Mining of Interaction Geometries in Collections of Protein Structures

Dissertation

with the aim of achieving the degree

Dr. rer. nat.

at the Faculty of Mathematics, Computer Science and Natural Sciences

submitted to the
Department of Informatics
of Universität Hamburg

submitted by

Therese Inhester

born in Göttingen

Hamburg, July 2017

Gutachter:
1. Prof. Matthias Rarey
2. Prof. Johannes Kirchmair

Tag der Disputation:
10.11.2017

Danksagung

An dieser Stelle möchte ich mich bei vielen Menschen bedanken, ohne deren Hilfe diese Arbeit nie entstanden wäre.
Allen voran möchte ich mich bei Prof. Matthias Rarey dafür bedanken, dass ich dieses interessante Thema in seiner Arbeitsgruppe bearbeiten durfte. Seine Tür stand stets offen für einen kurzen Ratschlag oder eine längere Diskussion. Ich danke ihm außerdem, dass er mir die Möglichkeit gegeben hat meine Arbeit auf internationalen Konferenzen vorzustellen. Darüber hinaus haben auch die weniger fachlichen Gespräche sowie die angenehme Arbeitsatmosphäre in seiner Gruppe meine Doktorandenzeit unvergesslichen gemacht. Vielen Dank dafür!
Ein großer Dank gilt außerdem der ganzen Arbeitsgruppe AMD für die tolle Zusammenarbeit und angenehme Arbeitsatmosphäre. Hier möchte ich mich vorallem bei Eva Nittinger für die tolle Zusammenarbeit rund um *NAOMI*nova bedanken. Kai Sommer danke ich für die Hilfe bei jeglichen Fragen zu QML und zur Koordinatengenerierung und unsere gemeinsamen Läufe um die Alster. Thomas Otto, Florian Flachsenberg und Stefan Bietz danke ich für die stets fundierte Beantwortung vieler, vieler Fragen. Rainer Fährrolfes danke ich für die Betreuung des Software Servers und das Bereitstellen der Reviewer-Software. Ich danke außerdem Matthias Hilbig, der mir gerade in den ersten Jahren meiner Doktorarbeit eine große Hilfe bei jeglichen Fragen und ein hervorragender Bürokollege war. Ein besonderer Dank gilt auch Melanie Geringhoff, die bei fast allen Problemen eine Lösung weiß.
Für die nette Arbeitsatmosphäre am ZBH möchte ich mich bei allen Mitarbeitern des Instituts bedanken, welche ich aus Platzgründen leider nicht alle namentlich erwähnen kann. Besonderer Dank gilt hier Prof. Johannes Kirchmair, der außerdem das Zweitgutachten meiner Arbeit übernimmt.
Des weiteren bedanke ich mich bei meiner Familie, allen voran bei meinem Vater, der mir das Studium der Bioinformatik überhaupt ermöglicht hat und mir stets mit Rat und Tat zur Seite stand.
Ein unbeschreiblich großer Dank gilt Alexander Thobe, der mich in jeder Lebenssituation stets angetrieben und unterstützt hat. Vorallem in der letzten Zeit der Doktorarbeit hat seine tatkräftige Unterstützt einen großen Beitrag zur Fertigstellung dieser Arbeit geleistet.

Abstract

The thesis at hand presents the development of new methods for the mining of interaction geometries in collections of protein structures. The binding between a protein and a small molecule or between two proteins is a fundamental event in all processes in living cells. Its complete understanding and manipulation is key in the field of structure-based drug design. The basis of molecular recognition are non-covalent interactions between atoms. Tools which can be used to mine the ever-growing data for specific spatial preferences of these interactions can help understanding the nature of molecular recognition. However, existing tools suffer from low variability and low precision of the used data and of the possible queries.

In this thesis, two methods have been developed which tackle the problem from two different perspectives. The first method, Pelikan, allows a user to search for specific interaction patterns in the interface between proteins and ligands. Using this methodology, bioisosters and chemoisosters can be found. This is particular useful if specific substructures in a ligand are to be replaced or a potential side-effect of a ligand is to be determined. The second method, *NAOMI*nova, calculates and presents distributions of interacting atoms in the vicinity of molecular substructures in collections of protein structures. From these distributions, preferred interaction directions can be deduced which is of major importance in the process of ligand optimization and affinity prediction during a drug design project.

For both methods, a serverless database system is used to efficiently store the relevant data. Fast and flexible retrieval systems have been developed for these database systems which go beyond existing methods. The retrieval system of Pelikan supports flexible as well as precise 3D queries on an atomic level. The *NAOMI*nova method handles user-defined molecular substructures and supports queries using different attributes of substructures and interacting atoms. In addition, the data used for the search processes is highly flexible and can be easily adapted. The correctness and the performance of the retrieval systems are demonstrated in this work and their applicability is shown in different examples. In addition, graphical user interfaces have been developed as part of this work which allow immediate and intuitive usage of the methods by life-science researchers.

Kurzfassung

Die vorliegende Arbeit beschreibt die Entwicklung von neuen Methoden zur Suche nach Interaktionsgeometrien in Proteinstruktursammlungen. Die Bindung zwischen einem Protein und einem kleinen Molekül oder zwischen zwei Proteinen ist ein fundamentaler Prozess in lebenden Zellen. Ein vollständiges Verständnis darüber warum zwei Moleküle binden und dessen Manipulation ist ein wichtiger Aspekt im Bereich des strukturbasierten Wirkstoffentwurfs. Die Basis von molekularer Bindung sind nicht kovalente Interaktionen zwischen Atomen. Methoden, die in der Lage sind die wachsende Menge an Daten nach geometrischen Präferenzen dieser Interaktion zu durchsuchen können helfen die treibenden Kräfte hinter diesen Bindungen besser zu verstehen. Existierende Methoden in diesem Feld bieten jedoch nicht genung Flexibilität und Präzision bezogen auf die möglichen Suchanfragen und in die analysierten Daten.

In dieser Arbeit wurden zwei Methoden entwickelt die das beschriebene Problem von verschiedenen Standpunkten betrachten. Die erste Methode (Pelikan) ermöglicht die Suche nach spezifischen Interaktionsmustern an der Schnittstelle zwischen Protein und Ligand. Damit können sogenannte Bioisostere und Chemoisistere entdeckt werden, die insbesondere dann von Nutzen sind, wenn spezifische Substrukturen in Liganden ausgetauscht oder mögliche Nebenenffekte eines Liganden vorhergesagt werden sollen. Die zweite Methode (*NAOMI*nova) berechnet geometrische Verteilungen von interagierenden Atomen im Umfeld von molekularen Substrukturen aus Proteinstruktursammlungen. Aus diesen Verteilungen können bevorzugte Interaktionsrichtungen abgeleitet werden, die beim Optimieren von Liganden und der Affinitätsvorhersage von Nutzen sind.

Beide Methoden benutzten eine server-lose Datenbank um die benötigten Daten effizient zu speichern. Hierfür wurden schnelle und flexible Suchverfahren entwickelt, die über existierende Methoden hinausgehen. Im Fall von Pelikan werden flexible und präzise 3D Anfragen auf atomarem Level unterstützt. Die Methode in *NAOMI*nova arbeitet mit benutzer-spezifizierten molekularen Substrukturen und unterstützt verschiedene Attribute von Substrukturen und interagierenden Atomen in den Anfragen. Zusätzlich können die für die Suche genuzten Daten angepasst werden. Die Korrektheit, das Leistungsspektrum und die Anwendbarkeit beider Methoden werden in dieser Arbeit demonstriert. Darüber hinaus wurden grafische Oberflächen entwickelt, welche eine intuituve Benutzung durch Forscher aus dem Bereich der Lebenswissenschaften ermöglicht.

Contents

List of Abbreviations

2D	two dimensional
3D	three dimensional
vdW	van der Waals
SQL	structural query language
RMSD	root means square deviation
EDIA	electron density for individual atoms
NMR	nuclear magnetic resonance
sec	secondary
PRP	potential result point
k-D trees	k-dimensional tree
id	identifier
GUI	graphical user interface
EC number	enzyme commission number
PDB	protein database
CSD	Cambridge structural database
CCDC	Cambridge crystallographic data center
DT	descriptor triangle
db-query	database query
Å	Ångström
min	minute

1 Introduction

Proteins are the workhorses of biochemistry. Almost no biochemical process could take place without their help. For example, during cell division, the foundation of life, proteins bind to and replicate the DNA. Proteins are also at the heart of our immune system: Antibodies are proteins, who precisely bind to a specific structures of germ particles and thereby lead to their destruction. Moreover, many drugs are based on small molecules binding to a specific protein to modify their function. For example, the blood pressure is increased by adrenaline binding to adrenergic β-receptors – proteins which are located in the cell membrane of different organs, e.g., the heart and the kidney. High blood pressure can be cured with small molecules binding to these receptors in the same way that adrenaline would, and thereby inhibit the unwanted binding of adrenaline. In all three of these examples, it is of major importance, that the proteins bind selectively only to very specific ligands. This specificity of the binding is often called 'molecular recognition', because the ligand is 'recognized' by the protein. These examples show that understanding and designing the molecular recognition between protein and ligand is key for controlling and understanding biochemical processes.

Already in 1894, Emil Fischer tried to understand why a protein and a small molecule specifically bind to each other. He postulated that the specific binding of a small molecule to a protein can be compared to a key which exactly fits to its lock [1]. Nowadays, it is clear that molecules can have a flexible structure and thus the key as well as the lock can adapt their structure to each other. However, the main idea of Fischer has been proven right: The shapes of the two binding molecules have to 'match' each other. Moreover, upon the binding of two molecules, non-covalent interactions between atom pairs of both molecules are formed. For this formation, the atoms have to be in close proximity to each other. Thus, one could say that in order to be able to build interactions not only the shape of the two molecules has to 'match' but also the position of their interacting atoms. Details about the nature of these interactions and their geometrical properties will be outlines in Section 1.1.

Back in the beginning of the 20th century, drugs were discovered without knowledge about the molecular processes leading to the diseases which the drugs were meant to cure. Most of the developed drugs were natural products and were used against infectious diseases [2]. Advances in molecular biology and genetics during the 20th century gave functional insights

into the molecular processes and led to the discovery of malfunctioning proteins as the cause of many diseases. Knowing this connection, researcher were able to specifically develop drugs which aim at modifying the function of these proteins.

In the 1980s, computational tools began to emerge in the field of drug design which is now known as the advent of the 'computer-aided drug discovery' field [3]. Herein, the 'structure-based' methods aim at analyzing the 3D structures of proteins and small molecules and predicting their specific mutual binding. Many success stories demonstrate the benefit of the structure-based drug design approach [4,5]. Among the first was the study of Erickson *et al.* in which an inhibitor for the HIV protease was designed based on the protein structure [6]. Today, the typical process of 'rational drug design' includes the identification of a target protein and the design of a small molecule which specifically binds to it guided by a plethora of different computer-based tools. This process will be explained in more detail in Section 1.2.

Despite these examples and the substantial progress which has been made since the beginning of using computational tools in drug-design projects, molecular recognition is still not fully understood [7] and there are still no methods available which reliably and correctly predict the binding between two molecules. Technological advancements in X-ray crystallography have led to the discovery of many molecular structures within the last twenty years [8, 9]. These two facts even increased the demand for computational methods which are able to handle the growing amount of structural data and filter out the relevant information in order to guide drug design projects.

The development of computational methods which help to analyze the geometrical preferences and characteristics of molecular recognition is the focus of this thesis. In the following sections, the theoretical background relevant for this work is given. At the end of this chapter, the motivation for the development of these methods is outlined.

1.1. Molecular Recognition in Proteins

Proteins are macromolecules which consist of several, covalently bound amino acids. In procaryotic cells, 20 different amino acids exist. Amino acids can be structurally divided into the backbone and the sidechain. The backbone is chemically equal in all amino acids and has an N- and a C-terminus. The covalent bond, also known as 'peptide bond' is formed between those termini of the amino acids' backbone. The side chains of the amino acids vary and are relevant for their chemical properties.

Proteins can be seen as the tools of live. Their function is mediated through the specific, often non-covalent binding to small molecules, other proteins, DNA, or RNA. Thereby, various cellular processes are triggered or inhibited, e.g., contraction of muscles, gene expression, or signal transduction. The resulting dimer of protein and ligand is also called 'protein-ligand

complex'. The binding happens at specific, often hollow-shaped areas in the protein. These areas are called 'binding sites'. In enzymes, the bound molecule is usually changed in a chemical reaction which is why the binding site is often called 'active site' in this context. The third term 'pocket' describes hollow-shaped areas in the protein where small molecules can bind.

In a healthy organism, the cellular balance between active and inactive states, growing and resting, surviving and dying are maintained by these molecular interactions. Malfunctioning proteins, however, may lead to imbalances in these equilibriums and thereby cause diseases. In these cases, drugs are often used to inhibit the binding of a protein to its counterpart in order to stop or reduce the activity of these malfunctioning proteins. This can be achieved by designing small molecules which non-covalently bind to the specific protein and simply occupy its binding site.

The specific, non-covalent binding of two molecules is driven by a negative free binding energy which consists of an entropic as well as an enthalpic term. Looking at the entropy, two main aspects have to be considered. On the one hand, the binding of two molecules reduces the disorder of the system. This is mainly a result of the reduced translational, rotational, and conformational freedom of both binding partners. On the other hand, the 'hydrophobic effect' leads to an increase in the total disorder of the system through binding. The hydrophobic effect describes the aggregation of hydrophobic substances in aqueous solutions. This effect applies if the two binding molecules have hydrophobic spots on their surfaces which are brought into close contact upon binding. In this case, the water molecules which used to surround these hydrophobic spots are released into the solvent and the entropy is increased.

The enthalpic term describes the binding energy of electrostatic interactions between oppositely charged atoms or groups of atoms. In the course of this thesis these interactions will be called 'atomic interactions' in order to avoid confusion with the general interaction between two molecules. The exact energetic contribution of an individual atomic interaction cannot be measured and is believed to be highly dependent on the chemical context [10, 11]. As for the entropy, two different processes have to be mainly considered. On the one hand, atomic interactions between water molecules and the binding site and the ligand are broken up. On the other hand, new atomic interactions are formed between the ligand and the binding site.

In total, only in those cases in which the binding is energetically more favorable than the unbound state, a binding occurs.
In the following, the interaction types most relevant in the field of drug design and their geometrical specifications are introduced. In general, the distance between two atoms which are involved in an atomic interaction is usually closer than the sum of the atom's van der Waals (vdW) radii but not as close as a covalent bond would be.

The 'ionic bond' is formed via the attractive force between two oppositely charged ions. In the context of proteins, this type of atomic interaction is relevant for the coordination of

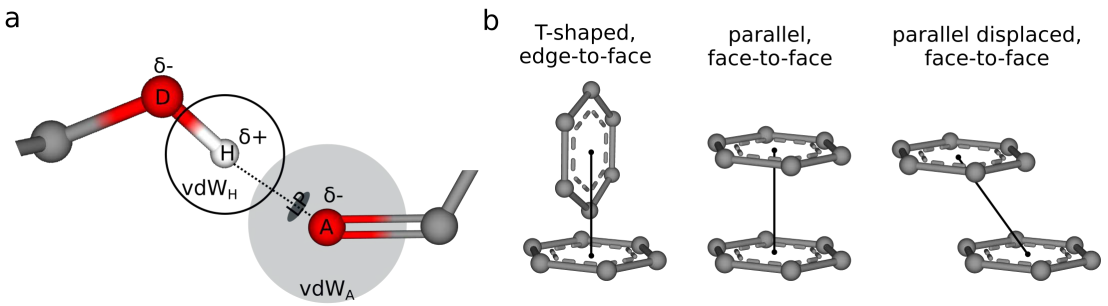


Figure 1.1.: Schematic depiction specific of atomic interactions. a) Schematic depiction of a hydrogen bond. D = hydrogen bond donor, H = hydrogen, A = hydrogen bond acceptor, vdW$_x$ = van der Waals radius of atom x, LP = electron lone pair. The black dotted line represents the hydrogen bond. b) Schematic depiction of the three different geometric arrangements of π-π interactions.

metal atoms. Additionally, charged atoms in side chains of amino acids, e.g., glutamic acid and arginine, can build ionic bonds. Since the positively charged atom is often bound to a hydrogen, these ionic bonds are also called 'ionic hydrogen bond' or 'salt bridge'.

Hydrogen bonds have been described as "by far the most important specific interaction in biological recognition processes" [7]. A schematic depiction of a hydrogen bond is shown in Figure 1.1a. Here, a positively charged hydrogen (depicted as H in Figure 1.1a) is covalently bound to a partially negatively charged heavy atom, the so called hydrogen bond donor (in the following denoted as donor, depicted as D in Figure 1.1a). The hydrogen is moreover in close proximity to another partially negative charged heavy atom, the so called hydrogen bond acceptor (in the following denoted as acceptor, depicted as A in Figure 1.1a). The partial charge of an atom is a result of its electro-negativity. Thus, in a biological context, only nitrogen and oxygen atoms are thought to be donors and only nitrogen, oxygen, and sulfur are considered acceptors of a 'classical' hydrogen bond. Several studies on protein structures have revealed that hydrogen bonds have strong geometrical preferences (for a review see [7]). Herein, not only the distance between hydrogen and acceptor but also the global geometry seems to be important. Bissantz et al. [7] report the optimal angle of donor–hydrogen$\cdots$acceptor to be above 150° and the typical distance between donor and acceptor to be about 2.9 Å. Looking at the hydrogen bond donor, the optimal position of the hydrogen should be in direction of the lone pair. The exact geometrical parameters, however, slightly differ depending on the element types of donor and acceptor [12, 13]. Moreover, different studies have shown that the exact geometrical preferences seem to depend on the chemical context, e.g., the functional group or the ring system both the donor and the acceptor are connected to [14–16]. Also for intra-molecular hydrogen bonds in small molecules, specific geometric patterns were found [17].

In recent years, more and more attention has been paid to so called 'weak hydrogen bonds' in the context of drug design. Here, donor atoms with less electro-negativity, e.g., carbons,

are investigated in more detail [18]. Also for these bonds, geometrical preferences were found in structures of protein-ligand complexes. However, compared to classical hydrogen bonds, these preferences seem to be more variable [12] and less easily differentiable from vdW contacts due to similar distances.

Another important group of atomic interactions involve aromatic rings. An overview of the different interaction types involving aromatic rings has been provided by Meyer *et al.* [19]. Briefly, the delocalized π electrons are attracted by positively charged atoms. In case of a cation-π interaction, a positively charged atom is positioned in a short distance on top of the π electrons. The main structure and energetics of this interactions have been reviewed by Me et al. [20]. Additionally, π-π or π-stacking interactions are known. Herein, the rings are believed to have a quadrupole moment [21]. This means that positive and negative charges are separated in direction of the ring normal. Mainly, three geometrical arrangements have been observed in which these aromatic systems attract each other: (1) T-shaped (or edge-to-face), (2) parallel displaced (or face-to-face), and (3) parallel (or face-to-face). Schematically, these three arrangements are depicted in in Figure 1.1b. Meyer et al. [19] reviewed different publications which suggest that the preferred geometry seems to be dependent on the substituends of the ring [19].

In addition, 'halogen bonds' have been in the center of research in last years. Examples of two recent studies are [22] and [23]. In this type of atomic interaction, an attractive force between a partially positively charged electron hole (so-called 'sigma hole') in the shell of an halogen atom and a Lewis base is formed. The electrostatic principles and preferred geometries of this interactions have been reviewed by Wilcken et al. [24]. A recent survey of halogen interactions on structures of protein-ligand complexes revealed that varying geometries can be observed depending on the involved amino acid [23].

Apart from the described interactions, many more exist which were reviewed by Bissantz *et al.* [7] and Meyer *et al.* [19]. In this section, only the atomic interactions most relevant in the context of structure-based drug design have been described. In summary, specific geometrical preferences have been reported for all of them which strongly depend on the respective chemical context.

1.2. Structure-Based Drug Design

In the field of structure-based drug design, the design process of drugs is based on the 3D structure of the target protein and the exploitation of the associated atomic interactions. At the beginning of such a process stands the identification of the target protein. A classical target protein is associated with a specific disease and carries out its function through binding a small molecule at a clearly definable binding site. In recent years, more and more attention has also been paid to protein-protein interactions, i.e. proteins which bind to other proteins

[25]. In both cases, the main goal is to modulate the protein's function by developing a small molecule which competes for this binding.

After the target protein has been defined, the 3D structure of the protein has to be elucidated. Here, mainly two different experimental methods are used: X-ray crystallography and nuclear magnetic resonance (NMR).

After the structure of the target protein has been determined, the binding site needs to be identified. This is most often the exact spot where the natural ligand binds. However, due to the principle of allosteric regulation, also other binding sites can be used [26].

This step is followed by the actual drug-design process, which can be divided into two subsequent steps:

- Lead identification.
- Lead optimization.

1.2.1. Lead identification

Computer-based approaches for the identification of a lead compound, i.e., a molecule which strongly binds to the target protein, can be divided into three categories [3]:

- Modification of a known ligand.
- Screening of a virtual library.
- *De novo* generation of ligands.

Modification of a known ligand. For this approach, knowledge about the binding pose of a ligand is required. This is often gained through co-crystallizing the target protein with the ligand. Chemical modifications of the ligand can then be applied and convert it into a more potent inhibitor. The binding affinity of this modified ligand can be assessed experimentally or virtually by using docking methods. Often, the latter method is used first and only promising compounds are later tested in the more cost- and time-intensive experimental assays. There are different docking methods, which analyze the structure of both the binding site and the ligand. These try to predict the binding pose and sometimes also the affinity [27]. Recent advances in docking methods have been reviewed by Yuriev *et al.* [28]. An integral part of these methods is a scoring function. Amongst other parameters, it evaluates the atomic interactions build in the interface.

Screening of a virtual library. Different libraries exist that can be used for virtual screening. Among them is the ZINC [29] database which is a set of purchasable molecules. Other sources for virtual sets of compounds are the National Cancer Institut (NCI, https://www.nih.gov/)

and PubChem (https://pubchem.ncbi.nlm.nih.gov) [30].

Screening algorithms can handle these large compound sets and detect those molecules which potentially bind to the defined binding site with high affinity. As for the docking methods, scoring functions are used to predict the binding of a small molecule. An overview about virtual screening methods is given by Lionta *et al.* [31].

De-novo generation of ligands. In this approach, small molecules which should bind to the binding site of the target protein are newly generated. Two different approaches exist: 'fragment based' and 'atom based' [32]. In 'fragment-based' approaches, small molecular fragments are first placed into the binding site. Then, in several steps, other fragments or functional groups are connected to the initial fragment. Thus the fragment 'grows' to a 'full' ligand. In every step, the binding is evaluated and only optimal solutions are kept. In structure-based approaches this evaluation is performed using scoring functions which also take atomic interactions into account. Atom-based approaches in principle have a similar workflow as fragment-based approaches. However, the building blocks are not chemical fragments or functional groups but atoms. More information about *de-novo* generation of drugs is provided in a review by Hartenfeller and Schneider [32].

The compound generated by these procedures are often subject to a 'hit-to-lead-optimization' which is comparable to the approach followed in the 'modification of a known ligand'. In this process, the compounds are chemically modified in order to improve their binding affinity.

1.2.2. Lead optimization

In this step, the identified leads are optimized in order to improve the ADMET properties of the molecules without reducing its affinity: (A) absorption, (D) distribution, (M) metabolism, (E) elimination, (T) toxicity. For some of these properties, the unintended binding of the lead compound to non-target proteins is a very important examination since this can lead to unwanted effects such as side-effects. For example, the binding of a drug to the hERG potassium channel has been shown to lead to cardiac arrhythmia in the past [33]. Moreover, the binding to the protein P-glycoprotein is associated with drug resistance as P-glycoprotein is able to export small molecules form cells [34]. Within the process of optimizing ADMET properties, similar techniques are applied as in the 'hit-to-lead-optimization' process.

After the lead optimization, the molecules are tested *in-vitro*, *in-vivo*, and finally in clinical trials.

1.3. Data Foundation for Analyzing Atomic Interactions

The previous sections showed that structures of proteins and small molecules have an important impact on the drug design process. On the one hand, they are used to perform statistical surveys and provide insights into the process of molecular recognition. On the other hand, they are the fundamental prerequisite in structure-based drug design endeavors. Two main libraries of molecular structures build the foundation of these tasks: (1) the Cambridge structural database (CSD) [35] and (2) the Brookhaven protein data bank (PDB) [36]. The CSD contains structures of small molecules whereas the PDB is a database for macromolecular structures, mainly proteins. The structures from the CSD and the majority of the structures contained in the PDB have been measured experimentally by X-ray crystallography.

Today, the commercial repository CSD contains more than 800 000 structures. Almost 130 000 structures are nowadays part of the PDB. The number of structures has increased exponentially over the last years and also the quality of the structures is constantly improving. In addition, the diversity of proteins in the PDB is increasing due to different structural genomics projects [37]. Given that the PDB is an open source repository, it displays a good means to investigate atomic interactions between proteins and other molecules.

However, there are some drawbacks and hurdles when working with structures from the PDB. First of all, since most of the structures have been elucidated with X-ray crystallography, only the parts of the molecules which contain electrons are resolved. Hydrogen atoms can only be detected in high resolution experiments, because here the binding electrons can be measured which are confined to a small spatial region. Secondly, again due to the experimental measurements, the exact orientation of δN and δO in asparagine and glutamine side chains cannot be determined. The same holds true for the aromatic ring in the histidine side chain. Thirdly, the crystallized molecule is in a non-native state. In solution, molecules are dynamic and there might be larger or smaller movements of some atoms. The structure in a crystal can therefore only be seen as a snapshot of the structure. Moreover, structural artifacts might occur due to the crystallization process, called packing effects. These mainly occur on the surface of the crystallized molecules where they are artificially in close contact to other molecules. Fourthly, errors might occur while fitting the molecular structure into the electron density. These can result in atoms being placed outside the electron density or inside too large bulbs of electron density.

For the first two problems, tools exist which compare all possible hydrogen positions and amino acid orientations and return the most probable solution. In this work, the tool Protoss [38] is used for this task. The third problem is relevant for those parts of the structure which can be in contact to neighboring molecules inside the crystal. It is therefore important to inspect the analyzed region of the protein for those contacts before drawing structural conclusions. The tool Ligand Protein Contact (LPC) [39], published in 1999, even provides an automated analysis of crystal contact in protein structures. However, the web service is

not active any more.

The fourth problem can be handled by checking the electron density support. Nowadays, structure factors have to be added to every newly deposited structure in the PDB. Here, the EDIA (electron density for individual atoms) is used to calculate the experimental support for protein structures [40, 41].

Besides these structural problems, the data collection provided by the PDB also suffers from a structural bias. On the one hand, the PDB contains only structures from proteins which are crystallizable. Especially trans-membrane proteins are thought to be difficult to crystallize and thus these proteins are underrepresented in the PDB. On the other hand, protein structures in the PDB are uploaded by researchers. Therefore, proteins which are subject of many research projects are overrepresented in the PDB.

However, keeping these difficulties in mind when working with the PDB and while generating statistics, the structures from the PDB can be used to deepen the knowledge about atomic interactions. Excellent reviews about the limitations of X-ray crystallographic data and possible validation approaches in the context of structure-based drug design have been written by Davis *et al.* [42], Deller and Rupp [43], and Warren *et al.* [44].

1.4. Motivation

In structure-based drug design the generation or identification of small molecules which are able to bind to a specific target protein by satisfying all interacting atoms in the binding site is one major goal. Within this context, there are especially four applications for which the knowledge about geometric preferences of these interactions is required:

- Parametrization of scoring functions.
- Assessment of ligand selectivity.
- Fragment-based drug discovery.
- Hit-to-Lead optimization / Lead optimization.

Parametrization of scoring functions. In applications like virtual screening or docking, scoring functions are essential to predict and evaluate the binding of a small molecule to a protein. The formation of atomic interactions is one major component here. The better the geometry of an atomic interaction, the higher is its estimated energy contribution. In this context, correct and precise geometric parameters should lead to a large predictive power of the method.

Assessment of ligand selectivity. For this task, the principle of chemoisosterism can be used. Chemoisosterism describes the property by which different protein environments can bind to

the same chemical fragment [45]. The binding of this fragment might involve more than one atomic interaction. It can be used to investigate why some drugs bind to several different proteins (polypharmacology) and thus draw conclusions about the selectivity of a ligand.

Fragment-based drug discovery. In order to find the initial fragment for a fragment-based *de-novo* design of a drug, the principle of bioisosterism can be used. Bioisosterism describes the property by which different chemical fragments can bind to the same protein environment. More than one atomic interaction might be involved in this binding.

Hit-to-Lead optimization / Lead optimization. In both optimization steps, the properties of a molecule are improved in order to increase its affinity to a target protein or to reduce its toxicity. Herein, the knowledge about preferred interaction geometries and their dependence on the chemical context are used in different ways:

- *Exchange of functional groups*. The affinity of a lead can be optimized by identifying suboptimal atomic interactions. Afterwards, the involved functional group in the molecule can be replaced by another group such that the resulting geometry is within optimal ranges for the new atomic interaction. In a similar way, the affinity to an unwanted target can be reduced.

- *Saturation of all atomic interactions*. There are atoms which can be involved in more than one atomic interaction at the same time. If the affinity of a known ligand should be optimized, the knowledge about the number of interactions and their directions can guide the exchange process in order to saturate all interactions.

- *Exchange of chemical fragments*. Here, the principle of bioisosterism is used in order to exchange chemical fragments in a lead molecule. In this context, a chemical fragment is larger than a functional group and several atoms of the fragment might be involved in atomic interactions. On the one hand, this exchange could improve the affinity. On the other hand, there are known chemical fragments which are associated with undesired molecular properties, e.g., toxicity. In this case, exchanging a chemical fragment could improve the molecular properties without reducing its affinity.

From a broader perspective, the required knowledge which is needed for the above mentioned tasks can be divided into two different groups. One the one hand, knowledge about larger chemical substructures and their spatial surrounding is needed to infer chemoisosters and bioisosters. Herein, the focus lies on a spatial interaction pattern rather than on one atomic interaction. Specific chemo- and bioisosteric cases are more relevant then statistics on large data sets in this context. This problem will be referred to as '**the search for interaction patterns**'.

All other described application scenarios require information about preferred interaction directions for specific substructures. Herein, the focus lies on interactions built by one specific atom of the substructure. Large numbers of cases are required in order to infer statistical relevant parameters. In the following, this problem will be called '**deduction of preferred interaction directions**'.

Automated tools which easily find the required geometrical information on sets of structural data are mandatory in order to comply with all above mentioned tasks. Due to the constantly increasing amount of available data, such as protein structures in the PDB, these tools need to handle large amounts of data efficiently and have to be able to comprehensively present the relevant data. The resulting information can then be used to improve the parametrization of scoring functions, automatically provide information about interchangeable functional groups, easily detect bioisosters and chemoisosters, and can be used to identify and characterize new types of atomic interactions. The focus of this work is therefore the development of two stand-alone tools. One of them enables searches for interaction patterns while the other one is able to deduce preferred interaction directions of molecular substructures.

1.5. Overview of Content

This thesis is structured in the following way:

Chapter 2 gives an overview of available tools for the search of interaction patterns and the derivation of preferred interaction directions and outlines their achievements and their limitations.
Chapter 3 describes the aims pursued by this thesis.
In Chapter 4, the methods used and developed in the course of this thesis are explained. Firstly, the used software library is introduced. Afterwards, the algorithmical concepts and data structures developed in the course of this thesis are explained.
In Chapter 5 the experiments used to evaluate the developed methods are explained.
Chapter 6 and 7 then show and discuss the results of the two developed methods.
Finally, Chapter 8 summarizes the achieved results and provides an outlook into future developments.

2
State of the Art

In this chapter, the relevant literature for the two identified problems is presented: the search for interaction patterns and the deduction of preferred interaction directions. At the end of each section the features and limitations of the different approaches are summarized.

2.1. Searching for Interaction Patterns

This section discusses software tools that aim at finding spatial patterns of atoms in sets of macromolecular structures. Among them are tools which search for interaction patterns which is one of the main focuses of this work. Moreover, tools are included which aim at searching spatial patterns of connected atoms in proteins. These tools are included because in principle they handle the same problem which is the search for a specific spatial pattern of atoms in a large set of macromolecules. During the comparison, special attention is paid to four aspects:

- Type of geometric patterns that can be searched.
- Data preparation and storage.
- Search algorithm.
- Presentation and postprocessing of results.

In the following sections, a set of tools which enables the search of spatial atomic patterns on macromolecular structures is presented with regard to these four aspects. Conceptually, this set of tools can be divided into two groups regarding their target structures: proteins and protein-ligand interfaces. The first category contains the tools 3DinSight [46], Erebus [47], Suns [48], ASSAM [49], IMAAAGINE [50], and PDBeMotif/MSDMotif [51]. Therein, 3DinSight was already published in 1998 and was, to my knowledge, the first tool enabling this kind of data mining. The other tools started to emerge about a decade later. These tools are mainly used for the comparison of structural features of proteins and for the deduction of their functionality. The second category is comprised of the tools CSD-CrossMiner [52],

PRDB [53], Prolix [54], and Relibase [55]. These tools have been developed between 2003 (Relibase) and 2016 (CSD-CrossMiner). Relibase is also available in a commercial version, called Relibase+. This version provides some additional features and supports different queries. Therefore, both versions will be treated as different tools in the following comparisons. These tools are mainly used in order to study the formation of interactions between proteins and small molecules. A detailed description for each tool can be found in Appendix A.1. Among the presented tools, Erebus, Suns, ASSAM, IMAAAGINE, Relibase, and PDBeMotif provide a web service which is still operating and freely accessible. All other tools have either never been publicly available (CSD-CrossMiner, PRDB, Prolix, Relibase+) or the service has been suspended (3DinSight).

Besides the listed tools, there are some commercial software solutions which have never been published in scientific journals: Psilo [56] (Chemical Computing Group Inc.) and PLDB [57] (Schrödinger LLC). According to the website, Psilo contains a database system which provides access to macromolecular structures. One feature of this database system are geometric queries between protein and ligand. PLDB provides searches for geometric parameters and interaction motifs. However, nothing can be said concerning their search mechanism nor their retrieval speed.

2.1.1. 3D queries

In this section, the set of tools is analyzed regarding their supported 3D queries and their means to define these queries.

In general, the supported 3D queries differ strongly between the different tools. In most of the tools, the 3D query consists of search objects which can be combined by distance constraints. These objects may be peptides, amino acids, ligand substructures, or atoms. This category contains the tools 3DinSight, IMAAAGINE, PDBeMotif, PRDB, Prolix, Relibase, and Relibase+. However, the precision of the 3D query among these tools is very different. In IMAAAGINE, a 3D query consists of distance constraints between up to eight amino acids. Herein, an amino acids is considered as one unit. Hence, the distance constraint cannot be defined for a specific atom of an amino acid. Similar reductions of amino acids to one unit are done in 3DinSight. On the opposite, in Relibase+ molecular substructures for the protein and the ligand can be defined and distance and angle constraints can be added for each pair of atom. In Relibase, this functionality is limited to inter-molecular distance constraints between pairs of atoms. In that comparison, tools like Prolix, PDBeMotif, and PRDB support semi-precise 3D queries as distance constraints for specific atoms can be defined for some of the possible search objects.

Besides the atom-level precision, the flexibility in terms of constraint ranges is different in the 3D queries of these tools. For example, in Relibase and Relibase+, distance constraints between atoms can be defined using a range of allowed distances. This is helpful if some

part of the 3D query should be exactly matched by the results whereas in other parts, some flexibility can be tolerated. Similar distance ranges can be used in 3DinSight, PRDB, and Prolix. In IMAAAGINE, only one tolerance value valid for all defined distances can be defined. Even less flexible is the search in PDBeMotif, where only fixed tolerances can be used for distance constraints.

Moreover, the tools in this first category differ in their means to define a 3D query. 3DinSight and PRDB only work with SQL queries. Hence, the definition of complicated geometrical patterns can be quite difficult. All other tools provide 2D graphical interfaces which allows the definition of search objects and mutual distances between the objects.

The second category contains tools which are able to search for exact geometrical patterns which are not defined via distance constraints but by complete substructures. This means that the query is defined by one or several molecular substructures in which every atom requires a valid 3D coordinate. These tools are Erebus, Suns, ASSAM, and again PDBeMotif. Erebus, Suns, and PDBeMotif can all be seen as precise on an atomic level because the complete pattern from the 3D query is searched and only small geometrical deviations are allowed. As explained in the following section, ASSAM reduces each amino acid of the query structure to one unit which represents the spatial orientation of the side chain in a very simplified way and can therefore not be seen as precise on an atomic level. All of these 3D queries are not flexible as no allowed geometrical deviation for specific parts of the query can be defined. PDBeMotif appears in both categories because both types of searches are possible.

The tools in this category highly differ in their provided ability to define a 3D query. Erebus and ASSAM require a 3D query in pdb file format. This is a convenient way if a specific part of a known protein should be searched. However, setting up 3D queries from scratch can be quite difficult. PDBeMotif provides here a set of predefined motifs and a graphical user interface to define amino acid sequences with specific backbone angles. However, for larger structures, this can be very difficult as backbone angles have to be known for each amino acid. In that respect, the tool Suns provides a very convenient way to define queries: a protein can be loaded and fragments of interest can be selected in a 3D viewer. The downside of this approach is, however, that no query can be constructed from scratch.

In general, the search for larger 3D structural units is more relevant on protein structures where specific folds of structural elements are searched. This is reflected by the fact that this category contains only tools which focus their search on protein structures.

The search supported by the tool CSD-CrossMiner fits to neither of the aforementioned categories. Here, the spatial arrangements of specific atom types is searched. The search elements are spheres around selected atoms. The radius of a sphere encodes the allowed

structural variance. The 3D query is classified here only as semi-precise because only predefined types of atoms can be used for the search, e.g., a carbon of a methyl group. Also the flexibility of the 3D query is only rated as semi-flexible. The reason is that the used spheres allow to define the geometrical flexibility of the position of an atom. However, its position cannot be defined differently with respect to the other atoms in the 3D query. As in the tool Suns, CSD-CrossMiner provides a 3D view in which structures of interest can be loaded and a 3D query can be defined visually by selecting specific atoms.

Besides the 3D query, only a few tools provide the possibility to combine the geometrical search with numerical or textual properties of the protein, the ligand, or the complex, e.g., the resolution, the organism. These are Relibase, Relibase+, 3DinSight, and PRDB.

In conclusion, it can be seen that Relibase+ is the only tool supporting precise and flexible 3D queries where every substructure can be drawn and any geometrical constraint can be added. The most convenient means to define 3D queries from existing 3D structures are provided by Suns and CSD-CrossMiner.

Data preparation

Almost all tools presented here have in common that they use relational databases to store the precalculated data. CSD-CrossMiner uses an SQLite database and Suns utilizes an in-memory database. For Erebus, no comment about the storage of data is made in the publication.

Almost all tools store general information about each used macromolecular structure in order to be able to reconstruct relevant information for the resulting hits, e.g., the resolution, the PDB code, and the release date. However, they strongly differ in their way to precalculate data which is used for the 3D search.

The tools PRDB and PDBeMotif calculate distances between atoms of small molecules and amino acids and store them in specific tables. Similarly, 3DinSight and Prolix store distances between C-α coordinates of amino acids. Thereby, 3DinSight uses all pairs of amino acids whereas Prolix uses only those amino acids which are in close proximity to a ligand.

ASSAM and IMAAAGINE store the macromolecular structures as graphs. Herein, each amino acid is transformed to a vertex in the graph. The edges between the vertices represent the mutual distances between specific points within the amino acids. In both tools the graphs are complete, meaning that all pairs of vertices are connected by an edge.

A third category of tools uses specific index techniques in order to rapidly reduce the number of possible results. Herein, Relibase and Relibase+ use topological fingerprints for ligands.

Prolix stores for each ligand the type of amino acids in its vicinity and its atomic interactions to amino acid types in a bit vector. In a similar way, CSD-CrossMiner stores the existence of specific atom types within a protein-ligand interface in a bit vector.

Comparable to a fingerprint technique is the data preparation of Suns. Here, the protein structures are divided into cubes with 15 Å side lengths. All molecular fragments of two to eight atoms within these cubes are enumerated and stored.

These different data storing techniques can only be judged in combination with the search mechanism. However, an obvious disadvantage would be a large size of the stored data. Unfortunately, the database sizes of the different tools are not given in the respective publications. Only the authors of Suns state that the storage of 24 218 protein chains requires 89 GB of memory. Given the fact that the complete PDB contains about 130 000 files nowadays, advanced hardware components would be required if searches on the complete PDB were to be performed with this approach.

Search mechanism

The search mechanism is strongly coupled to the data stored in the database. The tools ASSAM and IMAAAGINE which store all proteins as graph data structures also translate the query into such a graph data structure and carry out subgraph mining algorithms. ASSAM is doing this by building product graphs followed by a the clique detection algorithm of Bron and Kerbosch [58]. IMAAAGINE utilizes the algorithm of Ullmann [59].

The tools PRDB and 3DinSight directly store distances and angles between atoms and amino acids. Moreover, they only support queries in SQL. Thus, the query mechanism only consists of a database query. Unfortunately, nothing is known about the exact query mechanism which is used in PDBeMotif.

The remainder of the tools all pursue a similar strategy: Different techniques are applied in subsequent steps in order to reduce the number of possible results until a precise matching is performed in the final step. Here, Erebus first translates the query into pairwise distances between atoms. These distances are searched in the database in subsequent steps. In a final step, the complete match is constructed by combining the detected results for all distance constraints. The tools which store fingerprints, namely Prolix, CSD-CrossMiner, Relibase, and Relibase+ first use these descriptors before a precise matching step is performed. Prolix and CSD-CrossMiner store features of specific pocket attributes. Thus, the first step contains the detection of pockets which contain all relevant features. Relibase and Relibase+ store a fingerprint for ligands. Hence, this technique can only be used if the query contains a ligand substructure. In a similar way, Suns first detects all cubes which contain the fragments requested in the query. Afterwards, the correct positioning of the fragments within a cube is verified using the algorithm of Kabsch [60].

The most important aspects on which a search mechanism can be judged are speed and correctness. In regard of the latter, the tool Suns demonstrates poor performance. Due to the arbitrary fragmentation of the data in cubes, no hits which span cubes can be detected. Thus the results might be not complete. Slight problems concerning the correctness can also be observed with the tool CSD-CrossMiner. Here, a query consists of spheres which represent the position of atoms. During the search, the query is converted into distance constraints between spheres. The complete results are finally constructed from these distance constraints. Due to this procedure, it might happen that hits contain atoms which are positioned outside of the query spheres. However, given the sometimes high inaccuracy of the used structural data, this problem might be negligible in this application scenario. All other tools should in principle be able to find all correct results.

Concerning the retrieval speed, it is difficult to compare the tools because they can handle different geometrical queries. Moreover, some of the tools are not publicly available. Others are only reachable via web interfaces. The hardware they are running on is not known making runtime measurement difficult to compare. Hence, only the runtimes stated in the respective publications are taken into account here.

In general, tools which find their results in a short time or even support interactivity are more convenient for potential users. In total, runtimes between $<$1 s and 6 min are reached with the reviewed tools here.

Remarkably, Suns achieves these runtimes on a database containing only 272 non-redundant protein chains. For a database containing the complete PDB, these runtimes could be much higher.

In PRDB, geometrical queries have to be defined in SQL and are directly applied on the database. Here, runtimes between 0.03 and 398 s on the complete PDB are reported. These numbers directly reflect the response time of the database and thus can be seen as a score for the database design used in PRDB. The high runtimes result from queries which contain amino acid triplets with mutual distances. These distances are stored in the database and a six time self-union of this table is required for this query.

Furthermore, it is remarkable that for Prolix very short runtimes below 5 s for typical queries are reported. On the other hand, for ASSAM runtimes of about 6 min for typical 3D queryies are reported. Both tools were published in the same year (2012) but nothing is known about the efficiency of the used hardware components, respectively.

From this small overview it could be concluded that fingerprint based-techniques can be beneficial and can accelerate the following exact matching procedure. The storage of all possible data as done by PRDB does not automatically lead to fast retrieval times and reduces the variability of the supported queries.

Result presentation

All tools provide a list of results, including the PDB code of the hit. Besides this, several tools provide the possibility to inspect the results visually in 3D. These are all tools except from 3DinSight and PRDB. For 3DinSight and PRDB, however, no comment about a visual presentation of results has been made in the respective publications.

The tools Suns, ASSAM, and CSD-CrossMiner even provide the possibility to superimpose the results based on the geometrical query. This enables the direct identification of differences and common structural features among the results. In my opinion, a very important functionality in the context of interaction pattern searching.

Relibase+ only provides the possibility to superimpose similar proteins based on their sequence which is not very helpful if the results based on a 3D query should be analyzed in more detail. However, Relibase+ is the only tool which can be used to analyze crystallographic packing effects in macromolecular structures [61].

Prolix, Relibase, and Relibase+ also provide statistics on the resulting hits. In Prolix, the percentages of hits which contain a specific interaction are visualized. Relibase and Relibase+ present histograms showing the distribution of the measured distance and angle constraints within their ranges from the query.

The tool Suns argues that it is perfectly suited to support the refinement of queries. Once a query has been answered, all resulting structures can be seen in a superposition and specific structural features can easily be spotted. From that view, the previous query can be refined by adding or changing constraints.

This overview of additional features for analyzing the results is probably not complete. Here, only features which are reported in the respective publications and which were considered useful in the context of structure-based drug design are given. In my opinion, the extraction of important information out of the result set is a very important functionality. Results of a structural search are less meaningful if they cannot be compared structurally. In this context, two functionalities seem to be extremely helpful: the superposition of the resulting structures and the extraction of statistics.

A complete overview about the most important aspects of the different tools is given in Table 2.1. It is obvious that no tool has the optimal characteristic in all the different columns. These are first of all the correctness of the search and an atom-precise query with possible ranges for distance and angle constraints. Tools for the superimposition and visual inspection of the result should be provided and statistics on the resulting hits should be accessible. The specificity of the tool could be even increased if the analyzed data set could be defined and easily exchanged by the user. Thereby, one tool could provide the possibility to find bioisosteric replacements in a non-redundant set of protein-ligand complexes. At the same

	Toolname	Performance criteria							Search algorithm
		Correctness	Query variability			Data flexibility*	Superimposition of results	Results statistics	
			Atom-level precision	Constraint ranges	Non-geom. attributes				
Focus on proteins	3DinSight	yes	no	yes	yes	no	no	no	SQL query
	Erebus	yes	only protein	no	no	no	no	no	Incremental, start with all pairwise distances
	Suns	no	only protein	no	no	no	yes	no	Incremental, starting with fragment search
	ASSAM	yes	no	no	no	no	no	no	Subgraph matching
	IMAAAGINE	yes	no	no	no	no	no	no	Subgraph matching
	PDBeMotif/ MSDmotif	yes	only ligand	no	no	no	yes	no	?**
Focus on protein-ligand interfaces	CSD-CrossMiner	no	semi	semi	no	no	yes	no	Incremental, start with fingerprint
	PRDB	yes	yes	yes	yes	no	?**	?**	SQL query
	Prolix	yes	only ligand	yes	no	no	no	yes	Incremental, start with fingerprint
	Relibase / Relibase+	yes	yes	yes	yes	no	no	yes	Incremental, start with fingerprint
	optimal tool	yes	yes	yes	yes	yes	yes	yes	–

Table 2.1.: Tools which enable 3D searches of atomic patterns. The first part of the table contains tools which focus on the search on proteins. The tools in the lower part of the table focus on protein-ligand interfaces. *Here, data refers to the data set on which the search is performed. ** Information not provided in the respective publication.

time, this tool could be used to detect chemoisosteric replacements among a set of similar proteins. To my knowledge, a simple switch of the analyzed dataset is not possible in the above described tools. Finally, only a small amount of tools allows the combination of a geometrical query with constraints for textual and numerical properties of the protein, the ligand, or the protein-ligand complex. These additional constraints could help to precisely tailor a structural search to a users demand.

Hence, there is a lack of tools which can rapidly answer precise and flexible geometrical queries on an atomic level.

2.2. Deduction of Preferred Interaction Directions

In the last 30 years, several studies have been performed which all aimed at deducing interaction preferences for specific chemical fragments. A comprehensive overview on studies performed to deduce geometrical preferences in hydrogen bonds can be found in [16]. Moreover, several studies have been performed in order to detect interaction preferences in macro molecular structures more generally. Among these, some studies focused on fragments of ligands [62, 63], others focused on fragments of amino acids [64–67], and a third group investigated both, fragments derived from ligand and protein [68, 69]. [62, 63, 69] even used statistical models to calculate probabilities of interaction geometries.

Despite the large number of studies on the topic of preferred interaction directions, only very few research groups have published ready-to-use tools implementing their methodology for other users. The remainder of this section will focus on those studies for which a tool is or has been available and discuss their strengths and weaknesses.

To my knowledge, in 1990, Sirius [64] was the first tool which calculated preferred directions of atomic interactions in macromolecular structures. This work resulted in the 'atlas of protein side-chain interactions' [70] and has later been used to evaluate the binding of peptide inhibitors [71]. Sirius' basic idea was later extended in the tool X-Site [65] in 1996. One year later, the tool IsoStar [68] was published which still is the most powerful tool available. Its main advantage is that it uses CSD data on top of the commonly used PDB. In 2014 the tool GIANT [66, 72] has been published which in contrast to the other tools discussed here uses statistical models to analyze the molecular data.

Among those tools, GIANT is the only currently publicly available tool and can be accessed via a web service. IsoStar is commercially distributed as a stand-alone tool.

In the following, these tools are analyzed with regard to their data preparation and their data presentation. Moreover, a detailed description for each tool can be found in Appendix A.2.

2.2.1. Data Preparation

The main idea of data preparation is very similar in all four tools: specific molecular substructures are identified in a data set of macromolecular structures. These substructures are then transferred into a reference coordinate system and the positions of surrounding, interacting atoms or other substructures are recorded. The four tools reviewed here mainly differ in their way how the molecular substructures are defined.

In Sirius, complete side chains of amino acids are used as substructures and the positions of surrounding side chains are recorded. In X-Site, the molecular substructures is generated by breaking up each amino acid within a binding site of a ligand into overlapping three-atom fragments. For each fragment, the distribution of interacting atoms is then recorded. Herein, chemically equal fragments coming from different amino acids are combined.

GIANT follows a very similar approach. However, the main aim here is to classify spatial interaction preferences without mixing the data from different amino acids. To this end, each amino acid is decomposed into fragments of three connected atoms. The specification of each fragment includes specific atom names for each position within an amino acid and the amino acid type. Hence, fragment which derive from different amino acids are treated as different fragments even though they might be chemically equal. Surrounding atoms of ligand molecules are detected for each fragment. For each atom type of these interacting atoms, spatial propensity functions were calculated using a Gaussian mixture model. The results are regions around the defined fragment having a high probability of hosting a specific ligand atom. The calculated data is stored in a database.

In IsoStar, a predefined set of functional groups is used as molecular substructures. For each functional group, the distribution of relevant atoms as well as other functional groups is recorded. Herein, it is differentiated between the original structure of the functional groups, e.g., protein or ligand. The data is stored in different data files, one for each combination of functional groups.

Even though the tools use different ways to define their set of analyzed substructures, almost none of them supports the flexible extension of this set. Only for IsoStar, the tool IsoGen can be used to generate data files for custom defined substructures. However, this only works for the CSD data set and not for the PDB [73].

In all four tools, the resolution is used as only quality criterion of the analyzed structures. Herein, Sirius, X-Site, and IsoStar use an upper limit of 2.0 Å whereas GIANT uses all structures with a resolution below 2.5 Å.

Moreover, the set of analyzed macromolecular structures cannot be exchanged in the reviewed tools. Again, only IsoStar makes a small difference here since the CSD and the PDB can be used. However, it is not possible to use a set of interesting macromolecular structures, e.g., all structures derived from a molecular-dynamics simulation, and analyze the preferred

interaction directions therein.

2.2.2. Data Presentation

The presentation of the analyzed data and the means to tailor the presented data to a users specific request is an important aspect of the usability of a tool. Only a clear and comprehensive presentation can help researchers with sound decision making in the rational drug design process.

In IsoStar, the distribution of atoms around a central substructure can be presented in two different modi: the atoms are either displayed as dots or a contoured density surface is shown. The latter can be created only for a specific element type. For each dot, a backlink to the original structure is provided. Moreover, sets of presented atoms can be reduced to only those in a specific distance to the central functional group. Distance distributions can be visualized in a histogram. In the 3D view, arbitrary distances, angles, and torsion angles can be measured. Moreover, the data files for IsoStar can be used to identify regions in a protein binding site where chemical groups are likely to interact using the tool SuperStar [74, 75]. To this end, the atom distributions around functional groups are transferred and displayed in user-defined protein binding sites.

In GIANT, the probability of atoms being in specific regions around the predefined fragments can be visualized in a web service. For each cluster, additional information as a list of protein-ligand complexes which contain the specific interaction pattern are provided. However, no further means of data analysis are provided. It is therefore difficult to infer interaction geometries for specific groups from that presentation. In a second application, protein-ligand interfaces can be loaded and the calculated data can be transferred and visualized within this interface.

For Sirius, it is not exactly known how the data is presented and which means for data analysis are provided. The existence of a stand-alone tool is mentioned in the respective publication but has never been published or described separately. The same holds true for X-Site.

In summary, only IsoStar provides some further means to analyze the calculated data. Filter for the distance and element type can be applied and a backlink to the original structure is provided. This can be very helpful if the position for outlying atoms should be analyzed in detail. However, even more filter options are required for much more sophisticated analyses. In that respect, the differentiation between intra- and inter-molecular connections can be helpful. Filters for specific parts of the structure could help finding differences of preferred interactions between side chain atoms, backbone atoms, and ligand atoms.

Toolname	Flexibility of substructures	Flexibility of data	Quality criterion	Filter for data presentation	Backlink to original structure
Sirius	no	no	res. $\leq$ 2.0 Å	no	no
X-Site	no	no	res. $\leq$ 2.0 Å	no	no
IsoStar	semi	semi	res. $\leq$ 2.0 Å	distances and element types	yes
GIANT	no	no	res. $\leq$ 2.5 Å	no	no
optimal tool	yes	yes	atom-wise criterion	yes	yes

Table 2.2.: Tools which deduce preferred interaction directions for molecular substructures. res. = resolution.

As described a the beginning of this section, several studies have been performed in order to analyze the geometrical interaction preferences of specific chemical groups. Surprisingly, the number of tools providing this knowledge and allowing a convenient deduction of parameters is relativity small. The high number of citations of IsoStar (280, from http://scholar.google.com/, accessed May 2017) however convincingly shows that these tools are used in structure-based drug design applications.

Table 2.2 gives an overview about the most important features identified throughout the previous sections. The main shortcoming of all available tools is the insufficient flexibility of the analyzed data. This includes the used set of PDB files as well as the set of chemical fragments. In order to be applicable in diverse settings, a tool requires a flexible adaption of these data sets. Thereby, the current application scenarios of existing tools could be extended to identify new atomic interactions. In addition, specific sets of macromolecular structures could be analyzed. For example, all structures derived from a molecular dynamic simulation could be analyzed and those structures which exhibit a specific interaction could be detected. A second unsolved problem is the data quality. The presented tools all use the resolution of a structure as the only measurement of quality. However, if precise predictions should be deduced from a spatial distribution, it is of high importance to know about the experimental precision of the used atom positions and, if required, only those atoms which have a high experimental support should be used.

In summary, it can be seen that only a few tools exist which can help deduce preferred interaction directions. The existing tools demonstrate weaknesses in their flexibility of the used data and their means for data analysis.

3

Aims and Preconditions

The preceding chapters presented how knowledge about principles in molecular recognition can be used in the structure-based drug design process. Existing software solutions which can be used to create and deepen this knowledge were introduced and their shortcomings were outlined.

The main aims in this thesis were the development of new computational methods which are able to solve the two problems introduced previously: searching for interaction patterns and deduction of interaction preferences. During the development, great emphasis was laid on the following aspects:

- Consistent handling of data: Structural data of proteins and their bound small molecules has to be processed in a consistent way and stored such that it is accessible to geometrical searches.
- Reliable and correct retrieval system: Geometric searches should be performed on the data in order to deduce information relevant for the two mentioned problems. In this respect, the correctness of the retrieval system is a fundamental requirement.
- Retrieval speed: The time needed to handle different queries should be as small as possible. Ideally, interactive usage of the tools should be possible.
- High variability: The methods should be able to precisely mine a data set for the relevant spatial information. To this end, the query system needs high variability. Moreover, the analyzed data set should be variable.
- Usability: Methods which aim at providing information to guide the structure-based drug design process have to be used by researches working in this field, e.g., medicinal chemists. In that respect, the usability of the developed methods is of high importance. This includes an easy handling of the data, an intuitive generation of queries, and a comprehensive presentation of the results.

The methods presented in this thesis were developed from February 2013 to June 2017 in the research group Computational Molecular Design at the Center of Bioinformatics, Universität Hamburg. They are based on the NAOMI software library [76] which is jointly developed and maintained by the PhD students in the research group. The library mainly provides methods and data structures for the chemically consistent handling and parsing of molecules in C++. In Chapter 4, methods from this library which were used in this thesis are briefly explained and the respective publications are given. Besides, functionality from the Qt library (https://www.qt.io/) was used. The graphical user interfaces (GUI) developed in the course of this thesis are based on QtQuick and QML and use the 3D visualization library for molecules and proteins developed by the BioSolveIT.

In the course of this thesis, the method developed for the search of interaction patterns was published in the *Journal of Chemical Information and Modeling* [77]. Moreover, a brief description of the method was published in the *Journal of Biotechnology* [78]. The method for the deduction of interaction preferences was jointly developed with Eva Nittinger and has been submitted to the *Journal of Chemical Information and Modeling* [79]. At the time of thesis preparation, the manuscript was under revision. Additionally, this method has been used to infer interaction geometries of different functional groups. The results of this survey were published in the *Journal of Medicinal Chemistry* [16]. See Appendix G for a list of publications, talks, and posters presentations.

4
Methods

In this chapter the methods developed and used in the course of this thesis are presented.

In the first section, fundamental decisions on the storage of data are outlined. Afterwards, the previously developed methods of the software library NAOMI that were applied in the context of this thesis are briefly summarized. Section 4.3 describes the new methods and algorithms for the search of interaction patterns. This methodology is the basis of the tool Pelikan. In Section 4.4, the methods developed for the problem of deduction of interaction preferences are presented. These methods are at the heart of the software tool *NAOMI*nova.

4.1. Basic Decisions on Data Storage

In Chapter 3, the main aims for both methods developed in the course of this thesis have been outlined. The way of storing the relevant data is fundamental in order to be able to reach these goals. First of all, large data sets should be handled and fast searches should be possible. Secondly, the used data should be interchangeable. In principle, databases are perfectly suited to fulfill these requirements. They can store large data sets and provide a quick access to the data at the same time. As outlined in Chapter 2, databases are utilized also by several other method in this context.

Two fundamentally different types of databases exist: relational and non-relational databases. While relational databases have the advantage that structured data can be easily stored and queried, non-relational databases are more flexible. No precise table scheme has to be defined and any data can be stored immediately. The data which should be stored here is, however, highly structured. Structures of proteins can be seen as a set of points in 3D space. In this view, every point represents an atom which has specific attributes, e.g., its element type. Very early in the course of this thesis, it was therefore decided to use a relational database.

The next decision regarded the type of the used relational database for both problems.

There are mainly two different classes: Server-based and non-server-based databases. As the used data should be interchangeable, the choice fell on a non-server-based database, namely SQLite. Herein, all data is stored in one file and can be easily exchanged. Moreover, the SQLite database file is compatible with different platforms which contributes to the usability of the developed methods.

Besides these considerations, the use of an SQLite database has another advantage: The NAOMI library already contains methods to store and access molecules in an SQLite database. Thus, these methods can be used here.

4.2. Basic Methods of the NAOMI Library

In this section the basic methods from the NAOMI library which have been applied in this work are explained. In cases where the functionality has already been published, only the most important aspects are mentioned.

4.2.1. Parsing and Interpreting the PDB

Structures of protein-ligand complexes can be encoded in different formats. The most common formats which are supported by the PDB are the '.pdb' format and the '.mmCif' format. Both formats are supported by the NAOMI library and basically contain the same information. The NAOMI library contains two different routines to parse the information. The further handling and interpretation of this parsed data is equal and does not depend on the original file format. In the following, the term 'PDB file' will refer to a file from the PDB which has one of the two formats.

A PDB file basically contains a list of all atoms measured in the structure including their element types and their coordinates in Cartesian space. Additional information regarding the experimental conditions used to determine the structure is given in the header section. The exact location of bonds between atoms and their valence state is however often not given. In the NAOMI library, the localization of bonds and valence states of the atoms is determined by checking the pairwise distance between all atoms and a rule-based chemical model [80]. The exact perception of atom types and bonds for small molecules has been described by Urbaczek *et al.* [81]. For proteins, this process has been described by Bietz *et al.* [38]. The used chemical model is very strict, meaning that if no valid valence state can be determined for an atom, the complete molecule is discarded. During this procedure, all constructed molecules are first classified as protein. Herein, all connected atoms with the same 'residue number' from an input file are firstly interpreted as a residue. In a finalization step, some of these residues are classified as ligands. This is done if they are isolated. Additionally, chains consisting of less than six residues are interpreted as ligands. A known

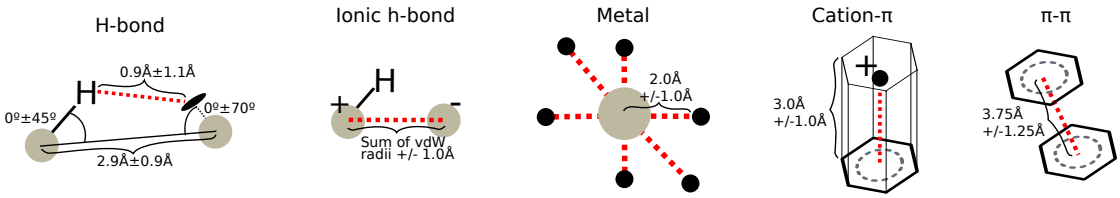


Figure 4.1.: Different types of atomic interactions and their geometrical constraints. This figure is adapted from [77]. Reprinted (adapted) with permission from [77]. Copyright 2017 American Chemical Society.

limitation deriving from this step is the categorization of covalently-bound ligands as part of the protein. Moreover, large sugar polymers are very often composed of more than five connected sugar molecules. Therefore, these molecules are also interpreted as protein in the NAOMI library. Another known shortcoming of the NAOMI library is that molecules with covalently bound metals cannot be built due to missing models.

The protein-ligand structures from the PDB are mainly derived from X-ray crystallography experiments where the position of protons can almost not be determined due to the lack of electrons. After the initialization of a protein-ligand complex, the positions of hydrogen atoms are calculated using Protoss [38] which is also part of the NAOMI library. Briefly, Protoss enumerates alternative hydrogen positions and protonation states of polar moieties in order to optimize the hydrogen bond network in molecular structures.

4.2.2. Calculation of Atomic Interactions

For the calculation of atomic interactions, a two-step method is used. First of all, atoms and rings which could in principle be involved in an interaction are determined. These are atoms which can be a hydrogen bond donor or an acceptor, which are charged (anion or cation), and all aromatic rings. In the next step, all detected atoms and rings are compared pairwise and the agreement of geometrical constraints for the different types of interactions are tested. The constraints for the different interaction types used here are shown in figure 4.1.

4.2.3. The EDIA value

The electron density for individual atoms (EDIA) is a measure for the experimental support of individual atoms in protein structures resolved by X-ray crystallography and can take values between 0.0 and 1.2. In order to calculate the EDIA, the experimentally determined electron density around each atom is analyzed. Atoms whose position is not well defined by electron density are assigned to a low EDIA value. This can be the case if atoms are

positioned outside of electron density or inside of large electron density bulbs. Atoms with an EDIA of 0.8 and higher are thought to be well supported by electron density. In general, the EDIA should only be calculated for structures with a resolution of or below 2.5 Å. A detailed description and evaluation of this method can be found in [41].

4.2.4. Superimposition of Atoms

In the NAOMI library, atoms from one set can be superimposed onto a set of other atoms using the algorithm of Umeyama [82]. Note that for this method, a unique assignment between atoms from the first and the second set are required. The quality of the superimposition is determined using the 'root mean square deviation' (RMSD), see formula 4.1. Herein, V and W are the two sets of atoms, v_i is the atom from V matching to atom w_i from W and $\vec{v}_i$ and $\vec{w}_i$ are the position vectors of atom v_i and w_i, respectively. n is the number of atoms in per set.

$$RMSD(V,W) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||\vec{v}_i - \vec{w}_i||^2} \tag{4.1}$$

4.2.5. Spatial Atom Index

The NAOMI library contains an interface to the nanoflann library [83]. This library enables the use of k-dimensional trees (k-D trees) which supports efficient distance queries on a k-dimensional set of points. The nanonflann library is especially optimized for the use of points in 2D and 3D.

4.2.6. The MolString

The molString is a unique string representation of molecules developed by Hilbig *et al.* [84]. It stores all atoms and bonds occurring in a canonized molecule and can be used to completely reconstruct a NAOMI molecule. Here, the molString is mainly used to efficiently store molecules in databases and to rebuild molecules upon request.

4.2.7. Databases

The NAOMI library contains several libraries which store information about molecules, proteins, and protein-ligand complexes in an SQLite database. Moreover, arbitrary numeric and textual properties can be stored in this database. Conceptually, the database is separated in different parts. Each part is in charge for a specific subset of tables. The main task of

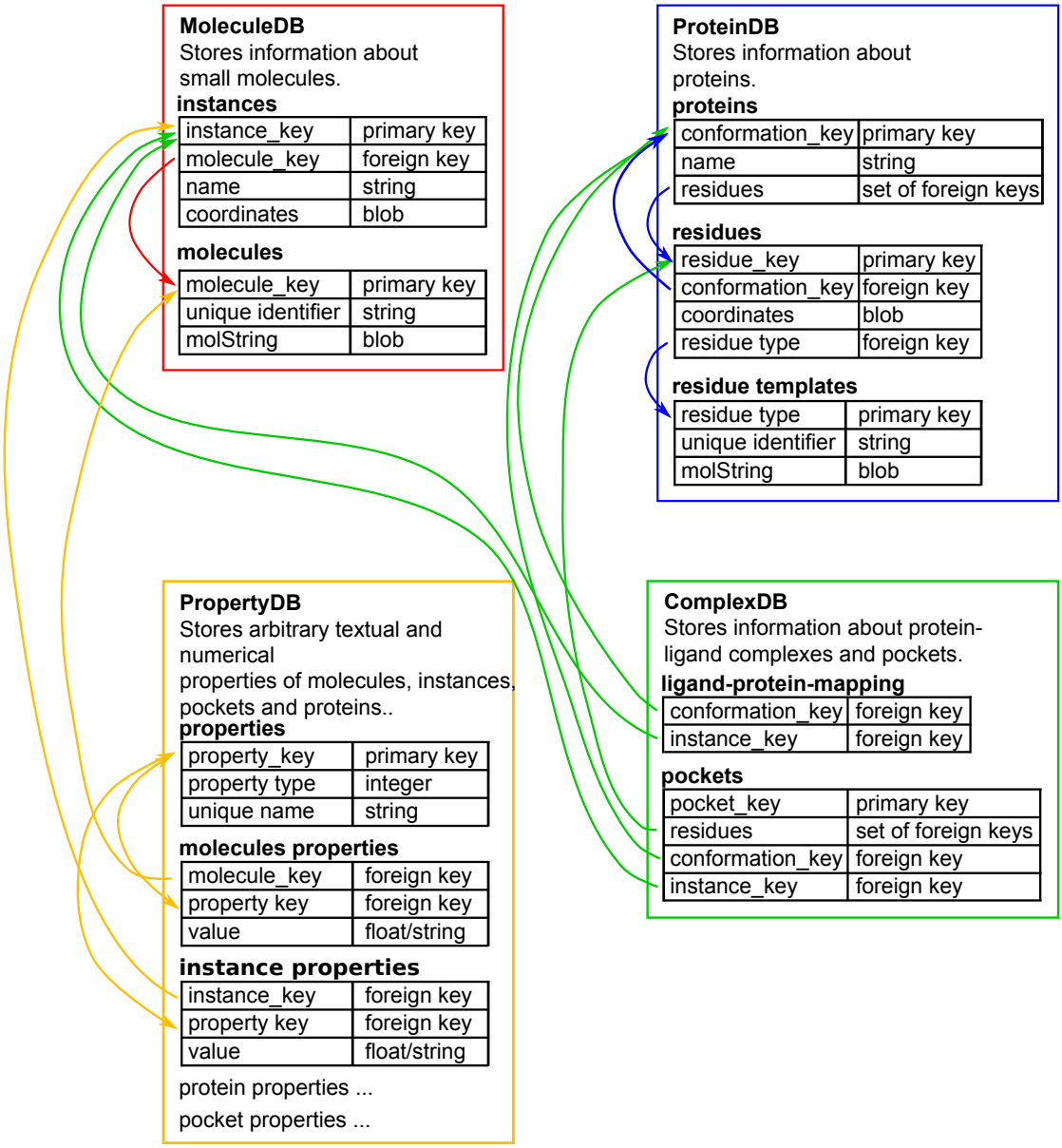


Figure 4.2.: Schematic depiction of different database tables in the NAOMI library. The red box contains the tables of the MoleculeDB, the blue box contains the tables of the ProteinDB, the green box contains the tables of the ComplexDB, and finally the yellow box contains the tables of the PropertyDB. Arrows indicate cross references between tables.

the database parts is to store the protein-ligand complex and to rebuild the desired structure upon request. A general overview about the different databases and their connections can be seen in Figure 4.2. In the following subsections, these parts are introduced in more detail.

MoleculeDB

The MoleculeDB handles the storage of small molecules. Its assembly is schematically depicted in the red box in Figure 4.2. Conceptually, each inserted small molecule is separated in its topology and its 3D coordinates. The topology of a molecule is represented by its MolString. For comparing molecular topologies, a unique string is generated using the SMILES language [85]. These values are stored in the table 'molecules'. The 3D coordinates of a molecule are stored in the table 'instances'. Two different keys, the instance key and a molecule key, are used to connect both database entries. Using the instance key, a molecule can be completely reconstructed. The conceptual details of the storage of small molecules has been explained in more detail by Hilbig *et al.* [84].

ProteinDB

The ProteinDB stores all information about proteins, including their different chains and amino-acids. It contains three different tables which are schematically depicted in the blue box in Figure 4.2. A protein is represented in these tables using a unique identifier (id), named conformation key. In the table 'proteins', a conformation key is associated with a name and a set of residues which build the protein's chain. Residues are identified using unique residue keys which are the primary keys of the table 'residues'. For each residue key, a set of coordinates and a residue type is stored. The residue type is comparable to the molecule key from the MoleculeDB. It refers to a unique chemical topology of a residue which is stored in the table 'residue templates'. The conformation key can be used to completely rebuild the protein. Moreover, the residue key can be used to rebuild only a specific residue of the protein. A more detailed description of the ProteinDB has been published by Schomburg *et al.* [86].

ComplexDB

The ComplexDB stores information about protein-ligand complexes and pockets. An overview about the structure of its tables is displayed in the green box in Figure 4.2. The table 'ligand-protein-mapping' maps conformation keys to instance keys. This mapping represents the small molecules and proteins which build a protein-ligand complex. Pockets are represented by a unique key, named the pocket key. In the table 'pockets', a pocket key is stored with three different foreign keys. First of all, a conformation key refers to the protein the pocket belongs to. Secondly, an instance key links to the reference ligand of the pocket in the MoleculeDB. Thirdly, a set of residue keys represents all residues which are part of the pocket. A more detailed description of the ComplexDB has been published by Schomburg *et al.* [86].

PropertyDB

The PropertyDB is used here to store arbitrary numerical and textual properties for molecules, proteins, and pockets. The structure of its tables is schematically displayed in the yellow box in Figure 4.2. A property is identified here using a unique key, named property key. For each property, a unique name and a type is stored in the table 'properties'. The property type encodes whether this property describes a feature of a molecule, an instance of a molecule, a pocket, or a protein. For each property type, another table stores the value of the property, a property key, and a foreign key pointing to the item the property belongs to. Depending on the table, this can be either a molecule key, an instance key, a pocket key, or a conformation key. In the yellow box of Figure 4.2, only the tables for molecule and instance properties are shown as representatives. This database is also intensively used by the software tool Mona [87].

4.2.8. Property Calculation for Molecules and Pockets

The NAOMI library provides methods to calculate physico-chemical properties of molecules and pockets. For molecules, this comprises properties as the volume, the molecular weight, or the count of heavy atoms. The same molecular properties are used in the software tool Mona [87]. Properties for pockets comprise descriptors for the topology of the pocket, e.g., its volume or its depth. These properties are calculated using the DoGSite algorithm [88].

4.2.9. Representation of Molecular Patterns

Within the NAOMI library, the concept of SMiles ARbitrary Target Specification (SMARTS) [89] has been implemented. A SMARTS is a linear representation of a molecular pattern. Within these patterns, atoms can be exactly described using various descriptors. Moreover, atoms can also be described more vaguely by combining atom properties with logical operators, e.g., 'N OR O'. Wild cards representing 'any atom' are also part of the SMARTS syntax. A recursive operator allows for the description of molecular environments around atoms. The NAOMI library contains functionality to interpret these SMARTS pattern and to search for matching atoms in molecules [90]. In this work, an improved version of the originally published SMARTS matching algorithm is used. Herein, the overall procedure is unchanged but the runtimes have been reduced by Robert Schmidt [91].

4.3. Pelikan - Searching for Interaction Patterns

In the following, the methods and concepts developed to search for interaction patterns in large sets of protein-ligand interfaces is explained. This includes newly implemented

algorithms as well as extensions to methods from the NAOMI library, already described in the Section 4.2. The method is a part of the software tool Pelikan. Conceptually, the method focuses on interaction patterns at the interface between proteins and ligands because most of the application scenarios presented in Section 1.4 are targeting this area. The methodology mainly comprises the following aspects:

- A query concept being able to represent different 3D patterns of atoms and different textual and numerical properties (see Section 4.3.1).
- Derivation and storage of relevant data from a set of protein-ligand interfaces (see Section 4.3.2).
- Efficient mining of the stored data for defined queries (see Section 4.3.4).

In the following, these aspects will be explained in more detail.

4.3.1. The Query

The query is composed of two different parts: a 3D query and a filter for textual and numerical properties. The 3D query contains geometrical objects of three different types: search points, point-point constraints, and angle constraints. It describes a spatial arrangement of atoms in Cartesian space. Every object which is part of a 3D query is equipped with a unique id. This id is used in various contexts in order to refer to a specific query object. Filters for numerical and textual properties can be added independent of the 3D query. These filters can describe various properties of the ligand, the protein, the pocket, or the protein-ligand complex, e.g., the resolution, the pocket volume, and the molecular weight of the ligand. In the following, all different query objects are introduced.

Search Points

Search points represent heavy atoms in a protein-ligand interface. Each search point has at least three different attributes. An overview about these attributes, their possible values, and their default configuration is shown in Table 4.1. The attribute 'molecule type' specifies to which molecule an atom should belong, e.g., protein or ligand. The exact values which are possible for this attribute are listed in the first row of Table 4.1. Each search point further has the attribute 'element' which holds information about the element type an atom should have. The possible types for the attribute 'element' are listed in the second row of Table 4.1. It comprises the organic set of atoms and all metals and halogens which frequently occur in protein-ligand complexes. The third attribute of all search points is the 'interaction type'. It describes the interaction type an atom can have e.g., donor or aromatic. The possible values of this attribute are listed in the third row of Table 4.1. These three attributes can

Attributes	Value type	Possible values	Default value
Molecule type	integer	any molecule, reference ligand, ligand, protein, metal, water	any molecule
Element	integer	any element, any element type of the organic subset, halogen, or metal (see table B.2 for the exact element types which can be used)	any element
Interaction type	integer	any type, acceptor, donor, aromatic, metal, cation, anion, hydrophobic	any type

Table 4.1.: Attributes of all search points of a 3D query in Pelikan.

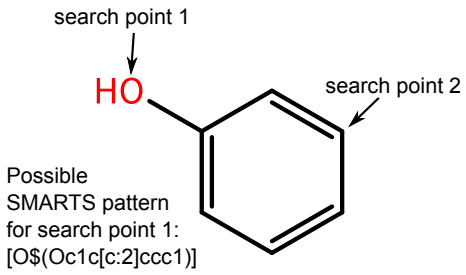


Figure 4.3: Example of a recursive SMARTS pattern describing the environment of a search point. The relative position of both search points within the molecules can be annotated using the search point labels in the SMARTS expression.

be defined for any search point. Depending on the value of the attribute 'molecule type', more attributes might be possible to define.

If the search point should only match atoms in small molecules, the attribute 'molecule type' should be set to 'reference ligand' or 'ligand' (the difference between these values is explained in Section 4.3.2). In this case, two additional attributes can be adjusted for the search point. Both are listed in Table 4.2. Firstly, the functional group an atom should be part of can be defined. The possible values of this attribute are shown in the first row of table 4.2. Secondly, a more detailed description of the environment of an atom can be defined using a recursive SMARTS pattern. Therein, the relative position of different search points can be defined using labels in the SMARTS pattern. An example for such a SMARTS pattern is displayed in Figure 4.3. In this example, search point 1 should match an aliphatic oxygen. Search point 2 should match an aromatic carbon. The relative position of both atoms in the molecule can be defined using the label of search point 2 in the SMARTS pattern describing the environment of search point 1: [O$(Oc1c**[c:2]**ccc1)].

If the search point should match only atoms which are part of a protein, the attribute 'molecule type' should be set to 'protein'. These search points have additional four attributes which are listed in Table 4.3. Firstly, the amino acid type can be chosen. As shown in the first row of Table 4.3, all 20 proteinogenic amino acids can be used here. Moreover, amino acid types are grouped by their chemical property. These can be used instead of a concrete

Attributes	Value type	Possible values	Standard value
Functional group	integer	any group, all other used functional group (see tabel B.1)	any group
Atom description	string	any valid SMARTS pattern describing one atom	empty string

Table 4.2.: Additional attributes of search points which only match atoms in small molecules of a 3D query in Pelikan.

amino acid type. The exact definition of each group is shown in Appendix B.4. The second additional property for search points matching only atoms in proteins is the location of the atom within the amino acid. Three different values are differentiated here: any location, backbone, and side chain. Moreover, the secondary structure element can be defined in which the matching atom should be located. The possible values for this attribute are listed in the third row of Table 4.3. Fourthly, the chemical environment of an atom matching the search point can be described using a recursive SMARTS pattern. Here, the same functionality applies as for search points matching atoms in small molecules. However, the SMARTS pattern may only involve one amino acid. If the amino acid type of a search point is set to a specific type, a fifth attribute can additionally be set. This attribute is the name of the atom. It refers to the specific atom name given to any atom of an amino acid within the PDB file. The exact names which are used for the specific amino acid types are listed in Appendix B.3.

Point-Point Constraints

Point-point constraints describe relationships between two search points. There are two different types: distance and interaction constraints. A distance constraints describes the minimum and maximum allowed distance between two search points. In principle, any distance value could be used here. However, with regard to the use cases mainly relevant for this method, the upper limit for both values is set to 15 Å. In an interaction constraint, an interaction type can be used to describe the relationship between two points. Here, any of the precalculated interaction types shown in Figure 4.1 can be used.

Per definition, there is a precise order of the search points within a point-point constraint. Thus, a point-point constraint always has a directionality from its first to its second search point. For referring to the specific search point object, the unique id of the search point is used. This order is necessary to exactly define angle constraints as explained in the the following section.

Attributes	Value type	Possible values	Standard value
Amino acid type	integer	any amino acid, alanine, arginine, asparagine, aspartate, cysteine, glutamine, glutamate, glycine, histidine, isoleucine, leucine, lysine, methionine, phenylalanine, proline, serine, threonine, tryptophane, tyrosine, valine, hydrophobic, polar, aromatic, acidic, basic, neutral	any amino acid
Atom location	integer	any location, backbone, sidechain	any location
Secundary structure	integer	any sec. structure, α-helix, β-sheet, no sec. structure	any sec. structure
Atom description	string	any valid SMARTS string describing one atom	empty string
Atom name (if amino acid type is given)	string	any name, list of specific atom names per amino acid type, see Appendix B.3	empty string

Table 4.3.: Additional attributes of search points which only match atoms in protein of a 3D query in Pelikan. sec. = secondary.

Angle Constraints

The third type of objects of a 3D query in Pelikan is the angle constraint. Angle constraints can be defined between any two vectors in the 3D query using two ids to other objects in the query and a minimal and a maximal allowed angle. Both, search points and point-point constraints may contain vectors. Figure 4.4a gives an overview about all vectors in search points. Only search points which have one of the following interaction types contain vectors: donor, acceptor, and aromatic. For donors, the vectors point from the heavy atom towards the hydrogen. For every hydrogen, one vector is contained in a search point. Similarly, the vectors of a search point with interaction type 'acceptor' point from the heavy atom in the direction of the electron lone pair. Again, a search point might contain several vectors if the matching atom has more than one lone pair. For aromatic search points, the normals of the ring plane in both possible directions are used as vectors. Figure 4.4b shows the second source of vectors in a 3D query. In principle, every point-point constraint contains exactly one vector. This vector reflects the direction from the first search point of the point-point constraint to the second search point.

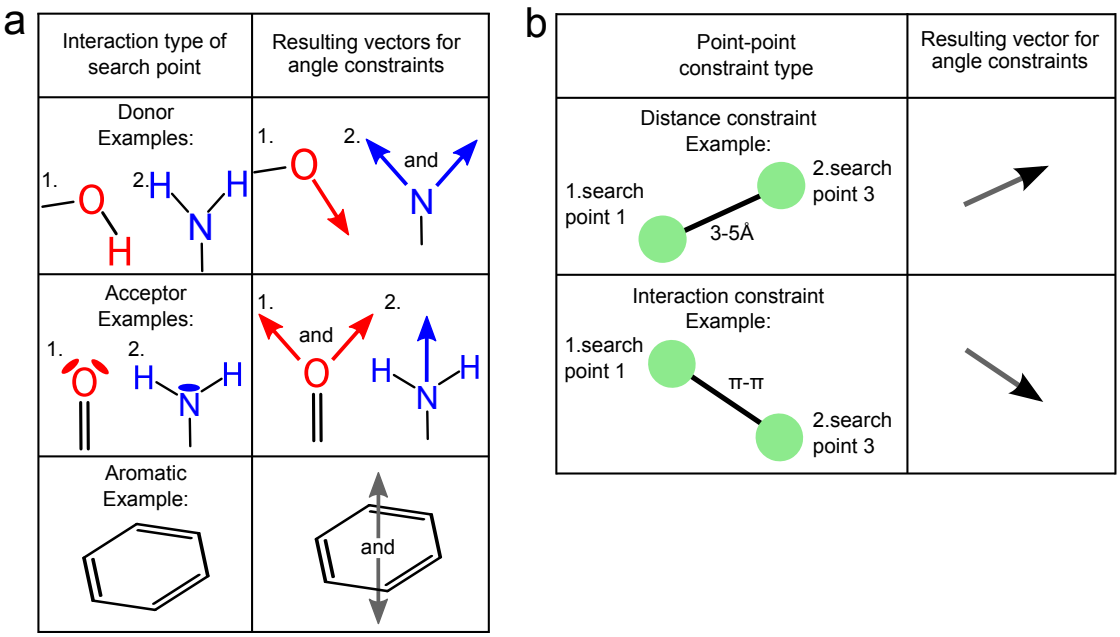


Figure 4.4.: Vectors of query objects in the Pelikan query system. Every pair of vectors can be connected with an angle constraint. a) Possible vectors of search points. b) Possible vectors of point-point constraints.

Filters for Textual and Numerical Properties

In addition to the geometrical query, numerical and textual filters can be added to a query. Filters can be defined for all properties stored in the PropertyDB database. A list of all stored properties is shown in Appendix B.5. Depending on the value type, these filters are either range filters where a minimal and a maximal allowed value is provided, or string filters. In the latter case, the given filter string is searched in the respective database entry as substring in a case-insensitive manner. Besides filters for specific properties, substructure filter for reference ligands can be defined using SMARTS patterns.

4.3.2. Calculation and Storage of relevant Data

In Pelikan, all relevant data is stored in an SQLite database. It contains all databases from the NAOMI library mentioned before, namely the MoleculeDB, the ProteinDB, the ComplexDB, and the PropertyDB. Additionally, the InteractionDB was developed in order to store data relevant for the spatial search process. In this section, first the structure of the InteractionDB is explained. Afterwards the process to build the complete database is described.

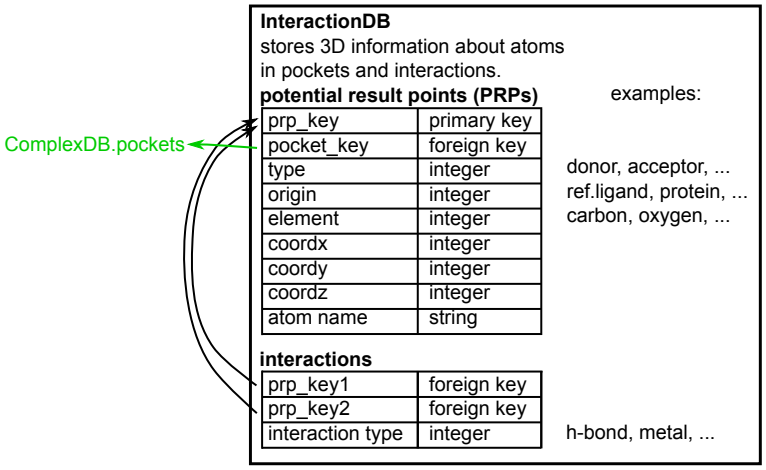


Figure 4.5: Overview over the tables of the InteractionDB used in Pelikan. Arrows indicate cross references between tables. Herein, black arrows represent cross references within the InteractionDB, whereas green arrows show cross references to the ComplexDB.

The InteractionDB

An overview about the table scheme of the InteractionDB is shown in Figure 4.5. The first table stores potential result points (PRPs) which are atoms with 3D coordinates. It is called 'PRP table'. A PRP is uniquely identified by its primary key, named prp_key. Note that the smallest valid prp_key is 1. For each PRP, the table stores its 3D coordinates, its interaction type, its element type, the molecule type it originates from (origin), and the name of its atom. In order to reduce storage space, the 3D coordinates are stored as integers. In a structural file from the PDB, there are at most six digits to represent the floating point number of the x, y, and z coordinates of an atom, respectively. Thus, each coordinate can be multiplied by 1 000 and casted to an integer by loosing at maximum the fourth decimal. The maximal integer which can be generated by this method is 999 999 000. This number can be represented using 4 bytes. According to the SQLite documentation [92], the storage of a floating point number would need 8 bytes. Hence, 12 bytes (4 bytes for each dimension) can be saved per PRP using this conversion of types. Additionally, a foreign key links back to the ComplexDB indicating the pocket the PRP is part of.

In the table 'interactions', atomic interactions are stored. An atomic interaction is always formed between two PRPs which are represented by two reference keys, prp_key1 and prp_key2, respectively. They point to the primary prp_key in the PRP table. Additionally, the type of the interaction, e.g., hydrogen bond or cation-π, is stored in this table.

Database Construction

The database can be constructed out of a collection of structural files from the PDB. Both file formats, pdb and mmCIF, are accepted. During the building process, the files are first read and interpreted using the functionality of the NAOMI library explained in section 4.2.1. As already mentioned, when reading files from the PDB using the NAOMI library, covalently

bound ligands as well as sugar chains are categorized as part of the protein. In order to resolve this shortcoming, a procedure was developed which identifies covalently bound ligands as well as ligands consisting of more than five residues. The methods is shown in Algorithm 1.

Firstly, all residues of a protein which might be a ligand are identified and classified as 'hetero residue' (see Line 3 in Algorithm 1). These are all residues which fulfill all of the following criteria:

- The residue is not a standard amino acid.
- The residue is connected to a protein chain (by definition of NAOMI) via a non-peptide bond.
- All atoms of the residue have valid 3D coordinates.

Algorithm 1 Detect additional ligands

```
1:  procedure getLigandsFromProtein(protein)
2:      newligands = empty molecule vector
3:      heteroresidues = findHeteroResidues(protein)
4:      components = groupConnectedResidues(heteroresidues)
5:      for all c ∈ components do
6:          if noChainBreakUponRemoval(c, protein) then
7:              molecule = createOneMolecule(c)
8:              addMoleculeToVector(molecule, newligands)
9:          end if
10:     end for
11:     return newligands
12: end procedure
```

After all potential ligands have been detected, connected residues are grouped into connected components by a breadth first search (see Line 4 in Algorithm 1). If no chain break is introduced by removing the complete component, all residues of one component are converted to one molecule and added to the vector of ligands. Note that the residues are not removed from the protein in order to maintain the chemical integrity of the structure. However, the respective residues in the protein are ignored in the further procedures.

Every ligand which consists of more than five and less than one hundred heavy atoms is considered a reference ligand here. These reference ligands are used to build pockets. A pocket is defined as all ligands, residues, water, and metals which have an atom-atom distance of less than 6.5 Å to any of the reference ligand's atoms. Pockets which contain no residue are discarded. Afterwards, all atomic interactions are calculated within each pocket using the NAOMI library (see Section 4.2.2).

In the last calculation step, different properties of the current data structures are determined.

Properties of the reference ligand and the pocket are calculated using the NAOMI library (see Section 4.2.8). Moreover, properties for the protein and the complete protein-ligand complex are extracted from the header section of the PDB file. In total, 39 different properties are collected for each input file. Therein, 18 properties are determined for the reference ligand, 13 for the pocket, four for the protein, and four for the protein-ligand complex. The exact properties which are calculated and their value ranges are shown in Appendix B.5.

In the next step, all calculated data is stored in the database. This includes the storage of all small molecules, protein chains, and pockets in the respective tables of the MoleculeDB, the ProteinDB, and the ComplexDB. All properties are stored in the respective tables of the PropertyDB using the reference key to either the molecule, the molecule instance, the protein, or the pocket assigned in the previous step, respectively. Each heavy atom in a pocket can be a PRP in a search and is thus stored as PRP in the PRP table of the InteractionDB. The interaction type of each atom, which is needed for the storage of a PRP, is determined using the NAOMI library (see the first step for identifying atomic interactions described in Section 4.2.2). It might happen that more than one interaction type is determined, e.g., the oxygen of a water can be donor as well as acceptor. In these cases, a separate PRP is stored for every detected interaction type. These PRPs are completely equal, except for their entry in the column 'interaction type'. The prp_key of each PRP can be seen as a unique id and will later be used to refer to a specific PRP.

For all detected atomic interactions in the pocket, both PRPs which take part in this interaction are stored in the interactions table of the InteractionDB. If more than one PRP has been entered into the database for one atom, only the PRP with the matching interaction type is used here.

4.3.3. The Triangle Descriptor

Since all atomic interactions in a pocket are stored in the database, interaction constraints can be queried directly. For distance constraints however, the 3D coordinates of all PRPs matching the respective search points of the constraint have to be processed pairwise. This is done by a self-JOIN of the table storing the PRPs. In principle, the run time of this query depends on the number of rows which have to be joined. The number of rows can be influenced by the attributes of a search point. Thus, two search points with element type 'manganese' at a specific mutual distance can be searched for much faster than two search points with element type 'carbon' because the latter appear much more often in the database than the former. In principle, a geometrical descriptor which rapidly generates a list of possible PRPs which occur in a specific geometrical form used in the query could help to overcome this problem. Different geometrical forms could be used here, e.g., a distance, a triangle, or a tetrahedron. Two main aspects have to be considered for this decision: the descriptor has to differentiate between different PRPs and the storage space should not be too large. Preliminary tests have revealed that a descriptor storing the existence of PRPs

in a specific distance to other atoms is ineffective in discriminating between different PRPs. To this end, a descriptor was developed describing the spatial surroundings of a PRP using different triangles.

In principle, the triangle descriptor is a set of bit strings. Each bit string represents the occurrence of all PRPs in a specific triangle. To this end, a set of spatial triangles was defined. For each descriptor triangle (DT), one bit string is calculated. The length of the bit strings is equal to the maximal PRP id. Hence, each position within the bit string corresponds to one specific PRP. If a PRP occurs in a specific DT, the bit at the respective position is set to one. Otherwise, it is zero. Hence, if a 3D query contains a triangle it can be mapped to the triangles used in the descriptor. If this mapping is successful, all PRPs occurring in the query triangle can be identified directly by identifying all bits which are set to one in the respective bit string.

In the following, the set of used DT is described first. Afterwards, the exact calculation of the descriptor is described.

Used triangles

Any one DT consists of three corners and three edges. One corner, in the following called the 'PRP corner', represents one specific PRP. The two other corners, in following called 'legs', represent the relative coordinates of two further atoms with respect to the PRP. The considered atom types for the legs of this triangle are:

$$Legs = \left\{ \begin{array}{l} \text{carbon protein, oxygen protein, nitrogen protein,} \\ \text{donor protein, acceptor protein, aromatic protein,} \\ \text{carbon ligand, oxygen ligand, nitrogen ligand,} \\ \text{donor ligand, acceptor ligand, aromatic ligand} \end{array} \right\}$$

The edges of the triangle correspond to the distances between the respective corners and are binned in the following groups:

$$TriangleEdges = \{\text{2.5-3.5 Å, 3.5-4.5 Å, 4.5-5.5 Å, 5.5-6.5 Å, 6.5-7.5 Å, 7.5-8.5 Å}\}$$

Using all combinations of two possible legs and three possible distance bins for the three edges of the triangle, a set of triangles is generated. Such a geometrical descriptor is only useful if it is able to discriminate between different PRPs, i.e., the bit string has no benefit if all bits are set to one. In our case, the majority of PRPs happen to occur in DTs with two 'carbon protein' legs. These DTs are therefore excluded from the descriptor. In total, 15 642 different DTs are used in this descriptor. Each DT is identified using a unique id.

Calculation of the triangle descriptor

The main aim of this step is to calculate 15 642 different bit strings d_{DTi}, one for each DT_i with i being the id of the respective DT. Since the process of database construction handles one protein-ligand complex after the other, the direct calculation of the descriptor is not feasible. To this end, the transposed descriptor is calculated first: For each PRP_j, a bit string d_{PRPj} coding for its occurrence in all DTs is determined. In principle, this descriptor could also be used to get all PRPs occurring in a specific DT by querying one specific bit in each bit string. However, experiments have shown that this procedure can be quite slow (data not shown here). Therefore, d_{PRP} is transformed after its complete calculation to d_{DT}. Thus, d_{PRP} is an intermediate step and it is destructed after the d_{DT} has been calculated.

In this section, first the calculation of all d_{PRP} is explained and the runtime of its calculation is outlined. Afterwards, the transformation of d_{PRP} to all d_{DT} is explained including a consideration of the runtime for this step.

Calculation of d_{PRP}

d_{PRP} stores the existence of a PRP in all possible DTs. d_{PRP_j} for the PRP with unique id j is defined as follows:

$$d_{PRP_j}[i] = \begin{cases} 1 & \text{if PRP with unique id j occurs in the } i_{th} \text{ DT} \\ 0 & \text{else} \end{cases}$$

$$\text{with: } 0 \leq i < 15\,642$$

This is done by calculating all DTs in which a PRP occurs using the spatial atom index of the NAOMI library (see Section 4.2.5). The procedure is described in Algorithm 2. First of

Algorithm 2 Calculate descriptor d_{PRP} for each PRP

```
1:  procedure CalcTrianglesForEachPRP(allPockets)
2:      for all p ∈ allPockets do
3:          tree = buildKD-tree(p.getAllPRPs)
4:          for all PRP_j ∈ p.getAllPRPs do
5:              allDTs = calcAllDTsForPRP(PRP_j, tree)
6:              d_PRPj = generateBitVector(allDTs)
7:              storeBitVectorInDatabase(d_PRPj)
8:          end for
9:      end for
10: end procedure
```

all, a k-D tree is built from all PRPs of one pocket in Line 3 of Algorithm 2. Afterwards, for each PRP_j, all DTs are calculated in Line 5 of Algorithm 2. To this end, seven distinct range queries on the k-D tree are performed in order to receive all neighboring PRPs within each distance of the distance bins. From this data, all possible DTs are constructed and d_{PRPj} is calculated. The final d_{PRPj} is then stored in the database using the PRP id as reference key.

Runtime analysis for calculating d_{PRP}

Herein, a tight upper boundary for the runtime of Algorithm 2 is determined. First of all, the runtime for one iteration of the outer loop in Line 2 is analyzed. Herein, the runtime of building the k-D tree in Line 3 is in $\mathcal{O}(n \log n)$, with n being the number of PRPs in one pocket [93]. The calculation of all DTs in Line 5 contains two main steps: range queries on the k-D tree and the construction of the DTs. The runtime of the former step is in $\mathcal{O}(n^{\frac{2}{3}} + m)$, with m being the number of returned points [93]. For the latter step, the distance between all pairs of resulting points has to be calculated which has a runtime of $\mathcal{O}(m^2)$. Adding these steps up leads to a complete runtime for Line 5 of $\mathcal{O}(n^{\frac{2}{3}} + m + m^2) = \mathcal{O}(n^{\frac{2}{3}} + m^2)$. This is done for each PRP. The complete runtime of the loop in Line 2 of Algorithm 2 is therefore in $\mathcal{O}(n^{\frac{2}{3}} \cdot n + m^2 \cdot n + n \log n)$. The term $n \log n$ can be neglected here because asymptotically it grows slower than $n^{\frac{2}{3}} \cdot n$. Thus, the complete runtime of the loop in Line 2 can be reduced to $\mathcal{O}(n^{\frac{2}{3}} \cdot n + m^2 \cdot n)$. It is not possible to compare the sizes of $n^{\frac{2}{3}}$ and m^2 because it depends on the size of the pocket and the distribution of PRPs in the pocket.

The complete runtime of Algorithm 2 is difficult to estimate because it depends on the number of PRPs per pocket. If most of the PRPs are in one pocket, the complete runtime would be equal to the estimated runtime for one iteration of the outer loop in Line 2. If, on the other hand, the PRPs are equally distributed over all pockets, the complete runtime would depend linearly on the number of pockets.

Calculation of d_{DT}

After d_{PRP} is calculated for each PRP, it is transformed into d_{DT}. In this process, a bit string is generated for each DT. The descriptor d_{DTi} for the i_{th} DT is defined as follows:

$$d_{DT_i}[j] = \begin{cases} 1 & \text{if } d_{PRPj}[i] == 1 \\ 0 & \text{else} \end{cases}$$

$$\text{with: } 0 \leq j < \text{max. unique id of PRP}$$

This calculation is performed by querying the respective bits from all d_{PRP} for each DT. The number of DTs is constant but for each query, each d_{PRP} has to be inspected. The runtime

of this step is thus in $\mathcal{O}(n)$ where n is the number of all PRPs.

The vector is stored in a compressed form in a specific table in the SQLite database using the unique id of each DT as primary key, respectively.

Extension of the triangle descriptor by new pockets

After a triangle descriptor has been constructed, it can be further extended. The extension mechanism is very similar to its construction. First of all, for each PRP_j, d_{PRPj} is calculated as described before. In the second step, the descriptor d_{DTi} for each DT is extended. To this end, the bit string of each d_{DTi} is extended such that the number of bits corresponds to the new maximal unique PRP id. Simultaneously, the unique ids of the new PRPs are set to one in each d_{DTi} if necessary.

4.3.4. The Search Process

During the search process, a query containing geometrical as well as textual and numerical filters is used to search for all pockets which match the given constraints. In general, a branch-and-bound strategy is applied. This means that the result space is reduced in different steps through the algorithm until a complete matching of the geometrical query is performed at the end. The process can be subdivided into six steps: (1) textual and numerical filtering, (2) detection of search points which are not involved in point-point constraints, (3) triangle descriptor request, (4) incremental search of point-point constraints, (5) hit construction, (6) SMARTS matching. Figure 4.6 schematically depicts the complete process and shows how the result space is reduced in each step. In the following, the different steps will be explained in detail.

Step 1 - Textual and Numerical Constraints

In the first step of the Pelikan algorithm, the filters for textual and numerical properties of the query are used. To this end, these filters are first divided by their property type (molecule, instance, pocket, or protein). This results in four distinct sets of filters. For each set, one query is executed, resulting in a list of the respective key type.

Besides the filters for properties stored in the PropertyDB, substructure constraints for the reference ligand can be defined as SMARTS patterns. These patterns are used in this step, to get a set of molecule keys which fulfill the SMARTS pattern using the substructure matching method of the NAOMI library.

Moreover, search points may contain substructure constraints for the reference ligands. In general, a very specific SMARTS pattern will lead to a large reduction of interim results in

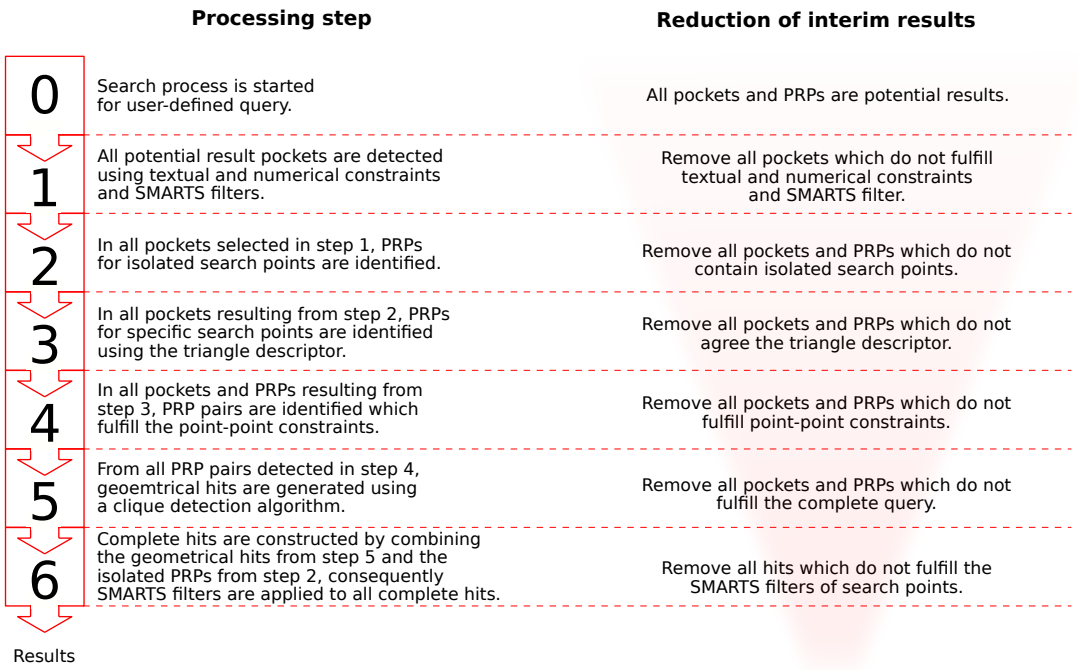


Figure 4.6.: Overview about the search process in Pelikan.

this step. An early reduction of results at this stage might reduce the time spend for the following steps. Based on this hypothesis, highly specific SMARTS patterns of search points of type reference ligand are additionally used in this step. A SMARTS pattern is defined to be highly specific here if it covers four or more atoms or if it contains other elements than carbon, oxygen, or nitrogen.

The complete first step ends with at most five different sets of keys. One set from each query of the property tables and one list of molecule keys from the SMARTS matching procedure. All resulting sets are transfered to their corresponding pocket keys. Finally, the intersection of these sets is generated which corresponds to the set of pockets which fulfill all textual and numerical constraints of the query. This set of pocket keys is transferred to the second step.

Step 2 - Querying isolated search points

In the second step of the process, results for the search points which are not involved in a point-point constraint are searched. This is done by one SQL query per search point on the table storing the PRPs. During this step, the list of allowed pocket keys is used if it has been generated in step 1. The resulting PRPs for each search point are stored and later passed to step 6. The list of allowed pocket keys is passed to step 3.

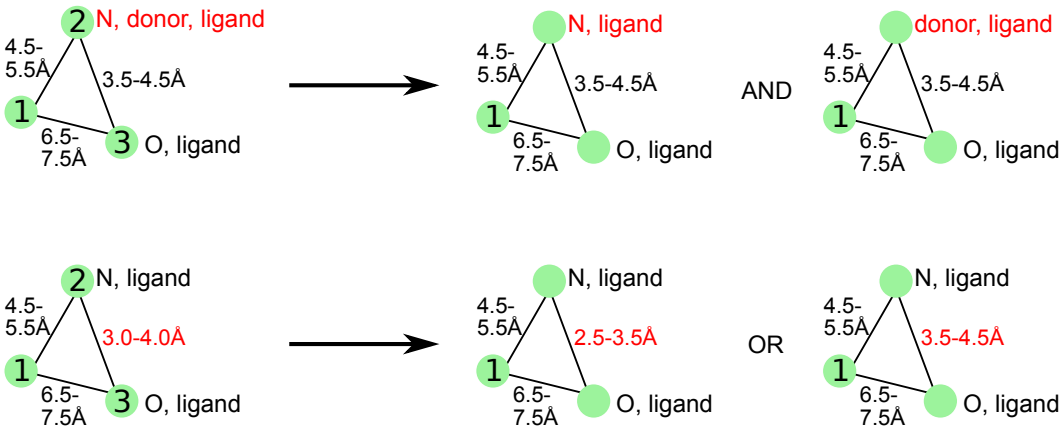


Figure 4.7.: Extraction of DTs from the 3D query and their combination. On the left, 3D queries are displayed. Search points are represented by green dots, black lines represented distance constraints. The properties highlighted in red lead to the generation of multiple DTs which are combined by logical operators. On the right, extracted DTs for search point 1 are shown. Their respective bit strings can either be combined using a logical AND or a logical OR.

Step 3 - Using the triangle descriptor

The input of the third step is a list of allowed pocket keys. In this step, the triangle descriptor is used to generate lists of PRPs for search points in the query, if possible.

Firstly, all triangles in the 3D query which are applicable to the triangle descriptor are identified. To this end, at least two corner points have to match the legs used in the triangle descriptor. Moreover, the distance ranges of the distance constraints must be within the distance range used in the triangle descriptor (2.5-8.5 Å). Then for each search point, all DTs are identified which are compatible with the query for that search point. Typically, each of these DTs fulfills only a subset of the query constraints. In order to cover all relevant constraints with the triangle descriptor the bit strings of non-contradicting DTs are combined with logical operators OR and AND. Figure 4.7 exemplary shows in which case which logical operator is used for the combination. Two bit strings are combined with a logical AND, if one search point is part of more than one DT. An example for this is show in the upper part of Figure 4.7. Herein, search point 2 is defined as nitrogen and donor from a ligand. Hence, two different DTs can be generated which both have to be fulfilled.

Two bit strings are combined with a logical OR, if at least one distance from the query spans different distance bins of the descriptor. This can be seen in the lower example of Figure 4.7. Herein, the distance range between search point 2 and 3 is defined as 3.0 to 4.0 Å. Again, two DTs can be generated from this information, but only one of the DTs has to be

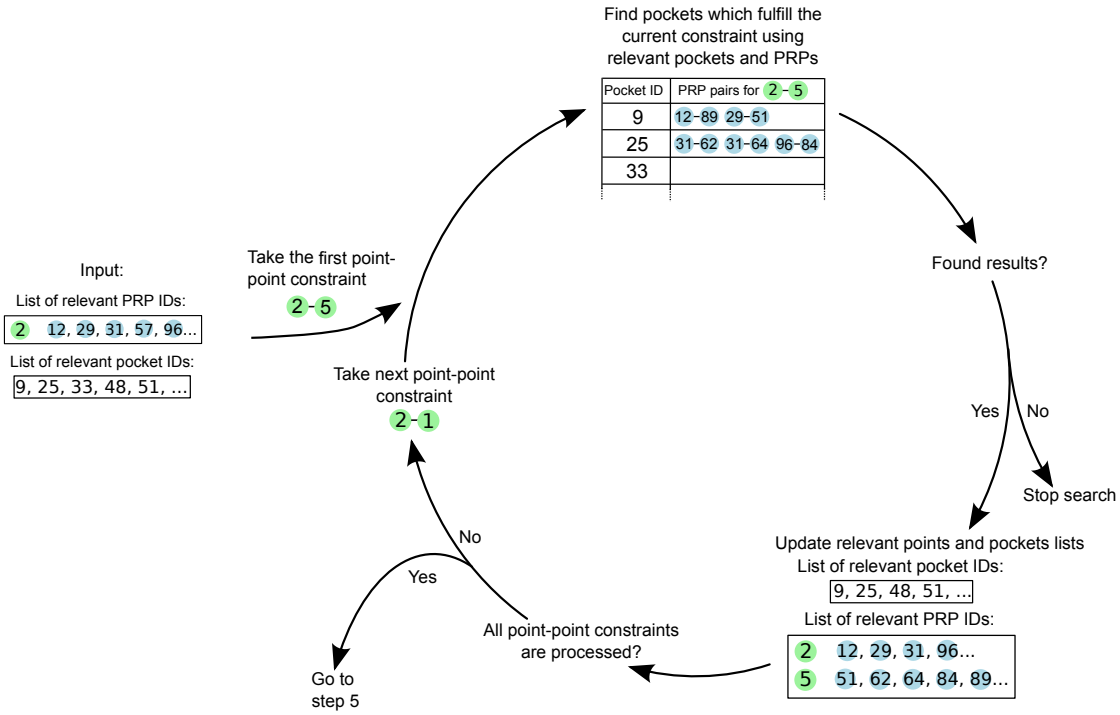


Figure 4.8.: Schematic depiction of step 4 of Pelikan's search process.

fulfilled.

During this procedure, only those PRP ids are used which are part of a pocket which occurs in the list of allowed pocket keys. At the end of step 3, sets of allowed PRP ids for the individual search points in the query have been generated, if the triangle descriptor was applicable. At the end of step 3, this list of allowed pocket keys is updated according to the detected PRPs for specific search points. This list together with the sets of PRPs are passed to step 4.

Step 4 - Sequential Querying of Point-Point Constraints

In step 4, all point-point constraints are processed in a sequential manner. This is schematically displayed in Figure 4.8. The order of this processing is chosen such that point-point constraints for which only a few results are expected are processed before those that probably have a large number of results. The number of results for a point-point constraints is the product of the estimated number of results for each search point. This estimation is done using a simple counter of elements and interaction types stored in the database.

For each point-point constraint, the loop shown in Figure 4.8 is run through. At the beginning of the cycle, all pocket and PRP pairs which fulfill the current point-point constraint,

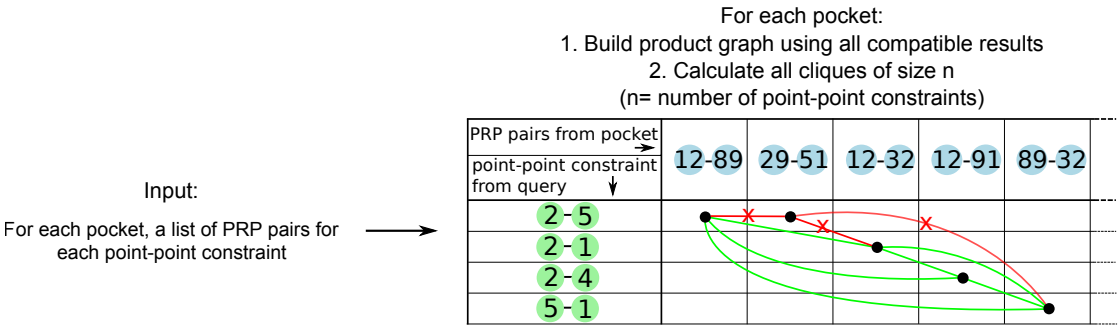


Figure 4.9.: Schematic depiction of step 5 of Pelikan's search process.

including the involved search points constraints, are identified. This is done using one SQL query. During this database query, the list of allowed pocket keys is used to only find results in relevant pockets. Moreover, if lists of possible PRPs for one or both of the search points exist, the lists are also used in the query. After a database query, the number of results are checked first. If no results are found, the process immediately stops. Otherwise, all lists are updated, which is the last step of the cyclic process. This means that the list of allowed pockets is set to the list of pocket keys detected in the previous query. Similarly, the lists of allowed PRPs for the search points involved in the previous query is set to the list of resulting PRPs, respectively. The idea behind this step is that the list are shortened after every query. This should reduce the runtime for the subsequent query.

After all point-point constraints have been handled in this step, the results are passed to the fifth step of the query process.

Step 5 - Clique calculation

The input for the fifth step are lists of PRP pairs for each point-point constraints in the query. In this step, the fulfillment of the complete query is verified. This is done with a clique algorithm on an edge-based product graph, similar to the work of Rascal [94]. Firstly, a product graph is constructed for each pocket. In Figure 4.9, the construction of the product graph is schematically displayed. Herein, a vertex is inserted in the graph for each detected PRP pair and the corresponding point-point constraint. Thus, a vertex has three attributes: first prp_key, second prp_key, and the unique point-point constraint id. Edges are added between all pairs of vertices which do not contradict each other. For example, two vertices which have the same unique point-point constraint id can never be connected by an edge. Moreover, consider a vertex which represents the match of the point-point constraints between search point 2 and 5 on the PRPs 12 and 89, respectively. This vertex could never be connected with a vertex matching search point 5 onto another PRP. Additionally, all angle constraints are checked in this step and two vertices are not connected if their calculated angle does not fulfill the required angle constraints. Each clique of size n in this graph is

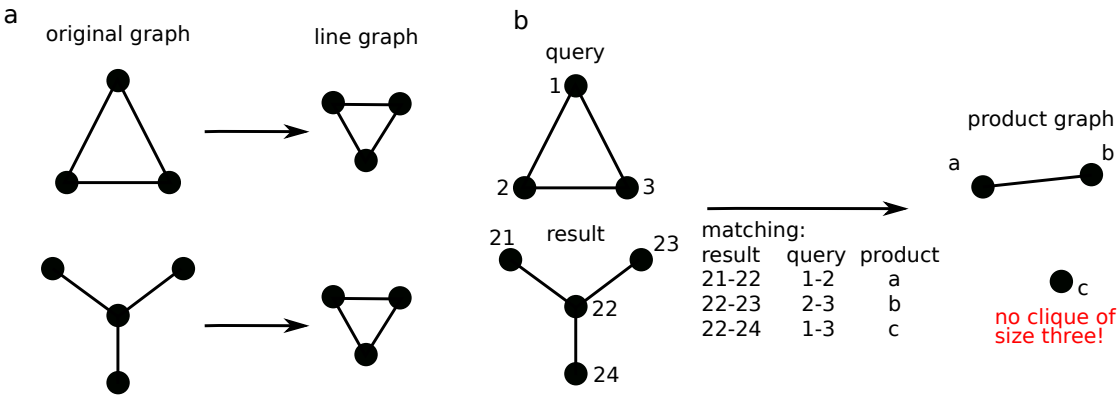


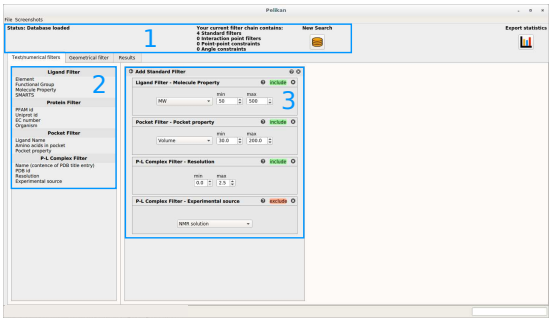
Figure 4.10.: Δ-exchange in the application of line graphs to solve the subgraph-isomorphism problem. a) Graphs are transformed to their line graphs by converting every edge to a vertex. Two vertices are connected in the line graph if they are incident on each other in the original graph. The two shown non-isomorphic graphs have isomorphic line graphs. b) Clique detection procedure in Pelikan. The Δ-exchange problem does not apply here because vertex labels are compared during product graph construction.

a valid geometrical match, with n being the number of point-point constraints. The clique detection is performed using the algorithm of Bron and Kerbosch [58] from the Boost Graph Library (http://www.boost.org/doc/libs/1_61_0/libs/graph/doc/index.html).

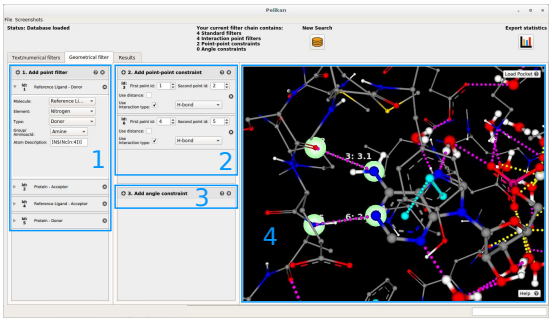
A known problem of the Rascal approach is the existence of so-called Δ-exchanges. This describes the phenomenon that two non-subisomorphic graphs can have subisomorphic line graphs. This can lead to a clique in the product graph even if there is no subgraph isomorphism between the original graphs. This problem is schematically depicted in Figure 4.10a. This problem does not apply in the clique detection procedure of Pelikan, because all search points and all PRP ids are compared during the construction of the product graph. Thus, the connectivity of the vertices is compared taking the mapping of point-point constraint ids and PRP pairs into account which excludes the occurrence of Δ-exchanges. An example is given in Figure 4.10b. Here, the vertex c in the product graph is isolated because its mapping of search points 1 and 3 to PRP 22 and 24, respectively, does not agree with the mapping of the other vertices a and b.

Step 6 - Match combination and SMARTS Matching

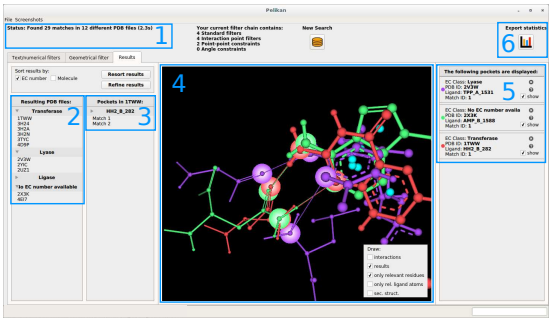
In the last step of the algorithm, the geometrical hits detected in step 5 are combined with the hits for isolated search points from step 2. All possible combinations of both results are generated and each combination is considered a valid hit. Finally, a SMARTS matching is performed for each hit using all SMARTS defined for search points. During this step, the specific labels in the SMARTS patterns describing the chemical relation between different atoms within the hit are taken into account. All hits which fulfill the SMARTS pattern build



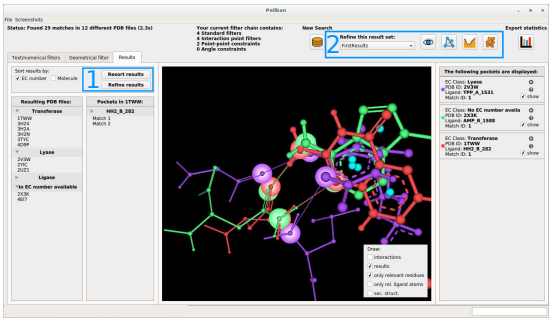
(a) First tab: textual and numerical filters can be defined.



(b) Second tab: Geometrical filters can be defined from scratch or using a pocket of interest as starting point.



(c) Third tab: Results of a search can be inspected visually.



(d) Third tab: Results can be refined by a subsequent search of a previous result set.

Figure 4.11.: Screenshots of the Pelikan GUI.

the final result set.

4.3.5. Pelikan

The method for searching interaction patterns in protein-ligand interfaces is part of the software tool Pelikan. Pelikan can either be used as command line tool or with a graphical user interface (GUI). A user guide for both is given in Appendix E. The GUI has mainly been developed in order to provide a convenient way to build databases, define queries, and to provide a comprehensive presentation of the results. In the following, the GUI is briefly introduced. Screenshots of the GUI are shown in Figure 4.11.

The GUI of Pelikan is logically divided into two parts. The upper part displays information about the current status of the program and the size of the filter chain (see Figure 4.11a, panel 1). Moreover, all buttons which can be used to start and stop a search are located in the upper part of the GUI. The lower part of the GUI contains three tabs.

The first tab is shown in Figure 4.11a. Here, different types of numerical and textual filter

can be chosen from a list (see Figure 4.11a, panel 2). The parameters of the chosen filter can be defined in panel 3 of Figure 4.11a.

The second tab, shown in Figure 4.11b, provides two different ways to generate a geometrical query. Herein, the three query components search points, point-point constraints, and angle constraints are managed in three different lists, shown in panel 1, 2, and 3 in Figure 4.11b, respectively. In these lists, query objects can be added, manipulated, and deleted. Moreover, a pocket of interest can be visualized and a query can be defined by selecting specific atoms or atomic interactions (see Figure 4.11b, panel 4).

The results of a search are displayed in the third tab of the GUI (see Figure 4.11c). In panel 1 of Figure 4.11c, the number of resulting pockets are shown. In panel 2, a list of all resulting PDB codes is given. This list is initially sorted by enzyme commission (EC) number but can also be sorted by molecule topology. A second list shows the resulting pocket of the currently chosen PDB structure (see panel 3 in Figure 4.11c). All selected pockets can be visualized and superimposed in the central view of tab three (see panel 4 in Figure 4.11c). The superimposition is performed using the NAOMI method described in Section 4.2.4. Panel 5 gives an overview about the currently displayed pockets and provides further information, e.g., the title of the respective PDB file. Statistics of the results can be exported using the button in panel 6 of Figure 4.11c.

The third tab furthermore provides a feed-back loop to refine the results of a search. To this end, a set of results can be 'remembered' by the GUI using the button in panel 1 of Figure 4.11d. When this button is pressed, a new set of buttons appears in the upper part of the GUI (see panel 2 in Figure 4.11d). These buttons can be used to perform a new search on the resulting PRPs, the pockets, or the PDB structures of a remembered result.

4.4. *NAOMI*nova - Deduction of Preferred Interaction Directions

In this section, the methods and concepts developed to search for preferred interaction directions of molecular substructures in large sets of protein structures are explained. This includes newly implemented algorithms as well as extensions to methods from the NAOMI library, already described in the Section 4.2. The main idea is to search for specific molecular substructures in a set of protein structures and collect all relevant, surrounding atoms. In the following, this molecular substructure will be called 'central substructure'. The relevant atoms in its vicinity will be called 'partner points'. The developed method works with complete protein structures including all chains and all small molecules defined in a file from the PDB. Due to this decision, the method is able to deduce preferred interaction directions for internal atomic interactions within the protein as well as for atomic interactions between protein and ligand. The method provides two possibilities to analyze the collected data. First of all, distributions of partner points around central substructures can be inspected

visually. Secondly, a link to the original structure will be provided for each partner point, called 'backlink'. This way, structural characteristics of a partner point distribution can be traced back to their origin.

The methodology mainly comprises the following aspects:

- A precise definition of molecular substructures and their representation (see 4.4.1).
- Calculation and storage of relevant data from a large set of protein structures (see 4.4.2).
- A query concept which represents a central substructure and surrounding interaction partners (see 4.4.3).
- Efficient mining of the stored data for specific substructure queries (see 4.4.4).

In the following, these aspects will be explained in more detail.

4.4.1. Definition of Substructures

In *NAOMI*nova, a substructure consists of five different items:

- **Template molecule**: A molecule with 3D coordinates. From this molecule, specific atoms can be selected which represent the substructure. These atoms are called 'template atoms'. There are two possibilities how this molecule can be defined: (1) a molecule file with 3D coordinates can be uploaded or (2) a molecules can be defined using a SMILES string [85]. In the latter case, the 3D coordinates are generated using Unicon [95].
- **Template atoms**: A set of connected atoms from the template molecule. These atoms will later be used as a template structure. All detected occurrences of a substructure will be superimposed onto the template atoms.
- **SMARTS**: A textual description of the substructure using the SMARTS language [89]. This description is used to detect all occurrences of the substructure in the used data set using the SMARTS matching method from the NAOMI library (see 4.2.9). The SMARTS concept used here is schematically depicted in Figure 4.12. In *NAOMI*nova, the SMARTS is logically divided into two parts: A fragment part and a surrounding part. The fragment part describes the actual atoms the SMARTS is matching. Herein, every atom corresponds to exactly one atom from the template atoms. Hence, the fragment part has to match the selected atoms from the template molecule. After a substructure has been found in the database, the detected atoms and the template atoms are assigned to each other based on their matching to a specific atom of the fragment part. Wildcards (SMARTS symbol '*') and OR combinations of atoms (e.g., [C,N]) are in principle allowed here. However, these ambiguities might

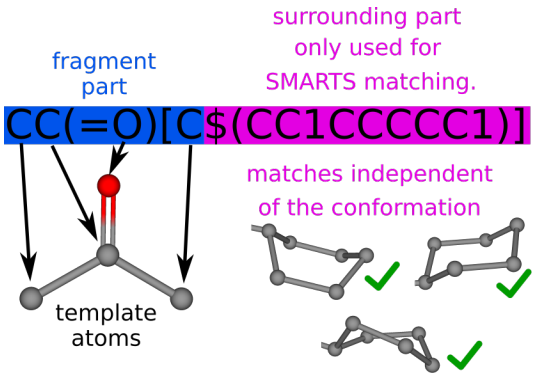


Figure 4.12: The SMARTS concept used in *NAOMI*nova. The fragment part is highlighted in blue, the surrounding part is highlighted in purple. Every atom described in the fragment part has to match exactly one atom of the template atoms. The surrounding part describes the chemical environment of an atom an matches independent of the conformation.

lead to structural changes due to different bond angles or bond lengths. Hence, a meaningful superimposition of the detected and the template atoms might not be possible in these cases.

The chemical environment of every atom in the fragment part can be described in more detail using the surrounding part. This can be done using a recursion in the SMARTS language, indicated with a '$'. Within this part, the complete syntax of the SMARTS language can be used. Even though the complete SMARTS will be used to find occurrences of the substructure in the data, the surrounding part is not relevant for the superimposition of the atoms. Hence, the surrounding part does not have to match the template atoms, it is only used to describe the surrounding of the atoms of interest. To this end, this part matches independent of the exact conformation as shown in Figure 4.12.

- **Name**: A unique name. Will be used as id of the substructure.

- **EDIA**$_{min}$: A minimal allowed EDIA value. Only atoms which fulfill the required minimal EDIA value of a substructure will be stored in the database.

4.4.2. Calculation and Storage of relevant Data

The data calculated here is stored in an SQLite database. The MoleculeDB, ProteinDB, and ComplexDB from the NAOMI library (see section 4.2.7) are used to store the relevant information about proteins and ligands. Additionally, partner points are stored in the database. These tables are governed by the PartnerPointDB which is described in the following section. The process of detecting central substructures and the storing of partner points will be described thereafter.

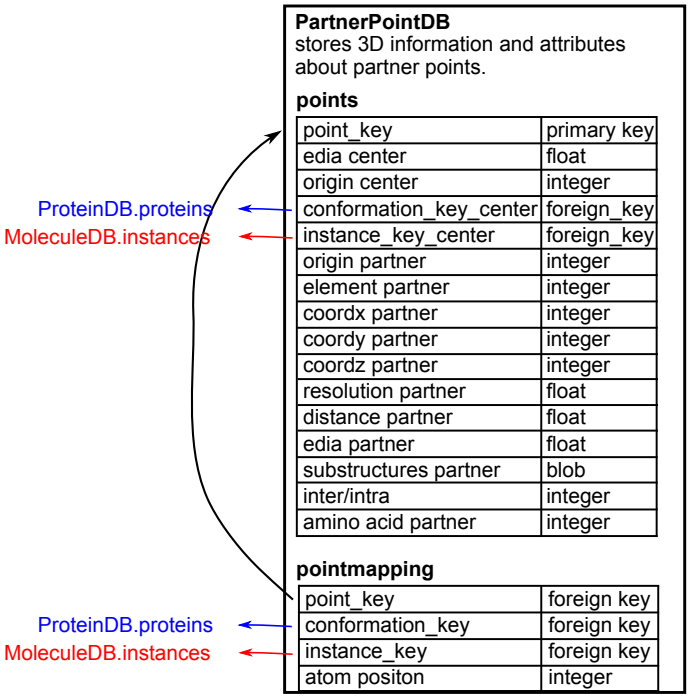


Figure 4.13: Schematic depiction of the two tables of the PartnerPointDB. Arrows indicate cross references between tables. Herein, black arrows represent cross references within the PartnerPointDB, whereas blue and red arrows show cross references to the ProteinDB and to the MoleculeDB, respectively.

The PartnerPointDB

The PartnerPointDB stores all partner points and their attributes. Its main purpose is to quickly find all partner points fulfilling the query constraints and to provide a link to the original atom for each partner point. Figure 4.13 shows a schematic depiction of the tables which belong to the PartnerPointDB. The table 'points' is generated for each added substructure. In this table, the main attributes of a partner point are stored, e.g., its 3D coordinates and its distance to the central substructure. Similar to the storing of the PRPs in the InteractionDB (see 4.3.2), the coordinates of a partner point are stored as integer values. Each partner point is identified via a unique id, called 'point_key'. This id is also used in the second table, called 'pointmapping'. Here, a reference to the original molecule and the exact atom position is stored for each partner point. These values are required if the original structure in which a partner point was detected should be displayed using the backlink functionality. Moreover, attributes of the central substructure for which a partner points was detected are stored in the table 'point', e.g., the combined EDIA value and a link to the original molecule. The first table 'points' is required for each filtering process, explained in Section 4.4.3. The second table of the PartnerPointDB 'pointmapping' is only used if a backlink to the original structure of a partner points is requested.

Database construction

The construction of a database consists of three steps. In a first step, PDB files are entered to the database. As for Pelikan, the two file formats '.pdb' and '.mmCIF' are accepted. The procedure works for apo protein structures as well as for holo structures, i.e., structures of protein-ligand complexes. Apo structures are simply handled as protein-ligand complexes without ligand. In the following, the procedure of constructing a *NAOMI*nova database is therefore explained only for structures of protein-ligand complexes. The perception of molecules from PDB files is explained in Section 4.2.1. Upon entering the structures into the database, the proteins and molecules are stored in the tables of the ProteinDB and the MoleculeDB. Secondly, substructures have to be registered. During this process, the SMARTS of each substructure is checked for validity and the name is checked for uniqueness. Afterwards, the central substructure is stored in a specific database table. The table contains one column for each of the five attributes of a central substructure. The name of the central substructure is used as primary key. In the third step, the partner points are detected and stored. A schematic overview about this process is presented in Algorithm 3.

In principle, all hits of each substructure in each protein-ligand complex are searched using a first loop over all protein-ligand complexes and a second loop over all substructures. In Line 6 of Algorithm 3 the SMARTS matching step is performed. Then, for each hit, the combined EDIA $EDIA_{hit}$ of the detected atoms and the affine transformation for its superimposition onto the substructure's template atoms is calculated. If $EDIA_{hit} < EDIA_{min}$ or if the RMSD of the transformation is larger then 0.2, the hit is discarded. Otherwise, partner points in the vicinity of the detected atoms are collected in Line 16. Within this function, all atoms of the protein-ligand complex which have a minimal distance to any of the detected atoms of below 4.5 Å are collected. Moreover, only atoms from the organic subset are considered as partner points here, except for carbon and phosphorus. These are atoms with an element type of oxygen, nitrogen, sulfur, halogen, or metal. This is done using the spatial atom index from the NAOMI library (see Section 4.2.5). To this end, a k-D tree is created for all relevant atoms of each protein-ligand complex already in Line 4 of Algorithm 3. Partner points are then detected using range queries on the k-D tree. Finally, for each partner point whose EDIA value is larger or equal than $EDIA_{min}$, all necessary attributes are calculated. These attributes are mainly the values stored in the table 'points' in the PartnerPointDB. After all central substructures have been detected in one protein-ligand complex, the data calculated so far is stored in the database.

The upper boundary of the algorithm's runtime is on the one hand determined by the runtime of the SMARTS matching algorithm. The problem of subgraph-isomorphism is known to be NP-complete [96] and thus all known algorithms have an exponential runtime relative to the size of the used graphs. On the other hand, all steps in the loop starting in Line 7 of Algorithm 3 are performed for each detected hit. If the absolute runtime is examined, it might therefore be possible, that the loop in Line 7 has a longer runtime than the SMARTS

Algorithm 3 Detection and Storage of Partner Points

```
1: procedure detectAndStorePartnerPoints(database)
2:     for all plc ∈ database.getPLCs do                ▷ plc = protein-ligand complex
3:         Let data be an empty list of partner points with attributes
4:         tree = buildKD-tree(plc.getAllOrganicAtoms)
5:         for all sub ∈ database.getSubs do            ▷ sub = central substructure
6:             hits = findAllHits(sub.SMARTS, plc)
7:             for all hit ∈ hits do
8:                 EDIA_hit = getCombinedEdia(hit)
9:                 if EDIA_hit < sub.EDIA_min then
10:                    continue with next hit
11:                end if
12:                rmsd, trafo = getTransformation(hit, sub.templateAtoms)
13:                if rmsd > 0.2 then
14:                    continue with next hit
15:                end if
16:                partnerPoints = getAllCloseAtoms(hit, tree)
17:                partnerPointsTrans = transform(partnerPoints, trafo)
18:                for all pp ∈ partnerPointsTrans do
19:                    EDIA_pp = getEdia(pp)
20:                    if EDIA_pp ≥ sub.EDIA_min then
21:                        pp_data = getAttributesForPartnerPoint(pp)
22:                        data.append(pp_data)
23:                    end if
24:                end for
25:            end for
26:            addDataToDatabase(database, data)
27:        end for
28:    end for
29: end procedure
```

matching step in Line 6 in case a large number of hits is detected. Which of those steps is finally more relevant for the overall runtime highly depends on the used SMARTS and the used protein-ligand complex.

Due to possible symmetries of a central substructure, an ambiguous behavior of Algorithm 3 might occur. Firstly, linear symmetry occurs in cases where the substructure's template atoms resembles a straight line, e.g., for the SMARTS 'O=C'. After transformation, the partner points are randomly distributed around the symmetry axis. No further projection of the partner points is performed here. Secondly, substructure symmetry might occur, e.g., for the SMARTS 'O=C(C)C'. In these cases, the SMARTS matching algorithm would detect several matchings on the same set of atoms. In principle, each hit is treated individually and the partner points for each hit are collected. However, if the detected atoms of a hit have been found before, all partner points are marked with a flag in the database. Thus, it is possible to distinguish between all hits of a substructure and only its arbitrary first hits.

Database extension

Besides building a new database, databases can also be extended by protein-ligand complexes as well as by new substructures. In case of protein-ligand complexes which are added to a database, the same procedure as explained in Algorithm 3 is used. The only difference is in Line 2. In this case, the loop covers only the newly added protein-ligand complexes.

In case new substructures are added to an existing database, also Algorithm 3 is executed. However, the loop in Line 5 uses only the new set of substructures. Besides this difference, an additional step is required when an existing database is extended by new substructures. During the database construction, attributes are calculated for each partner point in Line 21 of Algorithm 3. One part of this step is another SMARTS matching step: the substructures a partner points is part of are determined. For this step, all currently registered substructures are used. If a substructure is added after other substructures, it might be possible that an earlier detected partner point is part of the new substructure. Hence, after a new substructure has been added, the detected atoms are temporarily stored for each protein-ligand complex. If one of these atoms has earlier been detected as partner point, its attributes are updated.

4.4.3. The Query

In *NAOMI*nova, a query can be used to get a set of partner points with specific attributes. For such a query, a central substructure is mandatory. All other attributes of the query are optional. Table 4.4 gives an overview about all possible optional parameters, their value types, and their default values.

Attributes	Value type	Possible values	Default value
Central molecule type	set of integer	any subset of ligand, protein, metal, water	all possible values
Partner substructure	string	any substructure, all substructures stored in the database	any substructure
Partner molecule type	set of integer	any subset of ligand, protein, metal, water, metal	all possible values
Partner element	integer	any element, oxygen, nitrogen, sulfur, halogen, metal	any element
Partner amino acid	integer	any type, any used amino acid type (see table B.4	any type
Partner connection	integer	any connection, inter, intra	any connection
Partner location	integer	any type, aromatic, aliphatic ring, conjugated system	any type
Partner distance	float	any range between 0.0 Å and 4.5 Å	0.0-4.5 Å
Resolution	float	any range between 1.0 Å and 2.5 Å	1.0-2.5 Å
Minimal EDIA	float	-1 and any value between 0.0 Å and 1.2 Å (-1 means that no minimal EDIA value is used)	-1

Table 4.4.: Attributes of the query for partner points in *NAOMI*nova.

In addition to the parameter 'partner distance', the exact atom from the substructure's template atoms can be chosen. If this is done, the distance is measured only to this atom. Otherwise, the shortest distance to any of the substructure's atoms is used.

4.4.4. The Search Process

The search process only involves one database query since all partner points for a central substructure are stored in one table. The central substructure defined in the query gives the table name for the query. All other chosen parameters of the query are used in the WHERE clause of the query.

4.4.5. *NAOMI*nova

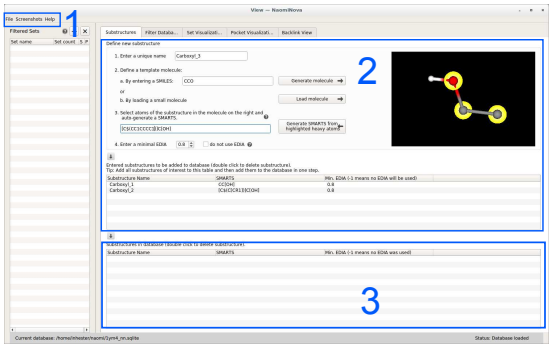
The method for deducing preferred interaction directions of molecular substructures is part of the software tool *NAOMI*nova. It can be used as a command line tool as well as in combination with a GUI. The usage of both is explained in detail in Appendix F. In the following, the main aspects of the GUI will be introduced.

The first step when working with *NAOMI*nova is to create or load a database. This can be done via the context menu of the window (see Panel 1 in Figure 4.14a). Conceptually, the GUI is separated into five different tabs. In the first tab, new substructures can be defined and added to the database (see Panel 2 of Figure 4.14a). Moreover, substructures which are already part of the database can be seen in a table view and can be deleted (see Panel 3 of Figure 4.14a). After substructures have been added, a query can be composed. This can be done in the second tab of the GUI, shown in Figure 4.14b. For filtering the collected partner points, a central substructure is mandatory (see Panel 1 in Figure 4.14b). Further attributes of the central substructure as well as of the partner points can be defined and a query can be submitted to the database (see Panel 2 in Figure 4.14b). After a query has been conducted, a set containing all received partner points is created.

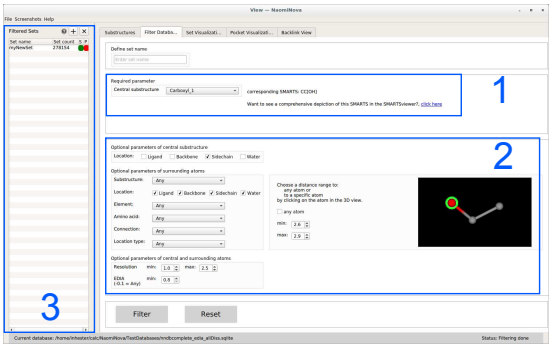
A list of all currently existing sets is shown on the left of the GUI (see Panel 3 in Figure 4.14b). These sets can be visualized in the two following tabs: the set visualization and the pocket visualization. In the former, a set of partner points can be visualized around the central substructure (see Panel 1 in Figure 4.14c). The partner points can be displayed as spheres or as a density grid. The calculation of the density grid is explained in the following section. Distances and angles can be measured by selecting any set of atoms and partner points. If the partner points are displayed as spheres, the color of each sphere encodes the element type of the partner point. The visualization and measurement can be controlled by different buttons in the upper part of the GUI (see Panel 2 of Figure 4.14c). The performed measurement can also be extended to all partner points by clicking on the button shown in Panel 3 in Figure 4.14c. The results are then displayed in a normalized histogram (see Panel 5 in Figure 4.14c). The process of histogram normalization is explained in Section 'Histogram normalization' below. Moreover, the filter properties used to generate the currently visualized set are displayed in Panel 4 of Figure 4.14c.

In the pocket visualization tab, sets of partner points can be visualized within a protein-ligand interface (see Panel 1 of Figure 4.14d). To this end, a protein and a ligand can be uploaded using the functionality provided in Panel 2 in Figure 4.14d. In the visualization, an atom can then be selected. If this atom is part of a substructure, the sets of partner points around that substructure can be displayed inside the pocket. Herein, partner points clashing with atoms from the protein or the ligand can be hidden. As in the set visualization tab, distances and angles can be measured here (see Panel 3 of Figure 4.14d)

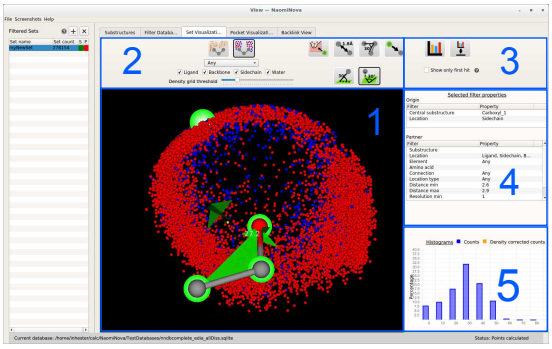
Finally, in the fifth tab, the original structures of the partner points can be visualized. To



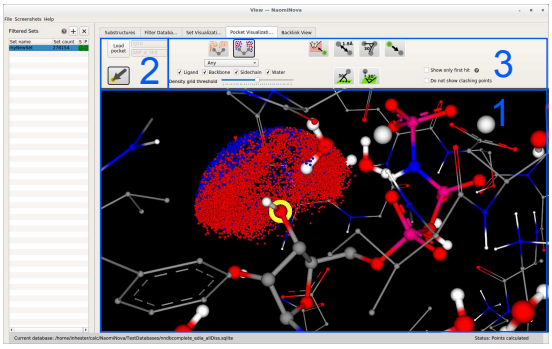
(a) First tab: Definition and addition of substructures.



(b) Second tab: Filtering for relevant partner points.



(c) Third tab: Visualization of partner points from a set around the central substructure.



(d) Fourth tab: Visualization of partner points from a set inside a protein-ligand interface.

Figure 4.14.: Screenshots of the *NAOMI*nova GUI.

this end, a partner point in the set or the pocket visualization has to be selected using the backlink functionality of *NAOMI*nova.

Density grid calculation

In the visualization tabs of the GUI, the partner points of a set can either be displayed as spheres or as a density grid. The density grid is an evenly spaced, rectangular grid with a value for the density of partner points assigned to each grid point. In the following, the calculation of this grid is explained in detail.

1. A grid covering the whole range of partner points with a spacing of 0.4 Å is calculated. The minimum coordinates of the grid (x_{min}, y_{min}, z_{min}) are set to the minimum over

all partner point coordinates:

$$\begin{aligned} x_{min} &= \min_{\forall pp \in P} x_{pp} \\ y_{min} &= \min_{\forall pp \in P} y_{pp} \\ z_{min} &= \min_{\forall pp \in P} z_{pp} \end{aligned}$$

Where P is the set of all partner points and x_{pp}, y_{pp}, and z_{pp} are the x-, y-, and z-coordinate of the point pp, respectively.
Starting from this minimal point, the grid is build with a spacing of d = 0.4 Å along all three dimension. The receiving maximal grid point (x_{max}, y_{max}, z_{max}) is then given by:

$$\begin{aligned} x_{max} &= \lceil \left(\max_{\forall pp \in P} x_{pp} - x_{min} \right) \cdot \frac{1}{d} \rceil \cdot d + x_{min} \\ y_{max} &= \lceil \left(\max_{\forall pp \in P} y_{pp} - y_{min} \right) \cdot \frac{1}{d} \rceil \cdot d + y_{min} \\ z_{max} &= \lceil \left(\max_{\forall pp \in P} z_{pp} - z_{min} \right) \cdot \frac{1}{d} \rceil \cdot d + z_{min} \end{aligned}$$

2. For each partner point, the eight grid points which correspond to the smallest box surrounding a partner point are identified. This box is represented by the minimal corner with the coordinates: (x_i, y_i, z_i) and its maximal corner with the coordinates (x_{i+1}, y_{i+1}, z_{i+1}). x_i is defined as:

   $x_i = \lfloor \frac{x_{pp} - x_{min}}{d} \rfloor \cdot d + x_{min}$, with x_{pp} being the x-coordinate of the partner point.

   x_{i+1} is defined as $x_i + d$. y_i, y_{i+1}, z_i, and z_{i+1} are calculated analogously.

3. The relative positions of the partner point with respect to the three dimensions inside the box $relpos_x$, $relpos_y$, and $relpos_z$ are calculated using linear interpolation. $relpos_x$ is defined as follows:

$$relpos_x = \frac{x_{i+1} - x_{pp}}{d}$$

   The values $relpos_y$ and $relpos_z$ are calculated analogously. All three values are between 0 and 1. A $relpos_x$ value of 1 means that the $x_{pp} = x_i$, whereas a $relpos_x$ value of 0.5 means that the x_{pp} is in between x_i and x_{i+1}.

4. In the last step, a value for each of the eight corners of the box is calculated representing its distance to the current partner point. In total, the assigned values for one partner point sum up to 1.

The following equations are used to calculate the values for each corner of the box. The used method is called trilinear interpolation,:

$$\begin{aligned}
\text{value}_{x_i,y_i,z_i} &= \text{relpos}_x \cdot \text{relpos}_y \cdot \text{relpos}_z \\
\text{value}_{x_{i+1},y_i,z_i} &= (1 - \text{relpos}_x) \cdot \text{relpos}_y \cdot \text{relpos}_z \\
\text{value}_{x_i,y_{i+1},z_i} &= \text{relpos}_x \cdot (1 - \text{relpos}_y) \cdot \text{relpos}_z \\
\text{value}_{x_i,y_i,z_{i+1}} &= \text{relpos}_x \cdot \text{relpos}_y \cdot (1 - \text{relpos}_z) \\
\text{value}_{x_{i+1},y_i,z_{i+1}} &= (1 - \text{relpos}_x) \cdot \text{relpos}_y (1 - \cdot \text{relpos}_z) \\
\text{value}_{x_i,y_{i+1},z_{i+1}} &= \text{relpos}_x \cdot (1 - \text{relpos}_y) \cdot (1 - \text{relpos}_z) \\
\text{value}_{x_{i+1},y_{i+1},z_i} &= (1 - \text{relpos}_x) \cdot (1 - \text{relpos}_y) \cdot \text{relpos}_z \\
\text{value}_{x_{i+1},y_{i+1},z_{i+1}} &= (1 - \text{relpos}_x) \cdot (1 - \text{relpos}_y) \cdot (1 - \text{linear}_z)
\end{aligned}$$

For each density grid point (x_k, y_m, z_n), these values are summed over all partner points. The final sum gives the color intensity of the displayed density grid. Using a control, a threshold can be determined defining the displayed density grid points.

Histogram normalization

In *NAOMI*nova, distances and angles can be measured for all partner points. These values are displayed in a histogram. *NAOMI*nova works with 3D data and thus, the bin values for spherical angles and distances have to be normalized by the volume they represent. As an example, a distance bin represents a spherical around the central substructure. The larger the diameter of this shell, the more volume it represents and the more atoms could be positioned within this shell.

For the normalization of distance histograms, the volume of a bin is calculated using the following equation:

$$V_{dist,k} = \frac{4}{3}\pi(r_{i+1}^3 - r_i^3)$$

Where r_i is the minimal and r_{i+1} the maximal distance of bin k. The value of the normalized distance bin is the count of partner points for the respective bin divided by $V_{dist,k}$.

Analogously, the volume of the kth angle bin is calculated using the following equation:

$$V_{angle,k} = \frac{2}{3}\pi|\cos(\alpha_i) - \cos(\alpha_{i+1})| \cdot R$$

Where α_i is the minimal and α_{i+1} the maximal angle of bin k. R is the distance cut-off value. The value of the normalized angle bin is then determined by dividing the count of partner points for the respective bin by $V_{angle,k}$.

5

Evaluation Strategy and Experiments

In this chapter the data sets and experiments used to evaluate the methods presented in this thesis are described. In general, three aims are pursued with these experiments. First of all, the correctness of the methods should be verified. Secondly, the performance of the algorithms is quantified in order to demonstrate its strengths and limitations. This includes the performance of the used databases. Thirdly, the value of the developed method are deduced by means of comparison to other existing software solutions. However, this last aspect could only be assessed for Pelikan since no comparable tool was available for *NAOMI*nova. In Section 5.2, the experiments performed with the Pelikan method are described. The experiments performed to validate the method used in *NAOMI*nova are presented in Section 5.3. The results of these experiments are shown in Chapters 6 and 7, respectively.

5.1. Hardware

Throughout this work, two different hardware settings are used to conduct the experiments. (1) A standard PC equipped with an Intel I5-4570 (3.2 GHz) processor, 16 GB of main memory, and a Samsung 950 pro PCIe solid state drive (512 GB, model nvme) with a btrfs file system (herein denoted as SSD). (2) A standard PC equipped with an Intel I5-4570 (3.2 GHz) processor, 16 GB of main memory, and a Seagate Desktop hard disk drive (500 GB, model ST500DM002) with an ext4 file system (herein denoted as HDD). Both platforms run with a standard configuration of a Linux openSUSE 13.1 distribution.

5.2. Pelikan - Experiments

5.2.1. Systematic Correctness

The correctness of the Pelikan method is verified here by analyzing the search results. In principle, there can be two types of wrong results. On the one hand, false positive results can be returned. This means, that there are results which do not fulfill the search criteria. False positive results can be detected by comparing each resulting hit with the attributes of the used query. On the other hand, there can be false negative results. In this case, there is a result in the data set but the method is not able to find it and thus it is not part of the result set. The strategy to detect this kind of error here is to specifically search for a known 3D pattern which is part of the database.

In order to check for both types of errors, a random set of 200 PDB files has been used to build a database. Afterwards, a query has been generated for every pocket in each protein-ligand complex. For query creation, the pocket has been randomly translated and rotated in 3D space as a first step. Then, a set of eight atoms has been collected. This set contained randomly picked atoms, each had a maximum distance of 8.5 Å to at least one other atom in the set. For each atom, a search point has been created using its element type, its origin, and its interaction type. For each atom pair with a mutual distance below 8.5 Å, a point-point constraint has been created with a random range around the measured atom-atom distance. Two angle constraints have been added to the query by randomly picking four point-point constraints. An angle constraint with a randomly chosen range around the measured angle has been introduced between the first and the second, and the third and the fourth picked point-point constraint. Finally, one property of the pocket, the protein, and the complex has been picked randomly, respectively. For each property, a textual/numerical constraint has been added to the geometrical query.

The query has then been used to find results on the database. Finally, each resulting match has been compared with the query in order to detect false positive results. Moreover, the pocket which has been used to create the query had to be part of the results in order to exclude false negative results.

5.2.2. Data Sets

For the following experiments, different sets of protein-ligand complexes are used. In general, the PDB has been used as source for protein-ligand complexes. For all experiments, files in pdb format have been used. From all files in the PDB (accessed November 2016), a set containing all protein-ligand complexes which contain at least one reference ligand has been compiled. This set will be named '$PDB_{complete}$' in the following. It contains 69 481 different files. Moreover, five sets of protein-ligand complexes with increasing sizes have been

created. To this end, 2 000, 4 000, 8 000, 16 000, and 32 000 files were randomly picked from $PDB_{complete}$, respectively. In the following, these sets will be named by their size.

5.2.3. Database Construction

In order to search for 3D atom patterns using the Pelikan method, a database has to be created first. In this experiment, the performance of the database construction process is analyzed. This includes the runtimes for creating a database and an analysis of the database's structure and the required disk space. In this experiment, a database is created for each of the data sets.

The construction process can be divided into two parts. In the first part, the protein-ligand complexes are added to the database. This step includes the determination of the pockets, the calculation of PRPs and non-covalent interactions, and their storage. In the second part, the stored PRPs are used to calculated the triangle descriptor. The runtime for first part clearly depends on the number of files added to the database. It is however not clear, if there is a constant amount of time needed to add one protein-ligand complex or if the run time for adding one protein-ligand complex increases with database size. This question should be answered by constructing a database using $PDB_{complete}$. Herein, the required runtime has been measured for each PDB file.

In a second experiment, the runtime for the construction of the triangle descriptor is assessed. To this end, the triangle descriptor has been calculated for databases of different sizes.

Existing databases can be expanded by adding new protein-ligand complexes. During this process, the new complexes are first added to the database. This process does not differ from the addition of protein-ligand complexes during database construction. Afterwards, the triangle descriptor is expanded. The runtime dependency of this step has been determined by expanding the different databases by 10% of their protein-ligand complexes.

5.2.4. Triangle Descriptor

The triangle descriptor has been introduced in order to be able to quickly reduce the search space. In general, such a descriptor can only be efficient if it is able to discriminate between elements. If, for example, all PRPs were part of all DTs of the descriptor, the descriptor would not help to reduce the search space. On the other hand, such a descriptor should generalize in order to reduce its size. For example, if there would be only one PRP for each DT, the number of different DTs is probably too large. Ideally, the PRPs are evenly distributed over the different DTs. Moreover, the amount of DTs a PRP is part of should not be too large and not too small. To this end, the triangle descriptor has first been characterized by means of counting the number of PRPs per DT and the number of triangles per PRP. In other

words, the bit density of all bit strings d_{DTi} and d_{PRPi} has been analyzed.

The ability of the triangle descriptor to accelerate a query is difficult to access because it highly depends on the query. For example, if the query contains a triangle with large distance ranges, the search points could be part of several DTs from the descriptor. Depending on the number of PRPs generated from this, the descriptor could be not very beneficial in this scenario. On the other hand, if a query contains a triangle with well defined search points (e.g., oxygen, donor, and reference ligand), the search points have to be part of several DTs of the descriptor. In this case, the descriptor would probably be more efficient.

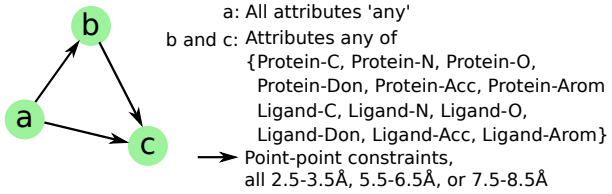


Figure 5.1.: Schematic depiction of the triangle queries used to determine the efficiency of the triangle descriptor. A green dot represent a search point. The arrow represent distance constraints. The attributes of the search points and the distance ranges are given in the text.

In order to analyze the mean acceleration potential of the triangle descriptor, the speed-up for hitting exactly one bit of the descriptor has been recored. The test queries used here are displayed in Figure 5.1. Each test query consists of three search points which are mutually connected by point-point constraints. One search point has no specifically defined attributes (see search point 'a' in Figure 5.1). The two other search points exactly correspond to one type of the leg types used in the triangle descriptor (see search points 'b' and 'c' in Figure 5.1). The set of test queries has been compiled by combining all possible attributes for these two search points. Additionally, three different distances for the point-point constraints have been used: 2.5-3.5 Å, 5.5-6.5 Å, and 7.5-8.5 Å. In total, the set contains 231 different queries. For each query, the gained speed-up has been calculated by using the following equation:

$$\text{speed-up} = \frac{\text{complete runtime, not using the triangle descriptor}}{\text{complete runtime, using the triangle descriptor}}$$

5.2.5. Query Retrieval Speed

The experiments described in this section should assess the runtime of the search process. The main purpose is to determine the time consumption of the different steps during the process. Moreover, the dependence of the runtime on different aspects of the algorithm should be assessed. In this set of experiments, three different influences have been tested:

- Attributes of the query: Topology, geometrical constraints, and properties of the query objects.
- The size of the database.

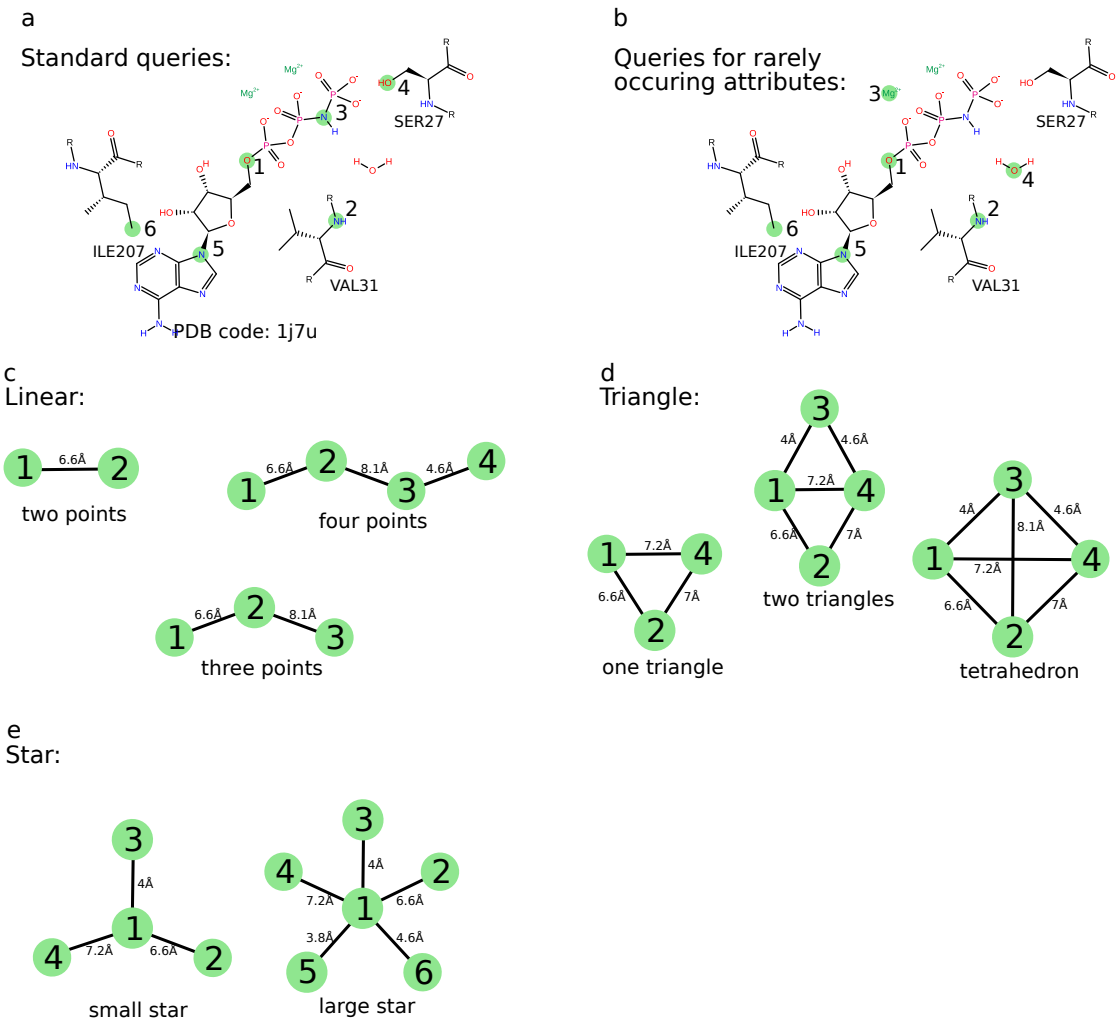


Figure 5.2.: Schematic depiction of different queries used to assess the query speed of Pelikan. a) Atoms from the binding site of ANP in PDB file 1j7u used in the queries are labeled with green spheres and their ids. b) Atoms from the binding site of ANP in PDB file 1j7u used in the queries with rarely occurring attributes are labeled with green spheres and their ids. c) Query topologies of linear queries. d) Query topologies of triangular queries. e) Query topologies of star-like queries. This figure is adapted from [77]. Reprinted (adapted) with permission from [77]. Copyright 2017 American Chemical Society.

- The hardware used during the search process.

In order to test these aspects, a set of test queries has been created. In general, all queries have been created such that they match atoms in the pocket of ligand ANP in the protein-ligand complex 1j7u [97] in order to make sure that at least one result is detected for each query. The used atoms are shown in Figure 5.2a and b. In the standard form, a query consists only of search points and distance constraints. The attributes of the search points are the

element type and the origin of the respective atom in 1j7u. The range of the distance constraints is the distance between the respective atoms in 1j7u $\pm$ 0.5 Å. In total, eight different standard queries exist. They are separated into three different topologies: linear, triangle, and star. For each topology, two or three different shapes exist, respectively. All shapes are shown in Figure 5.2c, d, and e. Starting from the standard queries, other queries have been constructed by changing one query attribute in different steps. For example, the influence of the search point properties has been tested by comparing the standard queries with queries where additional properties of the search points are used. In the first step, this has been the interaction type of each search point. In the second step, all possible attributes of the search point have been set to the respective properties of the matching atom in 1j7u. The exact queries are listed in Appendix C.

The influence of the database size has been determined by executing the same queries on database with different sizes. Moreover, the influence of the used hardware has been tested by comparing runtimes of the same queries on the SSD and the HDD settings.

5.2.6. Comparison with Relibase

In the last experiments, the Pelikan method has been compared with Relibase. Relibase has been chosen as competitor because it also focuses on protein-ligand complexes and it is able to search for precise geometrical queries at an atomic level. The focus of this experiment is the comparison of the search capabilities of 3D patterns. The definition of the 3D query slightly differs between the tools. Thus, for comparison geometric queries have been developed which can be used for both tools. The design of these queries has mainly been restricted by Relibase which is not able to work with intra-molecular distance and angle constraints. These features can only be used in Relibase+. Moreover, Relibase does not support the search for atomic interactions. To this end, three different queries have been designed which contain substructure constraints of the ligand, as well as of the protein. Both substructures are connected with distance constraints. The used queries are shown in Figure 5.3.

Due to the different design methods, the exact queries used for each tool slightly differ. In Relibase, molecular structures have to be drawn in 2D. For each structure, its original molecular structure, e.g., protein or ligand, has to be annotated. For pairs of atoms in the 2D view, distance constraints can be added. The resulting query almost looks like the examples displayed in the left panel of Figure 5.3. For Pelikan, the search points with their attributes and the point-point constraints with their distance ranges are shown in the right panel of Figure 5.3. For each query, the two different queries have the same chemical and geometrical meaning. Thus, the same hits should result from a search.

For query design and query execution with Relibase, the web service provided from the Cambridge Crystallographic Data Center (CCDC, http://relibase.ccdc.cam.ac.uk/) has been

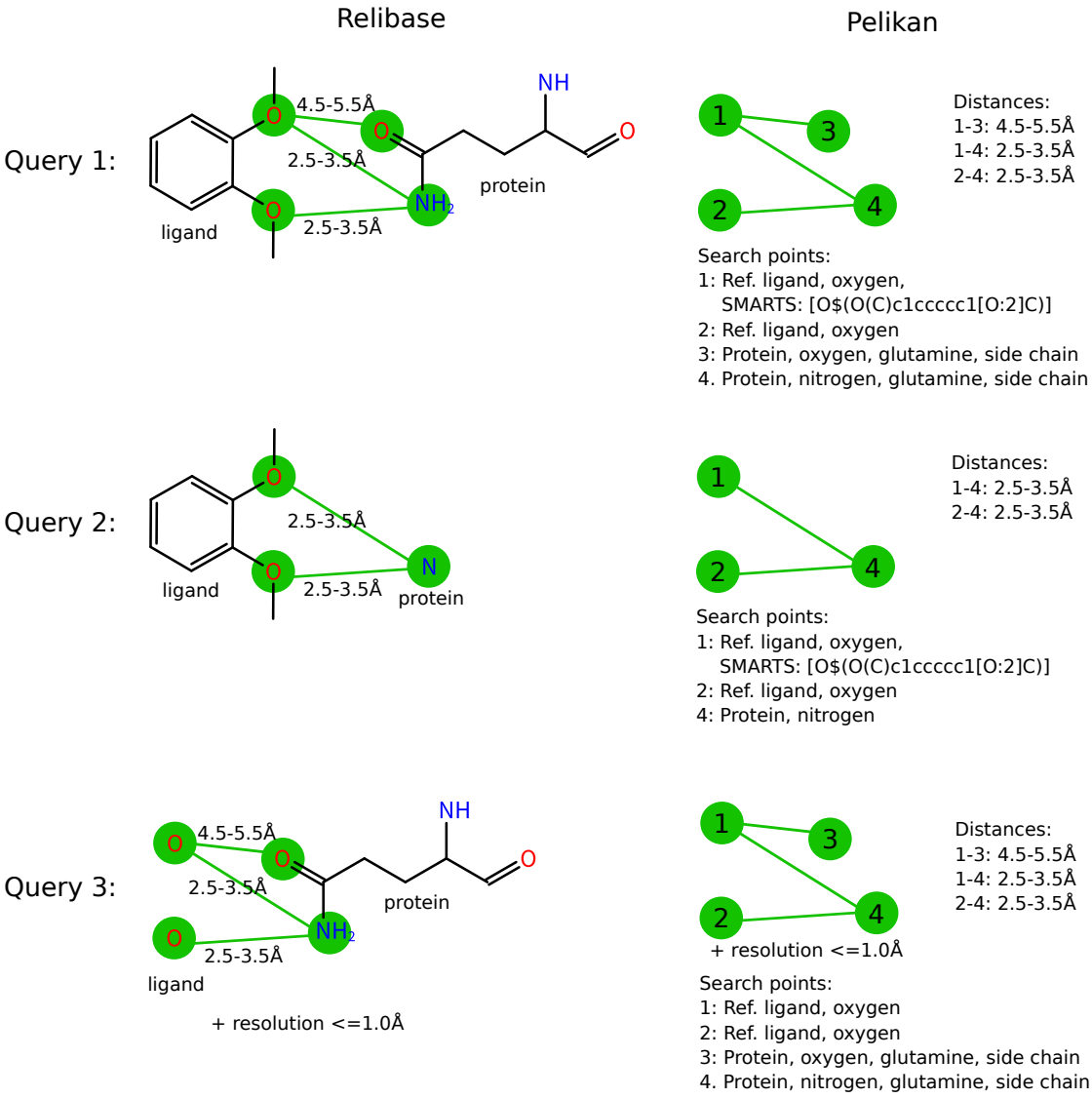


Figure 5.3.: Schematic depiction of three different queries used to compare Relibase and Pelikan. The queries used for Relibase are shown on the left. Here, structures for the ligand as well as for the protein are drawn as 2D molecular structures. Distance constraints are indicated with green line. The corresponding queries used for Pelikan are shown on the right. Here, search points are indicated as green dots and distance constraints are depicted as green lines.

used between march and june 2017. All test queries have been executed with Pelikan using the SSD hardware settings.

For all of these queries, the runtime and the exact results have been compared between Relibase and Pelikan.

5.3. *NAOMI*nova - Experiments

The typical workflow of *NAOMI*nova is the following: First a database out of protein structures of interest is created. Afterwards, substructures are registered and added to the database. Finally, the database can be mined for detected partner points around the substructures using different filters. The most important aspect is the correctness of this approach. Moreover, the runtimes of the different steps within the workflow are important for its ease of use. These aspects will therefore be tested here. The exact performed experiments are describes below. The results and their discussion can be found in Chapter 7.

5.3.1. Systematic Correctness

As for the Pelikan methods, two different errors can occur here: false positive and false negative results. In order to check for their existence, a set of 200 randomly chosen protein structures has been compiled. For each of the protein structures, a database has been constructed. Afterwards, three substructures have been defined by randomly choosing three atoms. For each atom, all neighboring atoms up to a depth of two bonds have been combined to one substructure. For each substructure, a SMARTS has been created. Together with an arbitrary name, the substructure has been registered and added to the database. Finally, an oxygen or nitrogen in the vicinity of the substructure has been randomly picked and was used to define the parameters of a query. To this end, its element and its distance to the substructure has been used. Among all partner points received with this filter, the accordance with the filter criteria has been checked in order to detect false positive results. Moreover, the atom which has been used to determine the filter criteria has been searched within the result set. Thereby, the existence of false negative results is checked.

5.3.2. Data Sets

The benchmarking of the different steps within the *NAOMI*nova workflow have been performed using different sets of protein structures. First of all, all structures from the PDB with a resolution better than 2.5 Å and for which an electron density file was available have been collected (accessed December 2016). In the following, this set will be called '$PDB_{2.5}$'.

It contains 56 807 unique PDB files. From this set, random sets containing 2000, 4000, 8000, 16 000, and 32 000 PDB files have been created, respectively. These sets have been used in order to determine how the runtime changes over a growing set of data. For each PDB file, the corresponding electron density file has been downloaded beforehand.

5.3.3. Database Construction

The runtime of constructing a database has been assessed using the $PDB_{2.5}$ data set. Herein, the time needed to add each PDB file has been recorded. The main interest is to determine an average runtime and to investigate whether this runtime changes during the process of database construction. Another important aspect here is to determine the most time consuming step.

In addition, the disk space required for a *NAOMI*nova database has been assessed by constructing databases for all used data sets. The main question here is to compare the increase of the disk space to the increase of the data sets. In theory, a linear dependency is expected here.

5.3.4. Adding Substructures

The process of adding substructures to a *NAOMI*nova database contains a SMARTS matching step, the superimposition of the matching atoms to the template substructure, and the collection of partner points in the vicinity of the matching atoms. As already mentioned in Section 4.4.2, the runtime of these steps depends on the used SMARTS pattern and on the number of hits detected for each SMARTS pattern. In principle, a more complicated SMARTS pattern requires more time for the SMARTS matching step but less hits are probably detected. On the other hand, a simple SMARTS would result in a fast SMARTS matching step but a large number of hits are detected. In order to verify these theoretical considerations, the following three different substructures have been added to the database:

- Unique name: Hydroxyethyl 1 – SMARTS: CC[OH]
- Unique name: Hydroxyethyl 2 – SMARTS: [C$(C[CR1])]C[OH]
- Unique name: Hydroxyethyl 3 – SMARTS: [C$(CC1CCCCC1)]C[OH]

For all three substructures, the $EDIA_{min}$ has been set to 0.8 and the template molecule has been defined using the SMILES 'CCO'. The SMARTS of these substructures all share the same fragment part but differ in their surrounding part. For these substructures, the runtime for the adding process on databases containing the different sets of PDB files have been recorded and compared.

5.3.5. Filtering

When working with *NAOMI*nova, the runtime of the filtering step is a very important parameter. In the typical workflow, a database is constructed once and all substructures of interest are also added once. Afterwards, several filter steps can be used to analyze the distribution of partner points. The runtime for this step strongly depends on the database and it is probably longer the more results are detected. In order to verify this hypothesis and to assess the runtime behavior of the filter step, three different filters have been used to query partner points from the three substructures added in the preceding experiments (see Section 5.3.4). The three queries are defined as follows:

1. Central substructure = current substructure, all other attributes are set to 'any'.
2. Central substructure = current substructure, element type partner point = oxygen, all other attributes are set to 'any'.
3. Central substructure = current substructure, element type partner point = oxygen, location of partner point = ligand, all other attributes are set to 'any'.

The queries differ in their precision. The first query demands all detected partner points of each substructure. The second and the third query should reduce the number of resulting partner points by asking for a specific element or molecule type of the partner points.

6

Results and Discussion of Pelikan

In this section, the results of the experiments for the Pelikan method (described in Section 5) are presented and discussed. The experiments cover the analysis of the Pelikan database, an analysis of the triangle descriptor efficiency, a benchmarking of the retrieval system for 3D queries, and a comparison with Relibase. If not otherwise stated, the experiments are performed using the SSD hardware settings (see Section 5.1 for the exact definition). Herein, a standard PC equipped with a solid state drive is used. The technical details of this platform are described in 5.2.5. All experiments reported here are performed after clearing the random access memory (RAM) cache. Thus, no accelerations due to cached database pages are to be expected.

In order to avoid confusion between complete queries exposed to Pelikan and queries defined in SQL and executed on the database, the latter will be explicitly called 'database queries' (db-queries) in the following. The term 'query' will always refer to a complete Pelikan query.

6.1. Database Construction

In this experiment databases for all datasets are created. An overview about the disk space used by these databases and the number of stored pockets and PRPs is shown in Table 6.1. The complete disk space of the database linearly growths with the number of protein-ligand complexes. About half of the size of each database is occupied by the triangle descriptor table. On average, one protein-ligand complex contains 3.8 pockets and 996 PRPs. Four different aspects of the database construction are analyzed in Figure 6.1. Figure 6.1a presents the runtimes for adding a single protein-ligand complex to the database without the calculation of the triangle descriptor for subsequent slices of 10 000 PDB files. For all data slices, the runtimes highly fluctuate as the lower and upper quartile of the values enclose a range of 4 to 6 s (see data slice 0-10 000 and data slice 50 000-60 000 in Figure 6.1a). Most probably, the fluctuation are a result of the different sizes of the individual complexes. 50% of the files can be added within 4 s, as indicated by the median (purple line in Figure 6.1a). However, there are many upward outliers which can be inferred from the fact that the mean is significantly

Database name	Size [GB]	Size of triangle descriptor table [GB]	Number of pockets	Number of PRPs	Size per complex [MB]
2 000	2.5	1.3	7 777	2 130 064	1.28
4 000	4.7	2.3	14 853	3 927 658	1.20
8 000	9.6	4.7	30 134	8 005 259	1.23
16 000	19	9.5	59 768	15 725 073	1.21
32 000	40	20.0	122 599	32 915 764	1.28
$PDB_{complete}$	84	42.0	264 930	69 718 823	1.24

Table 6.1.: Sizes of different Pelikan databases. The database name stands for the number of entered protein-ligand complexes in the database. The database '$PDB_{complete}$' contains 69 481 different protein-ligand complexes.

larger than the median (compare yellow square and purple line in Figure 6.1a). These outliers correspond to protein-ligand complexes with a large number of atoms. Overall, there is no large increase in runtime with increasing data. Thus, the runtime for adding protein-ligand complexes stays constant with increasing size of the database.

Figure 6.1b displays the breakdown of the computation time to the individual process steps. Even though the standard deviations are quite large, one can clearly see that the calculation of the pocket properties with the DoGSite algorithm and the calculation of hydrogen positions with Protoss are the most time-consuming steps in the process. The sole writing of calculated data into the database takes less than 10% of the complete runtime.

After protein-ligand complexes have been added to the database, the triangle descriptor is calculated. The runtime for this step is shown in Figure 6.1c for different databases. It can be seen that the computation time depends linearly on the database size (correlation coefficient of linear regression is 0.999). This result is in agreement with the theoretical considerations of the algorithm (see Section 4.3.3). It was concluded there that the runtime depends the distribution of the PRPs over the different pockets. The linear runtime behavior indicates that the PRPs are uniformly distributed over the different pockets.

The runtime for extending the triangle descriptor is shown for all test databases in Figure 6.1d. Again a linear dependency between the database size and the runtime can be observed (correlation coefficient = 0.999).

Taken together, a database containing $PDB_{complete}$ can be build within eight days (190 h). Therein, 36.5 h are spend for triangle descriptor calculation. The expected workflow here is to build a database once which can be used for different queries. If needed, protein-ligand complexes can be added later by using the extension procedure. During the development, the runtime for database construction was therefore considered to be less important than the retrieval time of queries.

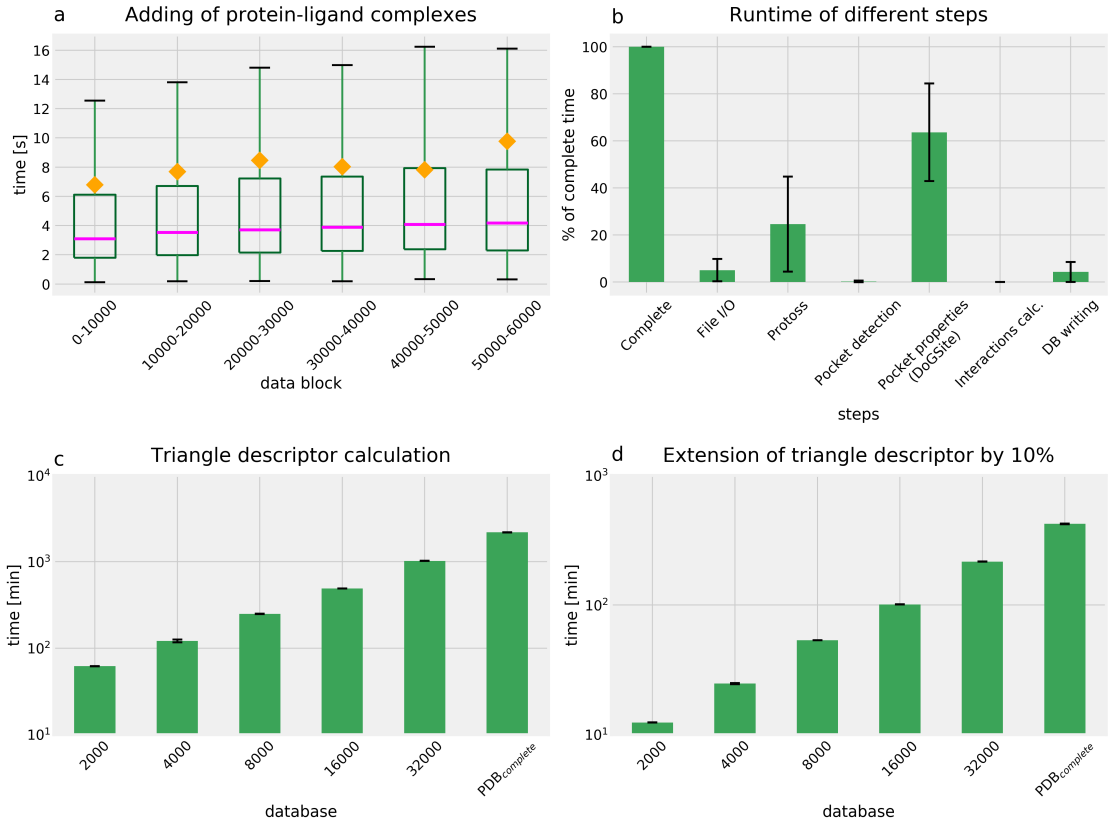


Figure 6.1.: Performance of the database construction process in Pelikan. a) Runtimes for adding one protein-ligand complex to a database shown in subsequent slices of data. Mean values are shown in yellow, the median is given as pink line. The box represents the lower and upper quartile of each data slice. The whiskers range from 5% to 95%. b) Runtime of different steps of the complete process of adding a protein-ligand complex to a database. Mean percentages of the complete runtime and standard deviations of three independent runs are shown. c) Runtime of the calculation of the triangle descriptor on different databases. Mean values and standard deviations of three independent runs are shown. d) Runtime of the extension of the triangle descriptor by 10% on different databases. Mean values and standard deviations of three independent runs are shown.

If the runtime was to be optimized, it would be promising to optimize the first step of the procedure in which the protein-ligand complexes are added to the database as this takes much more time than calculating the triangle descriptor. Within this step, the calculation of pocket properties using the DoGSite algorithm has been determined as the most time consuming step. Hence, a runtime optimization of this step would have great impact on the complete database building process. Moreover, the analysis shows that the entry of values into the database only takes 10% of the overall runtime. The overall runtime could therefore be reduced if all calculation steps for the PDB files are executed in parallel and if all resulting values are stored in a file. Afterwards, all values could be entered to the database in one step.

The size of a database containing $\text{PDB}_{complete}$ is about 84 GB. As stated on the SQLite homepage [92] (accessed February, 2017), the size limit for an SQLite database is 140 TB. Thus, the size of the Pelikan database is technically not a problem. However, since multi-user operations are not as well managed as in server-based databases [98], every user has to prepare a database by himself or has to copy it. In this scenario, a database of 84 GB is not very convenient. The triangle descriptor uses 50% of the complete database size whereas the remaining size is used for storing the protein-ligand complexes and the PRPs. The overall database size could therefore be reduced with two strategies: reduction of the triangle descriptor and a more efficient storing of the protein-ligand complexes. Concerning the latter aspect, there is only little potential as the stored data structures are already optimized and compressed if possible. In the further analysis of the triangle descriptor (see Section 6.3), it becomes clear that it could indeed be reduced by some triangles which do not affect the retrieval time for queries.

A similar descriptor with a triangle shape is also used by other software tools in the context of drug design, namely Recore [99] and Pharmer [100]. Here, geometric properties of small molecules are stored in triangles. In both tools, the triangles are stored as points in 3D. Therein, each side length corresponds to one coordinate. Using a tree data structure for the storage of these points, fast access to specific triangles is achieved by range queries. This approach has also been tested here but has proven to be not applicable. The size of the k-D tree data structure was ten times larger than the current table storing the triangle descriptor. Unfortunately, the required disk space for storing the triangles is neither given for Recore nor for Pharmer. Both tools work on small molecules and use a smaller number of atoms as triangle corners. Hence, it might be possible that the number of triangles is much smaller than in this application.

Another possibility is to store the triangles in a FastBit index [101]. This is an open source data processing library which stores data as compressed bitmap indices. Schlosser and Rarey have already shown that this library provides an efficient data handling for triangles in the context of structure-based drug design [102]. The only disadvantage of this approach would be the fact that the FastBit index is stored in a separate file and thus two different files would be required for Pelikan.

It has been confirmed that the large size of the triangle descriptor table does not influence the search speed for queries which do not use the triangle descriptor. This was concluded from experiments where the retrieval times for queries without triangles on a normal Pelikan database and on a database without triangle descriptor table were compared (see Figure D.1 in Appendix D).

6.2. Systematic Correctness

The experiment described in section 5.2.1 was performed independently three times. Neither false positive nor false negative results were detected.

The detection of false positive results used here is a very straightforward experiment. For all results, the agreement with the used query is checked. Detecting false negative results is not that trivial. For such a test, one has to know all correct results. These expected results can then be compared with all retrieved results. Unfortunately, such a test set of geometrical queries and expected results has never been compiled for this application scenario. Moreover, comparable tools in this area did not publish their methods for the verification of correctness. We therefore decided to implement a procedure where only one result is known. However, due to the random procedure of constructing a query, the size of the used data set, and the independent repetitions, even the correct detection of one expected result strongly points to a correctness of the method. However, there might still be rare cases in which the method is not correct which have so far not been detected by this test. In order to detect those, the developed test could be performed on a much larger data sets and could be repeated much more often.

6.3. Triangle Descriptor

In this section, the characteristics of the triangle descriptor and its ability to accelerate 3D queries is tested. To this end, the density of the descriptor is inspected first. Later, a set of triangle queries is used to asses the speed-up factors achieved by the triangle descriptor. These queries are called Δ-queries. The exact construction of these queries and the definition of the speed-up factor are explained in Section 5.2.4.

6.3.1. Descriptor Density

Ideally, each PRP should be contained only in a small number of DTs. At the same time, all PRPs should be equally distributed over the different DTs. As a consequence, each DT then contains the same small number of PRPs. In this scenario, the result space could immediately be reduced to a small number of PRPs if one DT is detected in a query. Each

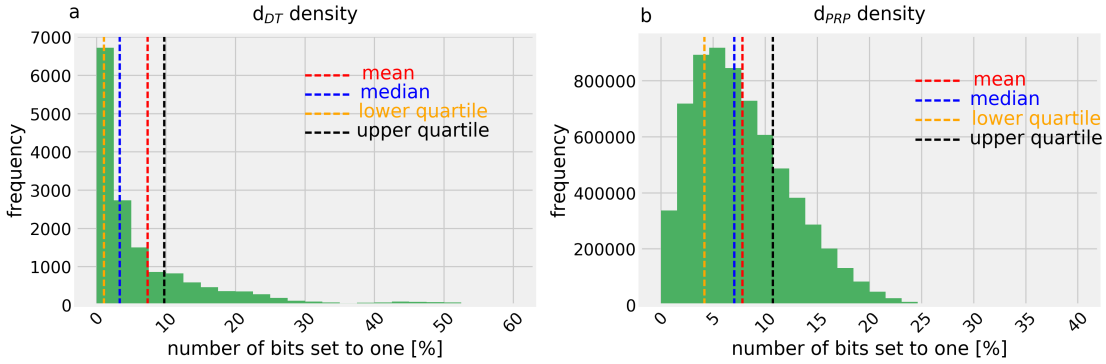


Figure 6.2.: Triangle descriptor density. a) Histogram showing the number d_{DT} density. This is the number of bits set to one in all d_{DTi} as a percentage of the bit string length. This describes the number of PRPs which occur in a DT as percentage of the complete number of PRPs. b) Histogram showing the d_{PRP} density. This is the number of bits set to one in all d_{PRPi} as a percentage of the bit string length. This describes the number of DTs a PRP occurs in, in percentage of the number of all DTs.

DT is then equally effective in this reduction. In this section, the density of the triangle descriptor is evaluated in comparison to the described optimal descriptor. To this end, two different properties are inspected: (1) Which fraction of PRPs are contained on average in a DT? This is equal to the percentage of bits set to one in d_{DT} and will be referred to as 'd_{DT} density'. (2) In how many DTs does a PRP occur? This count is equal to the number of bits set to one in d_{PRP} and will be referred to as 'd_{PRP} density'.

In Figure 6.2 both properties of the descriptor from the database $\text{PDB}_{complete}$ are displayed. The total count of PRPs in this database is 69 718 823. The triangle descriptor contains 15 542 different DTs. The d_{DT} density of all d_{DTi} is shown in Figure 6.2a. In total, there are 220 triangles for which no PRP has been found. Hence, for these DTs, $d_{DTi} = 0$. At most, 57% of the bits are set to one in a d_{DTi}. Moreover, 75% of all d_{DTi} contain less than 10% of the PRPs, as indicated by the location of the upper quartile in Figure 6.2a.
Figure 6.2b shows the d_{PRP} density. Only 251 PRPs do not occur in any DT. Hence, there are 251 bit positions in all d_{DTi} which are always zero. At maximum, one PRP occurs in about 6 299 different DTs. This is equal to about 40% of all DTs. That means that at most 40% of the bits are set to one in a d_{PRPi}. In total, 75% of the PRPs appear in less than 11% of the DTs, as indicated by the upper quartile in Figure 6.2b.

These numbers show that in principle the triangle descriptor is able to differentiate between the PRPs. There are no PRPs which occur in all DTs and there is only a small number of PRPs which occurs in no DT at all. Additionally, there is no DT in which all PRPs occur and only a small number of DTs are never found. However, within one query, more than one triangle might be involved. Since these triangles can be combined with AND and OR in an arbitrary query it is difficult to estimate a factor by which the number of PRPs are reduced by the triangle descriptor.

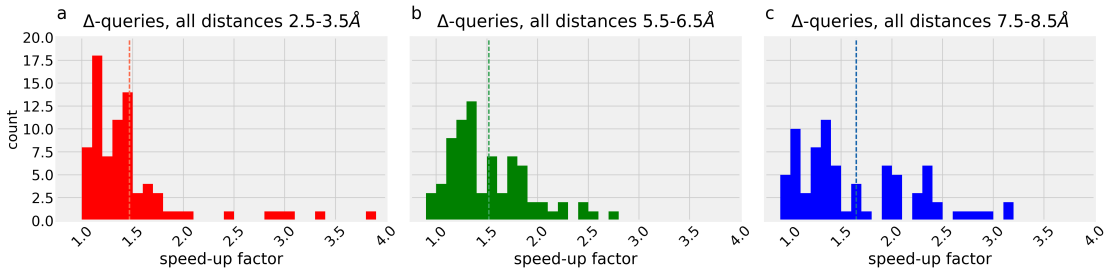


Figure 6.3.: Speed-up for Δ-queries using the triangle descriptor. Each histogram shows the speed-up factor for 77 different Δ-queries achieved by using the triangle descriptor. The vertical dotted line indicates the mean value for each plot. a) Speed-up factors for Δ-queries with side lengths of 2.5-3.5 Å. b) Speed-up factors for Δ-queries with side lengths of 5.5-6.5 Å. c) Speed-up factors for Δ-queries with side lengths of 7.5-8.5 Å.

6.3.2. Speed-up

In the next experiment, the speed-up gained by the triangle descriptor is measured using a set of triangle test queries. The construction of these test queries and the overall test set up is described in Section 5.2.4. The experiment was performed on the database 8 000. Figures 6.3a, b, and c show the speed-up factors achieved by the triangle descriptor for the three different groups of Δ-queries. These groups only differ in their distance constraint ranges which are 2.5-3.5 Å, 5.5-6.5 Å, and 7.5-8.5 Å, respectively. Overall, the smallest achieved speed-up factors are 1.02, 0.99, and 0.98 for the three different distance ranges, respectively. That means that the triangle descriptor never slows down the retrieval time drastically. The maximum recorded speed-up factor is 3.8 (see Figure 6.3a). On average, a speed-up of about 1.5 is reached here.

In the next step the exact reason for the speed-up is analyzed in more detail. With regard to the search procedure, the only steps which could be accelerated by the triangle descriptor are steps 4 and 5 of the search process (see Figure 4.6). In step 4, three different db-queries are generated for each Δ-query. One db-query is executed for each distance constraint. In step 5, the results for all distance constraints are combined to complete hits and verified using a clique-detection algorithm. The overall effect of the triangle descriptor therefore depends on the percentage of runtime spent for steps 4 and 5 of the search procedure. The amount of time spent in steps 4 and 5 relative to the complete runtime without using the triangle descriptor is shown in Figure 6.4. Two correlations can be observed: The percentage of runtime spent in step 4 negatively correlates with the overall runtime. Hence, the faster the Δ-queries, the more time is required for step 4 on average. For queries with a runtime below 2000 s, step 4 even seems to be the most time consuming step with a share of more than 60% (see Figure 6.4a). In some cases, almost 100% percent of the runtime is spent in this step. On the other hand, a positive correlation exists between the overall runtime and the percentage of time spent in step 5. At maximum, 78% of the complete runtime is spent in

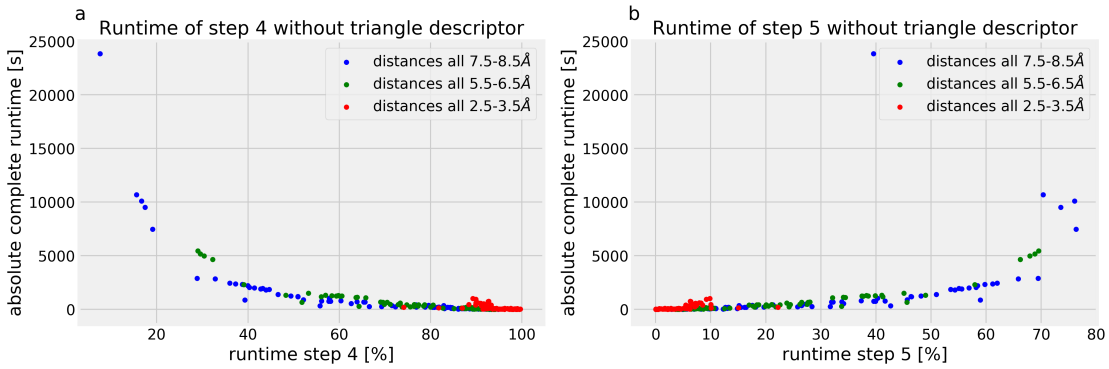


Figure 6.4.: Percentage of the complete runtime for Δ-queries spent in steps 4 and 5 of the search procedure if no triangle descriptor is used. Each data point corresponds to one Δ-query. The color of each dot codes the distance ranges of the respective Δ-queries. a) Percentage of time spend in step 4. a) Percentage of time spend in step 5.

step 5 (see Figure 6.4b).

Moreover, it can be seen that the Δ-queries with small distance ranges (2.5-3.5 Å, displayed as red dots in Figure 6.4) have shorter total runtimes compared to the Δ-queries with larger distance ranges (5.5-6.5Å and 7.5-8.5Å, displayed as green and blue dots in Figure 6.4, respectively).

In general, it can be concluded from these observations that step 4 and step 5 can be both highly time consuming steps in the search procedure. In principle, the triangle descriptor could therefore be able to reduce the overall runtime. From this perspective, if the runtime of long-running queries are to be accelerated, a speed-up of step 5 seems to be more effective than a speed-up of step 4. However, the expected overall speed-up highly depends on the fact that both steps are accelerated by the triangle descriptor at the same time. As an example, a Δ-query which has an overall runtime of 10 000 s is considered. Herein, step 5 requires 75% of the overall runtime. In this scenario, an optimization of step 5 alone can maximally lead to an overall speed-up of four (based on Amdahl's law [103]).

In the next step, the achieved speed-up is analyzed in more detail. Due to the sorting procedure at the beginning of step 4, the distance between search points 'b' and 'c' of a Δ-query is always used as the first db-query. Since the triangle descriptor only applies to search point 'a' of a Δ-query, no speed-up can be achieved for this first db-query. Thus, only the second and third db-query and step 5 can be accelerated in this experiment. To this end, the acceleration of the second and third db-query of step 4 and step 5 are compared to the overall speed-up achieved by using the triangle descriptor.

Figure 6.5 shows the speed-up factors for each of these steps plotted against the overall speed-up for each Δ-query. It can be seen that the second and the third db-queries are only

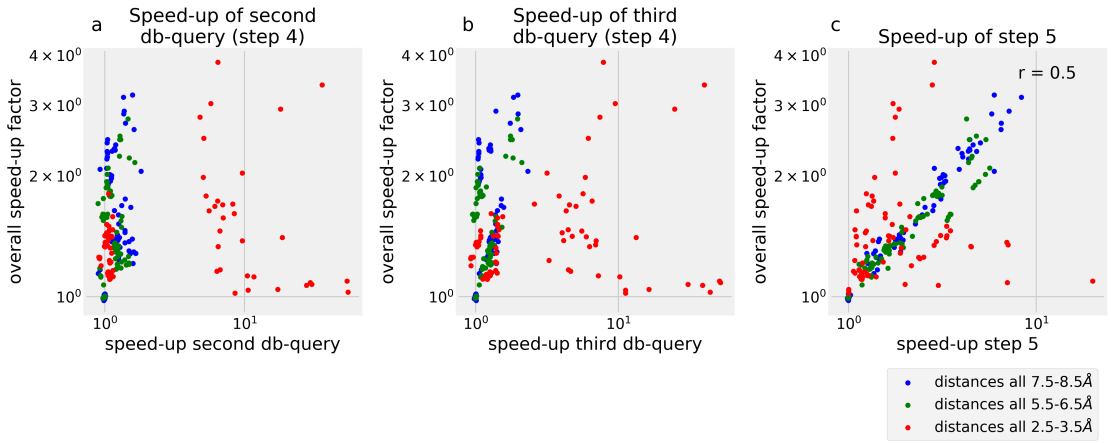


Figure 6.5.: Speed-up factors gained by using the triangle descriptor in different steps of the search procedure. Each data point corresponds to one Δ-query. The color of each dot codes for the distance ranges of the respective Δ-query. a)-c) Speed-up factors achieved in the second db-query of step 4, in the third db-query of step 4, and in step 5 plotted against the overall speed-up factor for the Δ-query, respectively. 'r' represents the correlation coefficient of a linear regression.

accelerated for some of the Δ-queries with the lowest distance range by the use of the triangle descriptor (see Figure 6.5a and b). Concerning Δ-queries with the larger distance ranges, the db-queries are almost not accelerated. In general, no positive correlation between the overall speed-up and the speed-up of the db-queries can be observed. This means that an acceleration of the second and third db-queries does not always lead to an overall speed-up of the entire triangle query. In contrast, a positive linear correlation can be observed between the speed-up of step 5 and the overall speed-up for the Δ-queries (correlation coefficient $r = 0.5$, see Figure 6.5c). Interestingly, the maximal speed-up factors reached for the second and the third db-queries (55 and 52, respectively) are much higher than the maximal acceleration achieved for step 5 (20), respectively.

A reason for the linear dependency between the overall speed-up and the speed-up of step 5 could be the runtime behavior of the clique detection algorithm. Clique detection in graphs is an NP-complete problem with an exponential runtime behavior [104]. Therefore, the reduction of a very large to a small input leads to strongly reduced runtimes in this algorithm. Here, the input for the clique detection are product graphs constructed from all PRP pairs detected in the earlier steps of the search procedure. One graph is constructed for each pocket. A closer look into this procedure reveals that both the number of graphs as well as the size of the graphs (in terms of vertices and edges) are reduced by using the triangle descriptor (no data shown). However, a more detailed analysis would be necessary to confirm this hypothesis.

In order to estimate the effect of the triangle descriptor, not only the speed-up factor as such, but also its ability to accelerate queries with respect to the absolute runtime is impor-

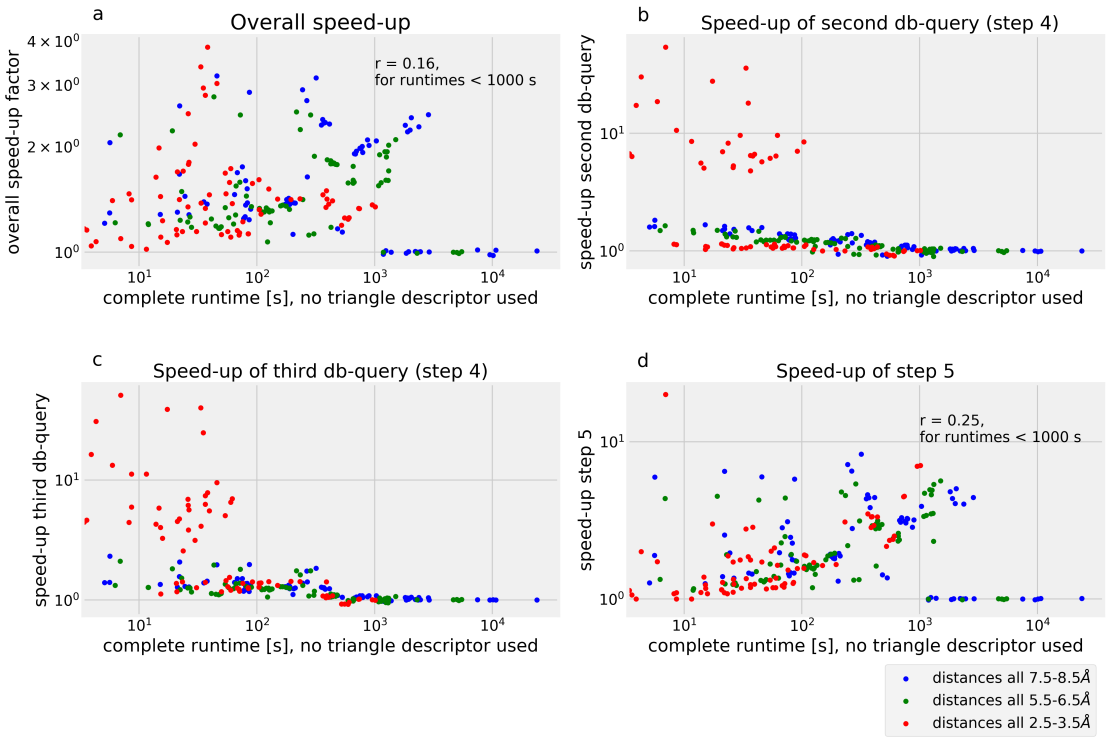


Figure 6.6.: Acceleration of the absolute runtime for Δ-queries due to the triangle descriptor. Each data point corresponds to one query. The color of each dot codes the distance ranges of the respective Δ-query. 'r' represents the correlation coefficient of a linear regression measured for all data points with a runtime < 1000 s. a) Overall runtime of Δ-queries without using the triangle descriptor plotted against the overall speed-up factor. b)-d) Overall runtime of Δ-queries without using the triangle descriptor plotted against the speed-up factors of the second db-query, of step 4, the third db-query of step 4, and step 5, respectively.

tant. For queries which already have a short runtime without using the triangle descriptor, a large speed-up factor is less important than for those with long runtimes. In Figure 6.6 the complete runtime of the Δ-queries without using the triangle descriptor is plotted against the overall speed-up factor and the speed-up of single steps, respectively.

For the overall speed-up factors, a slight positive linear correlation with the complete runtime can be observed up to a complete runtime of 1000 s (see Figure 6.6a, correlation coefficient r=0.16). This means that Δ-queries with a long overall runtime are more accelerated than those with a short runtime. However, this only holds true for Δ-queries up to an overall runtime of about 1000 s. Δ-queries with longer overall runtimes are almost not accelerated at all.

In line with previous observations, large speed-up factors can be observed for the second and third db-queries if the complete runtime is small (see Figure 6.6b and c). This seems to be only the case for queries with distance constraint ranges between 2.5 and 3.5 Å. As

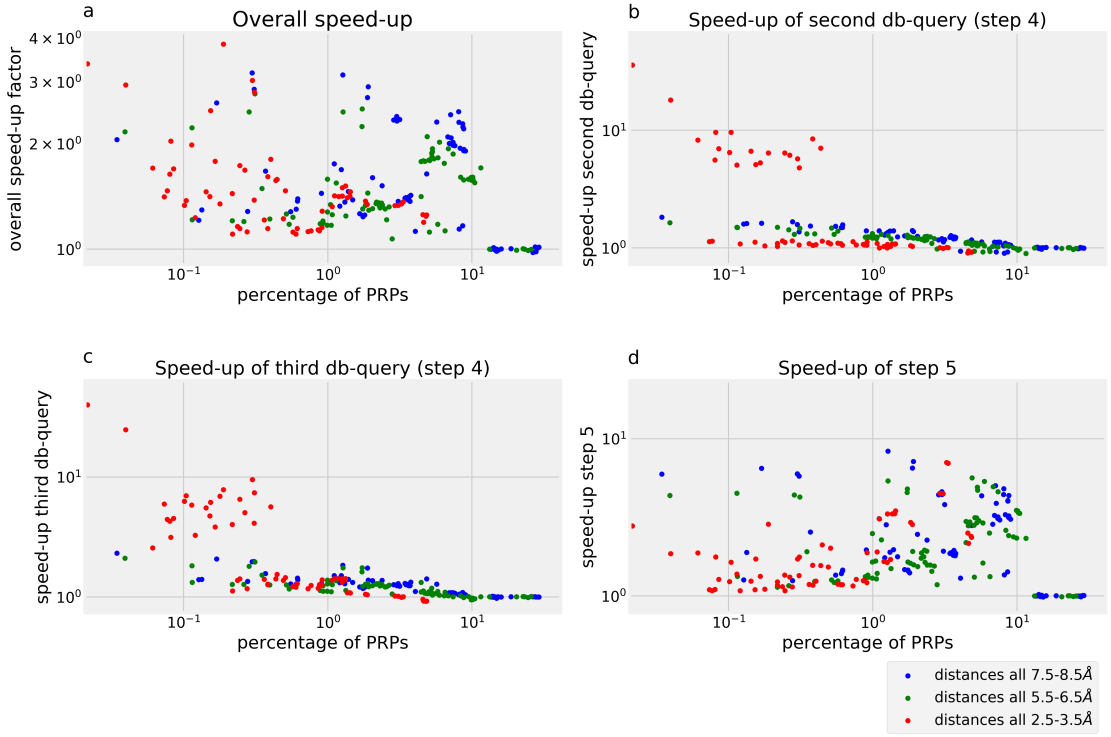


Figure 6.7.: Comparison between speed-up factors of Δ-queries achieved by using the triangle descriptor with the percentage of PRPs represented by the used DT. Each data point corresponds to one query. The color of each dot codes the distance ranges of the respective Δ-query. a) Overall speed-up factor plotted against the percentages of PRPs which are represented by the used DT. b)-d) Speed-up factor of the second db-query of step 4, the third db-query of step 4, and step 5 plotted against the percentages of PRPs which are represented by the DT.

in Figure 6.6b and c only red dots have y-values significantly larger than zero. In step 5, also Δ-queries with longer distance ranges are accelerated. The linear correlation for data points up to a complete runtime of 1000 s is even stronger than for the overall speed-up here (correlation coefficient r = 0.25, see Figure 6.6d). However, the factors by which the Δ-queries are accelerated are smaller than for the second and third db-queries.

Given these results, the question arises what the reasons are that for some queries a high speed-up factor was observed and for other queries almost no speed-up could be recorded. The test queries used in this experiment were designed such that exactly one bit (or one DT) of the triangle descriptor is used in each Δ-query. To this end, the speed-up was compared with the percentage of PRPs which occur in the respective DT for each Δ-query. The results are shown in Figure 6.7.

Figure 6.7a shows that only Δ-queries in which the used DT represents up to 10% of

the PRPs are accelerated. If DTs are used which represent more than 10% of the PRPs, Δ-queries are not accelerated at all. A very similar distribution can be observed for the speed-up of step 5 in Figure 6.7d. db-queries from step 4 are only accelerated if the used DTs represent less than 1% of the PRPs.

In summary, it seems that there are two different kinds of Δ-queries. On the one hand, there are Δ-queries in which the db-queries are accelerated by the use of the triangle descriptor. In these, the used DT only represents a very small amount of PRPs. In our experiment, this mainly applies to Δ-queries with distance ranges of 2.5-3.5 Å. The Δ-queries in this group already have a small overall runtime of less than 100 s without using the triangle descriptor. The maximally achieved speed-up factors are 55 for the second and 52 for the third query.

On the other hand, there are Δ-queries for which the use of the triangle descriptor results in an acceleration only of step 5. Here, the used DT represents up to 10% of the PRPs. In our experiment, the Δ-queries with distance ranges of 5.5-6.5 Å and 7.5-8.5 Å mainly belong to the this group. This group contains Δ-queries with an overall runtime of up to 1000 s without using the triangle descriptor. Hence, the second group contains more Δ-queries than the first. However, the maximally achieved speed-up factor for step 5 is only 20.

One possible explanation for this observation is the number of results produced by the queries. In general, the queries with a lower distance range return fewer results than those with higher distance ranges (see Figure D.2 in Appendix D). The triangle descriptor does not change the number of results but is able to reduce the number of interim results during step 4 and before step 5. In theory, if the number of interim results is already small for a Δ-query, a further reduction of these will not lead to a significant speed-up of step 5 due to the exponential runtime behavior of the clique detection algorithm. On the other hand, for larger interim results, a slight reduction could already lead to a strong acceleration in the clique detection.

Concerning db-queries, an acceleration is only observed for Δ-queries with small distance ranges. Moreover, it can be seen that the db-queries are only accelerated for those Δ-queries which result in a small number of results (see Figure D.2b and c in Appendix D). Due to the shell described by a distance constraint around an atom, it can be assumed that the number of interim results are probably smaller for Δ-queries with small distance ranges than for those with longer distance ranges. Given this assumption, it can be speculated that db-queries are only accelerated if the number of already small interim results are further reduced. However, in order to verify this hypothesis, further analyses of the number of interim results would be necessary.

From this analysis the following final conclusions can be drawn. First of all, the triangle descriptor is able to accelerate Δ-queries. The steps which are accelerated by the descriptor are the highly time-consuming steps in the overall search algorithm. The largest overall

speed-up factor achieved on the queries used here was found to be 3.7. This speed-up is a result of two effects. On the one hand, triangle queries which result in a small number of hits and which are already fast are accelerated due to a speed-up of the db-queries. In these cases, the targeted DT represents only a small number of PRPs ($\leq 1\%$).

On the other hand, Δ-queries which produce more hits can be accelerated by a speed-up of step 5 using the triangle descriptor. These Δ-queries have a runtime of less than 1000 s without using the triangle descriptor and the DTs involved here represent up to 10% of the PRPs. Δ-queries which have a runtime of more than 1000 s without using the triangle descriptor and which employ a DT representing more than 10% of the PRPs, are almost not accelerated at all.

It is obvious that the speed-up factors achieved for complete Δ-queries are much smaller than those achieved for the individual steps analyzed here. One reason for this observation is that only two out of six steps are accelerated by the triangle descriptor. Moreover, it could be seen in Figure 6.4 that for Δ-queries with an overall runtime of up to 2000 s, step 4 is the most time consuming step if no triangle descriptor is used. However, step 5 is the most accelerated part for exactly those Δ-queries.

In order to achieve larger speed-up factors, the following steps could be undertaken. First of all it could be favorable to reduce the number of PRPs represented by one DT. To this end, those DTs which represent more than 10% of the PRPs should be divided into more subclasses by adding more properties to the triangle legs or by reducing the distance ranges. On the other hand, triangles which only represent a few PRPs could be combined using a reverse strategy. Thereby, the overall speed-up factors of the triangle descriptors could be increased without increasing its disk space.

A second possible improvement is an optimized runtime behavior of the clique-detection algorithm. In Pelikan, the algorithm from Bron and Kerbosch implemented in the Boost Graph library is used. Preliminary tests have shown that other implementations for the clique detection problem achieve better runtimes, e.g., the algorithm 'cliquer' by Östergård [105].

However, these steps will only have an effect on those queries which do not produce too many results. The aim of the descriptor is to reduce the result space at an early stage in the algorithm. The effect of the descriptor on the overall runtime is therefore limited by the number of final results. In its current application, the Pelikan algorithm is used in a tool where all results can be visually inspected and compared. In such a scenario, a search which produces more than 1 000 hits is not useful anyway.

Triangular descriptors have already been used by others in order to accelerate the search for matching 3D atomic structures [106–108]. Unfortunately, these authors do not state which speed-up factors they obtain by using these descriptors. Sheridan *et al.* [109] even used distances between specific atom types to accelerate geometrical queries on molecules. Our

investigations, however, have shown that a distance descriptor was not able to discriminate between different PRPs and almost no speed-ups could be achieved for the queries. On the other hand, more complex geometric descriptors such as a tetrahedron are probably too selective and need much more disk space. Thus, the intermediate complexity of a triangle for a geometrical descriptor seems to provide a reasonable balance between discrimination and disk space.

6.4. Query Retrieval Speed

In this section, the runtime of the query mechanism is analyzed. The experiments performed here pursue two distinct goals. First of all, they are used to identify the characteristics of the query mechanism, including its strengths and its limitations. Moreover, guidelines will be deduced from the experiments which can be used to design fast 3D queries.

Given the variability of the query, different aspects of the query mechanism have to be considered. Firstly, different attributes of the query are tested using the query data set presented in section 5.2.5. Secondly, the influence of the database size is tested. Finally, the influence of the used hardware on the query runtime is tested. To this end, different test queries were executed using the HDD and the SSD hardware settings. The technical details of both platforms are described in 5.2.5.

The exact number of resulting hits for each run is listed are Table D.1

6.4.1. Query attributes

Topology

In the first experiment, the influence of the topology of the query on the runtime is tested. Figure 6.8a shows the runtimes of different queries which only contain search points. If a search point without any further geometric constraints is added to a query, an additional db-query is performed in step 2. In principle, subsequent db-queries could be accelerated by precious db-queries here as the pocket ids found in earlier db-queries are used for subsequent db-queries. However, the existence of more search points will not accelerate the first db-query. Hence, it is not surprising that more search points lead to longer runtimes if the search points are not connected by point-point constraints. The most time consuming step in these queries is step 2 of the search algorithm (see Figure D.3a).

In Figure 6.8b, the compared queries have the same number of search points. Herein, the

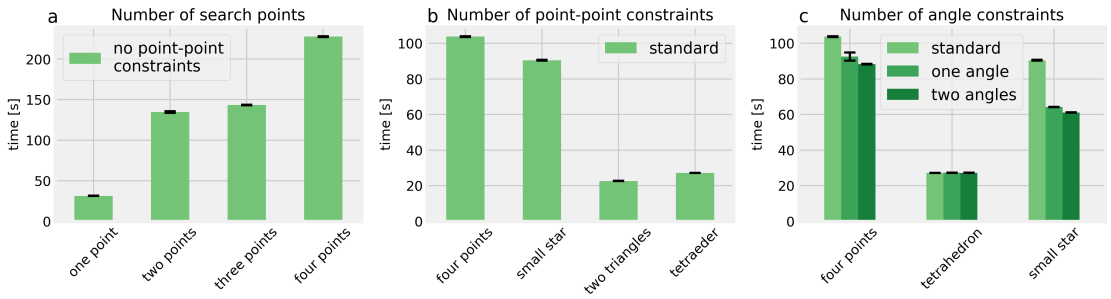


Figure 6.8.: Runtimes of different test queries having different topologies. a) Runtimes of test queries that only differ in their number of search points. b) Runtimes of test queries that only differ in their number of point-point constraints. c) Runtimes of test queries that only differ in their number of angle constraints.

number of distance constraints is growing throughout the used queries. Additional point-point constraints lead to a more specific query. Even if an additional db-query is performed for each point-point constraint in step 4, results from previous queries are used here. Hence, more point-point constraints lead to a reduction of runtime (see Figure 6.8b). Again, the database queries (step 4) are the most time consuming steps in these queries. The overall reduction of the runtime is mainly a result of an acceleration of step 4 (see Figure D.3b). Moreover, the runtime for step 5 is also reduced because fewer results are detected after step 4 and thus the clique detection runs on less data.

The addition of angle constraints does not influence the runtime of a query strongly (see Figure 6.8c) even though the number of resulting hits are reduced (see Table D.1). As an example, for the geometry 'small star' the number of resulting hits is reduced by 95% (from about $1.7 \cdot 10^6$ to about $8 \cdot 10^4$) comparing the standard case and the introduction of one angle constraint. However, the runtime is only reduced by 29%. The reason for this is the fact that angle constraints are checked in step 5 during the construction of the graph for the clique detection process. The addition of angle constraint could therefore only reduce the runtime of the clique detection step. This is the case for the queries on geometry 'four points' and 'small star'. However, step 4 is the most time consuming steps for the queries shown here, respectively (see Figure D.3).

Geometrical constraints

Next, the influence of the attributes of the geometrical constraints are tested. To this end, test queries which are completely identical except for distance ranges, angle ranges, and point-points constraints were generated.

In the first test, the size of a distance constraint range was changed (see Figure 6.9a). In the standard query, all distances have a range of 1 Å. Here, the range for one distance was

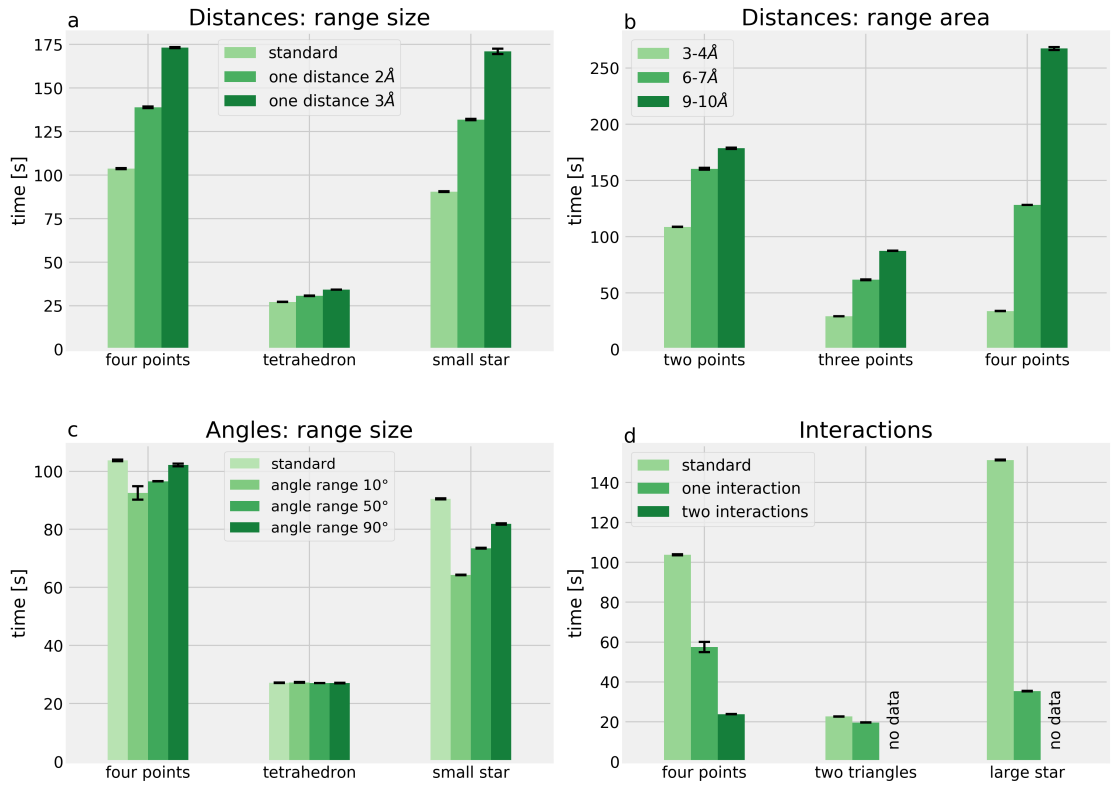


Figure 6.9.: Runtimes of different test queries having different geometrical constraints. a) Runtimes of test queries which only differ in their range size of one distance constraint. b) Runtimes of test queries which only differ in their range area of one distance constraint. c) Runtimes of test queries which only differ in their size of one angle constraint. d) Runtimes of test queries which only differ in their number of distance constraints which are converted to interaction constraints.

increased to 2 Å and 3 Å, respectively. In a second test scenario, the area of the distance range was changed from 3-4 Å in the standard case to 6-7 Å and 9-10 Å, respectively (see Figure 6.9b). In both cases, the number of hits increases with larger range distances and with larger range areas (see Table D.1). The increase in resulting hits is less pronounced for the changed distance range sizes in the geometry 'tetrahedron' as for the other cases. This agrees well with the slight increase in runtime for these queries (see Figure 6.9a). For all queries, the increase of the runtime is a result of a prolonged runtime of step 4 as well as step 5 (see Figure D.4a and b). Next, distance constraints were exchanged for interaction constraints in the queries. The resulting runtimes are shown in Figure 6.9d. Note that for the geometries 'two triangles' and 'large star' only one distance could be converted to an interaction constraint without losing all resulting hits. The introduction of interaction constraints in the queries leads to a strong reduction of results for the geometries 'four points' and 'large star' (see Table D.1). Also the overall runtime, and the runtime of step 4 and step 5 are reduced in these cases (see Figure D.4d). Concerning the geometry 'two triangles', the

runtime is only reduced slightly. In accordance with this, the number or results only decreases from 5 166 results in the standard case to 878 with one interaction. As a comparison, for the geometry 'large star', the number of hits are reduces from about $1 \cdot 10^7$ to about $1 \cdot 10^5$ by the replacement. Thus a reduction of the result by 89% is accompanied by a runtime reduction of 77%.

In Figure 6.9c the resulting runtimes for queries with different range sizes of one angle constraint are shown. Again, the overall runtime is only slightly influenced. This is due to the fact that only the runtime of step 5 is impacted by angle constraints. However, the most time consuming step here is step 4 (see Figure D.4c).

Properties

In this experiment, the influence of additional properties on the overall runtime is tested. Firstly, the influence of a more precise description of the search points was analyzed. The results of this experiment are shown in Figure 6.10a. It can be seen that more properties lead to strongly reduced runtimes. This is mainly due to decreased runtime in step 4 (see Figure D.5a). Accordingly, the number of resulting hits is reduced by the definition of search-point properties (see Table D.1).

Similarly, the addition of textual and numerical constraints leads to reduced runtimes and a reduced number of hits for different query geometries (see Figure 6.10b).

Next, the runtimes of queries are compared by replacing search points properties by rarely occurring attributes. As an example, the attributes 'nitrogen, reference ligand' of search point 3 in the standard case were exchanged by 'any element, metal' in the query called 'metal'. The results are displayed in Figure 6.10c. For the geometries 'four points' and 'small star', a clear reduction of the runtime can be observed. Concerning the geometry 'tetrahedron', the runtime is not strongly reduced by the the queries 'metal' and 'metal, water'. The reason for this effect is the triangle descriptor. In the standard case, every search point is part of several triangles. Hence, the number of possible PRPs for search point 1 can be reduced to $6 \cdot 10^5$ before any db-query is executed. The number of resulting hits in the standard case are 5 166. In comparison, if the attributes of search point 3 are set to 'any element, metal', the list of possible PRPs for search point 1 is increased to $8 \cdot 10^6$. This query results in only 45 hits. If additionally the molecule type of one search point is set to 'water', only one search point remains in a triangle used in the descriptor. Here, 23 hits are detected. In the last step, the element type of one search point is additionally set to 'phosphorus'. Here the triangle descriptor does not apply at all. However, the combination of these search points only rarely occur in the database (5 hits found) and thus the runtime is very fast. In conclusion, even though the number of results are reduced in the changed queries, the triangle descriptor does not work as effective as in the standard case and the runtimes are therefore not strongly changed.

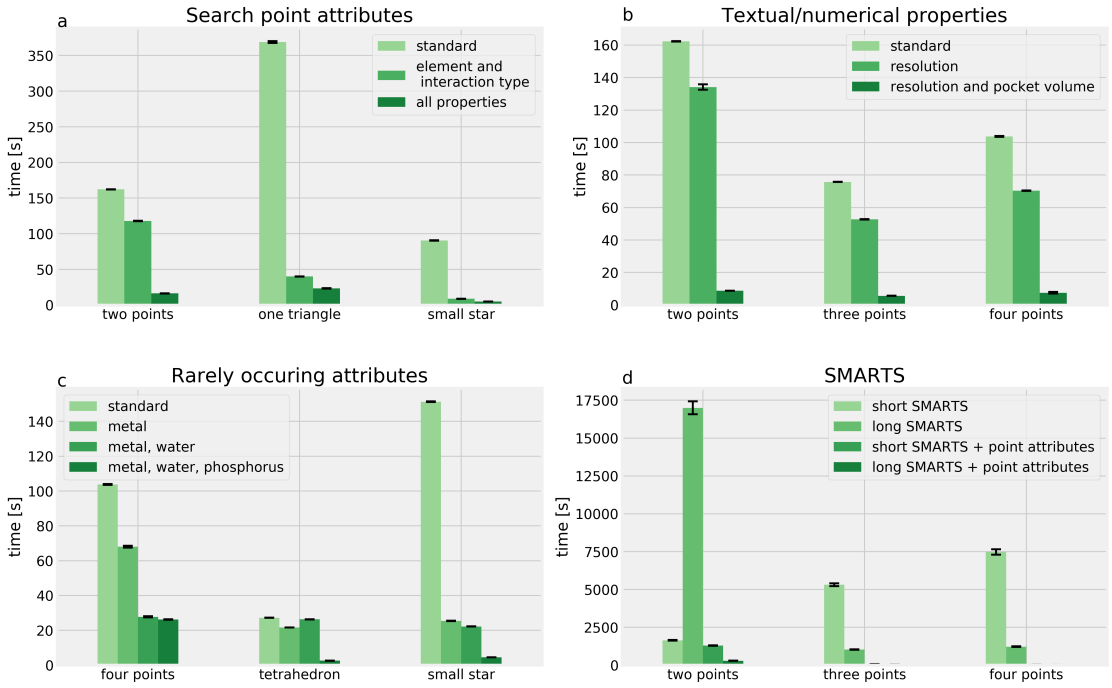


Figure 6.10.: Runtimes of different test queries having different additional properties. a) Runtimes of test queries which only differ in their attributes of all search points. b) Runtimes of test queries which only differ in their textual and numerical properties. c) Runtimes of test queries which only differ in their element and molecules types of search points. d) Runtimes of test queries which only differ in their SMARTS description and other additional properties of all search point.

As an additional property, the chemical environment of a search point can be described using a SMARTS pattern. Here, the runtimes of different queries with short and long SMARTS patterns were used to analyze their influence on the runtime. In the queries, every search point is either equipped with a long or a short SMARTS pattern. The short SMARTS patterns describe three, the long six to nine atoms in the chemical environment of the respective search point. The exact SMARTS patterns used can be found in Appendix D.

Because of their length, all long SMARTS are additionally used in step 1 where a SMARTS matching on all small molecule is performed. Concerning the short SMARTS, only the pattern for search point 3 is additionally used in step 1, because it contains the element phosphorus. The results of this experiment are displayed in Figure 6.10d. Overall, it can bee seen that using SMARTS patterns leads to much longer runtimes compared to other queries. If in addition to the SMARTS pattern, all possible properties of the search point are set, the runtimes are strongly reduced. Interestingly, for the geometries 'three points' and 'four points', using short SMARTS leads to longer runtimes as the use of long SMARTS. The opposite is true for the geometry 'two points'. Here, both queries without additional search point properties result in more than 1000 hits. Thus the SMARTS search procedure

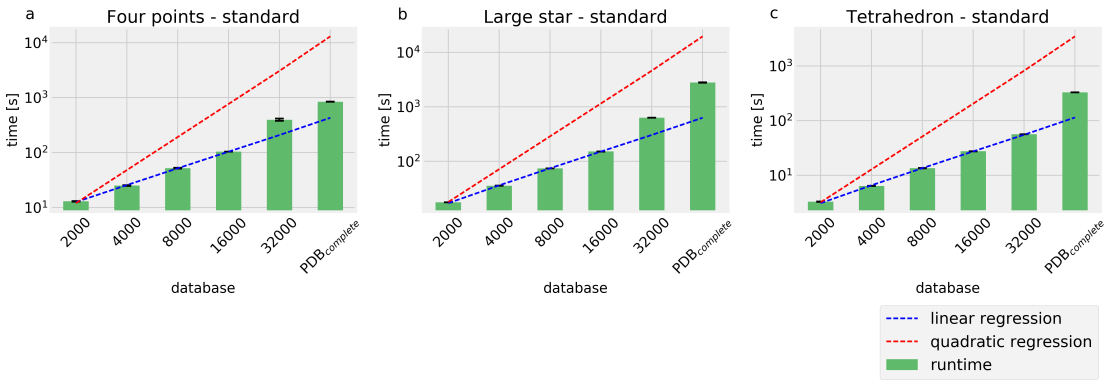


Figure 6.11.: Runtimes of different test queries on databases with different sizes. The mean runtime and standard deviations of three independent runs are shown as bar plot in seconds. A linear regression curve using the first four data points of each plot is shown as blue dotted line. In red, a quadratic regression line of the form $y = a \cdot x^2$ is shown. a is chosen such that the first data point lays on the regression line. Note that the quadratic function has a linear growth of 2 in this plot due to the logarithmic scale of both axes.

is stopped after 1000 hits were detected. These 1000 hits are reached faster for the short SMARTS than for the long SMARTS pattern. This then results in a faster runtime for the query containing short SMARTS for every search point. In case of the queries with 'three points' and 'four points', all resulting hits are below 1000. For all queries containing SMARTS pattern, step 6 of the search algorithm is the most time consuming step (see Figure D.5d).

In general, it can be concluded from the results so far that geometrically more specific queries lead to faster runtimes. However, specifically defined search points or distance constraints are more effective than specific angle constraints. Most of the time, the runtimes for step 4 is the most time consuming step. Using a SMARTS pattern can lead to strong increases of the runtime for a query. In these queries, the SMARTS matching procedure in step 6 is the most time consuming step.

6.4.2. Database Size

In the following experiment, the influence of the database size on the runtimes is assessed. To this end, three different queries were used on databases of different sizes. The results are shown in Figure 6.11. It can be seen that the runtime of the queries increases with database size. For each of the used geometries, the number of results almost doubles when the size of the database doubles (see Table D.1 and Figure D.7 in Appendix D). This indicates a linear connection between the number of results and the complete database size. For all three used types of queries, the runtime grows linearly up to a database size of 16 000 protein-ligand

complexes. This is reflected by linear fits to the data shown in Figure 6.11. For these fits, only the first four data points were used, respectively. In all three cases, the regression coefficient is 0.999. For larger database sizes, the runtimes deviate from this linear dependence, growing faster than linear. The overall growth is, however, slower than quadratic, which can be seen by comparison with the quadratic growth line plotted as guide to the eye in Figure 6.11.

For the queries analyzed here, step 4 is the most time consuming step. Within this step, database queries for all point-point constraints are performed. SQLite uses B-trees of database pages to store the data in the tables. Thus the runtimes of these statements are limited by the runtime complexities achieved on these B-trees. Each db-query contains a combined JOIN and WHERE statement. In SQLite, JOIN statements are implemented as nested loops. Hence the asymptotic runtime of this step is in $\mathcal{O}(n \cdot m)$ where n and m are the number rows of each of the joined tables, respectively [92]. In this case, a table is joined with itself. Hence, $n = m$ and the complete runtime for a JOIN is in $\mathcal{O}(n^2)$. If indices exist on the attribute used for joining, this runtime complexity could be reduced because the relevant rows can be found in $\mathcal{O}(log\ n)$ [110]. The complete runtime of the JOIN would then be in $\mathcal{O}(log\ n + v \cdot w)$, where v and w are the rows detected for the first and the second search point which have to be joined, respectively. However, even if indices are used, the runtime of the JOIN statement depends on the number of rows which have to be joined. Hence, in the worst case, $v = n$ and $w = n$ which still leads to $\mathcal{O}(n^2)$ as an upper boundary of the complete step. The execution of the WHERE statement is also performed as a loop inside the nested loops of the JOIN statement. The order in which these loops are combined depends on the sizes of the expected results from each step. In principle, if an index is used, all PRPs fulfilling a WHERE statement can be detected in $\mathcal{O}(log\ n + r)$, where n is the number of rows and r is the number of results.

Taken together, the asymptotic runtime behavior of a query on large databases is influenced by different key factors, for example the number of rows in the table and the number or results. These theoretical considerations show that the runtime of this step grows faster than linear and describes at most a quadratic function which is in line with the data observed in Figure 6.11.

6.4.3. Hardware

As step 4 is one of the most time consuming steps in the query mechanism, the speed of the underlying hardware is expected to have an impact on the overall runtime of the query. For comparison, runtimes for different queries on the HDD and SSD hardware settings were recorded. For these experiments, the database $\text{PDB}_{complete}$ was used. The results of the measurements are displayed in Figure 6.12. Overall, using the HDD setup increases the runtime for all queries. This is attributed to the higher reading speed of the SSD compared to the HDD, because the $\text{PDB}_{complete}$ database is larger than the available random access memory

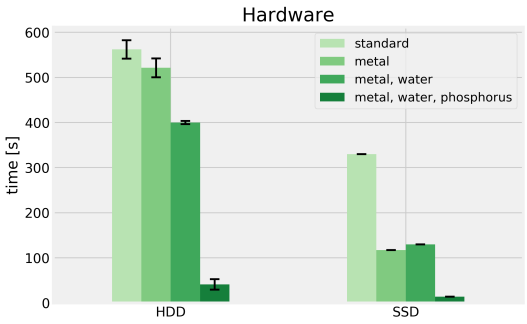


Figure 6.12: Runtimes of four different test queries on the hardware settings HDD and SSD. The database $PDB_{complete}$ was used. Mean values and standard deviations of three independent experiments are shown as bar plot.

(RAM) of 16 GB. The factor by which the runtime is increased varies between 4.4 (query 'metal') and 1.7 (query 'standard'). For databases smaller than $PDB_{complete}$, no significant runtime differences could be observed (data not shown). This reflects that not the complete but only pages relevant for the search are usually held in RAM. On a PC equipped with less RAM, the effect of a solid state disk would therefore probably kick in for correspondingly smaller database sizes. Here, the used PC had 16 GB of RAM. Moreover, the exact file system installed on the hard disk and the solid state disk can have an influence on the runtime.

Considering all runtime measurements, it can be concluded that the runtime of queries mainly depends on the number of resulting hits which is a result of the specificity of the query as well as of the size of the used database. In order to achieve fast runtimes for 3D queries, the following rules can be deduced:

- Geometrically more precise queries are faster than queries with a not well defined geometry. In this respect, ranges of point-point constraints have a stronger effect than ranges of angle constraints.
- Search point attributes which hit fewer PRPs lead to faster queries.
- Additional textual and numerical constraints reduce query runtimes.
- The use of smaller databases leads to faster runtimes.
- High reading speed of the hardware decreases the runtime of queries.

The Pelikan method has been mainly designed for the rapid search for specific queries in large sets of protein-ligand interfaces. In this scenario, only queries which result in less than 1 000 results are applicable. In experiments shown here, queries which result in less than 1 000 hits have a runtime of less than 50 s on a database containing 16 000 protein-ligand complexes, except for the queries containing SMARTS patterns. A second application of the Pelikan method are statistical analyses of coarse queries. In this case, more results than 1 000 are required in order to derive meaningful hypotheses. Herein, interactivity is probably not as important as correctness for a user. The experiments shown here demonstrate that Pelikan

has runtimes of up to 360 s in this scenario, except for the queries containing SMARTS pattern. Thus, the Pelikan method is able to rapidly detect specific geometrical patterns if interactivity is required. For the generation of statistical significant results, Pelikan is also able to find large number of results but requires longer runtimes.

Most of the results shown here were calculated on a database containing 16 000 different PDB files. The comparisons of databases with different sizes have shown that the runtime behavior grows slightly faster than linear with the database size. Given the rapid growth of the PDB in the last years, the approach chosen in Pelikan will loose its effectiveness on larger data sets. According to Allen and Owens, simple statements are extremely quickly on SQLite because of a small or non-existing overhead of network calls and server authentication procedures [98]. However, more complex queries can be faster on other relational databases systems which have a more sophisticated query optimizer and planner. The statement used for detecting PRP pairs in the database fulfilling the search point criteria belong to the latter group of more complicated queries. Hence, a database management system with a better query optimizer could achieve faster runtimes here. On a server-based database it might also be faster to query all point-point constraints in a parallel instead of a sequential process. Moreover, an even faster solid state drive could be used in order to further accelerate the queries.

Ideally, the Pelikan method would then be usable in two different ways: geometrical queries on $\mathrm{PDB}_{complete}$ could be answered using a web service which accelerates queries using a sophisticated query optimizer. For the analysis of more specific data sets, user can create their own SQLite database.

6.5. Comparison with Relibase

As a last experiment the performance and ability of the Pelikan method to search for 3D structures in protein-ligand interfaces is compared to Relibase. This experiment should help to contextualize the Pelikan approach within the landscape of existing tools. Among the currently published tools, only Relibase and Relibase+ allow for precise queries on an atomic level as Pelikan does. However, since Relibase+ is not publicly available, only Relibase can be used here. In contrast to Relibase+, Relibase does not allow for intra-molecular distances. Moreover, angle constraints are not possible. Hence, three different 3D queries were designed which contain only distance constraints between the ligand and the protein. The queries are shown in Figure 5.3.

The resulting hits and the retrieval times for the searches of all three queries on both systems are displayed in Table 6.2. Since the runtimes for Relibase highly fluctuated, the runtimes for Relibase are given as mean values from three independent experiments. For Pelikan, only the runtime of one exemplary run is shown since the fluctuation of runtimes using the same

Query	Relibase			Pelikan		
	hits	PDB structures	runtime	hits	PDB structures	runtime
Query 1	29	24	148 ± 60 s	30	24	62 s
Query 2	114	55	108 ± 74 s	218	52	66 s
Query 3	8	7	28 ± 6 min	13	7	113 s

Table 6.2.: Resulting hits detected by Relibase and Pelikan using three different queries. For each query, the number of resulting hits, the number of detected PDB structures and the runtime is given for Relibase and Pelikan, respectively. For Pelikan, a database containing $PDB_{complete}$ was used. Runtimes for Pelikan were measured using the SSD settings. For Relibase, the web interface provided by the CCDC was used (http://relibase.ccdc.cam.ac.uk/index.php, accessed between March and June 2017). Runtimes were measured using a stopwatch. Mean values and standard deviations of three independent experiments are shown.

query is very small.

For query 1, almost the same results were detected by Relibase and Pelikan. They only differ in the number of detected hits. Relibase found 29 hits whereas Pelikan detected 30. The reason for this is that the used query could match the same set of atoms twice. In the Pelikan query, search point 1 and 2 are interchangeable if both matching atoms fulfill the distance constraint to search point 3. This is the case in one pocket. Obviously, Relibase does not count these symmetric hits.

Concerning query 2, Pelikan found many more hits than Relibase. The reason for this is again the symmetry of the query. Pelikan counts every unique hit whereas Relibase seems to count only unique sets of atoms as hits. Moreover, Relibase found three PDB structures which were not detected by Pelikan, these are 2hm9, 3kwh, and 2xad. The PDB code 3kwh contains a protein-ligand complex whose structure has been determined using NMR. Thus several models for this structure exist. During the interpretation of the PDB file, only the first entry is used by the NAOMI library if several models exist. In this first model, the distance between the atoms which match search point 1 and 4 is too large for the used distance constraint (4.7 Å). Relibase uses all models of the structure. In other models, the distance between the matching atoms agrees with the distance constraints. The structure with PDB code 3wkh is deprecated and has been replaced by the structure 3oc0 in the PDB. Both Relibase and Pelikan find a hit in 3oc0. 3wkh seems to be still part of the Relibase database but not of the Pelikan database. The PDB code 2xad links to a structure which contains a glycopeptide. Relibase considers this glycopeptide as ligand and thus detects a hit. In Pelikan, this structure is considered as protein and thus no hit is detected in this structure.

Even though Relibase and Pelikan find the same number of PDB structures for query 3, the detected structures differ between both tools. Only one hit is identical: PDB structure 1kwf with ligand GLC. The remaining seven hits detected by Relibase are not detected by Pelikan.

PDB Code	Ligand	Reason why hit is not detected by Pelikan
3noq	EDO	Ligand contains only four heavy atoms.
4mty	GOL	Ligand oxygens are in two different molecules
2ayw	ONO 501	Ligand oxygens are in two different molecules
2ayw	ONO 601	Ligand oxygens are in two different molecules
1ylj	SO4	Ligand oxygens are in two different molecules
1lug	SUA	Ligand oxygens are in two different molecules
3k34	SUA	Ligand oxygens are in two different molecules

Table 6.3.: Hits which were exclusively detected by Relibase using query 3. For each hit, the PDB ID, the ligand name and the reason, which the hit was not detected in Pelikan is given.

Table 6.3 lists all these hits and gives a reason why the respective hit is not detected by Pelikan. In most of the cases, the two ligand oxygens of the query are part of two different molecules in the resulting hits. In the Relibase query, only the origin of an atom can be specified, e.g., ligand, protein, or water. If the drawn structure is not connected, it cannot be specified that two atoms should be part of the same molecule. In Pelikan, the reference ligand, which is used to define the pocket, is logically different from other small molecules within the pocket, called ligands. If in the the query 3 for Pelikan, the term 'reference ligand' is exchanged by 'ligand' for one of the oxygen, all hits listed in Table 6.3 could be detected, except for PDB structure 3noq. This hit could only be detected with Pelikan if for both oxygens, the term 'reference ligand' is replaced by 'ligand'. This means that Pelikan is in principle able to find all hits Relibase is detecting. However, the query used in Pelikan is more precise in a sense that a user has to define if points are part of the same reference ligand or part of different ligands. In Relibase, these cases can not be distinguished.

Moreover, there are six PDB structures which were found by Pelikan but not by Relibase. For each of these structures, one hit is exemplary shown in Figure 6.13. All six hits are valid since they agree with the used search constraints. The hits detected in PDB structures 3whi and 5jug are within covalently bound ligands. Relibase does not seem to interpret these structures as ligand which is why they are not detected by Relibase. A reason why the PDB structures 1i1w and 1g66 are not among the hits of Relibase could be that GOL is not interpreted as ligand. The molecule GOL is relatively small and could be interpreted as part of the solvent rather than as a ligand. However, this interpretation is very unlikely since Relibase also detected the ligand EDO in PDB structure 3noq (see Table 6.3) which is chemically very similar to GOL. Concerning PDB structures 4x5p and 2bzz, no possible explanation why these hit are not part of Relibase's hitlist could be discovered.

The comparison of the runtimes which Relibase and Pelikan needed to find all hits is difficult because Relibase can only be accessed via a web server. The runtime here highly depends on the connection and number of simultaneous users. Therefore, the queries were repeated several times on different days using the Relibase web interface. Overall, the runtimes for query 1 and 2, are mainly between 1 and 3 minutes using Relibase and Pelikan. For query 3, however, Relibase requires with 28 minutes about a factor 15 longer than Pelikan. Noticeably,

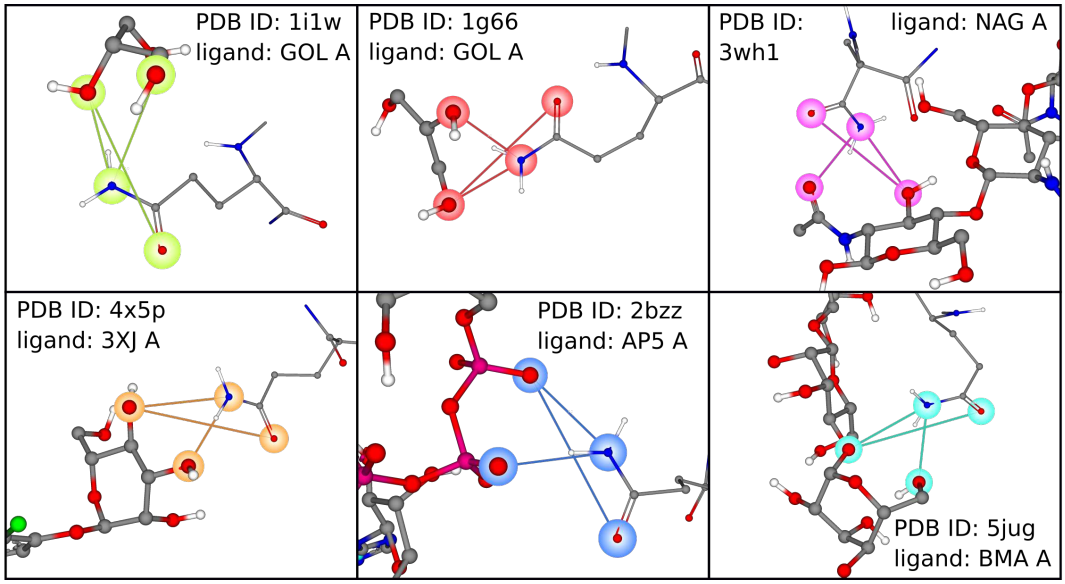


Figure 6.13.: Hits which were exclusively detected by Pelikan using query 3 on the database $PDB_{complete}$. The atoms matching the search points are highlighted with colored disks. The distance constraints are indicated by colored lines.

queries in Relibase seem to be much faster if a large ligand substructure is used instead of a large molecular structure from the protein.

Taken together, it can be concluded that Pelikan is able to find correct results in a runtime which is similar or faster than that of Relibase. For queries which contain more information about searched structures in the protein than in the ligand, Pelikan is even faster than Relibase. Moreover, Relibase does not find all hits which are detected by Pelikan for this query. Even if the additional geometric constraints of Relibase+ are taken into consideration, Pelikan offers more query flexibility as a large number of textual and numerical properties can be added to a geometrical query.

6.6. Application Examples

In this section two application examples are presented which demonstrate how Pelikan can be used to search for bioisosters as well as chemoisosters.

6.6.1. Bioisosters

In the first application example, chemical fragments which bind to a very similar sub pocket in proteins (bioisosters) are searched. Herein, the binding site of the protein factor X is used

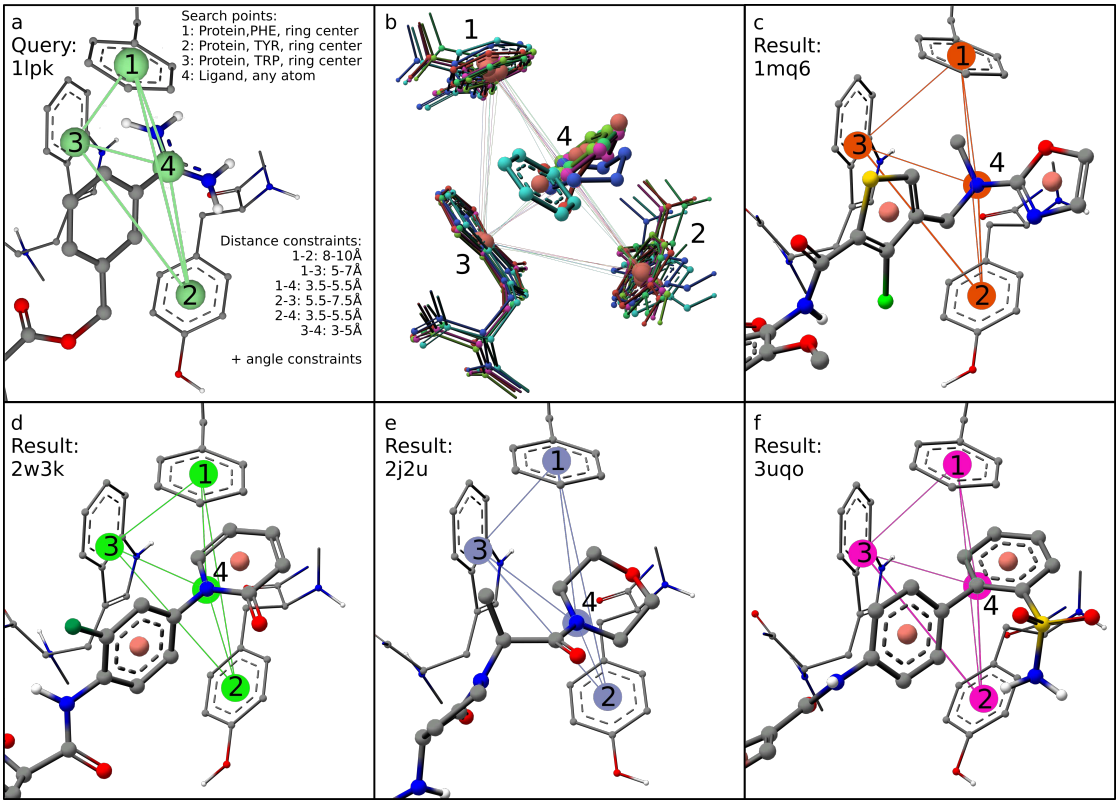


Figure 6.14.: Application example of Pelikan's ability to find bioisosters a) The pocket of 1lpk is used to define a geometrical query describing an aromatic cage. A search point is represented by a green sphere with their id written in black. Distance constraints are indicated by green lines. b) Superimposition of several resulting structures. c)-f) Examples of resulting structures. In each figure, the rings and atoms matching the search points are highlighted with colored disks and with the id of the respective search point.

as a starting point. Factor X is an enzyme involved in the blood coagulation process by converting prothrombin into its active form thrombin. In order to prevent thromboembolic disorders like stroke or thrombosis, different drugs have been developed which inhibit factor X. In the S4 pocket of its binding site, factor X features three aromatic rings which surround a hydrophobic space. This arrangement is called aromatic cage. In order to find different chemical moieties which bind in such an aromatic cage, a geometrical query was designed as shown in Figure 6.14a. The binding site of the structure 1lpk [111] was used here. A search point was defined for each aromatic ring. Each search point has three attributes: 'location = Protein', 'interaction type = aromatic ring', and its specific amino acid type. Additionally, a fourth search point was added to the query with only one attribute: 'location = Reference ligand'. The points are mutually connected with distance constraints representing the structure of 1lpk. Moreover, several angle constraints are added to the query such that the specific spatial arrangement of the rings is represented. The exact query can be found in

Appendix C.

Finding all occurrences of this query in the data set $PDB_{complete}$ took about 60 s on the SSD settings. In total, 213 hits in 66 different PDB files were found. In Figure 6.14b, a superimposition of several results is shown demonstrating the geometrical accordance of the results with the query. Most of the results are structures of the protein factor X. Exemplary, the resulting hits in 1mq6 [112], 2w3k [113], and 2j2u [114] are shown in Figure 6.14c, d, and e, respectively. Moreover, a structure from the bovine trypsine (PDB code 3uqo [115], Figure 6.14d) was found which contains a very similar subpocket as factor X. The four result examples presented in Figure 6.14 all contain different chemical moieties which occupy the space inside the aromatic cage. These moieties are bioisosters.

A search such as the one presented in this example can be used to find bioisosteric fragments and help generate ideas for placing or replacing specific fragments into subpockets. A similar search on proteins with aromatic cages has recently been performed in different small-molecule design projects [116, 117] showing the importance of such approaches.

6.6.2. Chemoisosters

In the second example, chemoisosteric protein environments are searched which are able to bind the same chemical fragment. Pelikan offers different possibilities to search for chemoisosters. If the exact definition of chemoisosterism is used, one could use Pelikan to search for a specific substructure in a reference ligand. In such a search, however, the results can contain binding sites which bind the fragment in different ways. Thus the concept of chemoisosterism can even be used in a more precise way. Pelikan can find specific substructures in reference ligands which build specific interactions with their chemical environment. In this example, protein environments which bind to an uracil fragment with four atomic interactions are searched. A schematic depiction of the query is shown in Figure 6.15a. The query has been designed using the ligand deoxythymidine in the PDB structure 2z1a (unpublished). The substructure is represented by search points 1-4. Search point 1 contains a SMARTS pattern, an attribute which describes the chemical environment of an atom. Moreover, the SMARTS pattern contains labels which refer to search points 2 and 3. In this way, the exact chemical relation between the search points 1, 2, and 3 is ensured. The exact SMARTS pattern is displayed in Figure 6.15a. Each search point which is part of the ligand substructure is connected via an interaction constraint to four different search points within the protein (search points 5-8). An angle constraint between search point 4 and 8 restricts the π-π interaction geometry to a face-to-face configuration. Moreover, two distance constraints between search point 1 and 4 and between 3 and 4 ensure the correct position of the ring center relative to the ligand substructure.

Using the data set $PDB_{complete}$, the search of this query took 90 s on the SSD hardware settings and revealed 160 matches in 43 different PDB files. About 50% of the resulting

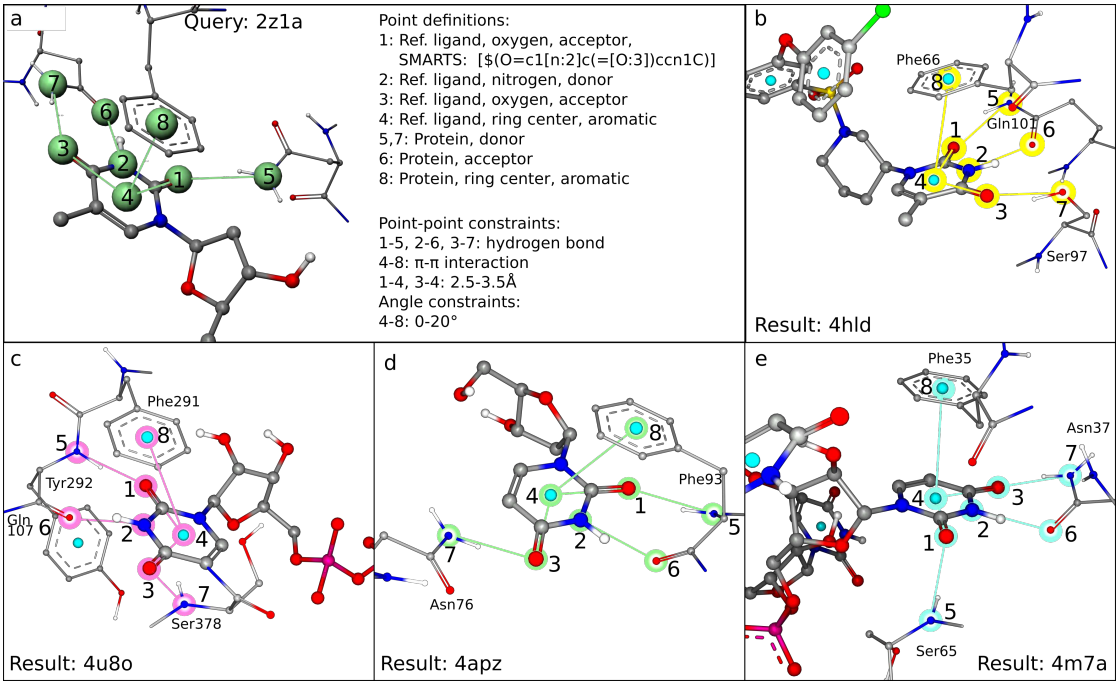


Figure 6.15.: Application example of Pelikan's ability to find chemoisosters. a) The pocket of 2z1a is used to define a geometrical query describing a uracil fragment binding to a protein with four specific atomic interactions. Search points are represented by a green spheres with their id written in black. Distance constraints are indicated by green lines. b)-e) Examples of resulting structures. In each figure, the atoms and rings corresponding to the search points are highlighted with colored disks and ids. This figure is adapted from [77]. Reprinted (adapted) with permission from [77]. Copyright 2017 American Chemical Society.

structures are thymidylate kinases from different organisms. Exemplary, the structure of PDB code 4hld [118] is shown in Figure 6.15b. The ligand 'uracil diphosphat' (UDP) occurs frequently in structures of the PDB and thus also several of the results here contained this ligand (see Figure 6.15c with PDB code 4u8o [119] as an example). Accordingly, several protein structures which naturally bind to UDP – as for example the dUTPase – are detected with different ligands (see Figure 6.15d, PDB code 4apz [120]). In Pelikan, small chains of nucleic acids are considered as ligands. Thus an RNA binding protein (Lsm) bound to the uracil fragment of a small RNA is part of the results (see Figure 6.15e, PDB code 4m7a [121]).

These results demonstrate Pelikan's ability to rapidly detect chemoisosters in a large set of protein-ligand complexes. Besides the demonstrated visual analysis of the hits found with Pelikan, a statistics report can be exported for each search. This report can be used to elucidate common structural aspects of all results, e.g., a common amino acid type of a specific search point.

7

Results and Discussion of *NAOMI*nova

In this section, the results of the experiments performed with *NAOMI*nova are presented and discussed. They mainly aim at proving the correctness of the *NAOMI*nova method and at showing its performance. The detailed conduction of these experiments is described in Section 5.3. All experiments described here were performed using the SSD hardware settings (see Section 5.1 for the exact definition).

7.1. Systematic Correctness

The correctness of the *NAOMI*nova method was tested by checking for false positive and false negative results. The experiment explained in Section 5.3.1 was performed independently three times. Neither false positive nor false negative results were detected.

As for the Pelikan method, it is difficult to completely exclude the existence of false positive and false negative results. For the former, all detected results were compared to the used filter properties. The test for the latter is even more difficult because all correct results had to be known for a query. This can be done on small examples but proofs to be difficult on a larger scale. The experiment performed here tests the retrieval of specific atoms. The substructure as well as the retrieved atom is chosen randomly. Hence, it can be concluded that in the majority of the cases, *NAOMI*nova works correctly and detects the correct results. However, there might be corner cases which rarely occur for which *NAOMI*nova does not work correct. These could be detected by increasing the set of used protein structures and by repeating the experiment more often.

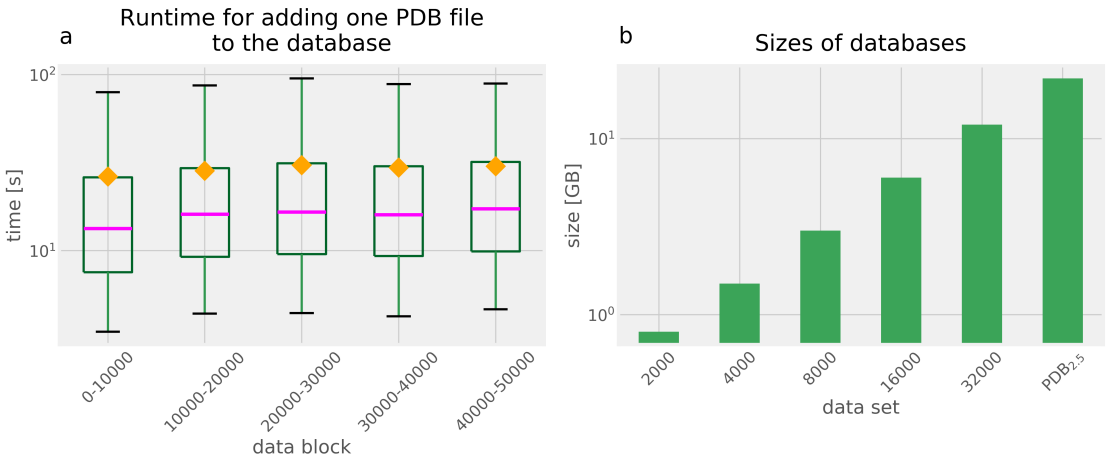


Figure 7.1.: Performance of the database construction process in *NAOMI*nova. a) Runtimes for adding one protein-ligand complex to a database shown in subsequent slices of data. Mean values are shown in yellow, the median is given as pink line. The box represents the lower and upper quartile of each data slice. The whiskers range from 5% to 95%. b) Sizes of *NAOMI*nova databases containing sets of protein-ligand complexes of different sizes.

7.2. Database Construction

A database was build using the $PDB_{2.5}$ data set in order to estimate the runtime for creating a *NAOMI*nova database. During the build process, the runtime required for adding each PDB files was recorded. The results of this experiment are shown in Figure 7.1a. Herein, the runtime characteristics are shown for subsequent data slices of 10 000 files. The mean value, displayed as a yellow square in Figure 7.1a, slightly fluctuates between 26 s (data block 0-10 000) and 31 s (data block 20 000-30 000). However, no continuous increase in the runtime is seen throughout the slices meaning that the runtime does not severely increase with database growth within the used regime. For each file, the most time consuming step is the calculation of the EDIA values (about 90% of the runtime, data not shown).

As expected, a linear dependency between the number of protein structures in the database and the used disk space can be seen (see Figure 7.1b). A database containing $PDB_{2.5}$ has a disk space of 20 GB.

7.3. Adding Substructures to the Database

The performance of the substructure-adding step is determined using three different substructures. The exact definition of these substructures can be found in Section 5.3.4. A schematic depiction of the SMARTS pattern for all three substructures is shown in Figures 7.2a-d. All three substructures share the same fragment part in their SMARTS which is

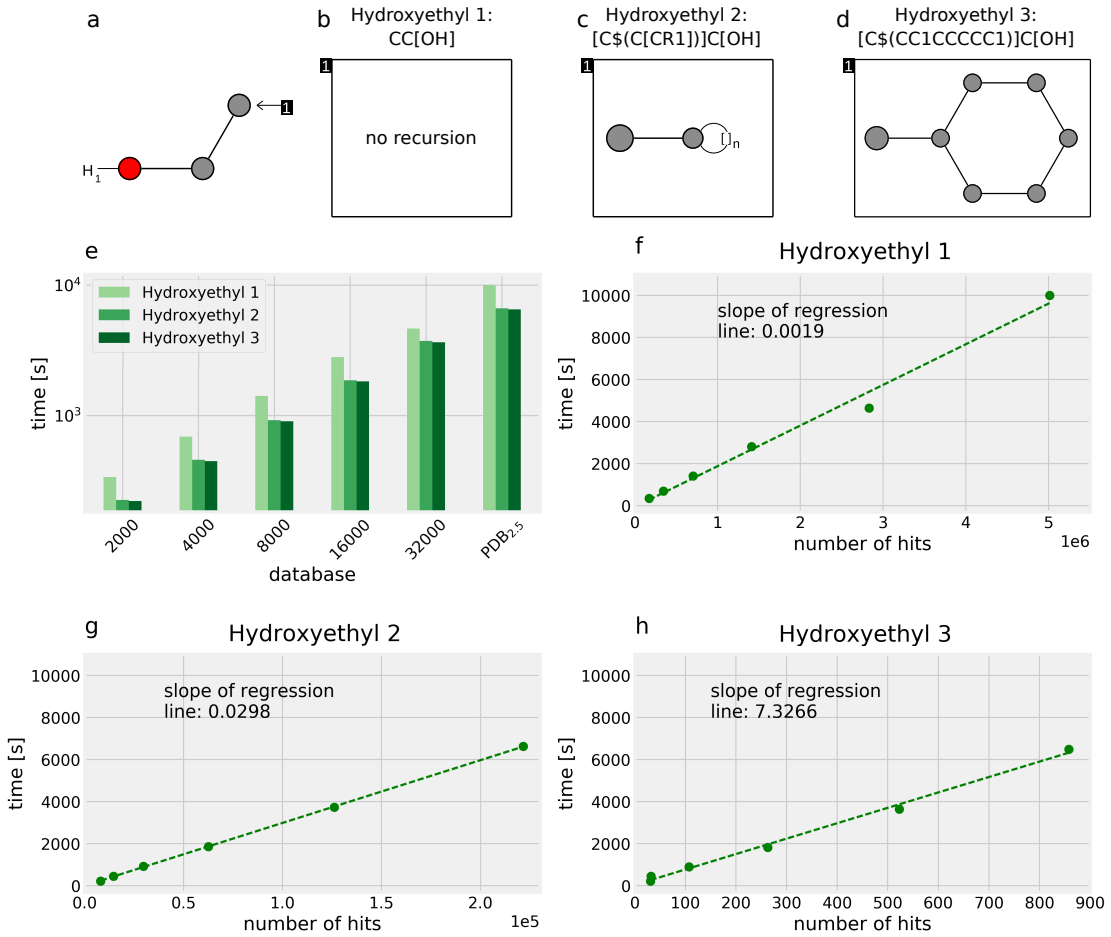


Figure 7.2.: Three different substructures are added to six different databases. a)-d) Schematic depiction of the three substructures. Pictures were generated with SMARTSviewer [122]. a) The fragment part of all three substructures. b) Surrounding part of Hydroxyethyl 1. c) Surrounding part of Hydroxyethyl 2. d) Surrounding part of Hydroxyethyl 3. e) Runtimes for the complete substructure-adding process on databases containing different data sets of protein-ligand complexes. f)-h) Number of detected hits in the SMARTS matching step plotted against the runtime for adding one of the three substructures to different databases, respectively. In each plot, a linear regression curve is shown as green dotted line.

'CCO' (see Figure 7.2a). They differ only in their recursive description of the first carbon (see Figures 7.2b-d).

The runtimes for adding each of the three substructures to databases containing different sets of PDB files are shown in Figure 7.2a. Overall, the runtime for adding Hydroxyethyl 1 is larger than for Hydroxyethyl 2 and Hydroxyethyl 3. The runtimes for Hydroxyethyl 2 and Hydroxyethyl 3 are almost identical. The complete procedure can be divided into two steps: (1) Preparation of protein-ligand complexes and EDIA values and (2) data collection. For Hydroxyethyl 1, the data collection takes about 50% of the complete runtime. For Hy-

droxyethyl 2 and Hydroxyethyl 3, this share is only about 20%. The data collection step again can be divided into the SMARTS matching procedure and handling of all detected hits. Interestingly, for Hydroxyethyl 1, the share of the SMARTS matching on the data collection step is only about 20%. For Hydroxyethyl 2 and Hydroxyethyl 3, the SMARTS matching requires about 78% and about 85% of the time for the data collection step. The reason here is probably that the for Hydroxyethyl 1, the SMARTS pattern matches very frequently and a large number of matches has to be handled. Hence, the handling of all results requires much more runtime than their detection. On the opposite, only a few results are detected for Hydroxyethyl 2 and 3. Hence, more time is required for their detection than for their subsequent preparation.

In Figure 7.2b, c, and d the overall runtime for the complete process of substructure-adding is plotted against the number of detected hits for each substructure, respectively. The correlation can be described with a linear regression line, indicated by the green dotted line in Figure 7.2b, c, and d. The lines strongly differ in their slope. This value indicates the time required for the detection and preparation of one hit. As expected, the slope for Carbonyl 1 is very small, indicating that the data collection step per hit is very fast here. However, the number of detected hits is very large which results in a long overall runtime. Interestingly, the overall runtimes for Hydroxyethyl 2 and Hydroxyethyl 3 are almost identical despite the much larger number of hits detected for Hydroxyethyl 2. Accordingly, the slope of the regression line for Hydroxyethyl 3 is much larger than the slope for Hydroxyethyl 2. This is a result of the more time consuming SMARTS matching per hit for Hydroxyethyl 3.

These results show that the runtime required for adding of a substructure growths linearly with the number of detected hits. Since every hit has to be handled individually, this behavior of the runtime cannot be changed.

For all three substructures, the first preparation of the data is one of the most time consuming steps. This step includes the reconstruction of protein-ligand complexes from the database and the reconstruction of the EDIA values for all atoms. During this step, the same reconstruction procedures are used for each of the protein-ligand complexes. Hence, even a slight improvement of the required runtime here can lead to much faster runtimes for the overall reconstruction step. This could be achieved by using a more efficient way of storing the reconstructed data. However, this procedure is only performed once every time new substructures are added to the database. Thus, it is favorable to add several substructures at a time.

For more complicated SMARTS pattern, the SMARTS matching is also an highly time consuming step. An acceleration of this step could therefore also lead to shorter overall runtimes. This could be achieved by using fingerprint techniques. Since the fragment part of the substructure description used here always describes a unique molecular fragment, fingerprints which store the occurrence of specific substructures in molecules can be applied. Using this fingerprint, the number of molecules in which the SMARTS pattern occurs could be

Filter criteria for partner points	Hydroxyethyl 1	Hydroxyethyl 2	Hydroxyethyl 3
no filter	$2.8{\cdot}10^7$ pps,	$7{\cdot}10^5$ pps,	2 235 pps,
	134 s $\pm$ 5 s	3.7 s $\pm$ 0.4 s	$<$1 s
element type = oxygen	$1.9{\cdot}10^7$ pps,	$4.6{\cdot}10^5$ pps,	1 636 pps,
	96 s $\pm$ 2 s	2.2 s $\pm$ 0.1 s	$<$1 s
element type = oxygen,	$2.4{\cdot}10^5$ pps,	$5.2{\cdot}10^4$ pps,	143 pps,
location = ligand	4.4 s $\pm$ 0.2 s	$<$1 s	$<$1 s

Table 7.1.: Runtimes and number of received partner points (pps) for three different filters on three different substructures based on the $PDB_{2.5}$ data set.

reduced rapidly and the exact matching algorithm has to be performed only on the relevant molecules. A similar approach is applied by Relibase and Relibase+ for substructure mining in small molecules.

However, the results presented in this section show that about $5{\cdot}10^6$ substructures can be detected and inserted into the database in about 160 minutes. In the typical use case, this step is performed only once and the database is afterwards used for several analyses. Hence, an interactive behavior is not as important as in the filtering step and runtimes of about 160 minutes are probably tolerable.

7.4. Filtering

As a last performance measurement, the runtime behavior for database filtering is tested. To this end, three different queries were tested on a database containing $PDB_{2.5}$ and the three substructures used in the previous experiment (Hydroxyethyl 1, Hydroxyethyl 2, and Hydroxyethyl 3). The queries differ in their exact definition. The first query only uses the substructure as a filter attribute whereas the second and third query use more specific selections of filter attributes for the requested partner points. A more detailed description of the queries can be found in Section 5.3.5. The used filter criteria as well as the number of resulting partner points, and the required runtime is shown in Table 7.1.

Overall, the runtime of a query is longer the more partner points are returned. However, up to $1.9{\cdot}10^7$ partner point can be retrieved in a reasonable short time of about 90s, which still supports an interactive use of *NAOMI*nova. The number of returned partner points depends on the substructure and how often it has been detected in the used data set. If substructures which are even more frequent than Hydroxyethyl 1 are used, even longer runtimes have to be expected. Given the growth rate of the PDB, this scenario is very likely in the coming years.

According to the statement of Allen and Owens, an SQLite database can achieve very fast runtimes for simple statements, e.g., simple SELECT statements [98]. It is therefore unlikely that a transfer of the *NAOMI*nova database to a similar server-based database would largely improve the runtime. Moreover, a weak influence of the hardware settings could be observed: the filter process for Hydroxyethyl 1 has an equal runtime on the SSD as well as on the HDD settings.

Other database architectures exist which could be useful in reducing the retrieval time. On the one hand, there are database managements systems which store the data row-wise instead of column-wise, as most of the traditional relational databases do. These database should be more optimized for reading data as for writing data which could be beneficial in the current scenario. One example of such a database is C-Store [123]. A second idea could be the use of a database which is able to execute an SQL query in a parallel way. In these databases, a single SQL query is executed by different processing units which can lead to much faster retrieval times [124].

Despite the longer runtime for the retrieval of large sets of partner points, the main strength of *NAOMI*nova is that the method is able to calculate partner point distributions for custom-defined substructures on custom defined data sets of macromolecular structures. To my knowledge, no other available tool provides this functionality. Even on large sets of protein-ligand structures, results for up to $1.9 \cdot 10^7$ partner points can be retrieved in about 90 s with *NAOMI*nova.

7.5. Application Example

In this section, one application example is presented which should demonstrate the usefulness of *NAOMI*nova for analyzing interactions geometries in proteins. Within this example, preferred interaction geometries of a specific functional group are analyzed on a large data set of protein structures. A more extensive presentation of *NAOMI*nova's capabilities has recently been published by Nittinger *et al.* [16]. In this study, interaction preferences for a large set of different functional groups has been deduced with the help of *NAOMI*nova.

Exemplary, the distribution of interacting atoms around the substructure Hydroxyethyl 1 (see Section 5.3.4 for its exact definition) in proteins' side chains is analyzed based on the $PDB_{2.5}$. In a first step, all partner points with an EDIA ≥ 0.8 and in a distance between 2.6 and 3.5 Å to the substructure's oxygen are filtered with *NAOMI*nova. This distance has been chosen in order to only hit atoms which build an atomic interaction with the central substructure. In total, 2 044 166 partner points are detected. Most of these points are nitrogens or oxygens (516 729 and 1 518 171, respectively). In Figure 7.3a, b, c, and d, the distributions for both element types are shown.

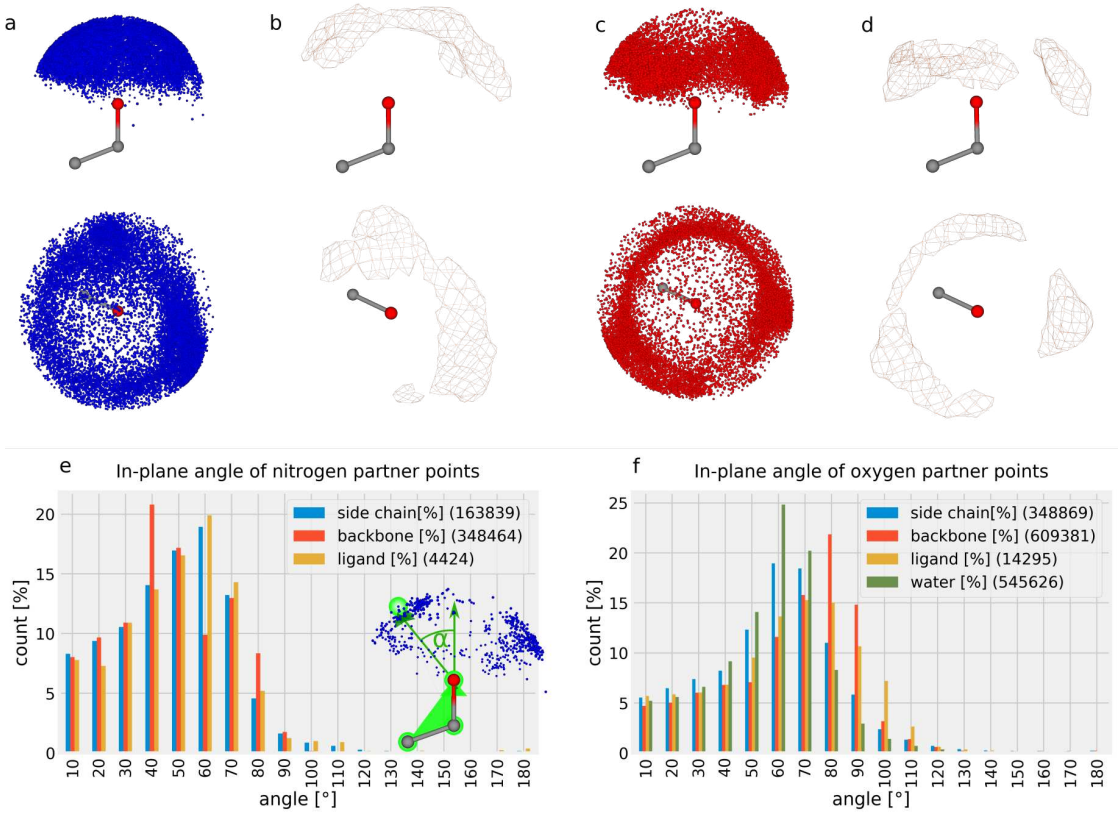


Figure 7.3.: Distribution of nitrogen and oxygen atoms around Hydroxyethyl 1 in proteins' side chains. Filter criteria in *NAOMI*nova: origin central substructure = side chain, EDIA ≥ 0.8, distance to oxygen between 2.6 and 3.5 Å, element type partner = nitrogen (a and b) or = oxygen (c and d), respectively. a), b) Side and top view of the distribution of nitrogens around Hydroxyethyl 1 in the protein's side chain. c), d) Side and top view of the distribution of oxygens around Hydroxyethyl 1 in the protein's side chain. In a) and c) partner points are displayed as sphere. In b) and d) partner points are displayed as density grid. e), f) Histogram showing the in-plane angle of partner points in percent of the total partner point count for different origins for nitrogen and oxygen, respectively. The value behind each category represents the absolute number of detected partner points. Schematically, the measurement of the in-plane angle is displayed on the right part of e).

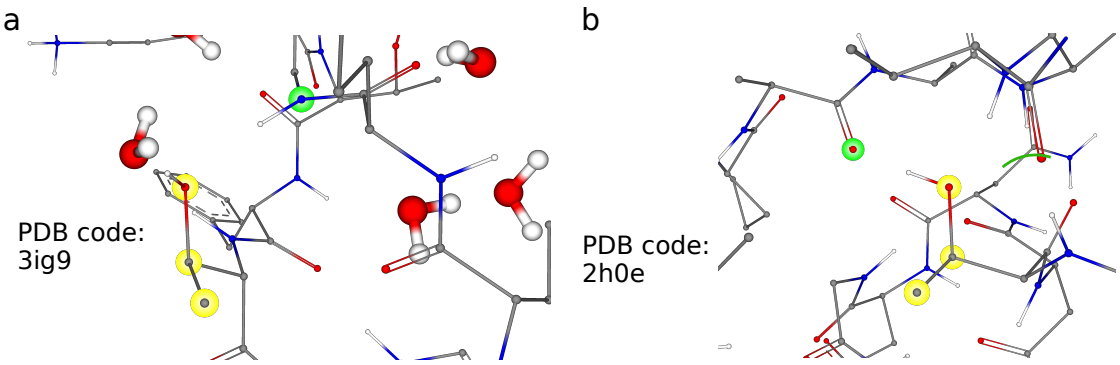


Figure 7.4.: Examples for original structures in which the Hydroxyethyl 1 in a protein's side chain provided by the backlink functionality in *NAOMI*nova. The atoms of the substructure are highlighted in yellow. The partner atom is highlighted in green a) In PDB structure 3ig9 [125] the substructure's oxygen acts as an acceptor in an atomic interactions with a nitrogen. b) In PDB structure 2h0e [126] the substructure's oxygen acts as a donor in an atomic interactions with another oxygen.

In Figure 7.3a and c, the distribution of partner points is displayed as spheres. However, the drawing of these sphere may be slow if two many partner points have to be drawn. Hence, if more than 15 000 partner points are detected, only randomly chosen 15 000 partner points are displayed. In Figure 7.3b and d, the distribution of the partner points is displayed as density grid. Here, all partner points are taken into account for the calculation.

From these visualizations, it can directly be seen that the distributions of the two element types are not equal. It seems that more nitrogens can be found on top of the substructure's oxygen than oxygens. In order to confirm this impression, the in-plane angle of all partner points was measured. To this end, the plane defined by the three atoms of the substructure was used as reference. Each partner point was first transferred to the defined plane in parallel to the plane's normal. Afterwards, the angle between its connection to the oxygen and C-O axis in the plane was calculated. Schematically, the performed measurement is depicted for one partner point in Figure 7.3e. The resulting in-plane angles separated by different origins of the partner points are displayed in Figure 7.3e and f. From these distributions, it can be seen that partner points from ligands and from the amino acids' side chains have their peak between 60° and 70° for nitrogen and oxygen. A strong difference can be seen for the distribution of partner points from amino acids' backbones. In this category, the nitrogens have their peak angle at about 40° whereas oxygens have their peak angle between 70 and 80°. According to other studies, the optimal angle for a hydrogen bond is believed to be at 60° in this measurement [16].

One possible explanation for these differences are steric hindrances of the backbone atoms which lead to sub-optimal interaction geometries. However, a more detailed look at the data also points to another possibility. The substructure's oxygen is part of an hydroxyl and can therefore act as donor and as acceptor in hydrogen bonds. Using the backlink functionality

in *NAOMI*nova, the original structures of different partner points from the backbone were analyzed. Selected examples are displayed in Figure 7.4. It can be seen that in atomic interactions with nitrogen, the substructure's oxygen always acts as an acceptor. Because of its planar geometry, the backbone's nitrogen can never act as an acceptor. On the other hand, when the substructure's oxygen is interacting with a backbone's oxygen, it always acts as an donor. Hence, the different in-plane angles might also be a result of the different functionalities within a hydrogen bond.

This example shows how the interactive tool *NAOMI*nova can be used to handle large sets of data and provide the relevant information to the user. Geometrical preferences of different substructures in different parts of the protein can simply be analyzed by visual inspection and by means of geometrical measurements. For each partner point, the original protein structure can be traced back which allows a convenient opportunity to detect additional features of specific points and rate their relevance to the current problem of interest.

8
Conclusion

In this thesis, two different methods for the mining of interaction geometries in macromolecular structures were developed. Pelikan is able to detect bioisosters and chemoisosters in collections of protein-ligand interfaces. *NAOMI*nova can be used to identify and visualize geometrical interaction preferences of specific substructures in protein-ligand complexes as well as in protein structures. For both tools, different experiments have been performed demonstrating their correctness and their performance for different input parameters. Both tools can be used to analyze atomic interactions and to get detailed insights into the nature of molecular recognition. Furthermore, it has been shown that these tools are highly beneficial for various applications related to drug design.
Pelikan as well as *NAOMI*nova are stand-alone tools which stably run on different platforms. Pelikan is a part of the software bundle provided by the 'AMD Software Server', Universität Hamburg (https://software.zbh.uni-hamburg.de/), and can be downloaded and used for free for academic use. In the near future also *NAOMI*nova will be a part of this collection of software tools.
In the following sections, the main achievements of these tools are highlighted. Afterwards, their limitations are presented. Finally, an outlook is given on potential improvements and further developments of both approaches.

8.1. Achievements

The main achievement of this work is the development of two stand-alone tools which enable the mining of interaction geometries in collection of macromolecular structures. Both tools present an outstanding combination of flexibility, specificity, and correctness in comparison to other existing methods. During the development, a focus has been laid to five specific aspects, listed in Section 3. In the following, for each of these aspects, the achievements of this work will be outlined in detail.

Consistent handling of data

Both tools use the NAOMI library for reading and interpreting the molecular structures from PDB files. Also the auxiliary tools like Protoss and DoGSite used for preprocessing of the data are based on these libraries. Hence, throughout this work the same fundamental chemical models are used which have shown to be consistent and deterministic [76]. The tool Protoss has been shown to achieve almost optimal results for hydrogen bond networks when compared with hand curated data sets [38]. A very important precondition when non-covalent atomic interactions are to be investigated. Moreover, Pelikan and *NAOMI*nova make use of the database concept from the NAOMI library. This allows an interchange of databases between different tools from the NAOMI library, e.g., with Mona [84, 87].

Both tools extend basic functionalities from the NAOMI library in order to make the relevant data accessible to the search procedure. This process is deterministic as from the same set of input data, the same database is constructed every time and the same results are received for identical queries. Dedicated tests for both tools ensure that information is not changed throughout the data collection steps.

Reliable and correct retrieval system

For both tools, the correctness of the search process was assessed by checking for false positive as well as for false negative results. Herein, random atoms were picked for which specific queries were generated. The correctness of the results for each query was verified. Several repetitions of this test without detected error lead to the conclusion that both methods are working correctly. Throughout the development, these test have brought to light different calculation and implementation errors which underlines their usefulness. For the final versions of both tools, no errors could be found.

For Pelikan, the correctness of a search was also compared with the tool Relibase. Also in this experiment, the search procedure proved to be correct.

Retrieval speed

As both tools are meant to be used interactively, the retrieval speed is of high importance. For both tools, the runtime of the retrieval system was analyzed in detail. The number of results could be identified as one of the most runtime-influencing parameter. In general, it can be concluded that the more precise a query, the fewer results are generated resulting in faster retrieval times. For typical queries, reasonable short runtimes could be achieved on a standalone PC which support interactive use. For Pelikan, a comparison with the tool Relibase elucidated advantages of its search mechanism for queries involving substructures of the protein are used.

High variability

Both tools developed in the course of this thesis provide flexibility with respect to both variable inputs: (1) The used data set and (2) the queries.

Both tools offer the functionality to generate a database out of any collection of PDB files. Since SQLite is used, databases can easily be exchanged between users and platforms. To my knowledge, no other existing tool provides this possibility. For *NAOMI*nova, also the used substructures can be defined by a user with the help of the powerful SMARTS language. This feature in combination with the variable set of used PDB files make *NAOMI*nova the only available tool supporting a complete variability of the used data.

Pelikan supports queries which are precise and at the same time flexible. The developed query system can be used to precisely define spatial arrangements of atoms in protein-ligand interfaces. Herein, no aggregation of specific amino acids or other chemical structures is done. Due to the allowed ranges of distance and angle queries, the precision of a query can be adjusted for parts of the query leading to more flexibility. This 3D query can be combined with a large variety of filters for textual and numerical properties of the ligand, the protein, the pocket, and the protein-ligand complex.

In *NAOMI*nova, the query variability has been realized by supporting filters for various properties of the central substructure and the partner points. Most importantly, the EDIA is integrated in *NAOMI*nova which provides the possibility to select only substructures and partner points which are experimentally well supported.

Usability

GUIs were developed for both tools which can be used to create databases, define queries and substructures, and inspect the results. In Pelikan, the 3D structures of the results can be superimposed on that of the query which helps to easily spot differences and similarities of all results. In *NAOMI*nova, different measurements can be done and the data of a complete set of partner points can be plotted as an histogram. Moreover, for each partner point the original structure can be visualized which is important to infer reasons for abnormal arrangements.

Both tools offer the possibility to export resulting data. In Pelikan, statistics on the the results can be written to a file whereas in *NAOMI*nova, all raw data of a measurement can be exported.

Due to the used database system, no server infrastructure is needed in both cases. After downloading the tools, a user can directly start to create databases or to search structures and distributions of interest.

8.2. Limitations

Besides the achievements presented in the previous section, both tools also have limitations. Based on the main source for the limitation, they can be divided into four different groups. First of all, there are limitations which derive from the used database technology. Secondly,

the use of the NAOMI library leads to restrictions. The third group deals with limitations derived from the used data. Finally, the specific algorithms implemented in both tools lead to restrictions. In the following, the limitations will be discussed in detail for each group.

Database technology

It has been decided to use SQLite because of its convenient way to interchange databases between different users and platforms. These advantages, however, come at a cost. In general, it could be shown that queries which lead to a high number of results lead to longer retrieval times. This is in part due to the runtime of the database queries. Given the growth of the available macromolecular structures, this will become more and more relevant in the near future. However, the targeted application scenario of Pelikan is to search for a specific pattern which leads only to a small number of results. Hence, even with growing data sets, the retrieval times might be sufficient. The limitation has a stronger effect on *NAOMI*nova since here the querying of large numbers of partner points is a typical task.

Moreover, the used SQLite databases are not optimized for multi-user purposes. Hence, if databases for Pelikan and *NAOMI*nova should be shared among several users, they have to be copied. Especially for databases containing large numbers of PDB files, this can be inconvenient. Updates have to be performed for each copy of a database.

NAOMI library

In principle, the NAOMI library is a very good basis for the development of computer-based approaches in the field of drug-design. However, the used data structures also limit the possible functionalities. First of all, a complex in NAOMI is based on a differentiation between protein and small molecules. In some cases, for example if peptides are present in a structure, this classification can be disadvantageous depending on the desired application. Once a molecule has been classified as a small molecule during the complex initialization process, it cannot easily be turned into a protein. Since the classification into small molecules and proteins is used in both tools, it would be desirable to have a more flexible classification procedure. Otherwise, unintended results might be detected or important results are never found.

Moreover, the NAOMI complex initialization process is not able to handle flexible structures. PDB files resulting from molecular dynamics simulations usually contain several coordinates for the same atom encoding the flexibility of the structure. As seen in the comparison between Pelikan and Relibase, this can lead to false negative results since only the first annotated structure is used in the NAOMI library. Also for structures derived from X-ray crystallography, alternate locations of atoms might be annotated in the PDB file which are not handled in the NAOMI library.

Data source

The main data source for both tools developed here is the PDB. The amount of deposited structures here is ever increasing and also the diversity of the proteins increased during the last years. However, there are still proteins who's structures have been elucidated very often and which are therefore overrepresented in the PDB. At the same time, there are proteins which are difficult to crystallize or which are no typical target of drug design projects. Structures of these proteins are often underrepresented in the PDB. This might lead to a bias in the performed analyzes and might have an influence on the drawn conclusions.

Concretely, if Pelikan is used to find specific interaction patterns, it is not directly clear if the results derive from different structures of the same protein or from different proteins. Even the inspection of all results might not get a clear decision of the former problem since the naming of proteins in PDB files is very inconsistent. In a similar way, such a bias might lead to wrong conclusion if in *NAOMI*nova a specific patch of atoms around a substructure is detected.

Specific limitations

In both tools, the structures of proteins and their ligands are treated as rigid structures. However, under physiological conditions, these molecules are flexible and also the binding between two molecules is not a rigid system. This flexibility is in part reflected by the possibility to define ranges for geometric constraints. However, a user has to know the size of movements in the particular region.

The position of hydrogens and the exact tautomeric states of all molecules are determined with the tool Protoss in this work. In addition, the optimal mesomeric forms of the molecule are determined by the NAOMI library during the initialization process. Afterwards, both tools handle these states as fixed and unchangeable. Hence, if a specific substructure is searched in *NAOMI*nova, delocalized bonds and hydrogen positions have to be handled with care. The SMARTS language provides the term '$\sim$' which matches any bond. However, expression for substructures can get very difficult if variable positions for bonds and hydrogens are considered. The same holds true if the environment of a search point is defined by a SMARTS patterns in Pelikan.

The tool Pelikan uses only the structural information of protein-ligand interfaces. This limits the use cases of the tool to projects dealing with the binding between a small molecule and a protein. Other interfaces such as protein-protein binding or intra-molecular binding in a protein cannot be investigated with the tools as presented here. Moreover, in Pelikan only the resolution of a structure can be used as quality criterion.

In both tools, the logical combination of filters is currently limited. All filter components are combined with a logical 'AND'. In some cases, a filter component can be negated and in other cases, an 'OR' combination is possible. The possibility to combine all filter components with the logical operators 'AND', 'OR', and 'NOT' would even increase the flexibility and

precision of the search process of both tools.

8.3. Outlook

With regard to the outlined limitations, there are several possibilities to improve Pelikan and *NAOMI*nova.

First of all, both tools could be transferred to server-based database systems in order to speed-up the retrieval times. Concerning Pelikan, a database management system with an advanced query optimizer would probably beneficial. Here, the database systems PostgreSQL (http://www.postgresql.org) or MySQL (https://www.mysql.com/) could be used. A first attempt to transfer Pelikan to PostgreSQL has already been done by A. Elvers in her bachelors thesis [127]. This work shows that a transfer is in principle possible but a specific optimization of the search process is still required to achieve good retrieval times.

For *NAOMI*nova, a database system which is optimized to handle large amounts of results would be a good choice. For example, a database system which supports the parallel execution of queries or which uses a different scheme for storing the data.

These transfers would have the beneficial side-effect that both tools could be made available via a web service and multiple users could be handled at the same time. However, this would reduce the flexibility of both tools as it would be more complicated to generate a database from a specific set of PDB files. Hence, the best way would be to support both applications: a user can generate specific databases using the SQLite technology. At the same time, large databases containing all relevant files from the PDB are provided via a web service.

The complex initialization procedure in the NAOMI library could be changed such that the classification into protein and small molecules becomes more flexible. This way, the classification of molecules for both tools could be driven by the user. For example, a list of molecular patterns could be provided by a user which defines all ligands or which defines small molecules which should not be categorized as ligands, e.g., molecules from the solvent. At the same time, the complex in the NAOMI library could be improved such that more than one 3D coordinate is stored for each atom. Thereby, alternate locations of atoms and flexible parts in structures could be represented. The search procedures of Pelikan and *NAOMI*nova could then work on all available 3D coordinates and the flexibility of a structure could in part be reflected.

In line with the above handling of structural flexibility, the handling of different mesomeric, tautomeric, and protonation states of molecules could be handled if the concept of SMARTS and its matching procedure were extended. For example, a term for a delocalized bond could be introduced into the SMARTS language. During the matching procedure, this would then match to all bonds which are part of a molecular system with delocalized bonds. This would

highly facilitate and improve the definition of substructures in *NAOMI*nova and the definition of search point environments in Pelikan.

The bias of the used data sets can of course not be changed immediately. However, it would be beneficial if both tools would indicate a potential bias if a large part of the results is derived from different structures of the same protein. One possibility to do this would be to classify all proteins upon database construction. In *NAOMI*nova the optimal way for such a classification would be a sequence similarity. However, in Pelikan only protein-ligand interfaces are used. The tool Siena which is able to identify similar binding sites [128] could be used for classification of all pockets here. In both tools, the results could then be displayed using these clusters. Moreover, a search could be limited to those structures which are similar or not similar to a specific structure or protein-ligand interface loaded by the user. All these functionalities would help to inform the user and manage the structural bias of a data set.

Specifically in Pelikan, there are four aspects which would improve this tool. First of all, the EDIA should be introduced such that only results having a specific EDIA value are returned. Secondly, the system should be extended to other interfaces, e.g., protein-protein interfaces. This would probably increase the database size. However, if the database is transferred to a more sophisticated database management system, this data extension may be tolerable. Thirdly, the handling of results can be improved in Pelikan. At the moment, resulting structures can be displayed in a 3D viewer and several results can be superimposed. However, if a large number of results is detected, a visual inspection is not feasible. To this end, it would be beneficial to have an algorithm which analyzes the results and detects spatial similarities outside the matching atoms. This information could on the one hand be used to cluster the results and provide a more comprehensive overview over the results. On the other hand, it could be used to suggest a refined 3D query. Finally, as already mentioned in Section 6.3, the setup of the triangle descriptor could be optimized in order to further speed-up 3D queries.

Also the tool *NAOMI*nova can be improved by three different aspects. Firstly, it would be helpful to extend the set of filters for partner point properties by 'is donor' and 'is acceptor'. In this way, the specific role of an atom in a hydrogen bond could be investigated. Secondly, the set of elements which can be used to filter partner points could be extended by 'carbon'. This would also allow the analysis of weak hydrogen bonds. However, the size of the database would largely increase because carbon atoms are very frequent in proteins and ligands. After a transfer to a server-based database system, this increase in data size is probably tolerable. Finally, it could be interesting to combine the used data with affinity values for protein-ligand complexes. Then, the distribution of partner points could for example be colored by their corresponding affinity values which allows a direct link between interaction geometry and affinity.

In both tools, the combination of filter components could be extended such that all components could be combined with the logical operators 'AND', 'OR', and 'NOT'. This would largely improve the flexibility of the possible queries.

This conclusion shows that Pelikan and *NAOMI*nova offer unique solutions for the fast and flexible search of interaction geometries in macromolecular structures. The possible extensions described above will further broaden their range of application and thus the benefit for researchers in the field of structure-based drug design.

Bibliography

[1] Emil Fischer. Einfluss der configuration auf die wirkung der enzyme. *Berichte der deutschen chemischen Gesellschaft*, 27(3):2985–2993, 1894.

[2] Jürgen Drews. Drug discovery: A historical perspective. *Science*, 287(5460):1960–1964, 2000.

[3] Amy C. Anderson. The process of structure-based drug design. *Chemistry & Biology*, 10(9):787 – 797, 2003.

[4] Bruce E. Maryanoff. Inhibitors of serine proteases as potential therapeutic agents: The road from thrombin to tryptase to cathepsin G. *Journal of Medicinal Chemistry*, 47(4):769–787, 2004. PMID: 14761180.

[5] Larry W. Hardy and Antony Malikayil. The impact of structure-guided drug design on clinical agents. *Current Drug Discovery*, 15:15–20, 2003.

[6] J Erickson, DJ Neidhart, J VanDrie, DJ Kempf, XC Wang, DW Norbeck, JJ Plattner, JW Rittenhouse, M Turon, N Wideburg, and al. et. Design, activity, and 2.8 a crystal structure of a c2 symmetric inhibitor complexed to hiv-1 protease. *Science*, 249(4968):527–533, 1990.

[7] Caterina Bissantz, Bernd Kuhn, and Martin Stahl. A medicinal chemist's guide to molecular interactions. *Journal of Medicinal Chemistry*, 53(14):5061–5084, 2010. PMID: 20345171.

[8] Jeffrey R. Deschamps and Clifford George. Advances in x-ray crystallography. *TrAC Trends in Analytical Chemistry*, 22(8):561 – 564, 2003.

[9] Jeffrey R. Deschamps. X-ray crystallography of chemical compounds. *Life Sciences*, 86(15–16):585 – 589, 2010. Emerging Research Technologies for Medications Development: Focus on Drugs of Abuse.

[10] Andrew M. Davis and Simon J. Teague. Hydrogen bonding, hydrophobic interactions, and failure of the rigid receptor hypothesis. *Angewandte Chemie International Edition*, 38(6):736–749, 1999.

[11] Hugo Kubinyi. *Hydrogen Bonding: The Last Mystery in Drug Design?*, pages 513–524. Verlag Helvetica Chimica Acta, 2007.

[12] Sunil K. Panigrahi and Gautam R. Desiraju. Strong and weak hydrogen bonds in the protein–ligand interface. *Proteins: Structure, Function, and Bioinformatics*, 67(1):128–141, 2007.

[13] Sanjay Sarkhel and Gautam R. Desiraju. Nh...o, oh...o, and ch...o hydrogen bonds in protein–ligand complexes: Strong and weak interactions in molecular recognition. *Proteins: Structure, Function, and Bioinformatics*, 54(2):247–259, 2004.

[14] J. E. J. Mills and P. M. Dean. Three-dimensional hydrogen-bond geometry and probability information from a crystal survey. *Journal of Computer-Aided Molecular Design*, 10(6):607–622, 1996.

[15] Zhiguo Liu, Guitao Wang, Zhanting Li, and Renxiao Wang. Geometrical preferences of the hydrogen bonds on protein-ligand binding interface derived from statistical surveys and quantum mechanics calculations. *Journal of Chemical Theory and Computation*, 4(11):1959–1973, 2008. PMID: 26620338.

[16] Eva Nittinger, Therese Inhester, Stefan Bietz, Agnes Meyder, Karen T. Schomburg, Gudrun Lange, Robert Klein, and Matthias Rarey. Large-scale analysis of hydrogen bond interaction patterns in protein-ligand interfaces. *Journal of Medicinal Chemistry*, 60(10):4245–4257, 2017. PMID: 28497966.

[17] Bernd Kuhn, Peter Mohr, and Martin Stahl. Intramolecular hydrogen bonding in medicinal chemistry. *Journal of Medicinal Chemistry*, 53(6):2601–2611, 2010. PMID: 20175530.

[18] Gautam R. Desiraju. C-h...o and other weak hydrogen bonds. from crystal engineering to virtual screening. *Chem. Commun.*, pages 2995–3001, 2005.

[19] Emmanuel A. Meyer, Ronald K. Castellano, and François Diederich. Interactions with aromatic rings in chemical and biological recognition. *Angewandte Chemie International Edition*, 42(11):1210–1250, 2003.

[20] Jennifer C. Ma and Dennis A. Dougherty. The cation-pi interaction. *Chemical Reviews*, 97(5):1303–1324, 1997. PMID: 11851453.

[21] M.R. Battaglia, A.D. Buckingham, and J.H. Williams. The electric quadrupole moments of benzene and hexafluorobenzene. *Chemical Physics Letters*, 78(3):421 – 423, 1981.

[22] Leo A. Hardegger, Bernd Kuhn, Beat Spinnler, Lilli Anselm, Robert Ecabert, Martine Stihle, Bernard Gsell, Ralf Thoma, Joachim Diez, Jörg Benz, Jean-Marc Plancher, Guido Hartmann, David W. Banner, Wolfgang Haap, and François Diederich. Systematic investigation of halogen bonding in protein–ligand interactions. *Angewandte Chemie International Edition*, 50(1):314–318, 2011.

[23] Suman Sirimulla, Jake B. Bailey, Rahulsimham Vegesna, and Mahesh Narayan. Halogen interactions in protein–ligand complexes: Implications of halogen bonding for rational drug design. *Journal of Chemical Information and Modeling*, 53(11):2781–2791, 2013. PMID: 24134121.

[24] Rainer Wilcken, Markus O. Zimmermann, Andreas Lange, Andreas C. Joerger, and Frank M. Boeckler. Principles and applications of halogen bonding in medicinal chemistry and chemical biology. *Journal of Medicinal Chemistry*, 56(4):1363–1388, 2013. PMID: 23145854.

[25] Duncan E. Scott, Andrew R. Bayly, Chris Abell, and John Skidmore. Small molecules, big targets: drug discovery faces the protein-protein interaction challenge. *Nat Rev Drug Discov*, 15(8):533–550, August 2016.

[26] Cody J. Wenthur, Patrick R. Gentry, Thomas P. Mathews, and Craig W. Lindsley. Drugs for allosteric sites on receptors. *Annual Review of Pharmacology and Toxicology*, 54(1):165–184, 2014. PMID: 24111540.

[27] R. D. Taylor, P. J. Jewsbury, and J. W. Essex. A review of protein-small molecule docking methods. *Journal of Computer-Aided Molecular Design*, 16(3):151–166, March 2002.

[28] Elizabeth Yuriev, Jessica Holien, and Paul A. Ramsland. Improvements, trends, and new ideas in molecular docking: 2012–2013 in review. *Journal of Molecular Recognition*, 28(10):581–604, 2015. JMR-14-0150.R1.

[29] Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. PMID: 26479676.

[30] Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. Pubchem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202, 2016.

[31] Evanthia Lionta, George Spyrou, Demetrios K. Vassilatis, and Zoe Cournia. Structure-based virtual screening for drug discovery: Principles, applications and recent advances. *Current Topics in Medicinal Chemistry*, 14(56):1923–1938, 2014.

[32] Markus Hartenfeller and Gisbert Schneider. Enabling future drug discovery by de novo design. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(5):742–759, 2011.

[33] Michael C. Sanguinetti and Martin Tristani-Firouzi. hERG potassium channels and cardiac arrhythmia. *Nature*, 440(7083):463–469, March 2006.

[34] Z. E. Sauna, M. M. Smith, M. Müller, K. M. Kerr, and S. V. Ambudkar. The mechanism of action of multidrug-resistance-linked P-glycoprotein. *Journal of Bioenergetics and Biomembranes*, 33(6):481–491, December 2001.

[35] Cambridge structural database; the cambridge crystallographic data centre, https://www.ccdc.cam.ac.uk/.

[36] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[37] Kamil Khafizov, Carlos Madrid-Aliste, Steven C. Almo, and Andras Fiser. Trends in structural coverage of the protein universe and the impact of the Protein Structure Initiative. *Proceedings of the National Academy of Sciences of the United States of America*, 111(10):3733–3738, March 2014.

[38] Stefan Bietz, Sascha Urbaczek, Benjamin Schulz, and Matthias Rarey. Protoss a holistic approach to predict tautomers and protonation states in protein-ligand complexes. *Journal of Cheminformatics*, 6(1):12, 2014.

[39] V. Sobolev, A. Sorokine, J. Prilusky, E. E. Abola, and M. Edelman. Automated analysis of interatomic contacts in proteins. *Bioinformatics (Oxford, England)*, 15(4):327–332, April 1999.

[40] Eva Nittinger, Nadine Schneider, Gudrun Lange, and Matthias Rarey. Evidence of water molecules – a statistical evaluation of water molecules based on electron density. *Journal of Chemical Information and Modeling*, 55(4):771–783, 2015. PMID: 25742501.

[41] Agnes Meyder, Eva Nittinger, Gudrun Lange, Robert Klein, and Matthias Rarey. Estimating electron density support for individual atoms and molecular fragments. *in preparation*, 2017.

[42] Andrew M. Davis, Stephen A. St-Gallay, and Gerard J. Kleywegt. Limitations and lessons in the use of x-ray structural information in drug design. *Drug Discovery Today*, 13(19–20):831 – 841, 2008.

[43] Marc C. Deller and Bernhard Rupp. Models of protein–ligand crystal structures: trust, but verify. *Journal of computer-aided molecular design*, 29(9):817–836, September 2015.

[44] Gregory L. Warren, Thanh D. Do, Brian P. Kelley, Anthony Nicholls, and Stephen D. Warren. Essential considerations for using protein–ligand structures in drug discovery. *Drug Discovery Today*, 17(23–24):1270 – 1281, 2012.

[45] Xavier Jalencas and Jordi Mestres. Chemoisosterism in the proteome. *Journal of Chemical Information and Modeling*, 53(2):279–292, 2013. PMID: 23312010.

[46] J An, T Nakama, Y Kubota, and A Sarai. 3dinsight: an integrated relational database and search tool for the structure, function and properties of biomolecules. *Bioinformatics*, 14(2):188, 1998.

[47] David Shirvanyants, Anastassia N. Alexandrova, and Nikolay V. Dokholyan. Rigid substructure search. *Bioinformatics*, 27(9):1327–1329, May 2011.

[48] Gabriel Gonzalez, Brett Hannigan, and William F. DeGrado. A Real-Time All-Atom Structural Search Engine for Proteins. *PLOS Computational Biology*, 10(7):e1003750, July 2014.

[49] Nurul Nadzirin, Eleanor J. Gardiner, Peter Willett, Peter J. Artymiuk, and Mohd Firdaus-Raih. Sprite and assam: web servers for side chain 3d-motif searching in protein structures. *Nucleic Acids Research*, 40(W1):W380, 2012.

[50] Nurul Nadzirin, Peter Willett, Peter J. Artymiuk, and Mohd Firdaus-Raih. Imaaagine: a webserver for searching hypothetical 3d amino acid side chain arrangements in the protein data bank. *Nucleic Acids Research*, 41(W1):W432, 2013.

[51] Adel Golovin and Kim Henrick. Msdmotif: exploring protein sites and motifs. *BMC Bioinformatics*, 9(1):312, 2008.

[52] Oliver Korb, Bernd Kuhn, Jérôme Hert, Neil Taylor, Jason Cole, Colin Groom, and Martin Stahl. Interactive and versatile navigation of structural databases. *Journal of Medicinal Chemistry*, 59(9):4257–4266, 2016. PMID: 26745458.

[53] Dominick Mobilio, Gary Walker, Natasja Brooijmans, Ramaswamy Nilakantan, R. Aldrin Denny, Jason DeJoannis, Eric Feyfant, Rupesh K. Kowticwar, Jyoti Mankala, Satish Palli, Sairam Punyamantula, Maneesh Tatipally, Reji K. John, and Christine Humblet. A protein relational database and protein family knowledge bases to facilitate structure-based design analyses. *Chemical Biology and Drug Design*, 76(2):142–153, 2010.

[54] Martin Weisel, Hans-Marcus Bitter, François Diederich, W. Venus So, and Rama Kondru. PROLIX: Rapid Mining of Protein–Ligand Interactions in Large Crystal Structure Databases. *Journal of Chemical Information and Modeling*, 52(6):1450–1461, June 2012.

[55] Manfred Hendlich, Andreas Bergner, Judith Günther, and Gerhard Klebe. Relibase: design and development of a database for comprehensive analysis of protein-ligand interactions. *Journal of Molecular Biology*, 326(2):607–620, February 2003.

[56] Psilo; chemical computing group inc., https://www.chemcomp.com/psilo-protein_structure_database_system.htm.

[57] Pldb; schroedinger llc, https://www.schrodinger.com/pldb.

[58] Coen Bron and Joep Kerbosch. Algorithm 457 Finding All Cliques of an Undirected Graph. *Commun ACM*, 16(9):575–577, September 1973.

[59] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.

[60] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sep 1976.

[61] Andreas Bergner, Judith Günther, Manfred Hendlich, Gerhard Klebe, and Marcel Verdonk. Use of relibase for retrieving complex three-dimensional interaction patterns including crystallographic packing effects. *Biopolymers*, 61(2):99–110, 2001.

[62] Ville-Veikko Rantanen, Konstantin A. Denessiouk, Mats Gyllenberg, Timo Koski, and Mark S. Johnson. A fragment library based on gaussian mixtures predicting favorable molecular interactions. *Journal of Molecular Biology*, 313(1):197 – 214, 2001.

[63] Riku Hakulinen, Santeri Puranen, Jukka V. Lehtonen, Mark S. Johnson, and Jukka Corander. Probabilistic Prediction of Contacts in Protein-Ligand Complexes. *PLOS ONE*, 7(11):e49216, November 2012.

[64] Juswinder Singh and Janet M. Thornton. Sirius. *Journal of Molecular Biology*, 211(3):595 – 615, 1990.

[65] Roman A. Laskowski, Janet M. Thornton, Christine Humblet, and Juswinder Singh. X-site: Use of empirically derived atomic packing preferences to identify favourable interaction regions in the binding sites of proteins. *Journal of Molecular Biology*, 259(1):175 – 201, 1996.

[66] Kota Kasahara, Matsuyuki Shirota, and Kengo Kinoshita. Comprehensive classification and diversity assessment of atomic contacts in protein–small ligand interactions. *Journal of Chemical Information and Modeling*, 53(1):241–248, 2013. PMID: 23186137.

[67] Ernesto Moreno and Kalet León. Geometric and chemical patterns of interaction in protein–ligand complexes and their application in docking. *Proteins: Structure, Function, and Bioinformatics*, 47(1):1–13, 2002.

[68] Ian J. Bruno, Jason C. Cole, Jos P.M. Lommerse, R. Scott Rowland, Robin Taylor, and Marcel L. Verdonk. Isostar: A library of information about nonbonded interactions. *Journal of Computer-Aided Molecular Design*, 11(6):525–537, 1997.

[69] Ville-Veikko Rantanen, Mats Gyllenberg, Timo Koski, and Mark S. Johnson. A bayesian molecular interaction library. *Journal of Computer-Aided Molecular Design*, 17(7):435–461, 2003.

[70] Juswinder Singh and Janet M. Thornton. *Atlas of Protein Side-Chain Interactions, Vols. I and II*. Oxford University Press, Oxford, U.K., 1992.

[71] Juswinder Singh, Jose Saldanha, and Janet M. Thornton. A novel method for the modelling of peptide ligands to their receptors. *Protein Engineering*, 4(3):251–261, February 1991.

[72] Kota Kasahara and Kengo Kinoshita. GIANT: pattern analysis of molecular interactions in 3d structures of protein–small ligand complexes. *BMC Bioinformatics*, 15:12, January 2014.

[73] Isostar user guide and tutorials 2017 csd release; the cambridge crystallographic data centre (ccdc), https://www.ccdc.cam.ac.uk/support-and-resources/ccdcresources/72d2b400918d4bffafe96677514ef413.pdf.

[74] Marcel L. Verdonk, Jason C. Cole, and Robin Taylor. Superstar: A knowledge-based approach for identifying interaction sites in proteins. *Journal of Molecular Biology*, 289(4):1093 – 1108, 1999.

[75] Marcel L. Verdonk, Jason C. Cole, Paul Watson, Val Gillet, and Peter Willett. Superstar: improved knowledge-based interaction fields for protein binding sites1. *Journal of Molecular Biology*, 307(3):841–859, March 2001.

[76] Sascha Urbaczek, Adrian Kolodzik, J Robert Fischer, Tobias Lippert, Stefan Heuser, Inken Groth, Tanja Schulz-Gasch, and Matthias Rarey. NAOMI On the Almost Trivial Task of Reading Molecules from Different File formats. *Journal of Chemical Information and Modeling*, 51(12):3199–3207, December 2011.

[77] Therese Inhester, Stefan Bietz, Matthias Hilbig, Robert Schmidt, and Matthias Rarey. Index-based searching of interaction patterns in large collections of protein–ligand interfaces. *Journal of Chemical Information and Modeling*, 57(2):148–158, 2017. PMID: 28128948.

[78] Stefan Bietz, Therese Inhester, Florian Lauck, Kai Sommer, Mathias M. von Behren, Rainer Fährrolfes, Florian Flachsenberg, Agnes Meyder, Eva Nittinger, Thomas Otto, Matthias Hilbig, Karen T. Schomburg, Andrea Volkamer, and Matthias Rarey. From cheminformatics to structure-based design: Web services and desktop applications based on the naomi library. *Journal of Biotechnology*, 261(Supplement C):207 – 214, 2017. Bioinformatics Solutions for Big Data Analysis in Life Sciences presented by the German Network for Bioinformatics Infrastructure.

[79] Therese Inhester, Eva Nittinger, Kai Sommer, Pascal Schmidt, Stefan Bietz, and Matthias Rarey. Naominova: Interactive geometric analysis of noncovalent interactions in macromolecular structures. *Journal of Chemical Information and Modeling*, 57(9):2132–2142, 2017. PMID: 28891648.

[80] Sascha Urbaczek, Adrian Kolodzik, and Matthias Rarey. The Valence State Combination Model A Generic Framework for Handling Tautomers and Protonation States. *Journal of Chemical Information and Modeling*, 54(3):756–766, March 2014.

[81] Sascha Urbaczek, Adrian Kolodzik, Inken Groth, Stefan Heuser, and Matthias Rarey. Reading PDB perception of molecules from 3d atomic coordinates. *Journal of Chemical Information and Modeling*, 53(1):76–87, January 2013.

[82] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Trans Pattern Anal Mach Intell*, 13(4):376–380, April 1991.

[83] Jose L. Blanco, David G. Lowe, and Marius Muja. Nanoflann, url=https://github.com/jlblancoc/nanoflann, 2011.

[84] Matthias Hilbig, Sascha Urbaczek, Inken Groth, Stefan Heuser, and Matthias Rarey. MONA – Interactive manipulation of molecule collections. *Journal of Cheminformatics*, 5(1):38, 2013.

[85] David Weininger, Arthur Weininger, and Joseph L Weininger. SMILES 2 Algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Modeling*, 29(2):97–101, May 1989.

[86] Karen T Schomburg, Stefan Bietz, Hans Briem, Angela M Henzler, Sascha Urbaczek, and Matthias Rarey. Facing the Challenges of Structure-Based Target Prediction by Inverse Virtual Screening. *Journal of Chemical Information and Modeling*, 54(6):1676–1686, June 2014.

[87] Matthias Hilbig and Matthias Rarey. MONA 2 A Light Cheminformatics Platform for Interactive Compound Library Processing. *Journal of Chemical Information and Modeling*, 55(10):2071–2078, October 2015.

[88] Andrea Volkamer, Daniel Kuhn, Thomas Grombacher, Friedrich Rippmann, and Matthias Rarey. Combining global and local measures for structure-based druggability predictions. *Journal of Chemical Information and Modeling*, 52(2):360–372, 2012. PMID: 22148551.

[89] Daylight Chemical Information Systems, Inc of Aliso Viejo, Ca. Daylight Theory Maunal, 2008.

[90] Hans-Christian Ehrlich and Matthias Rarey. Systematic benchmark of substructure search in molecular graphs - From Ullmann to VF2. *Journal of Cheminformatics*, 4(1):13, July 2012.

[91] Robert Schmidt. Efficient incremental search of variable molecular patterns. Master thesis in computer science, Universität Hamburg, 2017.

[92] Sqlite homepage; https://www.sqlite.org/.

[93] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[94] John W Raymond, Eleanor J Gardiner, and Peter Willett. RASCAL Calculation of Graph Similarity using Maximum Common Edge Subgraphs. *The Computer Journal*, 45(6):631–644, January 2002.

[95] Kai Sommer, Nils-Ole Friedrich, Stefan Bietz, Matthias Hilbig, Therese Inhester, and Matthias Rarey. Unicon: A powerful and easy-to-use compound library converter. *Journal of Chemical Information and Modeling*, 56(6):1105–1111, 2016. PMID: 27227368.

[96] Ronald C. Read and Derek G. Corneil. The graph isomorphism disease. *Journal of Graph Theory*, 1(4):339–363, 1977.

[97] D. L. Burk, W. C. Hon, A. K.-W. Leung, and A. M. Berghuis. Structural analyses of nucleotide binding to an aminoglycoside phosphotransferase,. *Biochemistry*, 40(30):8756–8764, 2001. PMID: 11467935.

[98] Grant Allen and Mike Owens. *The Definitive Guide to SQLite*. Apress, January 2011.

[99] Patrick Maass, Tanja Schulz-Gasch, Martin Stahl, and Matthias Rarey. Recore: A fast and versatile method for scaffold hopping based on small molecule crystal structure conformations. *Journal of Chemical Information and Modeling*, 47(2):390–399, 2007. PMID: 17305328.

[100] David Ryan Koes and Carlos J. Camacho. Pharmer: Efficient and exact pharmacophore search. *Journal of Chemical Information and Modeling*, 51(6):1307–1314, 2011. PMID: 21604800.

[101] Kesheng Wu. Fastbit: an efficient indexing technology for accelerating data-intensive science. *Journal of Physics: Conference Series*, 16(1):556, 2005.

[102] Jochen Schlosser and Matthias Rarey. Beyond the virtual screening paradigm: Structure-based searching for new lead compounds. *Journal of Chemical Information and Modeling*, 49(4):800–809, 2009. PMID: 19354328.

[103] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[104] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[105] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1–3):197 – 207, 2002. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

[106] David Ryan Koes and Carlos J. Camacho. Pharmer: Efficient and exact pharmacophore search. *Journal of Chemical Information and Modeling*, 51(6):1307–1314, 2011. PMID: 21604800.

[107] Ingo Schellhammer and Matthias Rarey. Trixx: structure-based molecule indexing for large-scale virtual screening in sublinear time. *Journal of Computer-Aided Molecular Design*, 21(5):223–238, 2007.

[108] Martin Jambon, Anne Imberty, Gilbert Deléage, and Christophe Geourjon. A new bioinformatic approach to detect common 3d sites in protein structures. *Proteins: Structure, Function, and Bioinformatics*, 52(2):137–145, 2003.

[109] Robert P. Sheridan, Ramaswamy Nilakantan, Andrew Rusinko, Norman Bauman, Kevin S. Haraki, and R. Venkataraghavan. 3dsearch: a system for three-dimensional substructure searching. *Journal of Chemical Information and Computer Sciences*, 29(4):255–260, 1989.

[110] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[111] Hans Matter, Elisabeth Defossa, Uwe Heinelt, Peter-Michael Blohm, Detlev Schneider, Andrea Müller, Silke Herok, Herman Schreuder, Alexander Liesum, Volker Brachvogel, Petra Lönze, Armin Walser, Fahad Al-Obeidi, and Peter Wildgoose. Design and quantitative structure–activity relationship of 3-amidinobenzyl-1h-indole-2-carboxamides as potent, nonchiral, and selective inhibitors of blood coagulation factor xa. *Journal of Medicinal Chemistry*, 45(13):2749–2769, 2002. PMID: 12061878.

[112] Marc Adler, Monica J. Kochanny, Bin Ye, Galina Rumennik, David R. Light, Sara Biancalana, and Marc Whitlow. Crystal structures of two potent nonamidine inhibitors bound to factor xa,. *Biochemistry*, 41(52):15514–15523, 2002. PMID: 12501180.

[113] Chad A. Van Huis, Agustin Casimiro-Garcia, Christopher F. Bigge, Wayne L. Cody, Danette A. Dudley, Kevin J. Filipski, Ronald J. Heemstra, Jeffrey T. Kohrt, Robert J. Leadley Jr., Lakshmi S. Narasimhan, Thomas McClanahan, Igor Mochalkin, Michael Pamment, J. Thomas Peterson, Vaishali Sahasrabudhe, Robert P. Schaum, and Jeremy J. Edmunds. Exploration of 4,4-disubstituted pyrrolidine-1,2-dicarboxamides as potent, orally active factor xa inhibitors with extended duration of action. *Bioorganic & Medicinal Chemistry*, 17(6):2501 – 2511, 2009. Special Issue: Natural Products in Medicinal Chemistry.

[114] Stefan Senger, Máire A. Convery, Chuen Chan, and Nigel S. Watson. Arylsulfonamides: A study of the relationship between activity and conformational preferences for a series of factor xa inhibitors. *Bioorganic & Medicinal Chemistry Letters*, 16(22):5731 – 5735, 2006.

[115] Anastasia Tziridis, Daniel Rauh, Piotr Neumann, Petr Kolenko, Anja Menzel, Ulrike Bräuer, Christian Ursel, Peter Steinmetzer, Jörg Stürzebecher, Andrea Schweinitz, Torsten Steinmetzer, and Milton T. Stubbs. Correlating structure and ligand affinity in drug discovery: a cautionary tale involving second shell residues. *Biological Chemistry*, 395(7-8):891–903, 2014.

[116] Kaspar Schärer, Martin Morgenthaler, Ralph Paulini, Ulrike Obst-Sander, David W. Banner, Daniel Schlatter, Jörg Benz, Martine Stihle, and François Diederich. Quantification of cation-π interactions in protein–ligand complexes: Crystal-structure analysis of factor xa bound to a quaternary ammonium ion ligand. *Angewandte Chemie International Edition*, 44(28):4400–4404, 2005.

[117] Andrei N. Lupas, Hongbo Zhu, and Mateusz Korycinski. The Thalidomide-Binding Domain of Cereblon Defines the CULT Domain Family and Is a New Member of the β-Tent Fold. *PLOS Computational Biology*, 11:e1004023, January 2015.

[118] Gabriel Martínez-Botella, James T. Loch, Oluyinka M. Green, Sameer P. Kawatkar, Nelson B. Olivier, P. Ann Boriack-Sjodin, and Thomas A. Keating. Sulfonylpiperidines as novel, antibacterial inhibitors of gram-positive thymidylate kinase (tmk). *Bioorganic and Medicinal Chemistry Letters*, 23(1):169 – 173, 2013.

[119] Isabel Da Fonseca, Insaf A. Qureshi, Ritcha Mehra-Chaudhary, Karina Kizjakina, John J. Tanner, and Pablo Sobrado. Contributions of unique active site residues of eukaryotic udp-galactopyranose mutases to substrate recognition and active site dynamics. *Biochemistry*, 53(49):7794–7804, 2014. PMID: 25412209.

[120] Javier García-Nafría, Jennifer Timm, Charlotte Harrison, Johan P. Turkenburg, and Keith S. Wilson. Tying down the arm in *Bacillus* dUTPase: structure and mechanism. *Acta Crystallographica Section D*, 69(8):1367–1380, Aug 2013.

[121] Lijun Zhou, Jing Hang, Yulin Zhou, Ruixue Wan, Guifeng Lu, Ping Yin, Chuangye Yan, and Yigong Shi. Crystal structures of the Lsm complex bound to the 3[prime] end sequence of U6 small nuclear RNA. *Nature*, 506(7486):116–120, February 2014.

[122] Karen Schomburg, Hans-Christian Ehrlich, Katrin Stierand, and Matthias Rarey. From structure diagrams to visual chemical patterns. *Journal of Chemical Information and Modeling*, 50(9):1529–1535, 2010. PMID: 20795706.

[123] Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik. C-store: A column-oriented dbms. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 553–564. VLDB Endowment, 2005.

[124] David DeWitt and Jim Gray. Parallel database systems: The future of high performance database systems. *Commun. ACM*, 35(6):85–98, June 1992.

[125] Li Qin, Andrei Fokine, Erin O'Donnell, Venigalla B. Rao, and Michael G. Rossmann. Structure of the small outer capsid protein, soc: A clamp for stabilizing capsids of t4-like phages. *Journal of Molecular Biology*, 395(4):728 – 741, 2010.

[126] Du-Kyo Jung, Youra Lee, Sung Goo Park, Byoung Chul Park, Ghyung-Hwa Kim, and Sangkee Rhee. Structural and functional analysis of PucM, a hydrolase in the ureide pathway and a member of the transthyretin-related protein family. *Proceedings of the National Academy of Sciences*, 103(26):9790–9795, June 2006.

[127] Antonia M. Elvers. Räumlich Anfragen von Proteinstrukturmustern: Portierung von SQLite zu serverbasierten relationalen Datenbanken. Bachelor thesis in computer science, Universität Hamburg, 2017.

[128] Stefan Bietz and Matthias Rarey. Siena: Efficient compilation of selective protein binding site ensembles. *Journal of Chemical Information and Modeling*, 56(1):248–259, 2016. PMID: 26759067.

[129] Judith Günther, Andreas Bergner, Manfred Hendlich, and Gerhard Klebe. Utilising Structural Knowledge in Drug Design Strategies: Applications Using Relibase†. *Journal of Molecular Biology*, 326(2):621–636, February 2003.

[130] M. Cris Silva-Santisteban, Isaac M. Westwood, Kathy Boxall, Nathan Brown, Sam Peacock, Craig McAndrew, Elaine Barrie, Meirion Richards, Amin Mirza, Antony W. Oliver, Rosemary Burke, Swen Hoelder, Keith Jones, G. Wynne Aherne, Julian Blagg, Ian Collins, Michelle D. Garrett, and Rob L. M. van Montfort. Fragment-based screening maps inhibitor interactions in the atp-binding site of checkpoint kinase 2. *PLOS ONE*, 8(6):1–15, 06 2013.

[131] Adam Biela, Nader N. Nasief, Michael Betz, Andreas Heine, David Hangauer, and Gerhard Klebe. Dissecting the hydrophobic effect on the molecular level: The role of water, enthalpy, and entropy in ligand binding to thermolysin. *Angewandte Chemie International Edition*, 52(6):1822–1828, 2013.

[132] Adam Biela, Michael Betz, Andreas Heine, and Gerhard Klebe. Water makes the difference: Rearrangement of water solvation layer triggers non-additivity of functional group contributions in protein–ligand binding. *ChemMedChem*, 7(8):1423–1434, 2012.

[133] Scott A. Wildman and Gordon M. Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of Chemical Information and Computer Sciences*, 39(5):868–873, 1999.

[134] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, Erik L L Sonnhammer, John Tate, and Marco Punta. Pfam the protein families database. *Nucleic Acids Research*, 42(D1):D222–D230, January 2014.

[135] UniProt Consortium. UniProt a hub for protein information. *Nucleic Acids Research*, 43(Database issue):D204–212, January 2015.

A

Tool Descriptions

A.1. Tools for Searching of Interaction Patterns

3DinSight [46]: 3DinSight has been published in 1998, the web server is not available any more. However, to my knowledge, it has been the first service which provided searches with distance constraints on macromolecules. The main focus of 3DinSight is to search for protein structures with a specific sequence pattern and annotated attributes, e.g., its organism. However, simple spatial queries are supported: Queries containing distance constraints between amino acids and nucleo bases can be answered. The query has to be entered in the structural query language (SQL). 3DinSight uses a relational database containing all structures of the PDB. Besides the atomic coordinates, distances between all pairs of amino acid and nucleic base are stored. The results of a search are presented in a list. From here, the original PDB entry can be viewed or different properties of the molecule can be visualized, e.g., which amino acids are involved in a α-helix.

Erebus [47]: Erebus searches specific atomic patterns in the complete PDB. The query has to be defined using the PDB format. Herein, atoms are defined by their element, their belonging to a specific amino acid or molecule, and their 3D coordinates. Using the terms 'ATOM' and 'HETATM', an atom can be explicitly defined as part of the protein or not, respectively. Using the atoms in the PDB file, similar patterns are searched in the protein structures of the PDB tolerating small geometrical deviations. For the search, both the protein and the query structure are converted to complete graphs. Thereby, the distance between atoms are annotated on the edges. Matching structures are then detected by applying an iterated sorting and filtering scheme. Herein, all pairwise atom distances are first searched in the set of protein structures before complete hits are constructed. For each hit, the spatial accordance of the hit to the query is checked. Only hits with small spatial deviations are accepted. Unfortunately, no comments are made about the retrieval times. The results are presented in a list with the original PDB code and the geometrical deviations from the query. The query can be superimposed on a specific result and visualized.
Using this approach, the detection of specific spatial patterns in proteins is possible. However, if the search involves small-molecules or water, the search could be difficult because of the inconsistent annotation of molecule names in PDB files. In addition, the definition of the query as a PDB file requires a template protein structure. Designing such a query from

scratch and can be very difficult.

Suns [48]: Suns is able to search for specific fragments in proteins similar to Erebus. However, Suns provide a graphical interface where a protein structure can be loaded and atoms or fragments of interest can be selected. Suns utilizes a database containing structural information of proteins for the search. The inserted protein structures are divided into cubes with side lengths of 15 Å. Inside this cube, all chemical fragments from two to nine atoms are collected. Using this approach, a database with 24 218 different protein chains has a size of 89 GB. During the search of a specific motif, first the cubes which contain all chemical fragments of the query are identified. Then, the respective cube is reconstructed and the arrangement of the fragments within the cube are aligned to the query using the algorithm of Kabsch [60]. As in Erebus, only hits with small spatial deviations are accepted. All hits are superimposed to the query and presented in a result window. Given this representation, similarities and differences can easily be spotted. On a database containing a set of 272 non-redundant protein chains, retrieval times between 100 and 10 000 seconds were reached. Even though short retrieval times are reached here, this approach has two clear disadvantages. Firstly, the size of the database is quite large given the exponentially increasing number of protein structures in the PDB. Secondly, due to the separation of data into cubes, no hits spanning multiple cubes can be detected. Thus, if a specific motif is by chance divided by the separation into cubes, it could never be found.

ASSAM [49]: The tool ASSAM is able to find amino acid chains with a specific 3D orientation in a set of protein structures. Herein, protein structures are represented as graphs and stored in a database. Every vertex in these graphs represents one amino acid. A vertex consists of two pseudo atoms which represent the orientation of the side chain. For each amino acid type, the exact position of the pseudo atoms is exactly defined by the positions of the side chain atoms. The graph is complete, meaning that all pairs of vertices are connected via an edge. The edge contains several distance values, representing the mutual distances between the two pseudo atoms of the vertices, respectively. A query in this context is a PDB file containing a peptide chain of up to twelve connected amino acids. For the search, the query peptide is converted into a graph structure as described before. Then a maximal common subgraph approach is used to detect all occurrences of the query within the database. Herein, a fast initial screen is performed using a clique algorithm. Afterwards, exact hits are detected using the algorithm of Bron and Kerbosch [58]. The resulting hits are presented as a list. Each hit can be visualized by super-imposing the query sequence onto the detected structure. Retrieval times should be around 6 min for a typical search.
Even though the protein and query structures are reduced to simpler graphs, the retrieval times are with six min quite high. Another shortcoming of this approach is the query design. Only connected peptide in a protein can be detected. However, due to the folding of the protein, there might be amino acids which are spatially in close contact but sequentially divided

by more than twelve amino acids. Moreover, due to the representation of amino acids by two points, the complete spatial variability of a side chain cannot be represented in this approach.

IMAAAGINE [50]: IMAAAGINE is based on the tool ASSAM. The methodology has however been slightly changed. Proteins are still represented as graphs in which every vertex corresponds to one amino acid. In IMAAAGINE however, every vertex only contains one pseudo atom which represents the functional part of the side chain. Again, each graph is complete. In this case, each edge represents only the mutual distance between the two vertices. A query here may contain up to eight amino acids. For each pair of amino acid, a mutual distance constraint can be added. Moreover, besides defining an exact type of an amino acid, its chemical function can be described, e.g., acidic, basic, or hydrophobic. For the search procedure, the query is converted into a graph. Thereby, every amino acid is converted into a vertex whereas every distance constraint between two amino acids is converted into an edge between the corresponding vertices. Then the subgraph isomorphism algorithm of Ullmann [59] is utilized to find all occurrences of the query in the database.
IMAAAGINE overcomes the problems of ASSAM, namely that only amino acids chains can be searched. However, the representation of amino acids has been reduced from two pseudo atoms to one pseudo atom which even reduces the possibility to represent the spatial variability of amino acids.

PDBeMotif/MSDmotif [51]: PDBeMotif is a search engine providing a large set on structural queries related to proteins. In principle a query consist of different query objects which can be connected by distance or interactions constraints. A search similar to the search possible in IMAAAGINE can be performed by generating several sequence motifs consisting of only one amino acid. These can then be connected by distance constraints. However, only distance constraints to amino acids which are within the same protein chain are taken into account here. Unfortunately, the exact search mechanism and the database construction is not explained in detail for this type of query. In addition to protein based queries, PDBeMotif provides the possibility to search for interaction patterns containing parts of the protein and small molecules. To this end, different types of atomic interactions and atom distances below 4.25 Å between protein and ligand atoms are precomputed and stored in the database. As query object, a small molecule can be sketched in 2D. Distance constraints up to 4 Å and interaction constraints can then be defined between exact atoms of the ligand and any protein based query object. Results are presented in a list and the resulting amino acids can be highlighted.
This approach allows a wide range of different query possibilities compared to the other presented approaches. However, only distances up to 4 Å can be used for the query. Another shortcomings of this software is its usability. The generation of spatial queries is complicated and no 3D template structure can be used here. Moreover, resulting hits cannot be super-imposed making it difficult to inspect the similarities and differences among the results.

CSD-CrossMiner [52]: The tool CSD-CrossMiner has been developed to perform spatial searches on two datasets: the CSD and the PDB. Here, we will focus on its capabilities to mine the PDB. CSD-CrossMiner uses an SQLite database in which data from protein-ligand interfaces are stored. For each ligand within a PDB file, this interface includes all atoms whose minimal distance to any of the ligand's atoms is below six Å. Inside this interface, atoms or substructures with specific pharmacophore features are identified, e.g., donor, acceptor, or hydrophobic. If necessary, the directionality of the feature is annotated. For example, a donor feature point includes the direction of the hydrogen atom and an aromatic ring feature point includes the direction of the ring normal. Moreover, the structure in which the feature has been detected is annotated, e.g., ligand or protein. For each interface, only these features are stored in the database. Hence, these features display the possible objects which can be used in a query. A query consists of a set of these feature points in 3D space. They can be defined using a 3D template structure. Every feature is represented by a sphere. The diameter of the sphere encodes the locational precision of each feature.
The search starts with translating the 3D query into distance constraints. Herein, the mutual distance between all feature points is measured. The sum of their feature radii reflects the tolerance of this distance. Then, using a fingerprint technique, all interfaces which contain the required set of features are identified. The relevant feature points of each identified interface are then subjected to a 3D search procedure in which their complete matching to the query is verified. This is described as a depth-first-search. Starting at the feature point involved in the highest number of distance constraints, the other feature points are included iteratively. In every step, a feature point is only included if the exact distance constraint is fulfilled. Finally, the algorithm of Kabsch is used to determine the best overlap of each hit with the query.
Results are presented in a 3D viewer super-imposed onto the query. Moreover, 2D depictions of the results are shown. Herein, all results are aligned to the query and can be easily compared.
The main advantage of the CSD-CrossMiner is its speed. Runtimes between 25 and 350 s for different example queries are given. These calculations have been performed in parallel on four different cores. Moreover, resulting hits are presented as soon as they have been detected and not after the complete search has been ended. A clear disadvantage of the CSD-CrossMiner is the reduction of the searchable objects to precalculated features. Moreover, due to the used search procedure, the resulting hits might not completely spatially agree with the query. Hence, single atoms might be positioned outside a query feature sphere.

PRDB [53] (Protein Relational Database): The PRDB also utilizes a database in order to provide spatial queries on protein-ligand interfaces from the PDB. The database contains several tables storing general information about the protein, e.g., its organism and data concerning its publication. In addition, several geometrical properties are stored in the database.

These are: Distances between two atoms, distances and angles between atoms and ring centroids, and distances and angles between two ring centroids. Herein, one partner of the measured parameter has to be part of the protein, the other one has to be part of the ligand. Moreover, only protein atoms and ring centroids are considered which are within 8 Å of any ligand atom. For amino acids which have at least one atom within 5 Å of any ligand atom, two additional tables exists. In the first table, mutual distances between the α carbon of the amino acids are stored. Analogously, the second table stored triplets of mutual distances between α carbon of the amino acids.
Queries can be designed in SQL and might cover all items stored in the database. The presentation of the result is not explained in the publication.
Even though the PRDB approach allows for the definition of distance and angle constraints in their query, it has clear disadvantages. First of all, queries containing more features than one distance or one angle constraint are highly complicated to design in SQL. Secondly, no substructure constraints can be added to the query. Thirdly, distances between amino acids are only measured at their α carbons. Thus, no precise, atom-wise distance query is possible. And fourthly, given all the precomputed data, the database has probably a quite large size. However, nothing is written about the database size in their publication.

Prolix [54] (Protein Ligand Interaction Explorer): Prolix facilitates the rapid mining of protein-ligand interactions in a large crystal structure database. The database contains information about all protein-ligand complexes from the PDB. Mainly, the information about non-covalent interactions between proteins and ligands are stored, e.g., the involved atoms. In addition, mutual distances between amino acids in the 'shell' around a ligand are stored. The shell contains all amino acids which are within 4.5 Å around any ligand's atoms. Herein, the distances between the α carbons of the amino acids are used.
A query in Prolix contains a ligand substructure and amino acids in its shell. Specific atomic interactions can be defined between an atom of the substructure and an amino acid. Distance constraints between amino acids can be added. Moreover, amino acids which should not be part of the shell can be defined.
The search algorithm contains three major steps. First of all, all ligands and their shells are detected in the database which contain the required amino acids and interaction types. Secondly, the required distance constraints between the amino acids are checked. Finally, complete hits are constructed using all possible combinations of matching amino acids for each ligand. Only those which fulfill all constraints are used. At some point within the search process, a subgraph matching has to be performed in order to detect ligands which contain the required substructure. Unfortunately, this point is not given in the publication. The result of a search are presented in list grouped by the protein.
The used fingerprint technique leads to fast runtimes between 2 and 5 s for typical queries. However, the fact that no constraints for exact atoms of amino acids can be defined strongly reduces the precision of the possible queries.

Relibase [55, 61, 129]: Relibase has already been developed in 2003 and provides a large variety of different geometrical queries. A commercial version, called Relibase+ and a free version, called Relibase exist. Both tools work with the same relational database which contains all structures from the PDB, including DNA and RNA structures. For structures which have been determined using NMR, all different models are incorporated. Ligands are defined as all small molecules and peptide chains of up to 20 amino acids here. For each ligand, a topological fingerprint is calculated using hashed values of all non-overlapping paths through the molecule up to a length of eight bonds.
Queries contain 2D representations of ligand and protein substructures. Specific atoms can be combined with distance constraints. Notably, allowed distance ranges can be defined here. In Relibase+, a distance constraint for any pair of atoms can be defined. In Relibase, only inter-molecular distance constraints are possible. Moreover, planes can be defined and angles constraints between any pair of plane or distance can be added to the query in Relibase+. Again, ranges of allowed angles are used here. This geometrical search can be combined with constraints for the resolution and the experimental procedure in both versions of Relibase.
The search mechanism starts with a substructure search on ligands using the topological fingerprint. Afterwards, all hits are subjected to an subgraph matching which compares the complete query to the selected ligand and its surroundings.
Results are presented in a list and can be visualized in a 3D viewer. Herein, the atoms matching the query are highlighted. In Relibase+, different results in similar proteins can be super-imposed based on the protein chains. As in Prolix, results are shown as soon as they have been detected and a user can start inspecting the results before the complete search is over.
The main advantage of Relibase+ is the high variability of the query. Any substructure can be drawn and distance and angle constraints can be added without many restrictions. However, the used search mechanism has the disadvantage that queries which do not contain large substructures of a ligand may have quite long runtimes (see Section 6.5 for example queries on Relibase). In principle however, Relibase+ allows geometrical queries which contain only parts of a protein. Moreover, results cannot be super-imposed based on the query which makes it difficult to spot similarities and differences among the results.

A.2. Tools for the Deduction of Preferred Interaction Directions

Sirius [64] / **X-Site** [65]: To my knowledge, Sirius and X-Site were the first tools which calculated interaction preferences of atomic interactions in macromolecular structures. For Sirius, the authors used a high resolution set ($\leq$ 2.0 Å) of 52 different protein structures from the PDB. In these structures they detected all interactions of amino acid side chains. Each pair of side chains was then transferred into a reference coordinate system using a template structure for one the detected side chains. For each combination of amino acids, a distribution could then be generated showing the preferred interaction directions. This

work resulted in the 'atlas of protein side-chain interactions' [70] and has later been used to evaluate the binding of peptide inhibitors [71]. For the tool X-Site, this principle was followed even further using a data set of 83 protein chains. Here, each amino acid within the binding site of a ligand was broken up into overlapping three-atom fragments. For each fragment, the distribution of interacting atoms was recorded. The resulting distributions were then transfered into other binding sites in order to evaluate the binding of a ligand.

These publications represent the first steps the community made to detect preferred interaction geometries within proteins and transfer the knowledge to the evaluation of the binding mode of a ligand. Obvious shortcomings of the approach like the small dataset and the fixed set of protein fragments can be explained by the small number of available high-quality structures and by restricted storage and compute power at the time of publication.

IsoStar [68]: IsoStar is a tool which visualizes the distribution of interacting atoms in the vicinity of predefined functional groups in the CSD and the PDB. Here, we focus on the PDB because we are focusing on interaction preferences in the context of proteins. From the complete PDB, all structure of protein-ligand complexes which were determined with X-ray crystallography and had a resolution $\leq$ 2.0 Å have been used. A ligand was defined as any non-peptide molecule of at least nine atoms and all peptides of up to ten amino acids. Within this data set, all non-covalent interactions (mututal atom distance $<$ 4 Å) between atoms of the protein and the ligand were detected. For each atom it was assessed, whether it belongs to one of 250 predefined chemical groups. Only if both atoms belong to one of the groups, the geometrical parameters were recorded. After data collection, all moieties which where detected for on of the chemical groups, were overlaid. This group is then called the central group. The distribution of surrounding atoms from other chemical groups (called contact group) was stored in a specific file format. Herein, only 'target atoms' are used, e.g., nitrogen, oxygen, hydrogen. Such a file has been generated for each combination of central chemical group and contact chemical group.

In a 3D viewer, these data files can be loaded and the distribution can be analyzed. The distribution can be displayed as spheres or as contour surfaces. Within the viewer, ranges for the distance between the central and the contact group can be added. Additionally, the originating structure can be traced back and visualized for each data point.

Data files for central groups which are not part of the initial set can be calculated using the programm IsoGen. However, this is only possible for the CSD data set [73].

To my knowledge, IsoStar has been the first tool which systematically collected spatial data around functional groups which are relevant in the field of drug design and provided a visualization for a convenient analysis. However, regarding the PDB, its functionality is limited to a predefined set of chemical groups which cannot be extended. Moreover, the means to analyze the data set are limited to a distance range constraint. Another shortcoming is the

data handling. Data is collected in separate files and each file has to be loaded individually. Despite these shortcomings, a large number of publications in which IsoStar has been used to analyze and improve ligand binding prove the usefulness of this approach (examples are [130–132]).

SuperStar [74, 75]: The tool SuperStar uses the data generated in the IsoStar approach and identifies regions in a protein binding site where chemical groups are likely to interact. To this end, a protein can be visualized in SuperStar. The IsoStar distributions are then transferred to a defined binding site showing regions where specific chemical groups could be located. The approach is similar to the tool X-Site. However, they differ in the used data set: SuperStar utilizes information from the CSD as well as from protein-ligand contacts detected in the PDB. X-Site only used side chain contacts within the PDB.

However, as for IsoStar, the chemical groups which can be displayed are limited to the predefined set used in IsoStar.

GIANT [66, 72]: GIANT is a recent attempt to generate an unsupervised classification of spatial interaction preferences without mixing the data from different amino acids. As set of 23 040 protein-ligand files with a resolution $\leq$ 2.5 Å from the PDB was used for this endeavor. From this dataset, proteins were clustered based on their protein chain, resulting in 3 219 different clusters. Each amino acid was decomposed into fragments of three connected atoms. The specification of the fragments included the specific atom names of the respective amino acid and the name of the amino acid itself. Hence, fragments, even if chemically equal, are treated as individual fragments. For each fragment, interacting atoms of the ligand are detected. Herein, a distance smaller than the sum of the van-der-Waals radii +1 Å was used as only criterion. For each atom type of these interacting atoms, spatial propensity functions were calculated using a Gaussian mixture model. In total, 8 022 combinations of protein fragment and atom type were found here.

The tool GIANT provides the possibility to visualize these propensity functions in the vicinity of specific amino acid fragments.

The separation of different side chain fragments provides the opportunity to individually analyze the interaction preferences of different side chains. However, in some cases, a combination could be beneficial. As for example for the carboxyl group of glutamic acid and aspartatic acid. Both are chemically almost identically amino acids. The calculation of propensity functions might be a good means to visualize geometrical preferences and present hots spots which might be hidden if simple distributions were shown. However, the possibility to trace back each data point to its original structure is lost by this calculation which could be helpful to find reasons for structural abnormalities. As for IsoStar, the set of analyzed fragments and atom types cannot be extended. This option might be less interesting for

amino acids but the extention by novel atom types might lead to the discovery of new atomic interactions.

B

Additional Attributes and Values for Pelikan and *NAOMI*nova

B.1. Functional Groups used in Pelikan

Aldehyde, ketone, amide, ester, azide, nitrile, guanidine, amidine, amine, alcohol, ether, pyrrol, thiophene, furane, phenyl, pyridine.

B.2. Element types used in Pelikan

Nitrogen, carbon, phosphorus, sulfur, oxygen, fluorine, chlorine, bromine, iodine, calcium, zinc, cobalt, iron, copper, manganese, nickel, magnesium.

B.3. Atom Names of Amino Acids used in Pelikan

Amino acid type	Atom names
alanine	C, CA, CB, CG, N, O
arginine	C, CA, CB, CG, CD, CZ, N, NE, NH1, NH2, O
asparagine	C, CA, CB, CG, N, ND2, O, OD1
aspartate	C, CA, CB, CG, N, O, OD1, OD2
cysteine	C, CA, CB, CG, N, O, SG
glutamine	C, CA, CB, CG, CD, N, NE2, O, OE1
glutamate	C, CA, CB, CG, CD, N, O, OE1, OE2
glycine	C, CA, CB, CG, N, O
histidine	C, CA, CB, CG, CD2, CE1, N, ND1, NE2, O
isoleucine	C, CA, CB, CG, CG1, CG2, CD1, N, O
leucine	C, CA, CB, CG, CD1, CD2, N, O
lysine	C, CA, CB, CG, CD, CE, N, NZ, O
methionine	C, CA, CB, CG, CE, N, O, SD
phenylalanine	C, CA, CB, CG, CD1, CD2, CE1, CE2, CZ, N, O
proline	C, CA, CB, CG, CD, N, O
serine	C, CA, CB, CG, N, O, OG
threonine	C, CA, CB, CG, CG2, N, O, OG1
tryptophane	C, CA, CB, CG, CD1, CD2, CE2, CD3, CZ2, CZ3, CH2, N, NE1, O
tyrosine	C, CA, CB, CG, CD1, CD2, CE1, CE2, CZ, N, O, OH
valine	C, CA, CB, CG, CG1, CG2, N, O

Table B.1.: Atom names of different amino acid types used in 3D queries in Pelikan.

B.4. Groups of Amino Acid Types used in Pelikan and *NAOMI*nova

Group	Amino acid types
hydrophobic	methionine, proline, alanine, leucine, tryptophane, valine, isoleucine, phenylalanine
polar	tyrosine, threonine, glutamine, glycine, serine, cysteine, asgaragine, lysine, arginine, histidine, glutamine, asparagine
aromatic	tryptophane, phenylalanine, tyrosine
acidic	glutamate, aspartate
basic	lysine, arginine, histidine
neutral	tyrosine, threonine, glutamine, glycine, serine, cysteine, asparagine

Table B.2.: Groups of amino acids used in 3D queries used in Pelikan and *NAOMI*nova.

B.5. Properties used in Pelikan

Property name	Description	Value range
Element	Count of each element type present in a reference ligand. The exact element types used are listed in B.2	0-100 (integer)
Functional group	Count of each functional group present in a reference ligand. The exact functional groups used are listed in B.1	0-100 (integer)
MW	Molecular weight	0-$1 \cdot 10^9$ (real)
Atoms	Number of heavy atoms	0-100 (integer)
TPSA	Topological polar surface ares	0-$1 \cdot 10^9$ (real)
logP	logP after Wildman and Crippen [133]	-1.0-1000 (real)
Volume	Volume of molecule	0-$1 \cdot 10^6$ (real)
Total charge	Total charge of molecule	
Acceptors	Number of hydrogen bond acceptors in molecule	0-100 (integer)
Donors	Number of hydrogen bond donors in molecule	0-100 (integer)
LipinskiDonors	Number of hydrogens bound to oxygen or nitrogen	0-100 (integer)

Table B.3.: Properties calculated for reference ligands in Pelikan, part 1.

Hetero	Number of hetero atoms in molecule	0-100 (integer)
AromAtoms	Number of aromatic atoms in molecule	0-100 (integer)
Halogens	Number of halogens in molecule	0-100 (integer)
Inorganic	Number of atoms not in organic subset	0-100 (integer)
LipinskiAcceptors	Number of nitrogens and oxygens	0-100 (integer)
RotB	Number of rotatable bonds in molecule	0-400 (integer)
CRTB	Maximum path of contiguous rotatable bonds	0-400 (integer)
Rings	Number of rings	0-99 (integer)
URFs	Number of unique ring families	0-99 (integer)
AroRings	Number of aromatic rings	0-99 (integer)
MaxRing	Biggest ring	0-100 (integer)
Ringsystems	Number of ringsystems in molecule	0-99 (integer)
AroRingsystems	Number of aromatic ringsystems in molecule	0-99 (integer)
MaxRSsize	Biggest ringsystem in molecule	0-100 (integer)
RS	Number of stereo centers in molecule	0-100 (integer)
EZ	Number of stereo bonds in molecule	0-400 (integer)
Cyclomatic Number	Cyclomatic Number of molecule	0-99 (integer)
Max Cyclomatic Number	Maximum Cyclomatic Number of molecule's ringsystems	0-99 (integer)

Table B.4.: Properties calculated for reference ligands in Pelikan, part 2.

Property name	Description	Value range
PFAM id	id of protein in PFAM database [134]	'PF'+five digit number
Uniprot id	id of protein in uniprot database [135]	string
EC number	enzyme commission number	four numbers, 0-99 (integer)
Organism	organism of protein	string

Table B.5.: Properties calculated for proteins in Pelikan.

Property name	Description	Value range
Ligand name	Name of reference ligand	no limit of length (string)
Amino acids in pocket	Count of each amino acid in the pocket	0-100 (integer)
Volume	Volume of pocket	0-5000 (real)
Surface	Surface of pocket	0-5000 (real)
Surface-Volume-Ratio	Ratio of surface and volume	0-5000 (real)
HeavyAtoms	Number of heavy atoms in pocket	0-1000 (integer)
Acceptors	Number of H-bond acceptors	0-100 (integer)
Donors	Number of H-bond donors	0-100 (integer)
Hydrophobicity	Hydrophobicity of pocket	0-1 (real)
Depth	Depth of pocket	0-5000 (real)
Enclosure	Enclosure	0-5000 (real)
Score	Simple DogSite score	0-5000 (real)
Metal	Number of metal atoms in Pocket	0-100 (integer)

Table B.6.: Properties calculated for pockets in Pelikan.

Property name	Description	Value range
Name	Content of PDB title entry	no limit of length (string)
PDB id	id of PDB file	no limit of length (string)
Resolution	Resolution of PDB file	0-4 (real)
Experimental source	Experiment type which has been used to generate the structure of the protein. The following types can be used: Unknown, NMR solution, NMR solid state, X-ray, fiber diffraction, neutron diffraction, electron microscopy, electron crystallography, solution scattering	enum (integer)

Table B.7.: Properties calculated for protein-ligand complexes in Pelikan.

C

Test Queries Pelikan

OnePoint - no point-point constraints
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any

TwoPoints - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - no point-point constraints
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any

TwoPoints - element and interaction type
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - all properties
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - resolution
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Additional filter: Resolution between 1.5 and 2.5 Å

TwoPoints - resolution and pocket volume
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Additional filter: Resolution between 1.5 and 2.5 Å, pocket volume between 650 and 750 $Å^3$

TwoPoints - 3-4 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 3 Å, max = 4 Å

TwoPoints - 6-7 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = , Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = , Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6 Å, max = 7 Å

TwoPoints - 9-10 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 9 Å, max = 10 Å

TwoPoints - short SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NCC)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - short SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NCC)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - long SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = Any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(O(CCO)POP)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NC(C(C)C)C=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

TwoPoints - long SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(O(CCO)POP)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NC(C(C)C)C=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

ThreePoints - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

ThreePoints - no point-point constraints
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any

ThreePoints - resolution
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Additional filter: Resolution between 1.5 and 2.5 Å

ThreePoints - resolution and pocket volume
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Additional filter: Resolution between 1.5 and 2.5 Å, pocket volume between 650 and 750 $Å^3$

ThreePoints - 3-4 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Am
ino acid location = Any
Distance constraint between search points 1-2: min = 3 Å, max = 4 Å
Distance constraint between search points 2-3: min = 4 Å, max = 5 Å//if set to 3-4 Å, no results were found

ThreePoints - 6-7 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any
Distance constraint between search points 1-2: min = 6 Å, max = 7 Å
Distance constraint between search points 2-3: min = 6 Å, max = 7 Å

ThreePoints - 9-10 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 9 Å, max = 10 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å

ThreePoints - short SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NCC)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NPO)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

ThreePoints - short SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NCC)]
Search point 2: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NPO)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

ThreePoints - long SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NC(C(C)C)C=O)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any,

SMARTS = [$(NC(C(C)C)C=O)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(N(P(O)(O)=O)P(O)(O)=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

ThreePoints - long SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NC(C(C)C)C=O)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NC(C(C)C)C=O)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(N(P(O)(O)=O)P(O)(O)=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

FourPoints - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - no point-point constraints
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any

FourPoints - resolution

Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Additional filter: Resolution between 1.5 and 2.5 Å

FourPoints - resolution and pocket volume
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Additional filter: Resolution between 1.5 and 2.5 Å, pocket volume between 650 and 750 $Å^3$

FourPoints - one distance 2 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 5.6 Å, max = 7.6 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - one distance 3 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 5.1 Å, max = 8.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - 3-4 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 3 Å, max = 4 Å
Distance constraint between search points 2-3: min = 4 Å, max = 5 Å//if set to 3-4 Å, no results were found
Distance constraint between search points 3-4: min = 3 Å, max = 4 Å

FourPoints - 6-7 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6 Å, max = 7 Å
Distance constraint between search points 2-3: min = 6 Å, max = 7 Å
Distance constraint between search points 3-4: min = 6 Å, max = 7 Å

FourPoints - 9-10 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined,

Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 9 Å, max = 10 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 3-4: min = 9 Å, max = 10 Å

FourPoints - one interaction
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Interaction constraint between search points 3-4: h-bond

FourPoints - two interactions
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Interaction constraint between search points 1-2: h-bond
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Interaction constraint between search points 3-4: h-bond

FourPoints - short SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NCC)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NPO)]
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional

group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - short SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NCC)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NPO)]
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group = any, Amino acid = SER, Atom name = OG, Sec. structure = No sec.structure, Amino acid location = Sidechain, SMARTS = [$(OCC)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - long SMARTS
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any , SMARTS = [$(NC(C(C)C)C=O)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NC(C(C)C)C=O)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(N(P(O)(O)=O)P(O)(O)=O)]
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(OCC(N)C=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - long SMARTS + point attributes
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any, SMARTS = [$(NC(C(C)C)C=O)]
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone, SMARTS = [$(NC(C(C)C)C=O)]
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location =

Any, SMARTS = [$(N(P(O)(O)=O)P(O)(O)=O)]
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group = any, Amino acid = SER, Atom name = OG, Sec. structure = No sec.structure, Amino acid location = Sidechain, SMARTS = [$(OCC(N)C=O)]
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - metal
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

FourPoints - metal, water
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 3-4: min = 6 Å, max = 7 Å

FourPoints - metal, water, phosphorus Search point 1: Original Molecule = Reference ligand, Element = Phosphorus, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 3-4: min = 6 Å, max = 7 Å

FourPoints - angle range 10° and **FourPoints - one angle**
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 2-3: 150-160°

FourPoints - angle range 50°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 2-3: 130-180°

FourPoints - angle range 90°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 2-3: 90-180°

FourPoints - two angles

Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 2-3: 150-160°
Angle constraint: between distance 2-3 and distance 3-4: 110-120°

SmallStar - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å

SmallStar - element and interaction type
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å

SmallStar - all properties
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group = any, Amino acid = SER, Atom name = OG, Sec. structure = No sec.structure, Amino acid location = Sidechain
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å

SmallStar - one distance 2 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 5.6 Å, max = 7.6 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å

SmallStar - one distance 3 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 5.1 Å, max = 8.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å

SmallStar - angle range 10° and **SmallStar - one angle**
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any

Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Angle constraint: between distance 1-2 and distance 1-3: 60-70°

SmallStar - angle range 50°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Angle constraint: between distance 1-2 and distance 1-3: 40-90°

SmallStar - angle range 90°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Angle constraint: between distance 1-2 and distance 1-3: 20-110°

SmallStar - two angles
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined,

Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Angle constraint: between distance 1-2 and distance 1-3: 60-70°
Angle constraint: between distance 1-3 and distance 1-4: 30-40°

LargeStar - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 5: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 6: Original Molecule = Protein, Element = Carbon, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 1-5: min = 3.3 Å, max = 4.3 Å
Distance constraint between search points 1-6: min = 4.1 Å, max = 5.1 Å

LargeStar - one interaction
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 5: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 6: Original Molecule = Protein, Element = Carbon, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Interaction constraint between search points 1-2: h-bond
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å

Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 1-5: min = 3.3 Å, max = 4.3 Å
Distance constraint between search points 1-6: min = 4.1 Å, max = 5.1 Å

LargeStar - metal
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 5: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 6: Original Molecule = Protein, Element = Carbon, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 4 Å, max = 5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 1-5: min = 3.3 Å, max = 4.3 Å
Distance constraint between search points 1-6: min = 4.1 Å, max = 5.1 Å

LargeStar - metal, water
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 5: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 6: Original Molecule = Protein, Element = Carbon, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 4 Å, max = 5 Å
Distance constraint between search points 1-4: min = 5.3 Å, max = 6.3 Å
Distance constraint between search points 1-5: min = 3.3 Å, max = 4.3 Å
Distance constraint between search points 1-6: min = 4.1 Å, max = 5.1 Å

LargeStar - metal, water, phosphorus
Search point 1: Original Molecule = Reference ligand, Element = Phosphorus, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location

= Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 5: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 6: Original Molecule = Protein, Element = Carbon, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 1-3: min = 3 Å, max = 4 Å
Distance constraint between search points 1-4: min = 5.2 Å, max = 6.2 Å
Distance constraint between search points 1-5: min = 4.5 Å, max = 5.5 Å
Distance constraint between search points 1-6: min = 4.2 Å, max = 5.2 Å

OneTriangle - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 6.7 Å, max = 7.7 Å

OneTriangle - element and interaction type
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 6.7 Å, max = 7.7 Å

OneTriangle - all properties
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Acceptor, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Donor, Functional group = any, Amino acid = VAL, Atom name = N, Sec. structure = Sheet, Amino acid location = Backbone
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Donor, Functional group

= any, Amino acid = SER, Atom name = OG, Sec. structure = No sec.structure, Amino acid location = Sidechain
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 6.7 Å, max = 7.7 Å

OneTriangle - resolution
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 6.7 Å, max = 7.7 Å
Additional filter: Resolution between 1.5 and 2.5 Å

OneTriangle - resolution and pocket volume
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 6.7 Å, max = 7.7 Å
Additional filter: Resolution between 1.5 and 2.5 Å, pocket volume between 650 and 750 $Å^3$

TwoTriangles - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 2.6 Å, max = 3.6 Å
Distance constraint between search points 1-3: min = 3.8 Å, max = 4.8 Å
Distance constraint between search points 1-4: min = 3.8 Å, max = 4.8 Å
Distance constraint between search points 2-4: min = 2.6 Å, max = 3.6 Å
Distance constraint between search points 3-4: min = 3.9 Å, max = 4.9 Å

TwoTriangles - one interactions
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Interaction constraint between search points 1-2: h-bond
Distance constraint between search points 1-3: min = 3.8 Å, max = 4.8 Å
Distance constraint between search points 1-4: min = 3.8 Å, max = 4.8 Å
Distance constraint between search points 2-4: min = 2.6 Å, max = 3.6 Å
Distance constraint between search points 3-4: min = 3.9 Å, max = 4.9 Å

Tetrahedron - standard
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

Tetrahedron - one distance 2 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 8.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å

Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

Tetrahedron - one distance 3 Å
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 5.6 Å, max = 8.6 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å

Tetrahedron - metal
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 5.7 Å, max = 6.7 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 4-3: min = 4.2 Å, max = 5.2 Å

Tetrahedron - metal, water
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å

Distance constraint between search points 1-3: min = 5.7 Å, max = 6.7 Å
Distance constraint between search points 1-4: min = 5.3 Å, max = 6.3 Å
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 2-4: min = 4.2 Å, max = 5.2 Å
Distance constraint between search points 3-4: min = 6 Å, max = 7 Å

Tetrahedron - metal, water, phosphorus
Search point 1: Original Molecule = Reference ligand, Element = Phosphorus, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Metal, Element = Magnesium, Interaction type = Metal, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Water, Element = Undefined, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 2-3: min = 9 Å, max = 10 Å
Distance constraint between search points 2-4: min = 4.2 Å, max = 5.2 Å
Distance constraint between search points 1-2: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 3-4: min = 6 Å, max = 7 Å
Distance constraint between search points 1-3: min = 4.2 Å, max = 5.2 Å
Distance constraint between search points 1-4: min = 5.2 Å, max = 6.2 Å

Tetrahedron - angle range 10 $^\circ$ and **Tetrahedron - one angle**
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 1-4: 60-70°

Tetrahedron - angle range 50°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any

Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 1-4: 40-90°

Tetrahedron - angle range 90°
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 1-4: 20-110°

Tetrahedron - two angles
Search point 1: Original Molecule = Reference ligand, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 2: Original Molecule = Protein, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 3: Original Molecule = Reference ligand, Element = Nitrogen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Search point 4: Original Molecule = Protein, Element = Oxygen, Interaction type = Undefined, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 6.1 Å, max = 7.1 Å
Distance constraint between search points 1-3: min = 3.5 Å, max = 4.5 Å
Distance constraint between search points 1-4: min = 6.7 Å, max = 7.7 Å
Distance constraint between search points 2-3: min = 7.6 Å, max = 8.6 Å
Distance constraint between search points 2-4: min = 6.5 Å, max = 7.5 Å
Distance constraint between search points 3-4: min = 4.1 Å, max = 5.1 Å
Angle constraint: between distance 1-2 and distance 1-4: 60-70°

Angle constraint: between distance 2-4 and distance 2-3: 30-40°

Test Query Bioisosters
Search point 1: Original Molecule = Protein, Element = Any, Interaction type = ring center, Functional group = any, Amino acid = PHE, Atom name = Any, Sec. structure = Any, Amino acid location = side chain
Search point 2: Original Molecule = Protein, Element = Any, Interaction type = ring center, Functional group = any, Amino acid = TYR, Atom name = Any, Sec. structure = Any, Amino acid location = side chain
Search point 3: Original Molecule = Protein, Element = Any, Interaction type = ring center, Functional group = any, Amino acid = TRP, Atom name = Any, Sec. structure = Any, Amino acid location = side chain
Search point 4: Original Molecule = Reference ligand, Element = Any, Interaction type = Any, Functional group = any, Amino acid = Any, Atom name = Any, Sec. structure = Any, Amino acid location = Any
Distance constraint between search points 1-2: min = 8 Å, max = 10 Å
Distance constraint between search points 1-3: min = 5 Å, max = 7 Å
Distance constraint between search points 1-4: min = 3.5 Å, max = 5.5 Å
Distance constraint between search points 2-3: min = 5.5 Å, max = 7.5 Å
Distance constraint between search points 2-4: min = 3.5 Å, max = 5.5 Å
Distance constraint between search points 3-4: min = 3 Å, max = 5 Å
Angle constraint: between search point 1 and distance 1-2: 10-40°
Angle constraint: between search point 4 and distance 1-4: 30-60°
Angle constraint: between search point 4 and distance 2-4: 40-60°
Angle constraint: between search point 2 and distance 1-2: 10-40°
Angle constraint: between search point 1 and distance 1-4: 30-60°

D

Additional Results Pelikan

Query name	Number of hits
FourPoints - resolution	286 356
FourPoints - resolution and pocket volume	28 601
FourPoints - 3-4 Å	26 015
FourPoints - 6-7 Å	944 122
FourPoints - 9-10 Å	2 593 420
FourPoints - standard	444 260
FourPoints - angle range 50°	225 116
FourPoints - angle range 90°	390 328
FourPoints - one distance 2 Å	874 413
FourPoints - one distance 3 Å	1 287 073
FourPoints - metal, water, phosphorus	5 377
FourPoints - metal, water	15 909
FourPoints - metal	26751
ThreePoints - resolution	230 879
ThreePoints - resolution and pocket volume	22 366
ThreePoints - 3-4 Å	44 231
ThreePoints - 6-7 Å	255 572
ThreePoints - 9-10 Å	495 512
ThreePoints - standard	355 815
TwoPoints - resolution	643 558
TwoPoints - resolution and pocket volume	44 423
TwoPoints - all properties	9 788
TwoPoints - element and interaction type	901 723
TwoPoints - 3-4 Å	217 507
TwoPoints - 6-7 Å	933 027
TwoPoints - 9-10 Å	1 336 873
TwoPoints - standard	961 933
FourPoints - one interaction	82 930
FourPoints - two interactions	11 412
FourPoints - angle range 10°	46 375
FourPoints - two angles range 10°	3 385
FourPoints - short SMARTS + point attributes	3
FourPoints - short SMARTS	364
FourPoints - long SMARTS + point attributes	3
FourPoints - long SMARTS	11
ThreePoints - short SMARTS + point attributes	29
ThreePoints - short SMARTS	606
ThreePoints - long SMARTS + point attributes	10
ThreePoints - long SMARTS	34

Table D.1: Exact number of hits of all test queries on a database containing 16 000 different PDB files using the tool Pelikan. Continued on next page.

Query name	Number of hits
TwoPoints - short SMARTS + point attributes	>1000
TwoPoints - short SMARTS	>1 000
TwoPoints - long SMARTS + point attributes	223
TwoPoints - long SMARTS	>1 000
FourPoints - no point-point constraints	>1 000 (enumeration stopped after 1000 hits)
OnePoint - no point-point constraints	610 935
ThreePoints - no point-point constraints	>1 000 (enumeration stopped after 1000 hits)
TwoPoints - no point-point constraints	>1 000 (enumeration stopped after 1000 hits)
OneTriangle - resolution	188 265
OneTriangle - resolution and pocket volume	13 190
OneTriangle - all properties	31
OneTriangle - element and interaction type	35 227
OneTriangle - standard	285 085
Tetrahedron - resolution	827
Tetrahedron - resolution and pocket volume	59
Tetrahedron - standard	1 199
Tetrahedron - angle range 50°	1 199
Tetrahedron - angle range 90°	1 199
Tetrahedron - one distance 2 Å	2 529
Tetrahedron - one distance 3 Å	3 439
Tetrahedron - metal, water, phosphorus	5
Tetrahedron - metal, water	23
Tetrahedron - metal	45
TwoTriangles - resolution	3 382
TwoTriangles - resolution and pocket volume	281
TwoTriangles - standard	5 166
Tetrahedron - angle range 10°	755
Tetrahedron - two angles range 10°	749
TwoTriangles - standard forInteractionTests	750
TwoTriangles - one interaction	878
LargeStar - standard	1 720 039
LargeStar - metal, water, phosphorus	36 336
LargeStar - metal, water	21 976
LargeStar - metal	71 926
SmallStar - all properties	45
SmallStar - element and interaction type	30 320
SmallStar - standard	1 050 439
SmallStar - angle range 50°	391 926
SmallStar - angle range 90°	696 369
SmallStar - one distance 2 Å	2 097 030
SmallStar - one distance 3 Å	3 123 997
SmallStar - angle range 10°	81 550
SmallStar - two angles range 10°	3 829
LargeStar - one interaction	115 862

Table D.1.: Exact number of hits of all test queries on a database containing 16 000 different PDB files using the tool Pelikan.

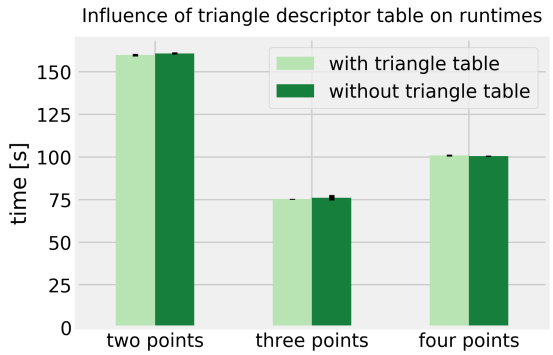


Figure D.1.: Retrieval times of queries which contain no triangle on the database 16 000 with and without triangle descriptor table. The triangle descriptor table does not influence the runtimes.

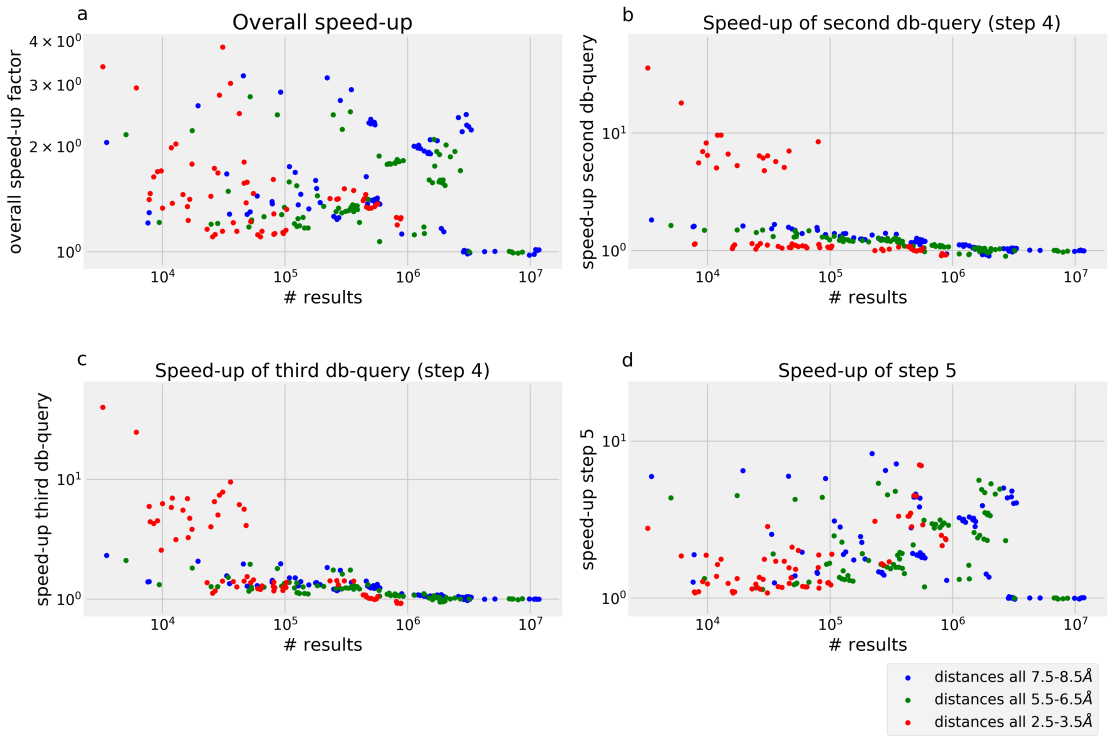


Figure D.2.: Comparison between speed-up factors of Δ-queries with the number of results (# results). Each data point corresponds to one query. The color of each dot codes the distance ranges of the respective query. For all experiments the database containing 8 000 different protein-ligand complexes was used. a) Overall speed-up factor gained with the triangle descriptor plotted against number of results. b) Speed-up factor of the second database query of step 4 gained with the triangle descriptor plotted against the number of results. c) Speed-up factor of the third database query of step 4 gained with the triangle descriptor plotted against number of results. d) Speed-up factor of step 5 gained with the triangle descriptor plotted against number of results.

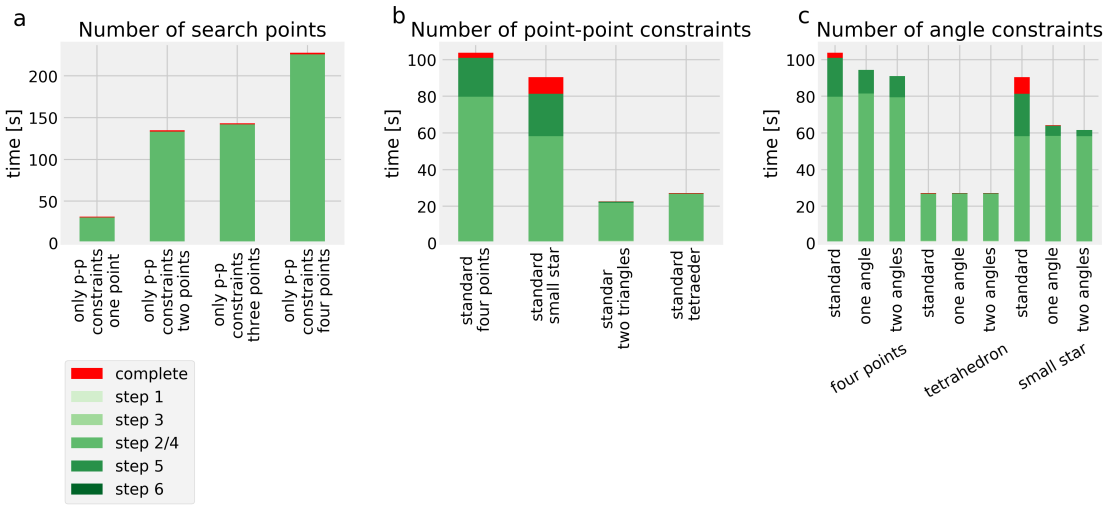


Figure D.3.: Runtimes of different test queries having different topologies separated by different steps of the algorithm. The complete runtime is the mean value of three independent experiments and shown in red bars. The single steps are shown in different colors of green as stacked bar. a) Runtimes of test queries which only differ in their number of search points. b) Runtimes of test queries which only differ in their number of point-point constraints. c) Runtimes of test queries which only differ in their number of angle constraints.

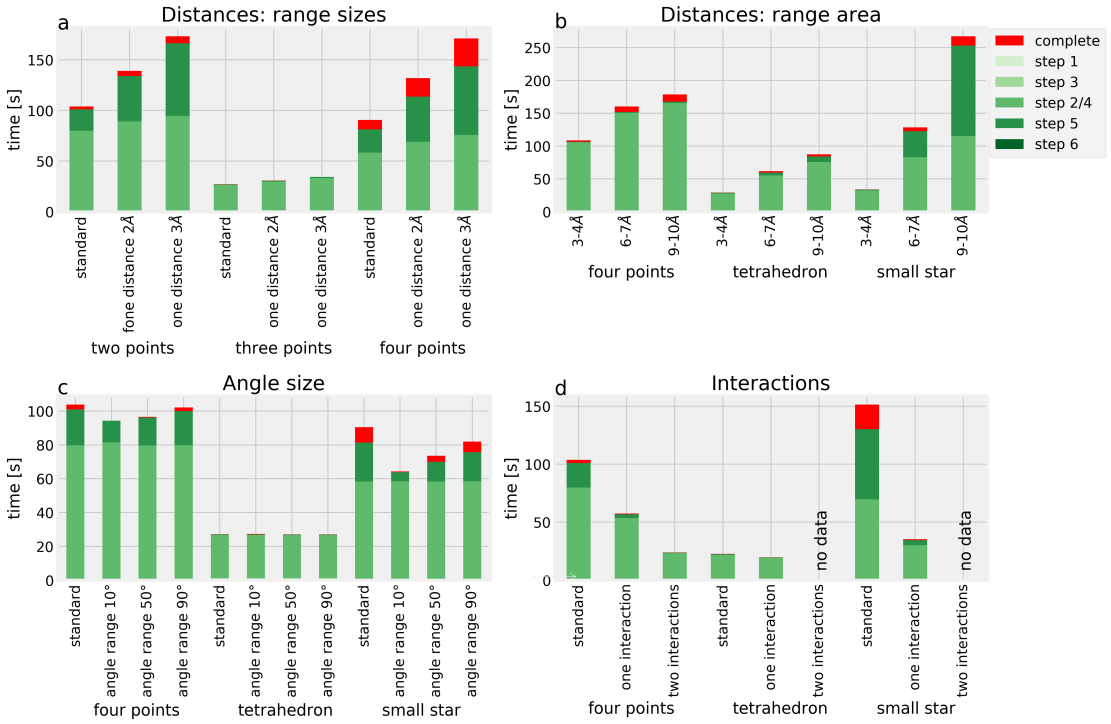


Figure D.4.: Runtimes of different test queries having different geometrical constraints separated by different steps of the algorithm. The complete runtime is the mean value of three independent experiments and shown in red bars. The single steps are shown in different colors of green as stacked bar. a) Runtimes of test queries which only differ in their range size of one distance constraint. b) Runtimes of test queries which only differ in their range area of one distance constraint. c) Runtimes of test queries which only differ in their size of one angle constraint. d) Runtimes of test queries which only differ in their number of distance constraints which are converted to interaction constraints.

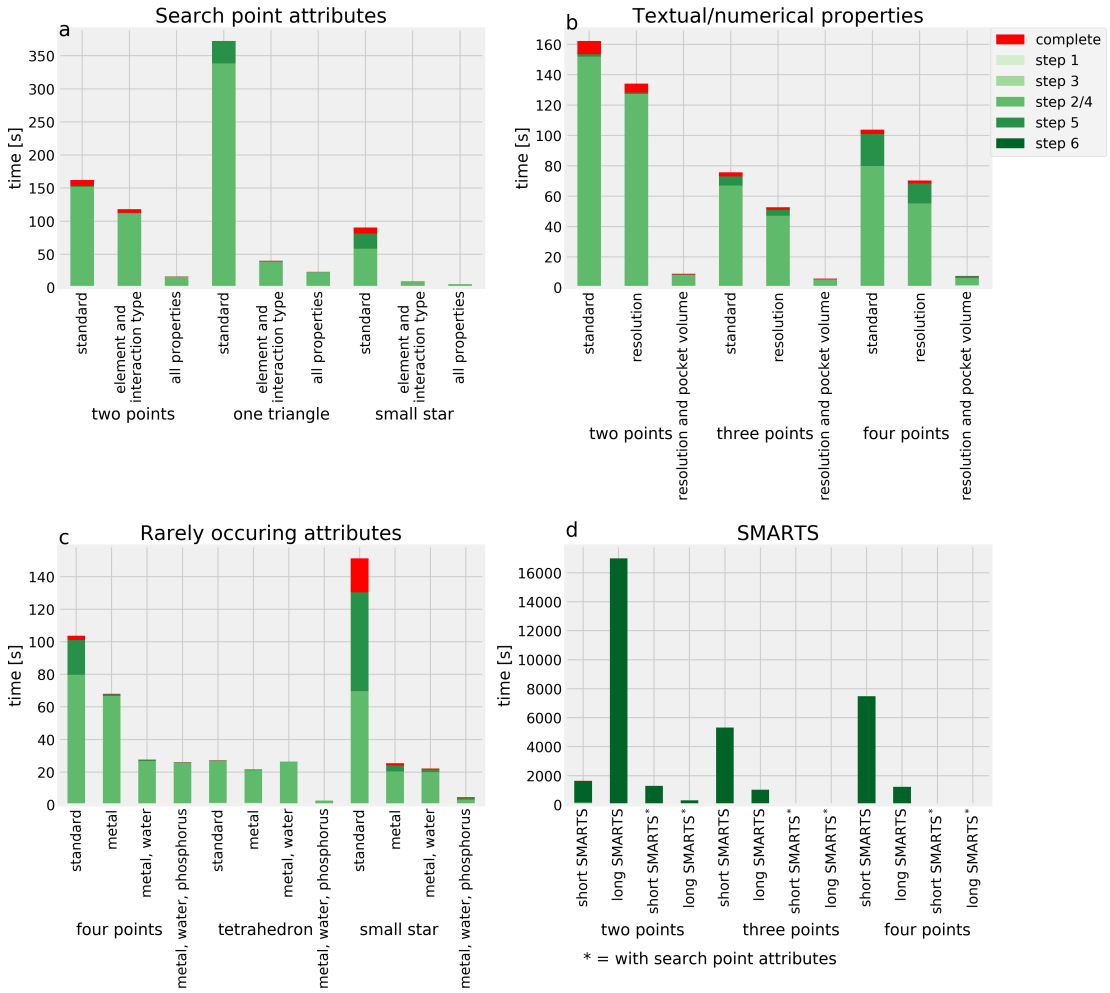


Figure D.5.: Runtimes of different test queries having different additional properties separated by different steps of the algorithm. The complete runtime is the mean value of three independent experiments and shown in red bars. The single steps are shown in different colors of green as stacked bar. a) Runtimes of test queries which only differ in their attributes of all search points. b) Runtimes of test queries which only differ in their textual and numerical properties. c) Runtimes of test queries which only differ in their element and molecules types of search points. d) Runtimes of test queries which only differ in their SMARTS description and other additional properties of all search point.

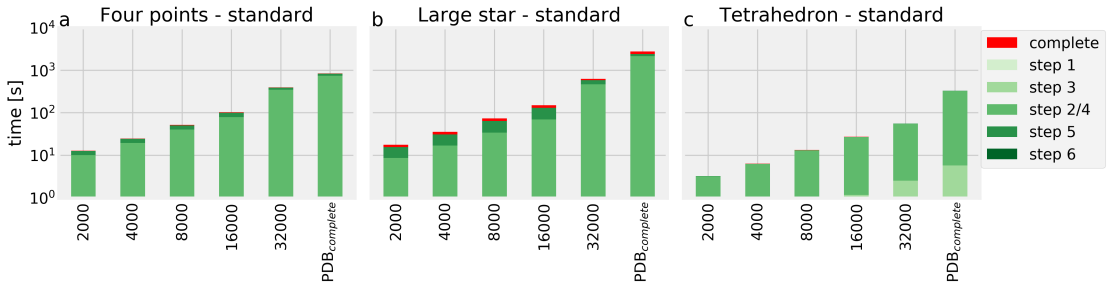


Figure D.6.: Runtimes of different test queries on database with different sizes separated by different steps of the algorithm. The complete runtime is the mean value of three independent experiments and shown in red bars. The single steps are shown in different colors of green as stacked bar. a) Runtimes of the query 'Four points – standard'. b) Runtimes of the query 'Large star – standard'. c) Runtimes of the query 'Tetrahedron – standard'.

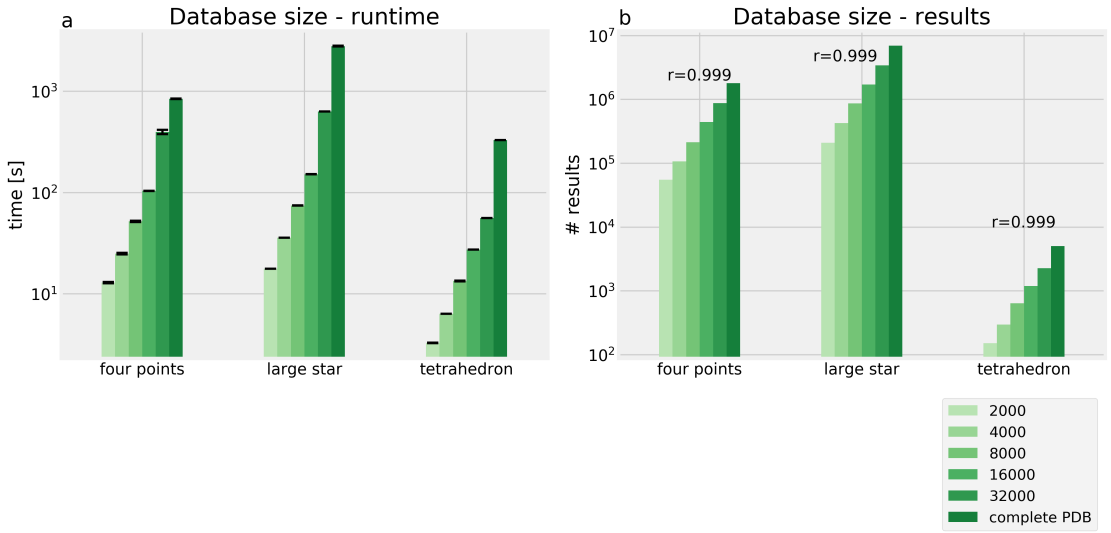


Figure D.7.: Comparison of runtimes and the number of results of different test queries on database with different sizes. a) Mean values and standard deviation of runtimes of different test queries from three independent runs shown as bars in seconds. b) Number of results received from three different test queries. 'r' indicates the resulting correlation coefficient of a linear regression.

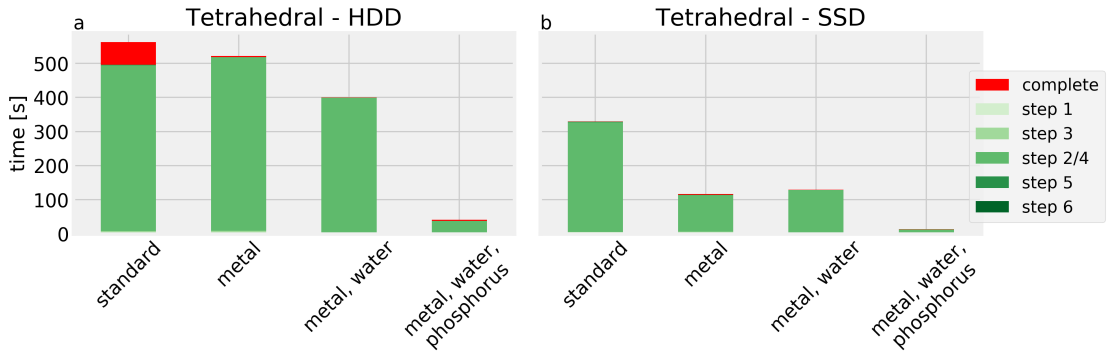


Figure D.8.: Runtimes of different test queries on different hardware settings using the $PDB_{complete}$ separated by different steps of the algorithm. The complete runtime is the mean value of three independent experiments and shown in red bars. The single steps are shown in different colors of green as stacked bar. a) Runtimes of different test queries using the HDD hardware settings. b) Runtimes of different test queries using the SSD hardware settings.

E

Pelikan User Guide

E.1. Starting Pelikan

Pelikan can be used to search for 3D interaction patterns in collections of protein-ligand complexes. It provides both, a graphical user interface (GUI) and can be used via command line parameters. Pelikan can be started from a console with:

```
./Pelikan --help
```

The program starts and lists up all possible options. In Table E.1 all options are shown and an explanation for each option is given.
Using the option '-c', the tool can be started as a command line tool. Otherwise, the GUI will open. In the console mode, two tasks can mainly be performed: a database can be created from a set of PDB files and a search for interaction patterns can be executed. In both cases, a database file has to be defined using the option '-o'. For database construction, PDB files can be defined as a folder (option -d) and as a list of several PDB files using the option '-i'. Exemplary, a command for creating a database using these options looks as follows:

```
./Pelikan -c -o testdb.sqlite -d /my/PDB/folder/
-i /my/PDB/folder2/abcd.pdb /my/PDB/folder2/efgh.pdb --recalc 3
```

The option '–recalc' defines which triangle descriptor table should be used. d_{DT} (option 3) has been proven to be most effective.
Using a similar command, an existing database can be extended by new protein-ligand complexes:

```
./Pelikan -c -o testdb.sqlite -d /my/PDB/folder3/
-i /my/PDB/folder2/ijkl.pdb /my/PDB/folder2/mnop.pdb
 --addDescriptor 3
```

Here, the new calculated PRPs are added to the existing triangle descriptor table.
A search for an interaction pattern can be performed using the following command:

```
./Pelikan -c -o testdb.sqlite -f myFilter.xml
--filtermode 3
```

-h [–help]	Show command line options.
-i [- -input] arg	Input pdb file(s), suffix is required. Several files can be declared by simply using spaces, e.g., -i a.pdb b.pdb .
-d [- -directory] arg	Parse all pdb files in folder.
-l [- -complexlist] arg	Parse all pdb files in this textfile. File must contain a list with paths to pdb files.
-o [- -database] arg	Open this database or, if it does not exist create this database. Information from complexes and substructures is written to this database.
-f [- -filter] arg	Use filter file to perform a search on the database.
-t [–development]	Open GUI in development mode, only for development.
-c [–console]	Start tool as command line tool.
–recalc arg	Recalculate the triangle descriptor: 1-simple, 2-recalculate complete table (point-row-descriptor), 3-recalculate complete Table (tri-row-descriptor), 4-recalculate complete Table (tri-point-descriptor))
–addDescriptor arg	Add new PRPs to triangle descriptor table: 1-simple, 2-add to table d_{PRP}, 3-add to table d_{DT}, 4-add to table d_{PRP-DT}(tri-point-descriptor))
–filtermode arg	Define which triangle descriptor should be used for the search process: 1- no descriptor, 2- d_{PRP}, 3- d_{DT}, 4- d_{PRP-DT}(tri-point-descriptor).
–license arg	Provide a license key.
–verbosity arg (=0)	Set verbosity level for status output during the run (0 = Quiet, 1 = Errors (default), 2 = Warnings, 3 = Info)

Table E.1.: Command line options for the console mode of Pelikan.

The option '-f' requires a specific file which contains a Pelikan filter. This can only be generated using the Pelikan GUI (see Section E.1.2). As before, the option '- -filtermode' can be used to defined the triangle descriptor table which should be used for the search. The parameter 3 is recommended here. The result of a search using the command line tool are different parameters of the search procedure, including the number of resulting hits and the required runtime. The actual results are not stored here. If the results should be inspected visually, it is recommended to perform a search using the GUI tool.

The GUI of Pelikan can be started with:

```
./Pelikan
```

Figure E.1 shows a screenshot of the GUI directly after its initial start.

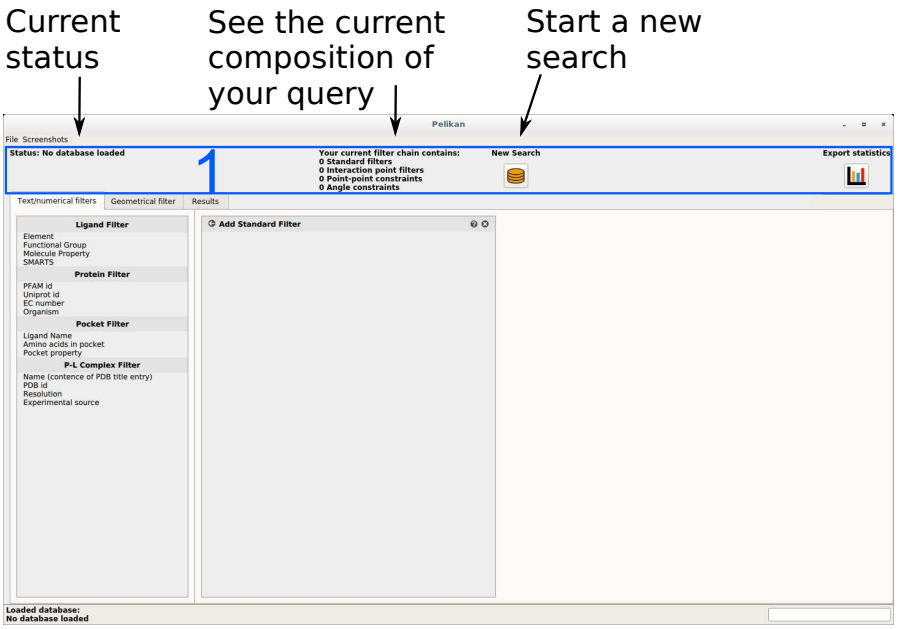


Figure E.1.: Screenshot of the Pelikan GUI directly after the initial start.

Conceptually, the GUI can be divided into two parts. In the lower part are different tabs which will be explained in the following sections. The upper part of the GUI offers a control panel (see Panel 1 in Figure E.1). Here, the current status of the GUI and the current composition of the query are shown. Moreover, a search can be started from here. In the following, the main tasks which can be performed with the GUI are explained: loading a database, creating a filter, inspecting the results of a search, and refining a search.

E.1.1. Load a Database

Pelikan stores all calculated data in an SQLite database. There are two ways how a database can be opened:

- Load an existing database via the context menu of the GUI: File → Open database
- Build a new database via the context menu of the GUI: File → Create new database. Afterwards a new dialog appears (see Figure E.2). Herein, two different attributes are required:

  1. Location: Name and path for storing the new database
  2. Source of complexes: Folder containing PDB files and/or a list of files. The file formats '.pdb', '.mmcif', and '.cif' are supported here.

Figure E.2.: Screenshot of the dialog for creating a new Pelikan database.

E.1.2. Generate a Filter

A query in Pelikan consists of two different filter types: Textual/numerical filters and geometrical filters. Both can be defined in two different tabs of the GUI and can be freely combined.

Filter for textual and numerical properties

Figure E.3 shows the first tab of the GUI. On the left, all possible filter attributes are listed (see Panel 1 in Figure E.3). They contain properties of the ligand, of the protein, of the pocket, and of the complete protein-ligand complex. By selecting a specific property from the list on the left, a filter is added to the list on the right (see Panel 2 in Figure E.3). Here, the exact parameters of a filter can be defined. All filters in the right list are part of the current filter chain.

Each filter element in the right filter list has three small buttons in its upper right corner:

- ? : Click here to get more information about the filter.
- include / exclude: Click here to change an including filter into an excluding filter and the other way around.
- x : Click here to delete the filter.

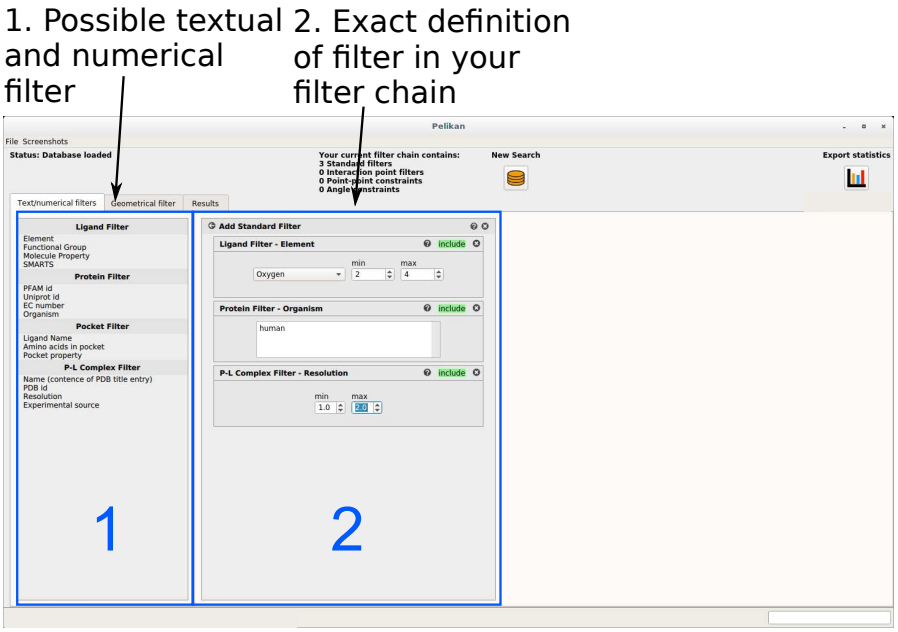


Figure E.3.: Screenshot of the first tab of the Pelikan GUI – Textual/numerical filter. (1) Textual and numerical properties for the ligand, the protein, the pocket, and the complete protein-ligand complex which can be used for a filter. (2) List of textual and numerical filter which are part of the current filter chain.

Geometrical filter

In the second tab of the GUI, the geometrical filter can be defined. A screenshot of this tab is shown in Figure E.4. In principle, a geometrical filter consists of three different elements: search points, point-point constraints, and angle constraints. Each of these elements is equipped with an unique identifier (id). In the following, these elements are briefly introduced and their creation from scratch is explained.

- Search point: A search point describes an atom which should be searched for. All search points of the current geometric filter are shown in the list in Panel 1 of Figure E.4. A search point can be added using the '+' symbol on the upper left corner of the search point list (see red square in Figure E.4). A search point has the following attributes:
  - Molecule: Molecule type of the atom, e.g., reference ligand, protein, or water.
  - Element: Element type of the atom.
  - Type: Interaction type of the atom, e.g., donor or acceptor.

  Depending on the attribute 'Molecule' of the search point, it might have further attributes:

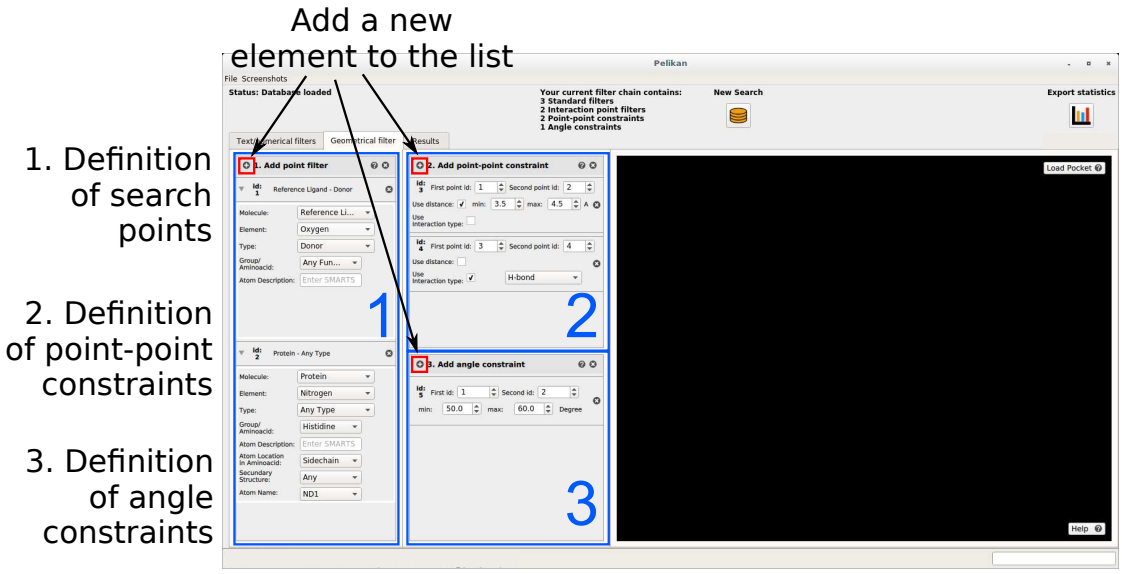


Figure E.4.: Screenshot of the second tab of the GUI – Geometric filter. (1) List of search points which are part of the filter. (2) list of point-point constraints which are part of the filter. (3) List of angle constraints which are part of the filter.

  - Functional group (only ligand and ref. ligand): Functional group an atom should be part of.
  - Amino acid (only protein): Amino acid an atom should be part of.
  - Atom description (only ligand, ref. ligand, and protein): SMARTS [89] pattern describing the searched atom. The relative orientation of other search points within the SMARTS pattern can be defined using atom labels referring to the search point ids, e.g., [C:2].
  - Location in amino acid (only protein): Backbone or side chain.
  - Secondary structure (only protein): Secondary structure the amino acid of the matching atom should be part of.
  - Atom name (only protein and only if amino acid type is set): Atom name of the matching atom. Corresponds to the atom name nomenclature from the PDB.

- Point-point constraint: Describes a distance or an interaction between two search points. All point-point constraints of the current geometric filter are shown in the list in Panel 2 of Figure E.4. A point-point constraint can be added using the '+' symbol on the upper left corner of the list (see red square in Figure E.4). A point-point constraint requires two ids of search points. The point-point constraint can have two different types:

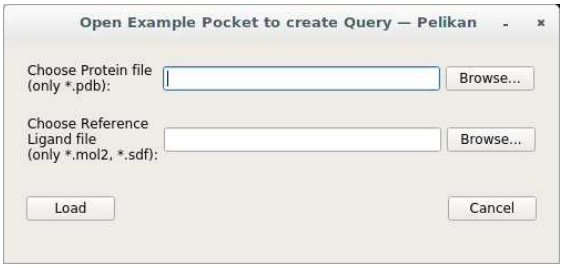


Figure E.5.: Screenshot of the dialog for loading a pocket in the 3D view for creating a 3D query.

  - Distance constraint: A minimal and maximal allowed distance in Å can be defined (see point-point constraint with id 3 in Figure E.4).

  - Interaction constraint: An interaction type can be defined, e.g., h-bond (see point-point constraint with id 4 in Figure E.4).

- Angle constraint: An angle constraint describes the angle between two vectors. All angle constraints of the current geometric filter are shown in the list in Panel 3 of Figure E.4. A vector can either be a point-point constraint or a search point. The vector for a point-point constraint starts at the position of its first search point and ends at its second search point. Only search points with an interaction type of 'donor', 'acceptor', and 'aromatic' can be vectors. For a donor, the vector points from the position of the heavy atom to the position of the hydrogen. If a donor has two hydrogens, two distinct vectors are associated with it. Accordingly, the vector of an acceptor points from the position of the heavy atom to the lone pair. If the atom has more than one lone pair, several vectors are generated. The vector of an 'aromatic' search point is the normal of the corresponding ring. It an angle constraint involves a search point which has more than one vector, a result is only returned for those atoms where at least one of the vectors fulfills all angle constraints.

Generate 3D query from pocket

Besides adding components of the 3D query using the '+' symbol in the respective list, a query can be constructed from a protein-ligand interface of interest. To this end, a pocket has to be loaded first. Click on 'Load Pocket' in the upper right corner of the visualization view. Then the dialog shown in Figure E.5 opens. Here, a protein file and a ligand file can be entered.

The loaded pocket is then displayed in the visualization view, as shown in Figure E.6. Search points can be added here by clicking on an atom. Properties such as the molecule of a search point and its element are automatically filled using the properties of the clicked atom. In the view, the atom is marked with a green halo.

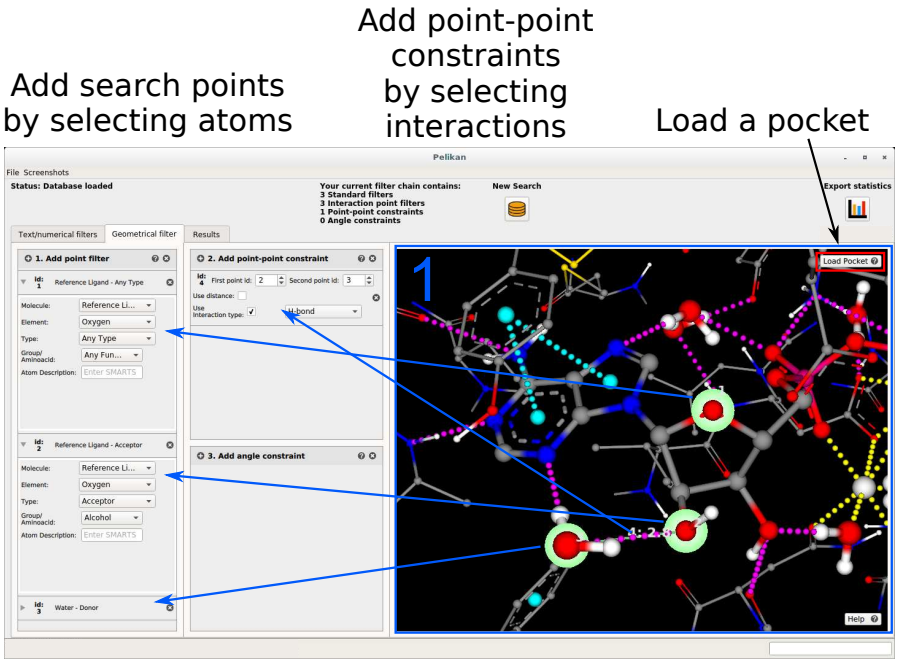


Figure E.6.: Screenshot of the second tab of the GUI – Geometric filter. (1) Search points as well as point-point constraints can be added to a query by clicking on a loaded pocket in the 3D view.

Point-points constraints can be added in two different ways:
1. Distance constraint between two search points: Click on one search point in the visualization view and move the mouse to the second search point. Release the mouse button when your are over the second search point. The used distance range will be automatically set to a range which includes the measured distance between both atoms.
3. Interaction constraint: Click on an interaction line. If both atoms which are involved in the interaction are already search points, only a point-point constraint will be added. Otherwise additional search points will be added. Note that the corresponding search point on an atom is only considered if the interaction type is correct. Otherwise a new search point and a distance constraint with distance 0 Å is added. The interaction type of the interaction constraint will be adjusted to the type of the clicked interactions line.

Save and load a filter

A filter, including textual and numerical components as well as 3D components, can be saved as xml file via the context menu: File → Save filter.
A filter can also be loaded via the context menu: File → Load filter.

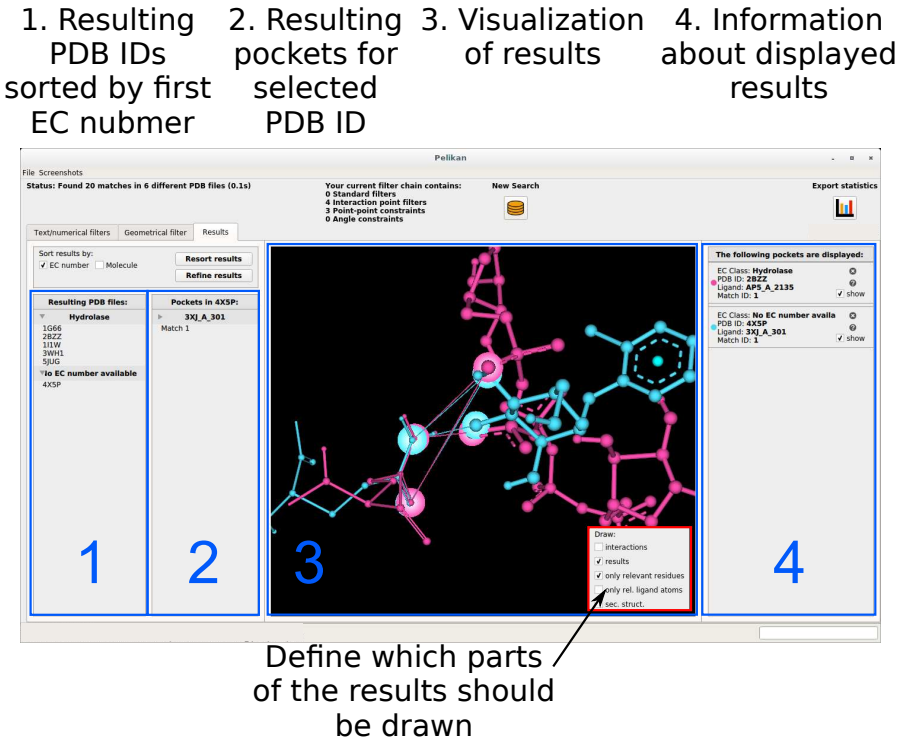


Figure E.7.: Screenshot of the third tab of the Pelikan GUI – Results. (1) Codes of resulting PDB files are listed. Initially, this list is sorted by the first EC number. It can also be sorted by molecule topology. (2) After a PDB code has been selected in the first list, all matching pockets of this PDB structure are shown. (3) Selected pockets are drawn in a 3D view. If several pockets are selected, they are automatically superimposed based on the atoms matching the search points. (4) Information about the currently drawn structures.

E.1.3. Inspection of Results

After a search has been performed, the results are presented in the third tab of the GUI: 'Results'. This tab is shown in Figure E.7. All codes of the detected PDB files are shown in the list in Panel 1 of Figure E.7. These are initially sorted by their first EC number. The detected pockets contained in a PDB file are shown in Panel 2 of Figure E.7 after a code from the first list has been selected. By clicking on a pocket in this second list, the corresponding hit is visualized in the central visualization view (see Panel 3 of Figure E.7). If several pockets are selected, they are automatically superimposed based on the atoms matching the search points. Using the box in the lower right part of the visualization view, the exact element which are drawn can be selected: interactions, results, only relevant parts of the ligand, only relevant parts of the protein, and secondary structure. The relevant parts are those parts involved in the matching.
All pockets which are currently drawn are shown in the list in Panel 4 of Figure E.7. Here, single pockets can be hidden, deleted from the visualization, and the title of the correspond-

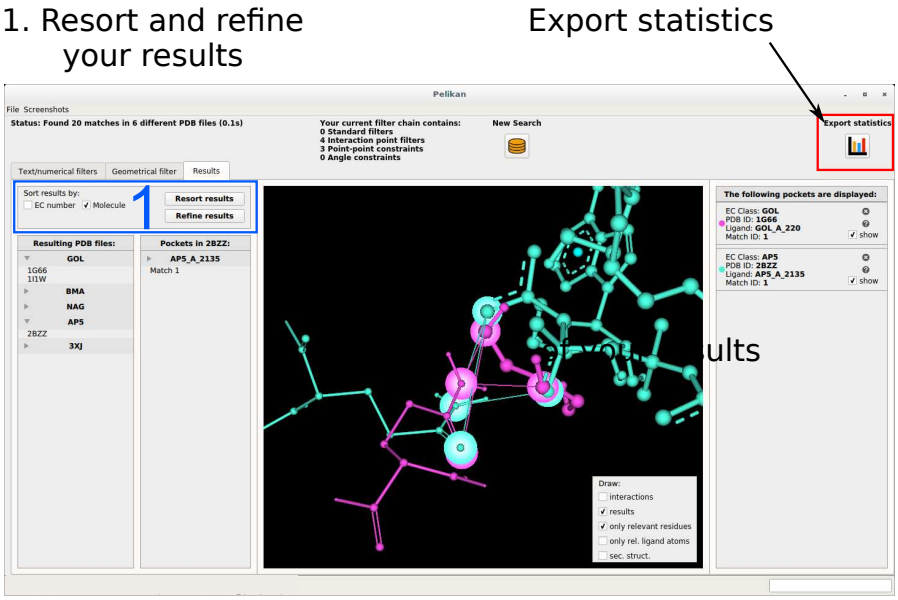


Figure E.8.: Screenshot of the third tab of the Pelikan GUI – Results. (1) Results from a previous search can be refined. Moreover, statistics of results can be exported.

Figure E.9.: Screenshot of the dialog for naming a specific result before it can be refined.

ing PDB file can be displayed as a tool tip from the '?' button.

The sorting of the results by the first EC number of their protein can be changed (see Panel 1 of Figure E.8). Here, the hits have been sorted by ligand topology.
Moreover, statistics of the results can be exported as text file using the button in the upper right corner of the GUI (see red square in Figure E.8).

E.1.4. Refine a Search

In order to refine a previous search, click on 'Refine results' in Panel 1 of Figure E.8. A dialog opens, as shown in Figure E.9. Here, a unique name can be entered. The currently displayed results will be assigned to this name in the following.

Afterwards, four new buttons appear in the upper part of the GUI (see Panel 1 in Figure E.10). Using these buttons, a new search can be performed only on result from a previous

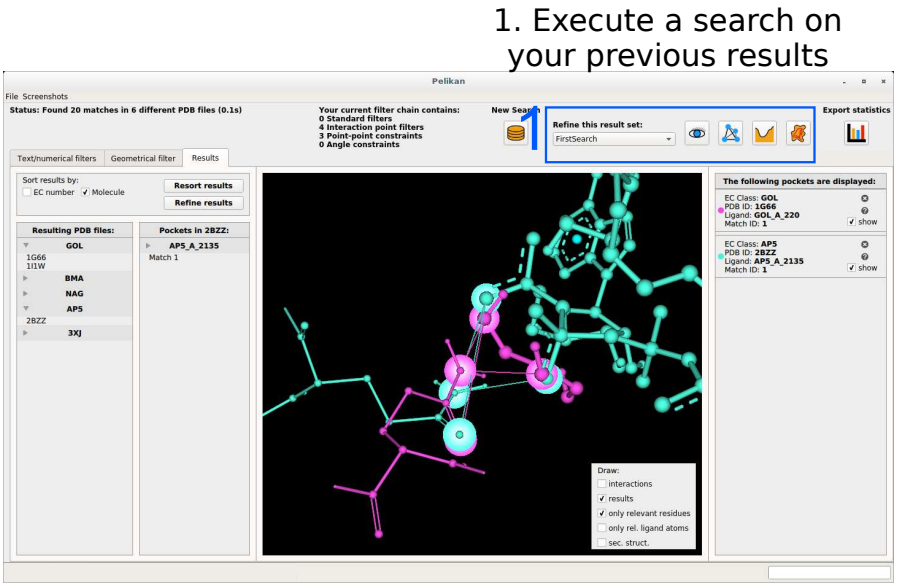


Figure E.10.: Screenshot of the third tab of the Pelikan GUI – Results, after a the results of one search have been assigned to a unique name. (1) A set of new buttons appears which can be used to refine a result.

search. The used previous results can be defined using the drop down menu. A click on the 'eye' button visualizes the selected results. Using the three neighboring buttons, a new search can be performed on all PRPs, on all pockets, or on all protein-ligand complexes from the selected results, respectively.

F

*NAOMI*nova User Guide

F.1. Starting *NAOMI*nova

*NAOMI*nova can be used to calculate and visualize geometric interaction preferences of molecular substructures. It provides both, a graphical user interface (GUI) and can be used via command line parameters. *NAOMI*nova can be started from a console with:

```
./NaomiNova --help
```

The program starts and lists up all possible options. In Table F.1 all options are shown and an explanation for each option is given. Most important is the option '-m' as it determines whether *NAOMI*nova is used as a GUI tool ('-m 0') or as a command line tool (-m 2). The

Command line option	Explanation
-h [- -help]	Show command line options.
-m [- -mode] arg	Mode of the tool. 0=Gui compiled (default), 1=Gui development (only for development), 2=Console.
-i [- -input] arg	Input pdb file(s), suffix is required. Several files can be declared by simply using spaces, e.g., -i a.pdb b.pdb .
-d [- -directory] arg	Parse all pdb files in folder.
-l [- -complexlist] arg	Parse all pdb files in this textfile. File must contain a list with paths to pdb files.
-o [- -database] arg	Open this database or, if it does not exist create this database. Information from complexes and substructures is written to this database.
-s [- -substructures] arg	File with substructures defined with a name, a SMARTS, a minimal EDIA, and a SMILES. Separated by blancks. One substructure per line
- -ediapath arg	Path for folder with electron density files.
- -csvpath arg	Path for folder with precalculated EDIA values as csv files.
- -license arg	Provide a license key.

Table F.1.: Command line options for the console mode of *NAOMI*nova.

option '-m 1' is relevant only for the development of *NAOMI*nova.
In the console mode, only two tasks can be performed: a database can be created from a set of PDB files and substructures can be added to the database. In both cases, a database file has to be defined using the option '-o'. For database construction, PDB files can be defined as a folder (option -d) and as a list of several PDB files using the option '-i'. A folder containing all electron density files can be defined using the option '- -ediapath'. Exemplary, a command for creating a database using these options looks as follows:

```
./NaomiNova -m 2 -o testdb.sqlite -d /my/PDB/folder/
-i /my/PDB/folder2/abcd.pdb /my/PDB/folder2/efgh.pdb
--ediapath /my/ED/folder/
```

If no folder for electron density files is given or if the electron density file for a specific PDB structure is not found in the given folder, the respective electron density file is downloaded from the PDBe web service.
Substructures can be added to a database using the option '-s'. Here, a file containing all substructures has to be defined. Each line in this text file has to look like this:

```
SMARTS name EDIAmin SMILES
```

The four attributes may only be separated by a blank. Using the following command, a text file containing several lines of substructure definitions can be added to a database:

```
./NaomiNova -m 2 -o testdb.sqlite -s subsfile.txt
```

Moreover, the tool can be used as GUI tool using one of the following commands:

```
./NaomiNova -m 0    or     ./NaomiNova
```

In the following, the main tasks which can be performed with the *NAOMI*nova GUI are explained.

Figure F.1 shows a screenshot of the *NAOMI*nova GUI directly after starting the program. In general, the main workflow for *NAOMI*nova is as follows: 1. Loading or creating a database 2. Defining and adding substructures 3. Filter partner points of interest 4. Inspect sets of partner points. Besides the loading and creating of databases, these different steps are distributed over the different tabs of the GUI: First tab – defining and adding substructures, second tab – filtering, third tab – visualization of partner point sets, fourth tab – visualization of partner point in protein-ligand interfaces. In the following, the different tasks and tabs are presented in more detail.

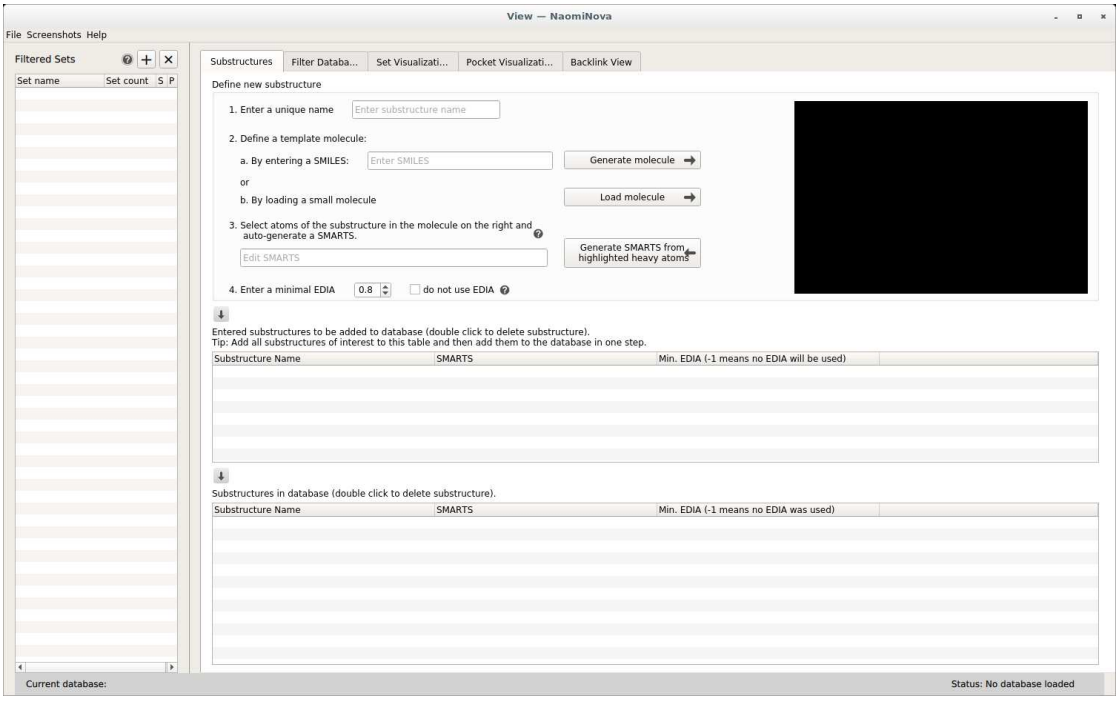


Figure F.1.: Screenshot of the *NAOMI*nova GUI directly after stating the program.

F.2. Loading and Creating a Database

*NAOMI*nova stores all calculated data in an SQLite database. There are two ways how a database can be opened:

- Load an existing database via the context menu of the GUI: File → Load database
- Build a new database via the context menu of the GUI: File → Create database. Afterwards a new dialog appears (see Figure F.2). Herein, three different attributes are required:

    1. Location: Name and path for storing the new database

    2. Source of complexes: Folder containing PDB files and/or a list of files. The file formats '.pdb', '.mmcif', and '.cif' are supported.

    3. Source of electron density: Folder which contains 2fo-fc electron density map files. Note that the electron density files have to be named exactly after their corresponding PDB code. Hence, for the PDB file '1j7u.pdb', the corresponding electron density file has to be named '1j7u.ccp4'. Moreover, EDIA values can only be calculated for PDB structures with a resolution of 2.5 Å and better.

Figure F.2.: Screenshot of the dialog for creating a new *NAOMI*nova database.

Additionally, missing electron density files are downloaded from the PDBe web service if the checkbox is selected.

By clicking on 'create', the database will be created. During the loading or construction of the database, the GUI stays active. However, no process which requires database operations can be started, e.g., filtering or adding of substructures.

F.3. Adding and Definition of Substructures

After a database is loaded, substructure can be defined and added to a database. This can be done in the first tab of the GUI. In general, the main area of the first tab is separated in three different panels: (1) New substructures can be defined (see Panel 1 Figure F.3), (2) Defined substructures are collected in a table before they are added to the database (see Panel 2 Figure F.3). (3) Substructures which are part of the database are listed in a table (see Panel 3 Figure F.3).

For the definition of a new substructure, four different attributes are required: a unique name, a template molecule, a SMARTS pattern, and an $EDIA_{min}$. They can all be defined in Panel 1 in Figure F.3.

- Unique name: name of the new substructure. Has to be unique.

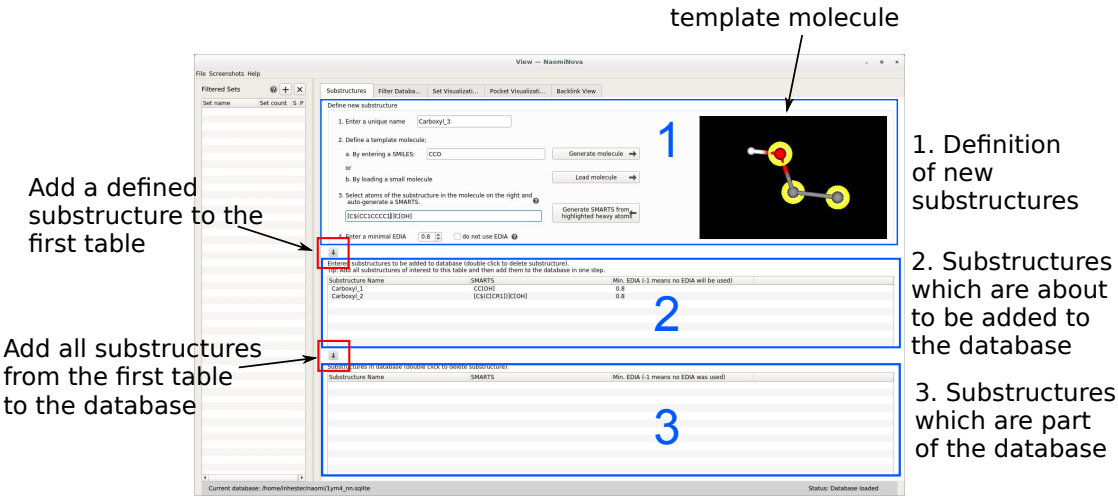


Figure F.3.: Screenshot of the substructure tab of the *NAOMI*nova GUI. (1) New substructures can be defined. (2) Defined substructures are collected in a first table before they are added to the database. (3) Table which contains all substructures which are added to the database. The buttons highlighted in red can be used to transfer substructures between the tables.

- Template molecule: Can be defined by uploading a small molecule from a file with 3D coordinates. Here, the file formats '.sdf' and '.mol2' are supported. Alternatively, a SMILES [85] can be entered. From this linear representation, a molecule with 3D coordinates is generated using UNICON [95]. The template molecules is then displayed in a 3D view (see F.3).
- SMARTS: A SMARTS pattern can be automatically generated by selecting atoms of interest from the template molecule. These atoms are highlighted in yellow. The SMARTS pattern can be manually adapted afterwards. Note that the SMARTS pattern without the used recursion has to match the highlighted atoms from the template molecule.

Using the button below Panel 1 in Figure F.3 (see red square), a defined substructure can be inserted into the first table, shown in Panel 2 in Figure F.3. In this process, the correctness of all substructure attributes are checked.
Finally, after all substructures have been defined, all substructures can be added to the database at once using the button below Panel 2 in Figure F.3 (see red square).

F.4. Filtering

After substructures have been added to a database, the collected data can be filtered for partner points of interest. This can be done in the second tab of the *NAOMI*nova GUI which is displayed in Figure F.4 In order to create a set of partner points, the following parameters

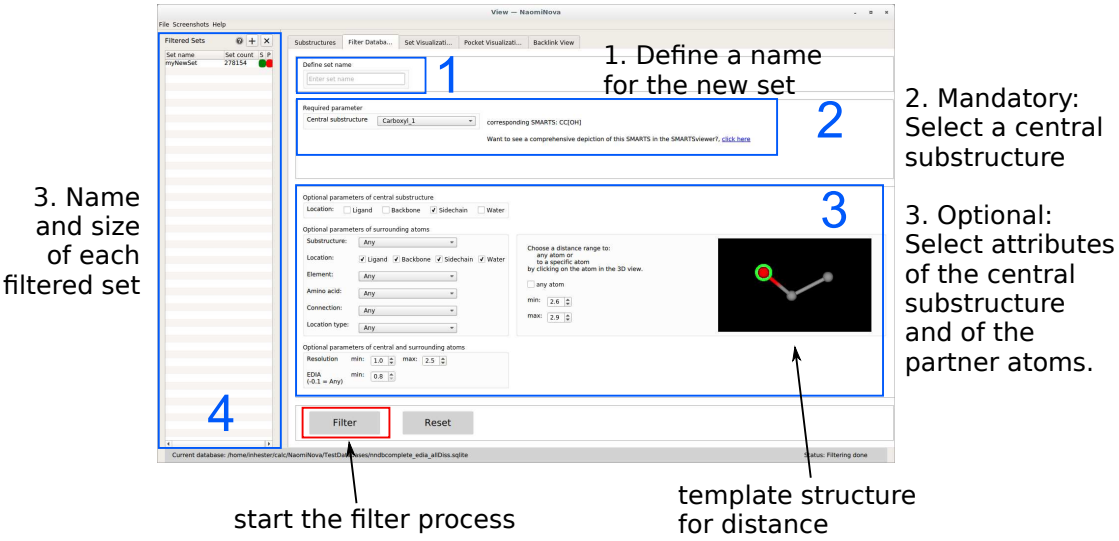


Figure F.4.: Screenshot of the filter tab of the *NAOMI*nova GUI. (1) The central substructure for a filtering process has to be defined. (2) Further properties of the central substructure and the partner atoms can be defined. The distance of a partner atom can either be calculated to any atom of the substructure or to a specific one (3) After the filtering process, the resulting sets of partner atoms are listed.

can be entered:

- Set name: Name of the new set. If not entered, the new set will be named 'myNewSet' (see Panel 1 of Figure F.4).
- Central Substructure: Unique name of a substructure from the database can be defined in a drop-down list (see Panel 2 of Figure F.4). Mandatory parameter.
- Attributes of central substructure: Optional parameter (see Panel 3 of Figure F.4). The following attributes can be defined:
  - Location (Ligand, Backbone, Sidechain, Water)
- Attributes of partner atoms: Optional parameter. The following attributes can be defined:
  - Substructure
  - Location (Ligand, Backbone, Sidechain, Water)
  - Element type
  - Amino acid type
  - Connection (intra or inter)

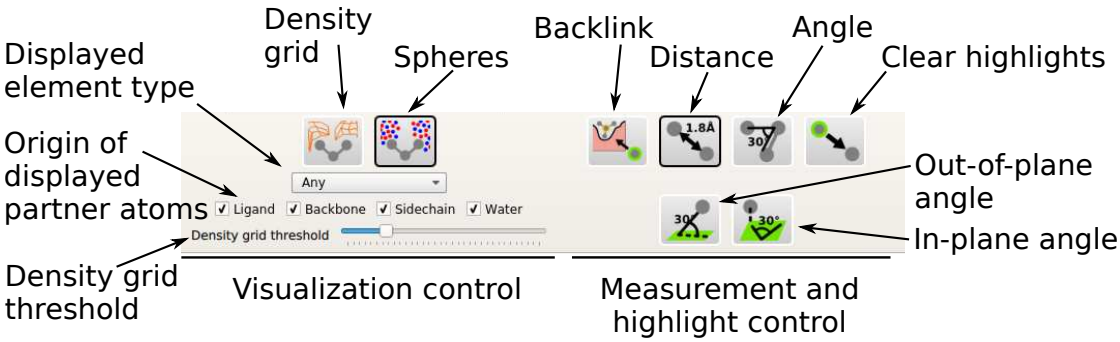


Figure F.5.: Screenshots of control panel in the substructure as well as in the pocket visualization tab.

- Location type (partner atoms is part of aromatic ring, aliphatic ring, or conjugated system)
- Distance to any atom of the central substructure or to a specific atom. This atom can be selected in the 3D view of the central substructure. The selected atom is highlighted in yellow (see Panel 3 of F.4).

The filtering process can be triggered by using the button on the bottom of this second tab (see red square in Figure F.4). After a set of partner points has been created, the name and the partner point count is listed on the left (see Panel 4 of Figure F.4).
Each set is marked with two colored dots. Both dots indicate in which view a set can be displayed. Herein, green stands for 'yes' whereas red indicates that a set can not be visualized. The left dot (category 'S') refers to the visualization in the **s**et view. Accordingly, the color of the right dot (category 'P') shows if the set can be displayed in the **p**ocket view.

F.5. Visualization

Sets which are displayed in the left list of the *NAOMI*nova GUI can be visualized in the set view, tab three of the *NAOMI*nova GUI, or in the pocket view, tab four of the GUI. Both tabs feature a set of control buttons which can be used to change the visualization of the current set or to perform measurements. The panel with the control buttons is shown in Figure F.5.

Conceptually, the panel can be divided into two parts:

- Visualization control:
  - Density grid button: Partner points of a set are displayed as density grid.
  - Spheres button: Partner points of a set are displayed as spheres.
  - Element types: Chose the element type of the displayed partner points.

  - Origin: Chose the origin of the displayed partner points.
  - Density grid threshold: Define the threshold of the displayed density grid.

- Measurement and highlight control (only active in sphere visualization):
  - Backlink button.
  - Distance measurement.
  - Angle measurement.
  - Clear highlights.
  - Out-of-plane angle measurement.
  - In-of-plane angle measurement.

In the visualization control, arbitrary combinations of the different options are possible. In the measurement and highlight control, only one button can be active at a time. In both parts of the control, a black border indicates that a button is active. Depending on which button is active, different measurements can be performed. In the following, the functionality of the buttons is explained for the set visualization tab.

F.5.1. Set Visualization Tab

In the set visualization tab, the sets are displayed in a central 3D view (see Panel 1 of Figure F.6 left). The two different visualization styles, i.e., density grid and spheres, are shown in the two screenshots in Figure F.6. The filter properties used to create the set currently displayed are shown on the right (see Panel 3 of Figure F.6 left).

Distance and angle measurements

If the sphere visualization of the partner points is chosen, distances and angles can be measured. In general, four different types of measurements are possible. Depending on the functionality which should be used, first the function has to be activated by clicking on the corresponding button.
In the following, the procedure for measuring the four possible parameters is explained:

- Distance: Activate the distance button in the upper part of the tab (see red square in Figure F.7a). Select any two atoms or partner points in the 3D view. The distance between the first and the second selected item is measured.

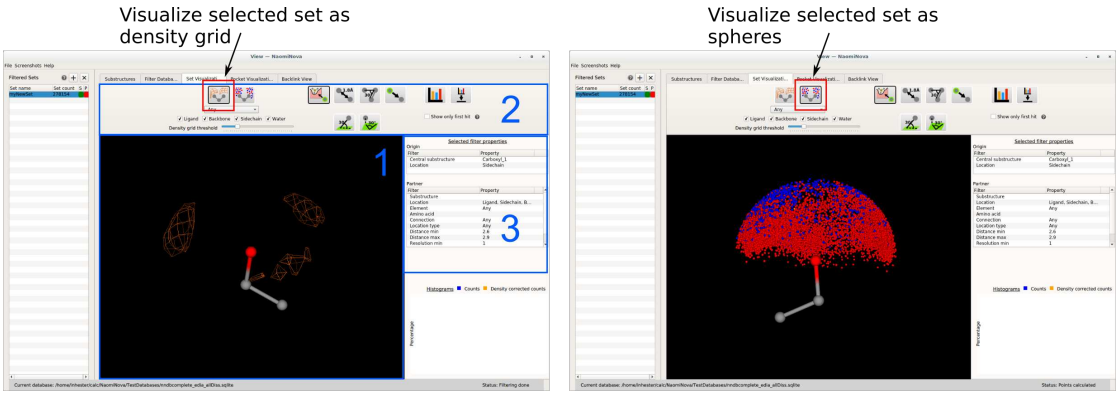


Figure F.6.: Screenshots of the set visualization tab of the *NAOMI*nova GUI. (left, 1) A set of partner atoms around a central substructure is displayed as density grid. (left, 2) The upper part of this tab contains several buttons which control the displayed data. The button in the red square is used to display all data of a set as density grid. (left, 3) All filter parameters for the currently displayed set are listed. (right) A set of partner atoms around a central substructure is displayed as spheres. The button in the red square is used to enforce this visualization mode.

- Angle: Activate the angle button in the upper part of the tab (see red square in Figure F.7b). Select any three atoms or partner points in the 3D view. The angle between the vectors from the first to the second clicked point and from the second to the third selected point is measured.

- Out-of-plane angle: Activate the out-of-plane angle button in the upper part of the tab (see red square in Figure F.7c). Select any four atoms or partner points in the 3D view. From the first three selected points, a plane is calculated. The fourth point is transformed into the plane along the plane's normal. Then, two vectors are constructed: from the first clicked point to the transformed fourth point and from the first clicked point to the original fourth point. The angle between these two vectors is measured.

- In-plane angle: Activate the in-plane angle button in the upper part of the tab (see red square in Figure F.7d). Select any four atoms or partner points in the 3D view. From the first three selected points, a plane is calculated. Herein, the vector from the first to the second selected point is the reference axis. The fourth selected point is transferred onto the plane using the plane's normal. Then, the vector from the first selected point to the transformed fourth selected point is defined and its angle to the reference axis is measured.

If the last selected point of the last measurement has been a partner point, the measurement can be extended to all displayed partner points. The this end, the histogram button in the upper right part of the third tab has to be selected (see red square in Figure F.7d). The calculated values are then displayed in a histogram (see Panel 1 of Figure F.7d). Moreover, the data plotted in the histogram can be exported using the button next to the histogram

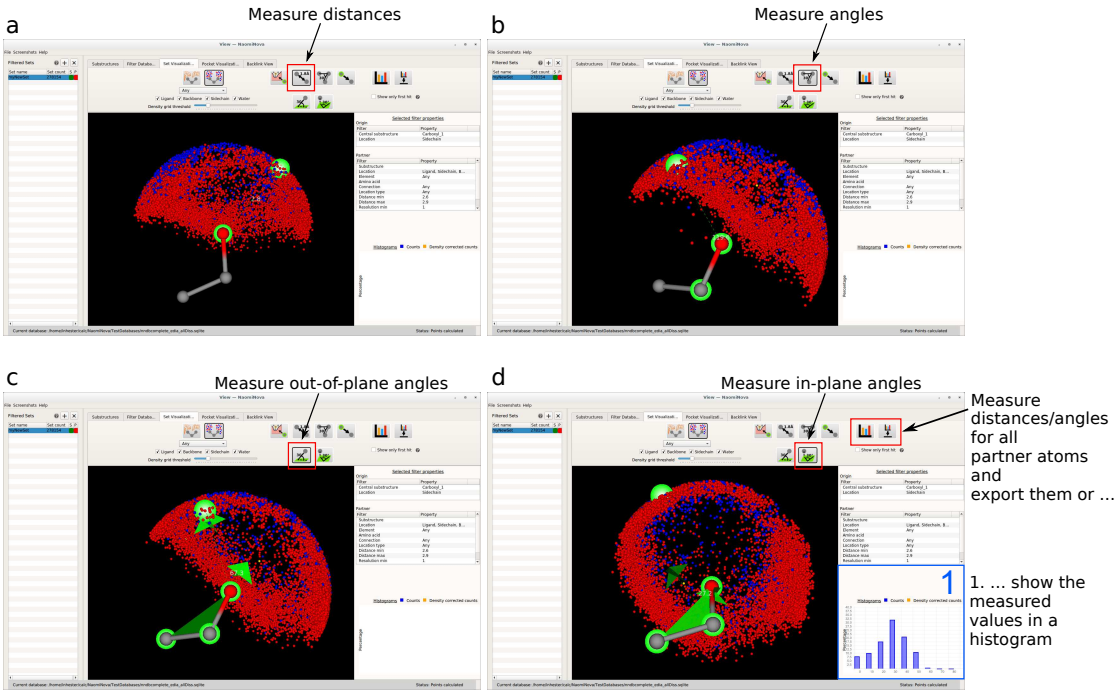


Figure F.7.: Possible measurements in *NAOMI*nova GUI, displayed in four different Screenshots of the set visualization tab. For each measurement, the respective mode has to be activated by clicking on a button. In each screenshot, the responsible button is highlighted with a red square. (a) Distances can be measured between any two points. (b) Angles can be measured between any three points. (c) Out-of-plane-angles are measured for four selected points. The plane is defined by the first three and the out-of-plane angle four the fourth is calculated. (d) In-plane-angles are measured for four selected points. The plane is defined by the first three and the in-of-plane angle four the fourth point is calculated. Using the histogram buttons, measurements can be displayed in the histogram in panel 1 and exported to a file.

button (see red square in Figure F.7d)

Backlink to original structure

For each displayed partner point, the original structure can be visualized. To this end, the backlink button in the upper part of the third tab has to be selected (see red square in Figure F.8 left). Then, any partner point in the 3D view can be selected. The GUI then automatically switches to the fifth tab – the original structure tab. Herein, the detected atoms of the substructure are highlighted in yellow whereas the atom represented by the partner point is highlighted in green (see Figure F.8 right).

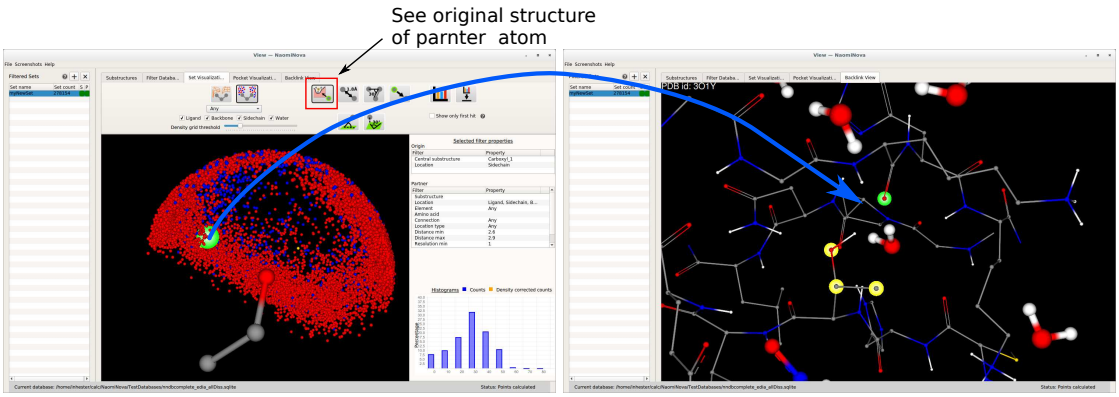


Figure F.8.: Backlink from a partner atom to its original structure in *NAOMI*nova GUI, displayed for the set visualization tab. (left) The selection mode of the GUI has to be set to the backlink function (see red square). Then, a specific partner atom of interest can be selected. (right) The original structure for a partner atom is displayed. The detected substructure atoms are highlighted in yellow. The partner atom is highlighted in green.

Show only first hit

Due to substructure symmetries, it might happen that the same set of atoms is matched by a substructure several times. By selecting the checkbox 'show only first hit' in the upper part of the third tab (see Panel 2 in Figure F.6), only the partner atoms from an arbitrary selected first hit are shown.

F.5.2. Pocket Visualization Tab

Sets which are displayed in the left list of the *NAOMI*nova GUI can be visualized in the pocket view, tab four of the *NAOMI*nova GUI. As for the set visualization tab, the upper part of this tab contains different buttons which can be used to control the displayed data and to measure distances and angles. Depending on the functionality which should be used, first the function has to be activated by clicking on the corresponding button.
In order to visualize a set in this tab, the following steps have to be done:

- Load a pocket: Load a protein and a ligand of interest by clicking on the 'Load Pocket' button in Panel 1 of Figure F.9b. A dialog, as shown in Figure F.9a, opens. Here, a protein and a ligand file can be defined in the upper two fields. Alternatively, a structure can be fetched from the PDB by entering its PDB id in the lower entry field. If the latter option is chosen, a new dialog appears after downloading the structure in which the ligand of interest can be selected. The loaded pocket will be displayed in the central 3D view.

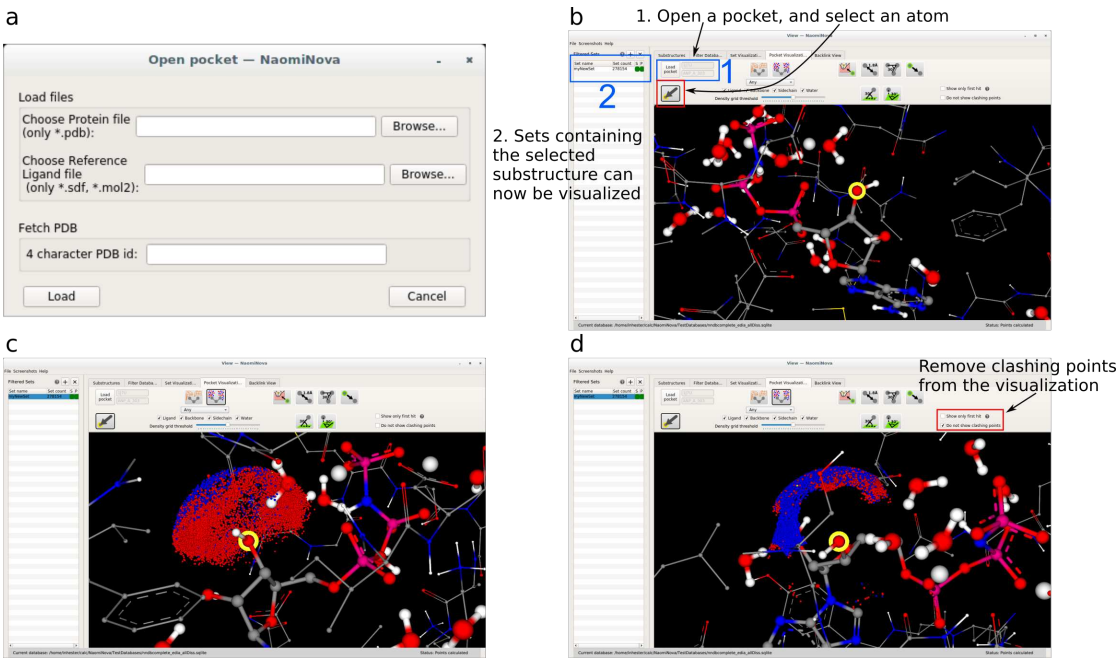


Figure F.9.: Screenshot of the filter tab of the *NAOMI*nova GUI. (1) The central substructure for a filtering process has to be defined. (2) Further properties of the central substructure and the partner atoms can be defined. The distance of a partner atom can either be calculated to any atom of the substructure or to a specific one (3) After the filtering process, the resulting sets of partner atoms are listed.

- Select an atom which should be part of the central substructure. To this end, the functionality for selecting an atom has to be activated first by clicking on the button below the pocket loading functionality (see red square in Figure F.9b). If this button is active, it is surrounded by a black square. The selected atom is highlighted in yellow. If this atom is part of a substructure for which a set currently exists, the color of its category 'P' will change from red to green in the left list (see Panel 2 of Figure F.9b).
- Select a set with a green dot in category 'P' in the left list. All data from this set will be transformed to the selected atom and displayed (see Figure F.9c).

Now, all display options and measurements are available as described for the set Visualization tab. However, histograms can only be calculated and shown in the set Visualization tab. As shown in Figure F.9d, partner points clashing with the atoms from the pocket can be removed by selecting the checkbox 'remove clashing points' (see red square in Figure F.9d).

G
Scientific Contributions

In this Appendix, my scientific contributions during the time of my PhD are listed. The contributions which are related to this thesis are highlighted in bold.

G.1. Publications

Therese Inhester, Eva Nittinger, Kai Sommer, Stefan Bietz, Matthias Rarey. NAOMInova: Interactive Geometric Analysis of User-defined Substructure Collections in Proteins. Journal of Medicinal Chemistry. 2017, 57 (9), pp 2132–2142.

Stefan Bietz, Therese Inhester, Florian Lauck, Kai Sommer, Mathias M. von Behren, Rainer Fährrolfes, Florian Flachsenberg, Agnes Meyder, Eva Nittinger, Thomas Otto, Matthias Hilbig, Karen T. Schomburg, Andrea Volkamer and Matthias Rarey. From Cheminformatics to Structure-Based Design: Web Services and Desktop Applications Based on the NAOMI Library. Journal of Biotechnology. 2017, ISSN 0168-1656, 261:207-214.

Eva Nittinger, Therese Inhester, Stefan Bietz, Agnes Meyder, Karen T. Schomburg, Gudrun Lange, Robert Klein, Matthias Rarey. A Large-Scale Analysis of Hydrogen Bond Interaction Patterns in Protein-Ligand Interfaces. Journal of Medicinal Chemistry. 2017, 56(6), pp 1105-1111.

Therese Inhester, Stefan Bietz, Matthias Hilbig, Robert Schmidt, Matthias Rarey. Index-based searching of interaction patterns in large collections of protein-ligand interfaces. Journal of Chemical Information and Modeling. 2017, 57 (2), pp 148-158.

Kai Sommer, Nils-Ole Friedrich, Stefan Bietz, Matthias Hilbig, Therese Inhester, Matthias Rarey. UNICON: A powerful and easy-to-use compound library converter. *Journal of Chemical Information and Modeling*. 2016, 56 (6), pp 1105-1111.

Therese Inhester and Matthias Rarey. Protein-Ligand Interaction Databases: Advanced Tools to Mine Activity Data and Interactions on a Structural Level. WIREs Computational Molecular Science. 2014, 4, pp 562-575.

G.2. Talks

Therese Inhester, Matthias Rarey. Fast Mining of Adaptable Interaction Patterns in Protein-Ligand Interface. 251st ACS National Meeting, 2016, San Diego, USA.

Therese Inhester, Matthias Hilbig, Matthias Rarey. Interactive Cheminformatics for Occasional Use in SMEs. 251st ACS National Meeting, 2016, San Diego/USA.

G.3. Poster Presentations

Therese Inhester, Eva Nittinger, Stefan Bietz, Matthias Rarey. NAOMInova: Intuitive Analysis of Interaction Geometries in Protein-Ligand Complexes. 2016, 12th German Conference on Cheminformatics, Fulda, Germany.

Therese Inhester, Matthias Rarey. Indexing Techniques and Algorithms to efficiently Mine Interaction Patterns in Large Sets of Protein-Ligand-Complexes. 2016, 1251st ACS National Meeting, 2016, San Diego, USA.

Therese Inhester, Matthias Rarey. Searching for Interaction Patterns in Protein Ligand Interfaces. 2015, 1st CSSB International Symposium, Hamburg, Germany.

Therese Inhester, Sascha Urbaczek, Benjamin Schulz, Matthias Rarey. Large Scale Generation of Small Molecule 3D Coordinates from Topology. 2013, 9th European Workshop in Drug Design, Siena, Italy.