of surface points, it also represents a "fuzzy" correspondence between atoms or atomic surface patches. This affects pairwise alignments too, but is amplified in the case of multiple alignments, where the number of corresponding points or atoms is reduced due to the intersection of the matchings. What remains in the case of atom alignment is generally a rather small common scaffold. Where such a common scaffold does not exist among all molecules, problems arise.

Identification of Active Conformers. In all but one case, the correct position of inhibitor 3TMN could not be identified using multiple surface alignment. This is due to the small size of 3TMN, which results in a rather small surface overlap with the other molecular surfaces. Here, atom alignment, indeed, produced better results. Except for two cases, with atom alignment an ensemble conformer of 3TMN could be positioned close to the experimental position. Due to its small size and the large flexibility of the other molecules, however, with 3TMN used as reference structure, the identification of satisfying multiple alignments failed for both surface alignment and atom alignment.

For tipranavir, atom alignment already failed to compute satisfying pairwise alignments in the ensemble test, i.e. the identification of an active conformer. Hence, we did not consider tipranavir for multiple atom alignment, but only for multiple surface alignment. But even if we ignore tipranavir, multiple surface alignment produced better results. First tests showed, that the results for multiple surface alignment could be improved if more than the 10 top-ranked pairwise alignments were used. This should also be the case for multiple atom alignment. However, we have not yet performed exhaustive tests to investigate this further.

Final Conclusion. For the considered experimental conformers of thermolysin inhibitors and HIV-1 protease inhibitors, the proposed multiple surface alignment algorithm produced excellent results, which are superior to those of multiple atom alignment.

With the exception of the thermolysin inhibitor 3TMN, multiple surface alignment of ensemble conformers also produced better results than multiple atom alignment. Partially, this is due to the fact, that pairwise surface alignment already produced better results than pairwise atom alignment. However, it is also due to the mentioned inherent problems of atom alignment with molecules showing dissimilar scaffolds.