Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Shape-based methods for Structure-based Fragment Growing

Kumulative Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat.
an der Fakultät
für Mathematik, Informatik, und Naturwissenschaften der
Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von

Patrick Penner
geboren in Bielefeld

Hamburg, den August 23, 2022

Erstgutachter: Prof. Dr. Matthias Rarey
Zweitgutachter: Prof. Dr. Andrew Torda
Drittgutachter: Prof. Dr. Peter Kolb
Tag der Disputation: 20.01.23

Acknowledgements

I want to thank Matthias Rarey for the opportunity to do my doctorate under his supervision. I was given a great project to work on and a lot of support to see it through to the end. I want to thank Pierre Ducrot for initiating the project and thereby creating this opportunity. I also want to thank my cooperation partners at Servier, first and foremost Virginie Martiny, for all the patience necessary to participate in and coordinate such a project. We had many fruitful discussions over the years with all the people involved and large parts of this project are a direct result. I want to thank the BioSolveIT, and especially Marcus Gastreich, for their continued support of the project.

I am very thankful to the members of my group who make up the great working environment. My project was absolutely dependent on the cooperative work ethic that maintains a significant amount of infrastructure in our group. I want to specifically thank Florian Flachsenberg for his guidance in structure-based drug design beyond his contributions to the project itself. I also want to thank Louis Bellmann for many in-depth discussions about things that never made it into the final version.

Zusammenfassung

Fragmentbasierte Medikamentenentwicklung ist ein einflussreiches Paradigma, das viele experimentelle und methodologische Ansätze revolutioniert hat. Computerunterstützte Ansätze in fragmentbasierter Entwicklung bestehen meist aus Software, die für vollständige Moleküle geschrieben wurde. Solche Programme gehen nicht auf den Unterschied zwischen Molekülen und Fragmenten ein. Sie sind auch oft nicht ausführlich statistisch validiert. In diesem Projekt haben wir ein neues Programm für strukturbasierte Fragmenterweiterung entwickelt, das auf diese Probleme eingeht.

Begonnen wurde auf der Ebene von Torsionswinkelbestimmung und dabei auf einem bestehenden System aufgebaut. Die Datenbasis der bestehenden Torsionswinkelstatistik wurde erneuert und signifikante Teile der Infrastruktur wurden umgeschrieben. Im Zuge dessen, konnte, durch die Integration von kürzlich entwickelter SMARTS Technologie, das System methodologisch signifikant verbessert werden.

Auf dieser Basis wurde ein geometrischer Algorithmus entwickelt, der besonders die gerichtete Natur der Fragmenterweiterung nutzt. Für die Bindetaschen und Fragmente wurden komplementäre geometrische Deskriptoren konzipiert, die schnell verglichen werden konnten. Die Qualität der Ergebnisse wurde durch, aus öffentlichen Strukturdaten extrahierte, Fragmenterweiterungen statistisch bestimmt.

Aus dem Algorithmus wurde durch den Zusatz mehrerer Komponenten ein produktives Programm für Fragmenterweiterung gebaut. Dazu war zum Beispiel eine Geometrie Optimierung der Ergebnisse notwendig. Es wurde auch ein einfaches Pharmakophor-Suchsystem integriert, um dem Nutzer tiefere Kontrolle über die Suche zu erlauben. Zuletzt wurde ein Ensemble-Flexibilitätsansatz implementiert, der Bindetaschen in mehreren Konformationen abbilden konnte. Das Programm wurde mit einem etablierten strukturbasierten System verglichen und die praktische Anwendung des Programms wurde anhand einer Fallstudie demonstriert. Die Entwicklung der Software fand in konstantem Austausch mit Industriepartnern statt und das Programm selbst wird bereits in mehreren Organisationen aktiv angewendet.

Abstract

Fragment-based drug design is an influential paradigm in pharmaceutical development. It has revolutionized experimental approaches and methodological concepts. However, computational support of fragment-based drug design often consists of software made for full-sized ligands. This generalization fails to address the differences between ligands and fragments. Furthermore, the validation of these methods is often not performed with statistically significant sample sizes. In this project, we developed a new tool for structure-based fragment growing that addresses these issues.

The tool was developed from the ground up by starting with the fundamental question of torsion angle preferences in small molecules. A long-standing torsion library system was regenerated with new data and large parts of the infrastructure reworked. In the course of this work, significant methodological improvements were made to the system using recently developed SMARTS technology, which led to chemically meaningful improvements in the torsion angle statistics.

On top of this we built the core of our tool, which is a novel shape-based algorithm that exploits the directionality of fragment growing. Shape descriptors could be generated symmetrically for pockets and fragments, and could be compared at high speeds. The quality of the results was measured using a set of fragment growing steps which were extracted from public crystal structure information.

A full-fledged modeling tool was built around the shape-based algorithm. This required a number of other components, such as geometry optimization. The workflow also included a pharmacophoric constraints system to search for fragments that generate specific interactions and an ensemble flexibility implementation to handle multiple conformations of a binding site. The tool itself was compared to an established structure-based method. Furthermore, the tool was applied in a case study to demonstrate a practical application. Our efforts have continually been close to active drug development projects and the tool is now in use at several organizations.

Contents

1. Introduction

A number of paradigms have defined the last 30 years of drug design. These have in general been driven by various technologies. High-throughput screening (HTS) was driven by assay miniaturization and automated detection[1]. A general speed-up in biological methods using high-throughput instruments led to an explosion of biological data that has spawned the field of omics data integration[2]. Improved accuracy in Isothermal Titration Calorimetry (ITC) enabled very precise experiments about protein-ligand interactions and an interest in thermodynamic lead finding and optimization[3], [4]. Interpreting ligand binding thermodynamics is a very complicated process[5], but it continues to deliver important insights to this day[6]. As these technologies spawned paradigms, the paradigms drove improvements in technology. One particularly influential paradigm is Fragment-based Drug Design.

1.1. Fragment-based Drug Design

The "SAR by NMR" paper published by Shuker et al. is widely credited as the starting point for FBDD[7]. As one of the defining papers for the paradigm, it demonstrates many aspects typical of FBDD. The authors initially screen several building-blocks or fragments that they intend to combine to form ligands. These fragments have micromolar or even low millimolar affinities and would therefore not be appropriate ligands in a drug design context in and of themselves. In this paper the authors generate a ligand by connecting two fragments with low affinities together with a linker to form a compound with nanomolar activity. To facilitate this linking process they require an experimental method that detects where fragments bind in relation to one another. In summary, the authors screen for low-affinity fragments that they elaborate with structure-based design into high-affinity ligands[7]. Although the "SAR by NMR" paper describes a nuclear magnetic resonance (NMR) focused technique, a lot of FBDD uses X-ray crystallography

as its source for structural information. X-ray crystallography provides the most complete picture of the fragment binding process and is by now a high-throughput method[8], [9]. Whether fragments can be tested by an experimental method also depends on the fragments themselves.

Designing fragment libraries is an important part of FBDD. Fragments are intended to be elaborated and must therefore be smaller than full-sized ligands. They are also often low-affinity binders, which means they must have sufficient solubility so that they can be screened at high concentrations[8]. These considerations have led to rule sets such as the "Rule of Three"[10] for fragments, which is an analogy to the "Rule of Five"[11] for ligands. Other considerations can be about the chemical diversity of the fragments in a set[12], their applicability to specific experimental methods[13], or even the shape diversity[14].

Fragments can be screened using several methods. The most popular methods are using biochemical assays, NMR methods, thermal shift assays, and X-ray crystallography[15]. Fragment screening with biochemical assays can use technology from HTS and generalize it to low-affinity binders[16]. Theoretical simulations show that NMR is very sensitive at detecting fragment binders that other methods may miss[17]. Thermal shift assays are a very simple method, but they may not be appropriate for all targets and often result in false positives[18]. Although X-ray crystallography may lag behind other methods in speed, it too is used as a high-throughput screening method[9]. These methods are rarely used in isolation and the identification, validation, and elaboration of fragment hits usually combine the information of multiple methods[17].

Having found fragment hits, there are several ways to combine them. The three main methodologies are fragment linking, fragment merging, and fragment growing[19]. Fragment linking means finding two fragment hits that bind to close positions on the target and using a chemical linker to make them into one ligand[7]. Fragment merging involves finding two fragments that bind overlapping positions and combining the important chemical features of both into one ligand[20]. In fragment growing, new functionalities are attached to a fragment based on the surrounding target environment to thereby grow a full-sized ligand. Fragment growing is by far the most common operation, but all three methods are in regular use[19].

1.1.1. Software in FBDD

The decision-making process that fragment elaboration goes through, and the design considerations preceding it, is often underreported in publications. Many modifications are simply "eye-balled"[21]. In silico methods sometimes guide or sanity check these design decisions. Most of the in silico methods used for fragment growing involve some kind of molecular docking engine[21]. In the simplest case, this may mean designing molecules based on fragment binding information and docking them into a binding site. Molecules designed this way may undergo several iterations based on feedback from the people involved in the project. Designers may also prioritize designs by criteria that cannot or are not coded into the scoring functions used in docking.

Because of the many repetitive steps involved, developers of computational FBDD methods have attempted to automate this process of iterative design using, for example, molecular docking engines and genetic algorithms as a metaheuristic sampling method[22]. Many different attempts have been made to abstract chemist-driven design processes. Such workflows focus on different aspects like statistical methods[23] or synthesizability[24].

Other methods that deserve mention are deconstruction-reconstruction approaches and binding site mapping with fragments. Deconstruction-reconstruction approaches deconstruct known ligands of a target, fragment them, and construct new potential ligands from these fragments. Fragment-based binding site mapping methods do not necessarily suggest new fragment hits or ligands but map the physico-chemical properties of a binding site using a set of previously defined fragments. An exhaustive discussion of the many different computational FBDD methods can be found in dedicated reviews[25]. Many of these methods are structure-based and therefore heavily dependent on the quality of the structure data presented to them.

1.2. Structure Data Quality

Structure elucidation of small molecules and macromolecules, whether for FBDD or otherwise, continues to present a challenge. Even disregarding experimental problems such as crystal formation, soaking failures, insufficient signal, and too much noise, converting an electron density derived from a clean X-ray experiment into a structure model is

not without ambiguity. How much ambiguity, what kind of ambiguity, and how this ambiguity is resolved are all measures of the eventual structure quality.

1.2.1. Quality Measures

A few structure quality measures were used throughout this project and will be introduced here. The most common measure of structure quality in drug design is resolution. The resolution of a structure is chosen by the crystallographer and is supposed to represent the range of Bragg spacings for which useful data has been collected[26]. Although there are valid criticisms to be made about the subjectivity of resolution, using it to filter structures is commonplace. There is some variation as to where to draw the line to acceptable structures, but many authors agree that "Once the resolution becomes worse than 2Å, [...] some published protein models appear to have been determined more by some crystallographer's imagination than by any experimental data"[26].

The R-factor or R-value is a measure of agreement between a measured diffraction pattern, for which an atomic model was built, and a diffraction pattern calculated for the atomic model[26]. The R-factor is used to refine atomic models and so to prevent overfitting a sibling parameter the R-free was introduced. The R-free is calculated by omitting a part of the data during refinement and then calculating the fit of the model to the data that was omitted. A large difference between the R-factor and the R-free is indicative of an overfitted model. R-factor and R-free are both very traditional measures of the quality of a structure model[26].

Occasionally it becomes necessary to determine the quality of fit for individual atoms. $EDIA$ is a method developed specifically for that purpose and in the context of molecular design[27]. $EDIA$ calculates whether single atoms are supported by surrounding electron density, while considering interference from neighbors and whether there is electron density further away that cannot be accounted for. This makes it a useful metric in the design of data sets of a statistical sample size[28], [29].

1.2.2. Structure Data Sources

The definitive source for structures of macromolecules is the Protein Data Bank (PDB). Established in 1971, it has since grown to over 190 000 structures at time of writing. PDB entries are of very heterogeneous quality and there are initiatives that try to standardize

them[30]. In practice, this means that for most applications PDB entries must be filtered by quality using criteria that are important to the problem at hand. Because the PDB is such an important source of bioactive conformations and protein-ligand complexes quite a few such data sets exist.

The PDBbind refined set[31] is one such quality filtered subset of the PDB. It is itself a subset of the larger PDBbind, which is a data set of PDB entries associated with binding affinity data[31]. The PDBbind and its subsets, the refined set and the core set, which makes up the CASF benchmark[32], are curated for the express purpose of scoring function development. This leads to a few decisions being made during curation, which may not be generally applicable in drug design, such as the exclusion of all covalent binders[31]. In using such a data set one must therefore be very aware of its intentions.

The Cambridge Structural Database (CSD) prides itself on containing "[...] all published organic and metal-organic small-molecule crystal structures"[33]. It is the definitive source for small-molecule crystal structures of very high quality. Filtering the CSD by quality is less important in drug design than filtering it by applicability domain, because it contains, for example, metal-organic structures, which are commonly avoided as drug candidates. The high quality of CSD structures does uniquely position it among structure data sources, especially because many insights derived from it seem to generalize to macromolecules, such as molecular interactions[34] or torsion angle preferences.

1.3. Torsions

Molecular conformations are defined by bond lengths, bond angles, and torsion angles. Bond lengths are defined as the distance between two atoms bound covalently or otherwise. Bond angles or valence angles are defined as the angles between two bonds, i.e. the angle between two atoms both bound to a third. Torsion angles are the dihedral angles between four atoms in a chain. Bond lengths and bond angles have very sharp energy optima that tolerate comparatively little change at room temperature. In drug design applications they are often either modeled as sharp harmonic potentials or explicitly set to idealized values. The energy landscapes of torsion angles are very heterogeneous and often contain multiple local energy minima that are likely to be populated at room temperature. It is largely torsions that facilitate molecular movement under physiological conditions.

Molecular movement is essential in many physiological processes. For example, Protein-receptors in cell membranes change their conformations to integrate an extracellular signal[35] and small molecule ligands adopt certain conformations to bind to their target[36]. There is some discussion as to how much conformational strain interactions with the target can compensate for. Nonetheless, the consensus is that most torsion angles in a molecule should be energetically favorable[37], [38]. To describe small molecule binding poses it is therefore imperative to be able to predict the likelihood of certain torsions.

1.3.1. Torsion Data

Torsion data is derived from databases of crystal structures, such as the PDB[39] and the CSD[33]. These represent experimental ground truths as far as their structure quality allows it. The PDB contains macromolecular structures and is therefore a valuable source of bioactive conformations[39]. Bioactive conformations are the eventual target of any conformer generation in drug design and should therefore be the subject of any validation. These can be used to validate generated conformations but must be pre-processed to ensure their quality[40]. The CSD is focused on crystals of organic and metal-organic small molecules[33]. Although these are not necessarily of bioactive conformations, CSD structures are generally of very high quality. It has been shown that geometric information learned from the CSD transfers well to the prediction of bioactive conformations[41]–[43].

Torsion angles are dependent on their chemical environment. The steric and electronic effects of torsions can be so impactful as to blur the line between the conformation and the configuration of a molecule[44]. Any kind of torsion angle statistic must therefore be associated with descriptions of the chemical environment in which the were generated. Some authors use custom keys made up of properties of the torsion substructure and its atoms to describe the chemical environment. Mogul, a molecular geometry searching application developed by the organization behind the CSD, uses the valences, atomic numbers, ring membership, etc. of the atoms involved in the torsion. These keys can then be arranged in a tree-like structure and matched against query structures[45], [46]. A more standardized and common choice is the molecular pattern language SMARTS[42], [47]–[49]. SMARTS are an extension of the ubiquitous SMILES language[50] that can be used to specifically describe almost arbitrarily complex substructures. Complex SMARTS may, however, become hard to interpret[51].

1.3.2. Evaluating Torsions

The likelihood of torsion angles can be used to evaluate conformations as well as generate them. A collection of quantum chemical (QC) methods represent the highest level of rigor and precision when it comes to evaluating torsions[52]. QC methods are computationally very expensive and thus suited to every task. Recently, a group at Pfizer has been precalculating torsion data with QC methods at scale using cloud computing and reinforcement learning[53], [54]. For QC calculations to converge in reasonable time frames, the torsions are usually considered as molecular fragments. When generating these fragments it also pays to keep delocalized effects in mind that may affect a torsion[55].

In terms of accuracy, QC methods are typically followed by force-field-based methods. Force-field-based methods attempt to calculate an energy for a given conformer using Newtonian approximations of atomic interactions. This energy is usually significantly less precise than a QC derived energy[52], [56]. Many conformer generators and docking procedures include some kind of force field in their workflow[56]. They are often seen as a reasonable compromise between speed and accuracy[40].

Knowledge-based methods are the fastest of the three methods but often settle for flagging likely and unlikely torsion angles[49]. Knowledge-based methods generate torsion angle likelihood statistics such as histograms, from crystal structure databases. These statistics are matched to chemical environments and used to evaluate torsions. Some methods try to approximate these statistics as smooth potentials to integrate them into large-scale virtual screening projects[57]. The speed of a method can still be a very defining factor in its popularity, like in the methods described below.

1.4. Molecular Shape

The shape of a molecule is defined by its conformation and its atomic volumes. A typical way of modeling atomic volumes is the hard-sphere model, in which every atom is considered to be a sphere of, for example, the van der Waals radius for the element of the atom. This leads to a collection of overlapping spheres in space that can be conceptually fused into one shape.

The shape of a molecule can influence many different properties of the molecule, such as its solubility. In fact, there are shape-based definitions of whether a polypeptide is a

protein that state a protein must have residues completely buried in its interior[58]. In drug design it has been shown that similarly shaped ligands may have similar activity, thereby extending the similarity principle into 3D space[59]. Furthermore, a binding process is predicated on the shape-complementarity of a ligand and a binding site[60]. The following introduces shape-based methods by example, but a representative fraction of shape-based methods can be found in dedicated reviews[34].

1.4.1. Shape-based Methods by Example

Rapid Overlay of Chemical Structures (ROCS) is the most widely recognized shape-based method. ROCS describes atom volumes with Gaussian functions instead of spheres, which has several advantages. Gaussian functions can be parameterized to produce very similar molecular volumes to hard-sphere models at high speeds[61]. Furthermore, Gaussian functions are smooth, continuous, and their overlap can be easily calculated. This means that two shapes defined by Gaussians can be optimized by gradient descent toward a local maximum of overlap, leading to rapid overlays of chemical structures[62].

ROCS demonstrates several traits typical of shape-based methods. Since the original publications on ROCS[63], a GPU implementation can now achieve over 1 000 000 overlays per second[64]. Shape-based methods, and ROCS in particular, tend to be very fast. Another shape-based method worth mentioning in this context is Ultrafast Shape Recognition, which achieved 5 000 000 comparisons per second on a CPU in the original publication from 2007[65]. The algebraic and comparatively simple methodology facilitates quick and accurate results.

Shape-based methods are very accurate considering their methodology. They often perform competitively in comparisons with more sampling intensive methods, such as molecular docking[42]. One of the strongest terms in a docking scoring function, and therefore one of the largest driving factors in pose generation, is interatomic clash[66]. Conceptually, shape complementarity can be seen as the reciprocal of interatomic clash. By optimizing shape complementarity shape-based methods extract one of the driving forces from molecular docking.

The third trait common to many shape-based methods is that they are intuitively visualizable. Although ROCS Gaussians are infinite functions, it is very simple to place hard-sphere shells at the origins of the functions that make up a molecular shape for the overlay of two structures. Intuitive visualization enables a user not just to discover

that some solution is better than another, but to trace why some solutions are better by visualizing the solutions and the steps taken toward them.

1.4.2. Diversity in Shape-based Methods

Not all problems can be solved by overlaying Gaussians. ROCS is fundamentally a method of overlaying small molecule structures and does not necessarily generalize, for example, to a structure-based context. Although methods exist that attempt to use Gaussian overlay and ROCS for this task, they run into several fundamental problems[67]. Placing target points in a binding site is not as straightforward as placing these target points on the atoms of another small molecule. Parameterizing the Gaussians at these target points also drifts away from a definition of molecular volume[67].

Successful structure-based applications have made use of shape-based methods such as surface comparison[68], [69]. Shape-based methods also explore different mathematical concepts[70]. Some authors even argue for using the diversity of shapes in a set of known ligands as a description of the receptor's flexibility[71]. Diversity in shape-based methods is therefore necessary to address specific problems and also develop new methodologies.

1.5. Validation in Computational Fragment-based Drug Design

Validation is an essential part of the process of developing new methodologies for drug design. Without validation, it is hard to prioritize the features of a method or even prove that they add any real value. Validation with statistical sample sizes is important to ensure that a methodology generalizes to more than just a handful of cases. In drug design the most definite validation is prospective, but justifying the process until then usually requires significant retrospective validation.

In Computer-Aided Drug Design (CADD) a large part of computational FBDD can be considered a subset of de novo drug design. De novo drug design can be defined as computationally building new molecules as opposed to virtually screening known molecules[72]. Virtual screening validation is typically performed on curated data sets of labeled molecules to describe whether a method can pick out molecules labeled active for a particular target rather than molecules labeled inactive[73]. Because de novo design can create never before seen molecules, validation has to be performed differently.

An intuitive way to perform validation of de novo design is by actually synthesizing and testing the molecules generated prospectively. This has the advantage of exactly replicating the intended use case and producing a ground truth as exact as the experiment allows. A handful of successful compounds can make a very compelling case to the right audience and so this method of validation is quite popular[24], [74], [75]. Its disadvantages are low sample sizes and heavy user bias. Experimental testing is resource intensive and so all examples cited included some kind of selection process performed by a user. This introduces a hard to quantify bias into the validation, which can easily overshadow the method's inherent performance. The cost of experimental validation also often leads to very few compounds being selected[24], further diminishing the statistical power of such validation.

A retrospective way to validate a de novo workflow is to try and generate molecules similar to known actives. Many de novo workflows use probabilistic sampling. This means that even if known active compounds are contained in the sampling space, a probabilistic approach may not always find them or even encounter them in particularly large spaces. To circumvent this problem, authors present the most similar compounds generated by their workflows as evidence that the workflow can generate realistic molecules[72], [76], [77]. Sometimes the similarity of these compounds is described by recognizable similarity metrics. These metrics vary from publication to publication.

Sommer et al. with their de novo design tool NAOMInext are some of the few authors that attempt to exactly replicate a statistical sample size of known molecules[78]. NAOMInext was developed in the context of fragment growing and so the validation consists of fragment growing steps from literature which are made up of a smaller input fragment in a binding site and a target ligand pose that the workflow tries to achieve. The data set in question was published by Malhotra et al.[79] but not necessarily for this purpose. Malhotra et al. attempt to evaluate how many fragment hits change their binding pose upon being elaborated into a full-sized ligand[79]. This results in very extreme examples of fragment growing steps, such as a trifluoromethyl as the input fragment and a steroid scaffold with a trifluoromethyl substituent as the ligand. In this case the steroid scaffold, which is many times bigger than the trifluoromethyl, would be the extension a fragment growing tool has to predict. This is not necessarily representative of fragment growing in practice.

In summary, validation in computational FBDD is performed very heterogeneously and each of the methods presented has advantages and disadvantages. This should not

distract from the fact that new methods should attempt the most precise validation available to establish themselves in the larger context of the field.

1.6. Thesis Content

In this work, we developed a structure-based fragment growing workflow called FastGrow using a shape-based algorithm for interactive use and rapid iteration. Although a significant amount of infrastructure was already present in the NAOMI C++ library used to implement it[80], we developed our algorithm from the ground up, starting at torsion angle preferences and conformations of druglike molecules, and moving on to the topics of molecular shape, intermolecular interactions, and geometry optimization. All parts of the workflow were validated extensively using a data set of automatically extracted growing steps from public crystal structure data that represented realistic fragment growing scenarios.

The following chapter will consider the torsion angle preferences of druglike molecules. It will build on previous works of the Torsion Library[49], [81]. Advances in SMARTS technology have the potential to significantly impact the Torsion Library's correctness. Integrating these advances into the tooling of the Torsion Library and improving the curated data will be the main focus of this chapter[D2].

Chapter 3 will discuss the creation of an automatic pipeline to mine fragment growing steps from crystal structure data. Besides consideration of structure quality, the fragment growing steps will also be chosen such that they reflect realistic and specific scenarios.

Chapter 4 will outline the methodology and validation of the shape-based algorithm at the core of FastGrow, as well as the custom conformer generation necessary. The advantages and disadvantages of the design will be discussed in some detail. Besides performance validation, we will also present robustness experiments and runtime analyses[D1].

Chapter 5 will combine the shape-based algorithm with intuitive pharmacophoric constraints to model intermolecular interactions, an ensemble flexibility approach, as well as geometry optimization for the generated fragment poses. We will explore how these features impact the performance and demonstrate use cases such as growing with water replacement and growing a micromolar fragment hit into a nanomolar ligand[D3].

Chapter 6 will draw a few overarching conclusions resulting from the project as a whole. These will discuss integrating torsion angle preferences from several perspectives. Furthermore, the conclusions will discuss the applicability of shape algorithms to the fragment growing context and the importance of facilitating user interaction with computational FBDD software.

Chapter 7 will envision improvements that could be built based on the current project. These will range from improving primitive growing operations, prioritizing new suggestions by FBDD specific criteria to optimizing the sampling process.

2. Torsion Angle Preferences

A correct description of torsion angle preferences is important to most three-dimensional drug design operations. For our project, we had to be able to retrieve torsion information very rapidly, which led to our choice of a knowledge-based method. We chose the Torsion Library which was developed in cooperation between the Universität Hamburg and Roche[49]. In the course of the project, some inconsistencies in the Torsion Library structure came to light, in large part due to the advancement of SMARTS comparison technology. This led us to generate a new version of the Torsion Library, which addressed these issues.

2.1. The Torsion Library

The 2013 Torsion Library as conceived by Schäerfer et al.[49] is itself based on a legacy of previous work by Klebe et al.[43] and further work on what was then called MIMUMBA by Sadowski et al.[48]. This latter version was developed with the conformer generator OMEGA in mind[42], as was the original 2013 Schäerfer Torsion Library. The last major update of the Torsion Library before our work was in 2016 by Guba et al.[81]. Guba et al. were motivated by occasions where torsions that seemed reasonable to modelers were marked as unlikely by the statistics in the Torsion Library. This mislabeling of torsions could be described as a larger statistical trend, which helped identify problematic patterns. Our work largely addressed inconsistencies in the hierarchical SMARTS structure and their consequences.

2.1.1. Torsion Library SMARTS

The Torsion Library uses SMARTS to describe the chemical environments of torsion angles. The four atoms of the torsion angle are marked with the SMARTS labels :1 - :4. The torsion angle stays the same whether it is measured from label :1 - :4 or :4 -

:1. The labels are mostly used to identify which molecular substructures match which parts of the SMARTS expression. Labels are not necessarily consistent in patterns describing the same or variations on the same substructure. The labels :2 and :3 are of particular importance because they match the atoms of the actual rotatable bond and must be present in every SMARTS pattern in the Torsion Library. The labels :1 and :4 are present in all torsion rules except for those where the 1st or 4th torsion element is replaced by an electron lone pair.

The Torsion Library makes two types of additions to the basic SMARTS syntax[47]: hybridization states and the nitrogen lone pair primitive. Hybridization states can be specified with ^1, ^2, or ^3 to mean sp, sp^2, or sp^3 hybridization, respectively[49]. This SMARTS extension is also found in other systems such as Open Babel[82]. Hybridization specifiers are effectively valence specifiers with geometry checks and cannot be considered an actual expression of the underlying atom orbitals. Nitrogens with lone pairs can be specified with N_lp[49]. This primitive requires a trivalent nitrogen to retain its lone pair, which is detected by a geometry check. Non-planar nitrogens pass this geometry check. If a torsion pattern contains only 3 labeled atoms, the Torsion Library will try to match a calculated lone pair position as the fourth point. The lone pair position is calculated using idealized VSEPR molecule geometry[83]. It is important to note that both SMARTS extensions are geometry dependent and so should only be used when matching 3D conformations.

Torsion Library SMARTS are arranged in a multi-level hierarchy[49]. At the top are 7 hierarchy classes that only encode the elements of the atoms forming the rotatable bond to match. One of the hierarchy classes is a catch-all class that accepts any rotatable bond, regardless of the elements of the two atoms. Below the hierarchy classes are the torsion subclasses. These must always be more specific than their parent class. Torsion subclasses may also be nested in other torsion subclasses. They often have names that label them as expressing a particular substructure relevant to drug design. Furthermore, torsion subclasses that are on the same level must be sorted from specific to generic. Below them in the hierarchy are torsion rules. They contain the actual angle preference histograms. Torsion rules must also be more specific than their parent class and torsion rules that are on the same level are also sorted from specific to generic. Torsion rules that are on the same level as torsion subclasses are matched first, except for in the generic hierarchy class where the single torsion subclass is matched first[49].

2.1.2. Torsion Library Data and Statistics

Small-molecule conformations to compute torsion angle preferences for torsion rules are taken from two data sources: the CSD[33] and the PDB[39]. The CSD is a database of high-quality small-molecule crystal structures. The structures are nonetheless filtered by quality according to criteria originally defined in Schärfer et al.[49] and translated for use with the CSD Python API by Penner et al.[D2]. The PDB is a database for structures of macromolecules that optionally contain small-molecule ligands. These ligands are extracted from their macromolecule receptors and filtered for quality in a workflow defined by Penner et al.[D2].

There are two different matching behaviors in the Torsion Library[49]: selective matching and unselective matching as shown in Figure 2.1. Selective matching takes a rotatable bond and its structural context as input and tries to match the most specific torsion rule in the Torsion Library. It does this by finding the appropriate hierarchy class, always entering the first torsion subclass it matches, and reporting the first torsion rule that it matches. If the procedure follows these rules and the Torsion Library structure is as described above, then the first torsion rule a selective matching finds will be the most specific. Selective matching is used to classify torsion angles into the categories: relaxed, tolerable, and strained. Unselective matching is used to populate torsion angle preference histograms of torsion rules. Unselective matching does not necessarily match any hierarchy classes or torsion subclasses before matching the torsion rules inside of them. When populating torsion rules with statistics, all torsion rules try to match all rotatable bonds in the conformations used as a data source. The torsion angles of all rotatable bonds that match are included in the histogram for a particular torsion rule[49].

Histograms in the Torsion Library have 36 bins that are 10° wide[49]. Each torsion rule has a set of two histograms per data source. One histogram has bins that begin and end at multiples of 10°, for example: [−170°, −160°). The other has its bins centered on multiples of 10°. The latter "shifted" histograms are preferred for visualization[49]. Peaks can be detected using a peak detection as described in Schärfer[49] but are in any case set manually. Both the simplicity of the peak detection as well as the nature of the data require an expert in the loop. Peaks are further parameterized by 2 tolerances, which correspond to the traffic light classification scheme. Torsion angles found within the first tolerance are considered relaxed (i.e. green). Torsion angles within the second tolerance are considered tolerable (i.e. yellow). Torsion angles outside of all tolerances are considered strained (i.e. red)[49].

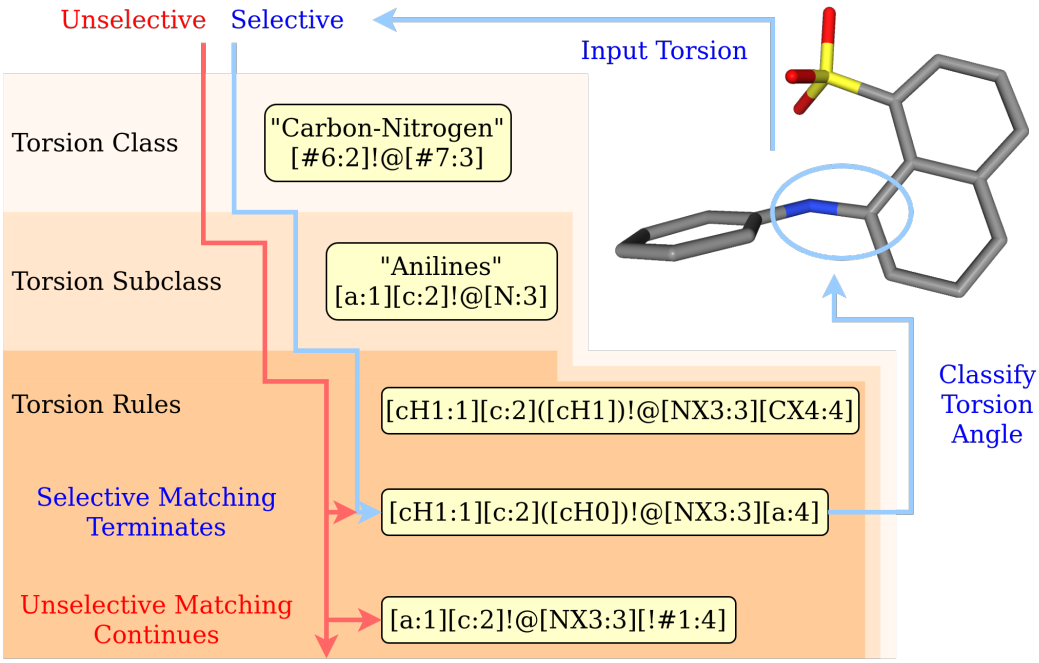


Figure 2.1.: Matching behavior of a torsion in the Torsion Library. The molecule is 8-anilino-1-naphthalene sulfonate (2AN) from PDB file: 2ANS. All matching procedures pass through the SMARTS hierarchy. Selective matching terminates at the first matching torsion rule, whereas unselective matching continues. Subclasses are subsets of the torsion class or subclass above them in the hierarchy. Torsion subclasses and torsion rules that are on the same level are sorted from specific to generic. All 3D molecule images were made with the NGL viewer[84]. Reprinted with permission from [D2]. Copyright 2022 American Chemical Society.

2.2. SMARTScompare

SMARTScompare is an algorithm that compares SMARTS expressions by enumerating the atom and bond types the individual SMARTS nodes can match[85]. With this information in hand, SMARTScompare can, for example, calculate similarity metrics or detect if one SMARTS is more specific than another. A more specific SMARTS is in this case defined as one that matches only a subset of the chemical space defined by another SMARTS. These SMARTS can be said to be in a subset relationship, where the more specific SMARTS defines a subset of the chemical space[85]. SMARTScompare has applications other than in the Torsion Library[51] and must be extended slightly to compare Torsion Library SMARTS patterns.

2.3. Semiautomated Improvement of Torsion Rules with SMARTScompare

2.3.1. SMARTS Hierarchy Improvements

To correctly match Torsion Library SMARTS patterns, SMARTScompare needed to be made aware of the labels in the Torsion Library patterns. Otherwise, SMARTScompare could have compared patterns after shifting the position of the rotatable bond or unrelated nodes in the environment. All labels available during a comparison of two patterns were mapped to their counterparts and occasionally flipped if the pattern was symmetrical[D2].

Two properties of the Torsion Library hierarchy could be checked with SMARTScompare[D2]. The first was whether the SMARTS of torsion subclasses and torsion rules were always more specific than (i.e. in a subset relationship with) those of their parent class. This was done by traversing the Torsion Library and recursively checking all members of a torsion subclass or hierarchy class. The nature of these errors in the hierarchy meant that they could not be corrected automatically so SMARTScompare was only used to detect them. The second was whether torsion rules that are on the same level or torsion subclasses that are on the same level were always sorted from specific to generic. These errors could be corrected automatically by implementing a sorting algorithm that placed more specific elements before more generic elements. All problems detected and corrected by SMARTScompare were also considered manually. Especially

the corrections of patterns that were not more specific than their parent classes had to be performed with their chemical meaning and intention in mind.

2.3.2. Data Sources

The torsion angle preferences that were present in the Torsion Library since the publication by Guba et al.[81] were generated from CSD and PDB conformations extracted in the original publication in 2013[49]. A new subset of high-quality CSD ligands was generated with the CSD Python API according to filter criteria that can be found in Penner et al.[D2]. The PDB data used in Schärfer et al. was a proprietary subset of the PDB[49]. We used an automated workflow to extract high-quality ligands from the PDB and annotate them with $EDIA$ values[D2]. Annotating the ligands with $EDIA$ values instead of filtering by $EDIA_m$ values meant that single well-resolved torsions of molecules that had less resolved parts could still be evaluated.

2.3.3. Torsion Tools

In the course of this work, the tools of the Torsion Library were reworked extensively. An overview of the Torsion Library ecosystem is shown in Figure 2.2. The new version of the TorsionPatternMiner was used for all hierarchy checks using SMARTScompare and to generate new torsion angle statistics based on the new data sets. The TorsionAnalyzer was reworked to be useful both as a command line tool as well as a GUI application. The original Qt application created in Schärfer et al.[49] had been difficult to maintain for a while, which meant that it was often unreasonably difficult to classify torsion angles of input molecules. The TorsionAnalyzer was therefore rewritten as a command-line application. This command-line application was then wrapped in a web application to create a web-based GUI.

2.3.4. Evaluation

The changes we made in Penner et al.[D2] were largely to the SMARTS hierarchy. A full accounting and explanation of all steps performed can be found in the Supporting Information of Penner et al.[D2]. Some of these changes corrected trivial syntactic errors, but quite a few of them were impactful semantic errors. One example was an incomplete Benzamidine pattern. The error in the pattern led to unselective matching behavior

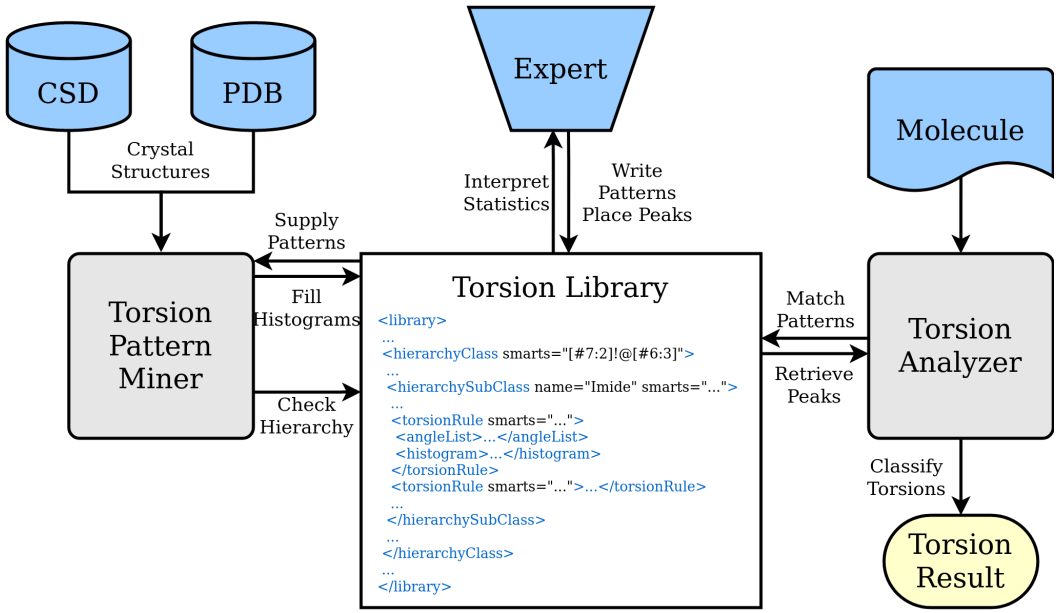


Figure 2.2.: Overview of the Torsion Library ecosystem. All inputs, the crystallographic databases, the symbolic expert, as well as a molecule to classify, are in blue. The two tools that interact with the Torsion Library are in gray. The TorsionPatternMiner is used to process the crystallographic data, as well as to check the consistency of the SMARTS hierarchy. The TorsionAnalyzer is the application-oriented tool which uses the Torsion Library to classify molecules. A part of the Torsion Library XML demonstrating the file contents has been placed in the center. Reprinted with permission from [D2]. Copyright 2022 American Chemical Society.

that completely overshadowed the torsional preferences of a bis-ortho substituted Benzamidine with less specific matches. Fixing the pattern led to a complete inversion of the peaks for that particular torsion rule.

Another example was an incorrect sorting of patterns involving bonds between nitrogen and sulfur. A generic pattern that accepted any kind of nitrogen and any kind of aliphatic sulfur had been placed before a collection of patterns defining very specific valence states of the nitrogen and the sulfur. Only the generic pattern was ever matched by a selective matching procedure, which meant that many molecules were classified according to a more generic description of sulfur-nitrogen bonds rather than the intended specific ones. By correctly sorting the generic pattern after the more specific ones all patterns could be matched in selective matching. This led to a more precise description of the chemical environments of torsions.

The Torsion Library as a whole remained quite stable according to the parameters defined by Guba et al.[81]. Figure 2.3 shows that a few more patterns that produce more than 40% strained alerts have appeared. These 6 patterns selectively matched fewer than 9 torsions in the CSD data set and could not be considered a statistically significant problem. The change of matching behavior measured by this metric was thus much smaller than the change from Schärfer et al.[49] to Guba et al.[81].

2.4. Further Quality Considerations

A derived criterion to keep in mind when working on the Torsion Library is the consequences for conformer generation. Conformer generation is not a particularly sensitive evaluation of a torsion library[42]. This is probably because no conformer generator completely samples the torsional space. Figure 2.4 shows benchmark runs of the platinum diverse set[40] using the Conformator[41] and different version of the Torsion Library. All versions produce very correlated results. The correlation between the 2016 version and the 2022 version is a little stronger ($r^2 = 0.90$) than the correlation between the 2013 version and the 2016 version ($r^2 = 0.79$). This suggests that the change from 2013 to 2016 was larger than from 2016 to 2022.

Improving the quality of Torsion Library matching behavior by minimizing strained alerts can have the unintended effect of making the tolerances of peaks too wide to handle torsions that were falsely classified as strained. It is somewhat difficult to assess

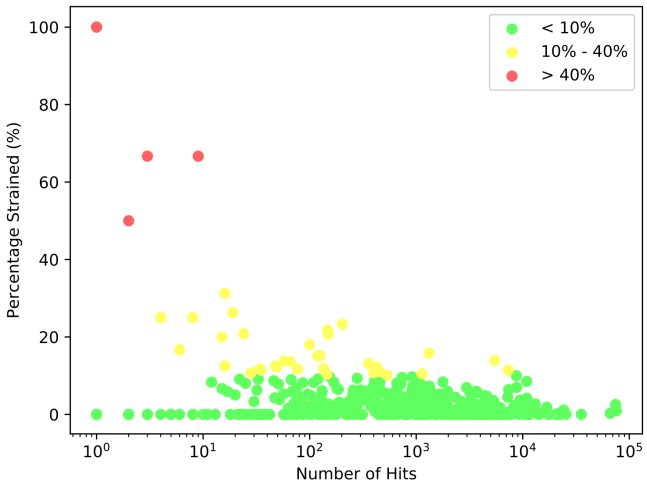


Figure 2.3.: Percentage of rotatable bonds marked as strained with respect to the number of total matches for all patterns of the current Torsion Library after all hierarchy checks, sorting, and manual corrections. The point at 100% strained matches and $10^0(1)$ hits is overplotted and actually represents three patterns. Reprinted with permission from [D2]. Copyright 2022 American Chemical Society.

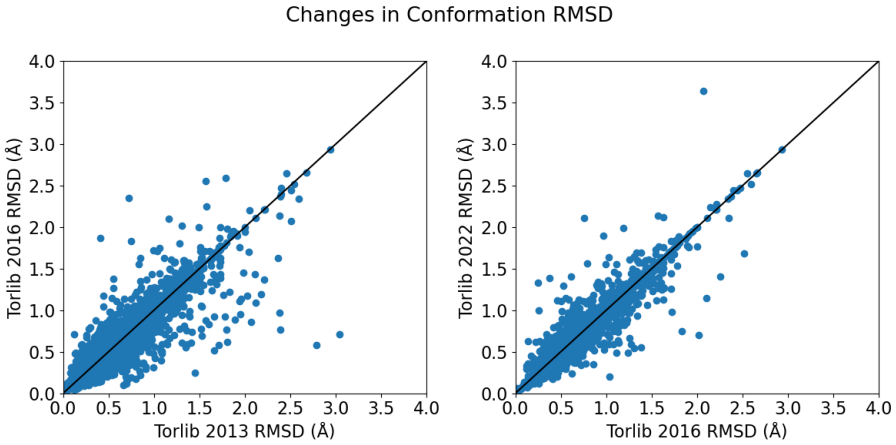


Figure 2.4.: Benchmark runs of the platinum diverse set[40] using the Conformator[41] and different version of the Torsion Library. Every point is one small-molecule. The x and y axes are the RMSD of the best conformer generated for a molecule to its bioactive conformation using the respective Torsion Library version.

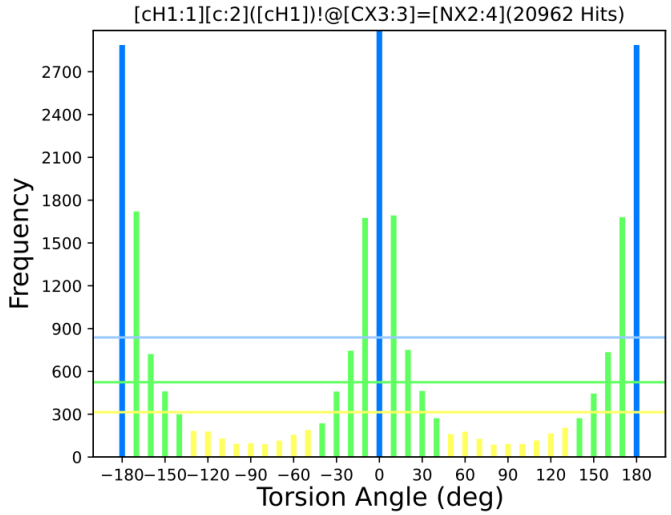


Figure 2.5.: Histogram of the torsion angles from unselective CSD matches for the pattern "[cH1:1][c:2]([cH1])!@[CX3:3]=[NX2:4]". Bars colored in blue represent peak positions. Bars colored in green and yellow represent the extent of the first and second tolerances, respectively. The three colored horizontal lines represent the minimum population necessary for a bin to be a peak/in the first tolerance/in the second tolerance according to the rules in the peak detection algorithm.

which torsions are falsely classified as relaxed or tolerable due to too wide peaks. One approximation is to use the population percentages that are defined for the peak detection algorithm to see whether tolerances are justifiable. In theory, a bin denoting a peak has to be filled with at least 4% of the total population of all bins in the histogram. Similarly, all bins within the first tolerance should contain 2.5% and all bins within the second tolerance should contain 1.5% of the total population of the histogram[49]. Bins not meeting these criteria may result in a false classification of torsion angles into relaxed or tolerable.

Figure 2.5 shows a histogram generated for the pattern "[cH1:1][c:2]([cH1])!@[CX3:3]=[NX2:4]" in an unselective matching of the CSD. All bins of the histogram are within the second tolerance of either the peek at 0° or 180°. 22 bins in this histogram do not have enough population to justify their inclusion in either the first or second tolerance. Almost 15% of the selective matches using the CSD, meaning those that represent torsion angle classifications, hit one of the 22 bins with insufficient population. This may be intentional and justifiable but deserves a second look.

2.5. Outlook

The Torsion Library and the ecosystem surrounding it have become quite complex. It may be reasonable to scale down some of the complexity for maintainability reasons. There are a few instances of complexity that may not be necessary. For example, in the generic hierarchy class, the single torsion subclass is matched before the torsion rules instead of the other way around as in all other hierarchy classes. This could be solved by placing all torsion rules that are on the same level as that torsion subclass into a different subclass and making sure the current subclass is matched first. Furthermore, 57 patterns receive fewer than 100 matches in an unselective matching of the CSD and 4 patterns are never matched at all. It is very questionable if these patterns can be supported by a knowledge-based approach if not enough data is present.

Protomers and tautomers have continually been an issue in the Torsion Library. Although NAOMI[80] has tautomer canonicalization, it cannot handle all issues automatically. For example, several substructures are modeled differently in the CSD and the PDB. The CSD tends to model Guanidine substructures with a charge. Ligands extracted from the PDB by NAOMI do not have charged Guanidines. The charge itself is not present in relevant patterns, but it changes what protonation states are reasonable for a molecule. It is difficult to enumerate these problematic substructures because the only way this phenomenon manifests is by fewer matches coming from one of the data sources. Nonetheless, it would improve the quality of Torsion Library matches to find a consistent way to handle as many of these substructures as possible.

Although SMARTScompare can detect many inconsistencies in the SMARTS structure, we could also demonstrate an inconsistency in the semantic meaning of a SMARTS pattern that could not be syntactically detected[D2]. It would therefore be very beneficial to establish a consistent rule set for writing SMARTS in a unique and human-readable way. Consistent human-readable SMARTS would make semantic errors far easier to detect.

Unpublished work on a torsional potential using the statistical distributions of torsion angles in torsion rule histograms has been ongoing for a while. Having the license to publish the exact bin counts of the histograms means that torsional potentials could be generated far more precisely than before. There is a need for a continuous description of torsion angle preferences[66], which could now be much more precise.

There is still much to do when it comes to the feature set and consistency of the Torsion Library. It has stood as a robust pillar of torsion angle preference information for quite some time now. Maintaining this robustness must, however, be an intentional effort.

3. Fragment Growing Validation

Validation in FBDD is performed very heterogeneously. Developing a new fragment growing workflow, however, requires constant validation to ensure the usefulness of the method. Ideally, this validation is performed with statistically meaningful data sets. In contrast to Sommer et al.[78], we did not use the data set by Malhotra et al.[79] due to its occasionally unreasonable fragment growing steps and decided to extract a fragment growing data set from public crystal structure data, which we called the Fragment Growing Validation Data set (FGVD). An overview of all its subsets can be seen in Table 3.1.

Name	Size (test cases)	Purpose
Self-growing Set	3967	Sanity checking and parameterization
Cross-growing Set	425	Performance measurement and comparison
Interactions Set	252	Validate pharmacophoric constraints
Water Replacement Set	162	Validate water replacement
Ensemble Growing Set	246	Validate ensemble flexibility approach

Table 3.1.: Overview of all data sets in the FGVD, their size, and their purpose. One test case involves one full growing procedure.

3.1. Structure Data

The first consideration we made before creating the data set was the data source. The data source chosen was the PDBbind and specifically the PDBbind refined set[31] so that some confidence could be placed in the crystal structure models. Throughout the project, we generated validation sets on the 2018, the 2019, and the 2020 versions of the PDBbind refined set. We found that, because the PDBbind is hand-curated, the refined set implicitly prioritized structures relevant to drug design.

3.2. Self-growing

The first validation set we generated in this project was the self-growing set. The concept behind the self-growing set is analogous to self-docking[31]. In self-docking, the validation task docks a ligand back into its own binding site after removing it. In self-growing, the validation task is reattaching a previously removed fragment to its ligand in that ligand's binding site[D1].

An overview of the self-growing set generation workflow is shown in Figure 3.1. Self-growing test cases were generated based on the ligands of the PDBbind refined set[31]. All ligands were randomly cut at single bonds and the two resulting structures were evaluated. The larger structure was considered the ligand core and the smaller the putative fragment. The fragments had to fulfill properties, both structural and physico-chemical, such as not being completely solvent exposed and "Rule of Three"[10] compliant[D1]. Only one fragment was extracted for each ligand in the PDBbind refined set[31]. If the current cut produced a duplicate fragment, the iteration continued to the next random single bond. Fragments extracted this way were stripped of their 3D information and saved as SMILES[50] strings. The input for a self-growing test case is therefore the ligand core, its binding site, and the fragment as a SMILES string. The SMILES strings were usually processed into a fragment screening database, which was the actual source of fragment information.

The success criterion was to reattach the fragment provided as a SMILES in a pose with less than 2 Å root-mean-square deviation (RMSD) to the fragment's atoms in the crystal structure. Only the atoms of the fragment were included in the RMSD calculation. The percentage of successfully regenerated poses was reported. In the latter FastGrow publication[D3], 95% confidence intervals were calculated on the percentage of successfully generated poses. This could be done by assuming that the binomial distribution of a less than 2 Å criterion approached a normal distribution at larger sample sizes, as was the case in all data sets of the FGVD[86].

The self-growing set had 3299 test cases and 1189 unique fragments at the time of the RVM publication[D1] and based on the PDBbind refined set v.2019. It grew to 3967 test cases and 1637 unique fragments at the time of the FastGrow publication[D3] based on the PDBbind refined set v.2020. The self-growing set was not featured in the FastGrow publication[D3], but it was regenerated as part of the whole FGVD. Self-growing was used for two purposes throughout the project. The first was sanity checking.

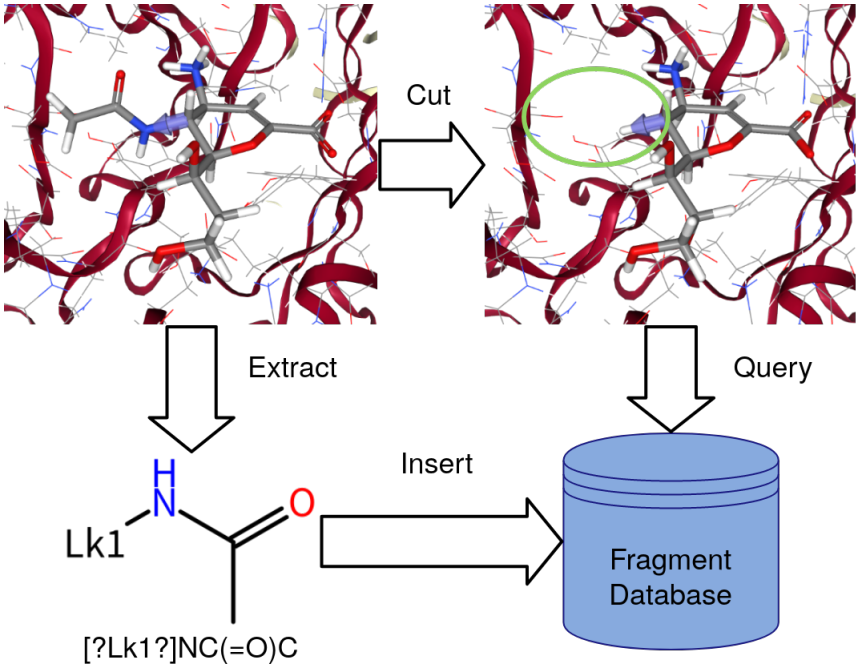


Figure 3.1.: Overview of the self-growing validation. A bond is chosen, according to filter criteria, shown in the top left image as a blue arrow. Ligands are cut at this bond in their pockets and used as queries to see if they can be reconstructed. The fragments cut off of the ligands are extracted and inserted into the fragment database to be queried. "Lk1" represents the linker resulting from the cutting procedure. Reprinted with permission from [D1]. Copyright 2020 American Chemical Society.

Failing test cases in the self-growing set are the easiest to correlate with methodology because they are simpler than the other test cases described below. The second purpose was parameterization. Although we cannot call the self-growing set a true training set because of its overlap with the data sets described below, all parameterization was done on this data set. While self-growing was therefore of less importance in describing performance, it was very important in method development.

3.3. Cross-growing

The cross-growing set was the main data set for measuring performance. Cross-growing is in this context analogous to cross-docking[31]. In cross-docking, a ligand from one structure is docked into a different structure of the same binding site. In cross-growing, we wanted to find pairs of ligands in different structures of the same binding site with a common core that only differed by one fragment attached to the core by a single bond. If we then used the core and binding site from the first structure and attached the fragment from the second structure to that core, we would grow the ligand of the second structure. By making sure we only used 3D information from the first structure we could use the second structure purely as a reference structure and see if we could generate the ligand from the second structure in a reasonable pose. This is closer to a real application than self-growing because we use the first structure, which represents a known structure to grow from, to extrapolate to the second structure, which represents the unknown structure that would result from experimentally validating the proposed growing.

An overview of the cross-growing set generation workflow is shown in Figure 3.2. Cross-growing test cases were made by first generating sequence identical and aligned ensembles of binding sites in the PDBbind refined set. A maximum common substructure (MCS) search was performed on every pair of ligands in the same binding site[85]. Each MCS was checked if it fulfilled the properties of a common core and if the fragments that would result from that common core fulfilled certain properties. The properties in detail can be found in the RVM publication[D1]. The input was therefore very similar to self-growing: the common core and the binding site from the first structure, as well as the fragment as a SMILES from the second structure. The success criterion in cross-growing was growing the fragment atoms into a pose with less than 2 Å RMSD to the second structure[D1]. Once again only the fragment atoms were used to calculate the RMSD.

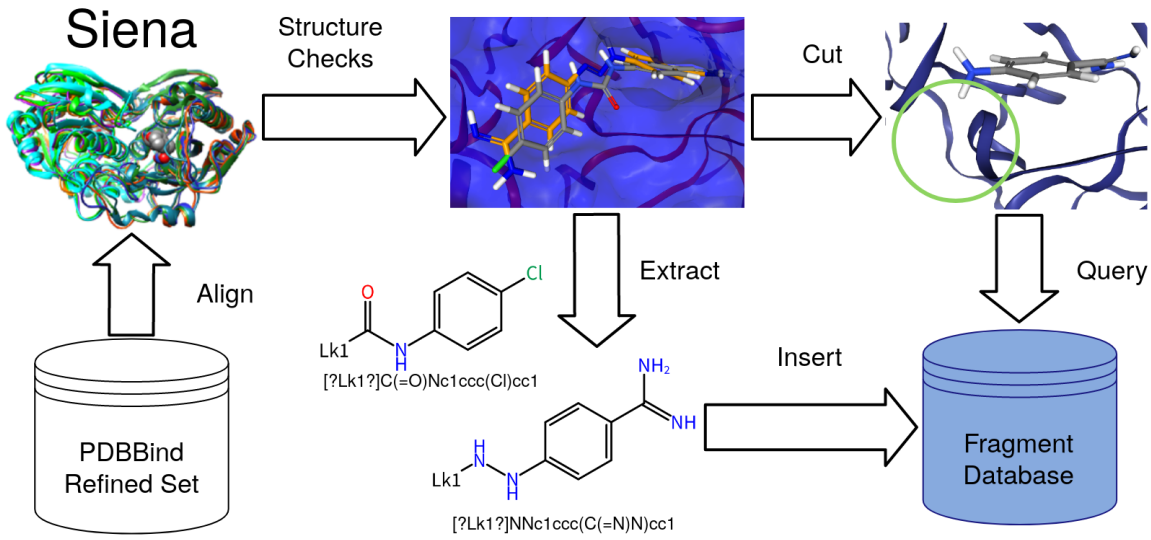


Figure 3.2.: Generation of the cross-growing set. Functionally equivalent pockets from the PDBBind Refined Set are aligned to binding site ensembles. These ensembles pass through structural checks to ensure the ligands have a common core and the binding mode of that core has not changed significantly. The variable parts attached to the common core are extracted as fragments and the common core, as well as the pocket, can be used as a query to find them again. The examples are from the PDB codes 1bju and 3gy2. Reprinted with permission from [D1]. Copyright 2020 American Chemical Society.

The cross-growing set had 326 test cases and 155 unique fragments at the time of the RVM publication[D1] and based on the PDBbind refined set v.2019. It grew to 425 test cases and 176 unique fragments at the time of the FastGrow publication[D3] based on the PDBbind refined set v.2020. The cross-growing test cases were used for performance measurement in the RVM publication[D1] and as the basis of performance comparison with established methods in the FastGrow publication[D3]. The data set was generic enough to generalize to different methods without significant effort.

3.4. Specialized Validation Sets

Several specialized validation sets were created to validate specific features of a fragment growing workflow. All were derived from the cross-growing set. The least invasive modification was generating protein-ligand interactions for test cases of the cross-growing set to validate their inclusion in fragment growing.

3.4.1. Interactions

Protein-ligand interactions for cross-growing test cases were generated according to interaction models based on crystal structure statistics[87] and the JAMDA scoring function[66]. Only interactions that were stable across both structures in a cross-growing test case were used. The 252 cross-growing test cases that had stable interactions according to the interaction models were run separately and added the interactions to the input. The validation task was extended to not only calculate and RMSD to the reference structure but also to see whether the input interactions were conserved in the growing process[D3]. By checking whether interactions were stable across both structures, the choice of interactions was biased slightly. This could have influenced the pose reprediction performance. For that reason, the focus of analyses using the interactions set was placed on the conservation of interactions rather than improvements in pose generation.

3.4.2. Water Replacement

Replacing water molecules with grown fragments became an interesting use case during development. It was validated as a special case of conserving interactions, where the

pharmacophoric constraints were positioned on waters in the binding site to be replaced. A validation data set for this was generated by finding waters in the cross-growing set that would be displaced upon performing the growing step. If the fragment displacing the water was able to generate a similar interaction to the one the water had with the binding site, this interaction was used in displacing that water. The 162 cross-growing test cases that displaced waters with interactions were run separately and the water displacing interactions were added to the input. The success criterion stayed the same, growing a pose of the fragment atoms with less than 2 Å RMSD, as in the unmodified cross-growing set.

3.4.3. Ensemble Growing

Proteins may change their conformation upon ligand binding and this change in conformation may be specific to the ligand[88]. This can lead to situations where a binding site structure crystallized with one ligand may not be geometrically compatible with another known ligand. One way some methods deal with this is by accepting multiple binding site structures that as an ensemble describe the flexibility of the system. To validate such a feature we created the ensemble growing set. Ensembles were generated using SIENA[89] and test cases of the cross-growing set. Sequence identical and RMSD clustered structures of binding sites were extracted for each test case. A minimum of two binding sites, excluding the reference binding site, had to be found for a test case to be included in the ensemble growing set. A maximum of 5 structures were chosen by SIENA for every test case. 246 such test cases could be generated. The success criterion for this validation task was the same as the cross-growing set: generating a pose for the fragment atoms with less than 2 Å RMSD.

3.5. Validating Performance w.r.t. Binding Affinity

The PDBbind was initially also chosen as a data source because of its binding affinity information. The idea was that chemical series that only varied one fragment could be extracted and the affinities of these transformations compared. The filtering procedure had to be more strict than in the cross-growing set to ensure that one perfectly equal common core was present across the whole chemical series. After having built this workflow, it became apparent that there were simply too few such chemical series in

the PDBbind to make meaningful statistical statements. The chemical series that were found were usually biased toward particular targets such as the Carbonic Anhydrase.

Another attempt was made by abandoning the requirement for structural data and mining affinity data from the binding affinity database ChEMBL[90]. Although the ChEMBL contained a wealth of data, using this data was not straightforward. Even after applying standard ChEMBL quality filtering[91], the data set still had undesirable properties. Many "chemical series" that could be extracted from the ChEMBL only had 2 entries. However, some systems had massive chemical series and would therefore skew the resulting statistic considerably. All of this led to a situation where none of the methods that were run to validate fragment growing could achieve more than random performance and it was questionable whether it was the method's fault or the data set's fault.

There was no shortcut available to a well-curated data set. Curation of such a data set, however, would have gone significantly beyond the scope of this work. Data sets for similar tasks in virtual screening are built over decades and are still far from perfect[73], [92], [93]. Creating such a data set will take a significant dedicated effort, which should address not only the usual issues of data source and quality but also how this specifically relates to molecule fragments instead of ligands.

3.6. Outlook

The Fragment Growing Validation Data set was generated using a pipeline of software based on the NAOMI framework[80] and the PDBbind refined set[31]. It was updated twice in the course of this work, from PDBbind refined set v.2018 to v.2019 and then to v.2020, and can be continually updated. Both of the updates performed led to significant growth of the data set. In the latter case, for example by a 30% increase in the size of the cross-growing set. This resulted in more statistically significant results. Furthermore, methods that achieve a stable performance across multiple versions of the FGVD could be considered more stable overall. Maintaining and continually updating this data set has already shown promise and may continue to do so.

The FGVD's impact could be further increased by applying more methods to it and publishing the results. Two methods were benchmarked using the FGVD in this work, but a broader application could serve to standardize performance benchmarks in the

context of structure-based fragment growing. The FGVD is focused on the primitive operation of attaching fragments in correct positions and so may not be ideal for probabilistic enumeration workflows. Regardless even these workflows need robust single-step fragment attachment which could be validated with the FGVD. Although the FGVD was generated specifically to validate FastGrow, it generalizes well to other software and could provide a standard to the otherwise heterogeneous validation in computation FBDD.

4. Molecular Shape for Fragment Growing

A shape-based algorithm for fragment growing needed to fulfill a few criteria. Going for a structure-based approach meant distancing the method from mainstream methods, such as the aforementioned ROCS. The algorithm had to describe both the shape of fragments as well as the shape of binding sites. These shape descriptions had to be easily comparable so that their shape-complementarity was readily measurable. Furthermore, the intrinsic directionality of a fragment growing was a unique feature of the scenario that could be taken advantage of.

In parallel to our development of what we will call the Ray Volume Matrix (RVM), a group at Allergan and Vitae Pharmaceuticals Liu et al.[75] had a very similar idea to our descriptor. The resulting publication described a larger workflow in which the shape-based descriptor was included as a pre-filter for possible directions to extend a molecule. The descriptor generation procedure is similar to the "first intersection" method we will characterize below. The authors validated their workflow prospectively by generating and testing inhibitors for 11β-Hydroxysteroid Dehydrogenase Type 1. To their credit the authors tested and published 37 hits generated by their workflow. Although the methods have a lot of similarities, our descriptor includes several methodological improvements and a more detailed characterization of the descriptor itself, isolated from a workflow.

4.1. The Ray Volume Matrix

The first step toward a shape-based workflow is describing the shape of the components involved. For a fragment growing scenario, the components are the following: a binding site, i.e. a subset of a proteins amino acids within a certain distance to a ligand, a ligand in that binding site, an exit vector that defines what part of the ligand fragment will be attached to, and a collection of fragments that could be attached to a ligand with a defined attachment point themselves.

4.1.1. Descriptor Generation

Figure 4.1 gives an overview of descriptor generation for a neuraminidase structure. The exit vector, which was usually a cut bond, defined the directionality of a growing. This direction was the central axis of our descriptor, the Ray Volume Matrix (RVM). The exit vector could be extended as far into the pocket as necessary to describe its depth. Usually, this depth was set to a fixed 10 Å. The next step was to sample the pocket shape at intervals along the depth axis. At every such interval, a defined number of rays at regular angle intervals to each other were shot out into the pocket perpendicular to the central axis. These rays intersected with the pocket atoms and thereby described the pocket shape. The fixed depth of the descriptor, the number of rays generated along the central axis, and the bin size of the width ranges were all parameters fitted to the self-growing set introduced in Section 3.2.

A similar procedure was performed for fragments. The exit vector was replaced with the bond to the attachment point of the fragment, the depth was usually the same fixed size as the pocket, and the rays sample the fragment atoms. One difference between pocket descriptor generation and fragment descriptor generation was that fragments needed 3D conformations before a descriptor could be generated for them. Conformer generation for fragments was limited to sampling within the cylinder of space defined by the descriptor. The algorithm was adapted from the Conformator[41]. In parameterization, it also became clear that the maximum number of conformations for each fragment could be set to lower than one would conventionally choose for full-sized ligands[D1]. A maximum of 10 conformations were generated per fragment throughout this project.

Two different methods of using ray and atom intersection information were explored. One method used the first intersection point of a ray and a pocket atom to decide that after this point no more space was available in the direction of the ray. The corresponding intersection point for a fragment was the last intersection point of a ray and the fragment atoms. The other method used all intersections of a ray and pocket atoms up to a fixed distance from the central axis to calculate ranges in which the ray was not intersecting with a pocket atom. The second method, which was termed "width ranges", could detect side pockets that were hidden by pocket atoms which the first method, called "first intersection", could not. The two methods were also different in how they saved intersection information. First intersection saved only the perpendicular distance of the first intersection point to the central axis. Width ranges were saved as long bitstrings,

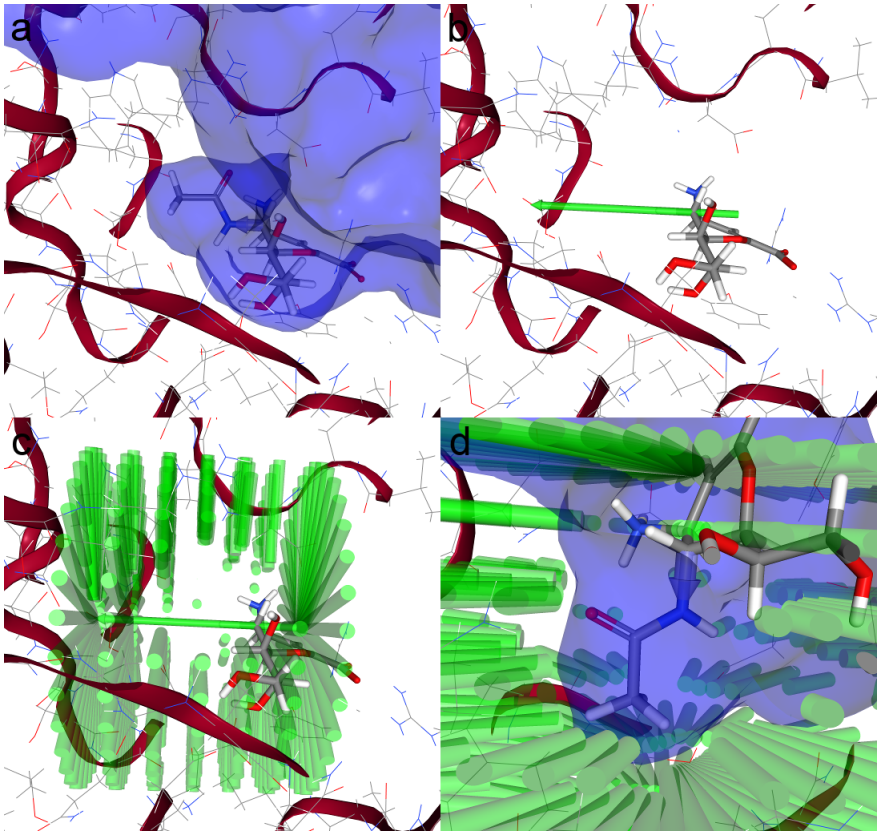


Figure 4.1.: Descriptor generation for a part of a neuraminidase inhibitor in its binding pocket (2qwd). The full inhibitor is shown in (a). The acetylamide fragment that the blue arrow is pointing at is cut off and a query is generated for the remaining part of the inhibitor. The green arrow in (b) represents the descriptor depth. The green cylinders shown in (c) start at the first intersection point between a sampling ray and a pocket atom. The points of first intersection are complementary to a protein surface created by NGL Viewer[84] as shown in (d). Reprinted with permission from [D1]. Copyright 2020 American Chemical Society.

which discretized intersections to bins that corresponded to bits and set these bits to one or zero if they were intersecting an atom or not.

4.1.2. Descriptor Comparison

The descriptors for pockets and fragments using the two different methods also had to be compared in different ways. First intersection descriptors would go along corresponding rays in the pocket and the fragment to see if the last intersection point of a fragment was further away than the first intersection point of a pocket. This would imply that the fragment is trying to fill space in the pocket that is not available and would be counted as clash.

Width range descriptors were compared by AND-ing the bitstring that was the pocket descriptor and the bitstring that was the fragment descriptor together. Bits at the same position in both descriptors corresponded to the same point in space. If both of these bits were set to one, this meant both the pocket and the fragment filled this space. This would lead to clash if the fragment was placed in the pocket. All bits that were set in both descriptors were counted as instances of clash.

Measuring clash is the simplest way that the RVM was used to determine whether a fragment fits into a pocket. Two other parameters could also be easily calculated using the RVM, close contacts, and filled volume. Together clash, close contacts, and filled volume were used to create a pose scoring function. Thus the descriptor was not just a clash filter but could also be used to prioritize some fragment poses over others. The pose scoring function was a weighted sum of either the range sizes (i.e. 1.5 Å of clash along one ray) or the bin counts of the descriptors. The weights of these sums were determined by optimizing the pose scoring function for the self-growing set. The subsequent validation by the self-growing set was therefore influenced by the fitting procedure.

4.2. Validation

Validation of the method was performed on the self-growing and cross-growing data sets. In preparation for running fragment growings, the unique fragments of each data set were made into a screening database. This began with generating up to 10 conformations for each unique fragment. A descriptor was then calculated for every single conformation and stored in a screening database with the conformations themselves. Generation of

the descriptors for the pockets was performed in the course of the fragment growing and included in the runtime. These pocket descriptors were then rotated to simulate rotation around the exit vector of a fragment and used to query the screening database. The runtime was measured up to the end of the comparison of all rotated pocket descriptors with all the fragment descriptors in the screening database. The actual attachment of the fragments was not included in the runtime analysis, which in this case was focused on the descriptor comparison itself.

4.2.1. Self-growing

Self-growing was introduced in Section 3.2 and involved replacing previously cut-off fragments back into their native binding site. Most of the fragments could be replaced into their own binding site with less than 2 Å of clash by all methods. First intersection lagged behind two different parameterizations of width ranges by only being able to replace 96% of fragments into their own binding site. This suggested that maybe first intersection was not able to find all necessary sub pockets to place the fragments. The differences between the pose quality, the percentage of pose with an RMSD of less than 2 Å, of the two methods in self-growing were minor.

4.2.2. Cross-growing

The trend of first intersection not being able to replace all fragments continued in the cross-growing validation. Whereas width ranges were able to generate poses with less than 2 Å of clash in 95-99% of the test cases, first intersection only managed 90%. The differences in pose quality were once again minor.

4.2.3. Runtime

Both variants of the RVM were very fast. First intersection screened around 70 000 conformations per second and the faster width ranges variant screened around 30 000 conformations per second. Although it was slower than first intersection, our further work focused on the faster width ranges variant because it was able to find occluded subpockets and its speed would not be the bottleneck of a fragment growing workflow. Width ranges were therefore a good compromise between speed and performance.

4.3. Discretization Considerations

A few other effects were discovered in the course of analyzing the RVM. The RVM can be seen as a cylindrical grid. Its cylindrical nature makes it easy to rotate, which is used in the screening process, but it also leads to a few unique discretization considerations beyond those of cubic grids.

Any kind of discretization to a grid necessarily involves rounding to the nearest grid point and potentially incurring rounding errors proportional to the granularity of the discretization. The severity of these rounding errors was investigated in a series of robustness experiments that involved translating and rotating the ligand that was grown from inside its binding site. The pose quality seemed to drop significantly faster than the percentage of fragments that could still be placed in the binding site, which suggested that the RVM-based growing could compensate for some amount of transformation. Significant drops in performance could be seen at 1.5Å of translation and at any amount of rotation.

There is also a difference in how extensively the RVM can sample clash with respect to the clashes orientation in spaces. Clash oriented perpendicular to the growing direction, i.e. in the direction of the sampling rays, can be perceived as larger by the RVM than clash oriented in the growing direction as shown in Figure 4.2. Another thing to consider is that the absolute distance between two sampling rays increases as the distance from the central axis increases. This should be considered when parameterizing the number of rays to ensure that rays do not miss whole pocket atoms at the outer edges of their sampling.

4.4. Coloring the RVM

Although the RVM was very successful at generating good poses for fragments, several examples could be found in which these poses egregiously disregarded protein-ligand interactions[D1]. One method to integrate interaction awareness directly into the shape descriptor is by "coloring the shape". This is a typical next step for shape-based algorithms[94].

Besides generating the usual RVM descriptors, fragments and pockets would also generate RVMs that contained polarity information instead of shape information. Fragment

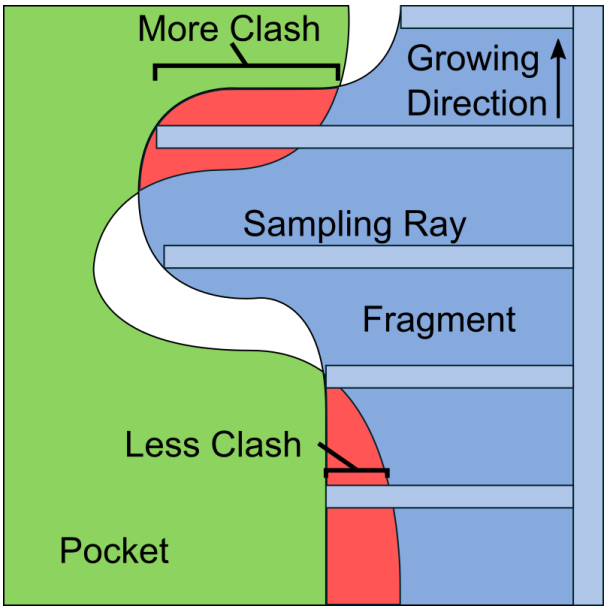


Figure 4.2.: Differences in how clash is sampled by rays depending on whether the clash's principal component is in the growing direction or perpendicular to it. The light blue rectangles represent the sampling rays of the fragment or building-block descriptor. The two areas of clash between the pocket and the fragment or building-block are similar, but the sampling rays perceive the clash very differently. The clash perpendicular to the growing direction is perceived as far more severe than the the clash in growing direction. Reprinted with permission from [D1]. Copyright 2020 American Chemical Society.

and pocket atoms would set polar grid points in their respective descriptors and a cut-off distance between polar grid points could be parameterized to see if polar groups in the fragment could be oriented toward polar groups in the pocket. This term could be integrated into the pose scoring function of the RVM workflow in the same way as the close contacts term but would instead describe polar contacts.

These attempts were discontinued for three reasons. The first reason was that even after parameterization the performance gain was minimal. The parameterization of the pose scoring function resulted in a very noisy optimization surface[D1] and the improvement of pose generation achieved by coloring the shape was well within that noise. The second reason was that to achieve this negligible increase in pose generation performance both the memory requirements and the comparison times were doubled. The close contact term in the pose scoring function of the RVM is the most expensive to calculate because it involves more than a direct comparison of bits with the same index in the bit strings of the width ranges. The polarity term was calculated in the same way as it required the same kind of distance-based implementation, in this case a distance between polar groups. This led to an almost doubling of the comparison time. The doubling of the memory requirements came from having to calculate a second bitstring that contained polarity information instead of shape information. The third reason was intuitive usability. A cylindrical grid is a difficult geometric entity to intuitively reason about. Although areas of complementary polarity are an intuitive concept, depending on where they are in a descriptor similarly sized complementary areas on a bitwise level would be weighted differently due to the discretization considerations above. For these reasons alternative methods of integrating interactions into the workflow were pursued in the following.

4.5. Outlook

An aspect of shape screening not pursued further in this work was the application of RVM-based workflows as prefilters for more complex methods. Clash detection in RVM descriptor comparison is the fastest part of an already very fast process and could save the time it takes a more complex method to perform a similar clash analysis. To validate RVM comparison for this purpose, one would simply have to prove a correlation to a more complex method's clash detection.

5. FastGrow

A fully-fledged molecular modeling tool needs to be made up of more than just a shape search for the purposes of computational FBDD. The inability of a shape-based algorithm such as the RVM to perceive interactions is particularly problematic in the context of FBDD. Section 1.1.1 and Section 1.5 discuss a number of other computational FBDD tools and their validation. These methods tend to have one central algorithm and build upon it to pursue certain strategies or to address common needs[23], [72], [95]. For these reasons we built a workflow around the RVM shape search we called FastGrow.

5.1. FastGrow Workflow

Although the RVM pose generation was producing many high-quality poses, the sampling was still bound to the discretization of the RVM. This led to poses that were often slightly outside the optimum. To fix this FastGrow required geometry optimization. FBDD is a very protein-ligand interaction focused paradigm, which is reflected in FastGrow through a system of pharmacophoric constraints. In cross-growing validation, we noticed that some test cases could not be completed successfully because of changes in protein conformation between the first and second structure. We needed some form of flexibility description to address this. However in adding all these features we still wanted the software to remain interactive and be conducive to iterative approaches, which meant tightly controlling runtime.

5.1.1. Shape-based Search

Having analyzed the performance of RVM shape searches, we chose the following setup. We decided to pursue the "width ranges" version of the RVM as a compromise between speed and performance. Discretization and parameterization of the RVM descriptors were taken from the "coarse width ranges" configuration of the RVM publication[D1].

We also continued to generate a maximum of 10 conformations per fragment, which had proven to work well in parameterization and validation.

5.1.2. Geometry Optimization

The role of geometry optimization was filled by JAMDA[66]. JAMDA is a docking scoring function that follows in the footsteps of other well-known scoring functions, such as ChemScore[96] or PLANTS[97]. Its novelty lies in its particularly stable limited step length optimization algorithm[66]. JAMDA scoring was slightly modified for use with FastGrow. Because JAMDA was used to optimize poses after fragment growing, we added penalty terms to the movement of input atoms. Atoms that are part of the input core or ligand are usually assumed to already be in the correct position and moving them would go against this assumption. It has been shown that this assumption holds true for the majority of fragment growing cases analyzed by Malhotra et al.[79]. The atoms of the attached fragment were optimized without restraint.

5.1.3. Interactions

We implemented a simple set of pharmacophoric constraints to model interactions. Pharmacophoric constraints were made up of a point, a tolerance radius, and a type. The three most prominent types were "Hydrogen Bond Donor", "Hydrogen Bond Acceptor", and "Hydrophobic". Pharmacophoric constraints were usually generated from interactions, such as those in the interactions validation set.

When screening with pharmacophoric constraints, every fragment considered as a hit had to at first have at least enough pharmacophore features to match the number and type of the constraints. In the next step, the position of these pharmacophore features was compared to the pharmacophoric constraints in space to see whether they fell within the tolerance radius.

5.1.4. Ensemble Flexibility

Having seen cases in the cross-growing set that required a flexible description of the binding site, we implemented an ensemble flexibility approach. In ensemble flexibility, the flexibility of a binding site is described by a collection of conformers of that binding

site. These can be from a variety of sources such as molecular dynamics trajectories[98]. In most of our application, these were derived from crystal structure information extracted by SIENA[89]. These structural ensembles could be passed to FastGrow and were screened at the same time. Separate scores were calculated for every member of the ensemble and the best score was taken as the representative one. The best score was chosen to reflect the use case ensemble that flexibility was supposed to handle in FastGrow, which is a case where one particular change in conformation strongly influenced a growing. This highly influential conformational change should conceptually lead to a very different score. An alternative would have been to combine the scores in some way, possibly as a weighted sum. This would have weakened the signal of a large conformational change and was therefore not pursued.

5.2. Validation

Validation of FastGrow began by establishing a base line. The self-growing and cross-growing validation sets were updated for the FastGrow publication[D3]. The feature-specific validation sets (interactions, water replacement, and ensemble growing) were first generated for the FastGrow publication[D3]. The changes in performance caused by the update of the data set were generally minor and within the 95% confidence intervals that could be calculated for runs before and after the data set update. This baseline was used as a comparison in feature specific validation and later in the comparison of FastGrow to docking.

5.2.1. Interaction Modeling Validation

Interaction modeling was validated using the interaction data set of the FGVD. This data set included stable interactions for 252 cross-growing test cases that could be used as pharmacophoric constraints. Using such pharmacophoric constraints to screen generally improved the performance of pose generation[D3]. This result was expected because some information from the reference structure must be used to analyze the stability of interactions across structures as described in Section 3.4.1. This information will implicitly bias such an analysis.

The more interesting analysis in this context was therefore whether the modeled pharmacophoric constraints led to the interactions being maintained in the output poses. Of

the three prominent pharmacophore types "Hydrogen Bond Acceptor", "Hydrogen Bond Donor", and "Hydrophobic", recreating hydrophobic interactions was the easiest task for the shape-search without pharmacophoric constraints. The hydrophobic types as defined by ChemScore[96], which is also the definition used by JAMDA[66], seemed to correlate most with the shape-complementarity of RVM comparisons. Hydrogen bonds were not maintained well with just an interaction unaware shape-search, which we had observed qualitatively in the RVM publication[D1] but had now measured quantitatively. Hydrogen bonds could be maintained far better with pharmacophoric constraints. Especially hydrogen bond donors' geometries improved with geometry optimization after using pharmacophoric constraints. It is largely here that the geometry optimization proves measurably beneficial[D3].

5.2.2. Water Replacement

The scenario of replacing water in a fragment growing came up in conversation with users of FastGrow. This could be achieved with the pharmacophoric constraints implementation as it was. By placing pharmacophoric constraints at the position of a water molecule, a fragment could be coerced into replacing the water with the specified interaction type. This was validated with the water replacement validation set. Although comparing the pose generation quality of the water replacement cases with their corresponding test cases in a purely shape-based growing did not show a very significant improvement, there were clear improvements in single cases, suggesting that water replacement is a useful but situational feature[D3]. The size of the water replacement validation, which is less than half the size of the cross-growing set, may have also contributed to problems in detecting a statistically significant improvement.

5.2.3. Ensemble Flexibility

Ensemble flexibility was validated using the ensemble growing validation set. Pose generation quality did not improve significantly as a result of screening against more structures. In absolute numbers, there were more ensemble growing test cases that improved than performed worse. It is a known phenomenon that using an ensemble to perform molecular docking can reduce the performance of the docking. The additional structural information and the tolerance it provides may lead to false positive results[98]. Nonetheless, a few test cases did noticeably improve using ensemble growing and some

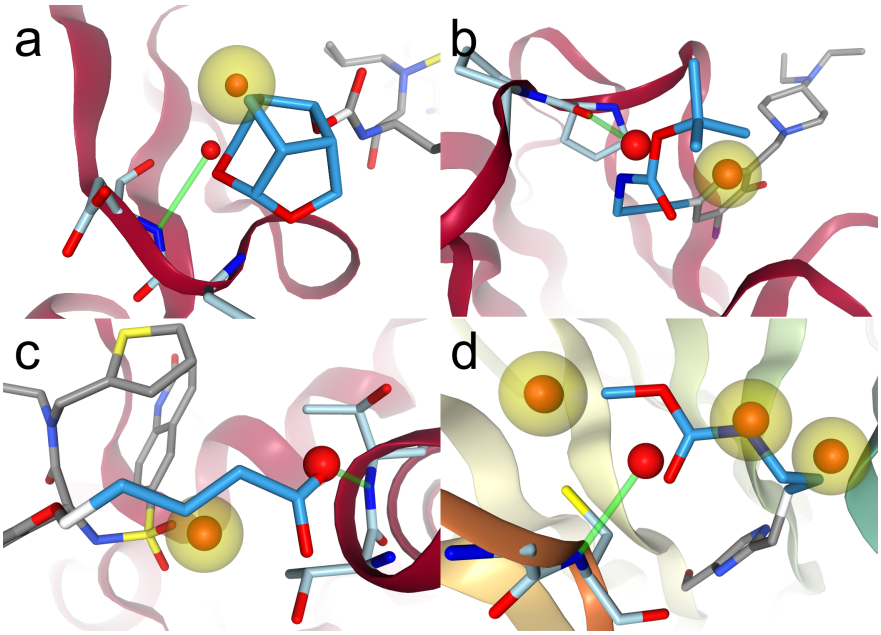


Figure 5.1.: Water replacement test cases that were extracted from the cross-growing set. The replaced waters are shown with the interactions that were used in the query to replace them. Yellow spheres are hydrophobic interactions and green cylinders hydrogen bond interactions. Binding site residues are light blue. The ligands are non-native to the binding site. The darker blue part of a ligand will be grown in a water replacement test case. (a) the ligand of 5ULT in 3GI6 (Gag-Pol polyprotein). (b) the ligand of 4AGO in 4AGM (Cellular tumor antigen p53). (c) the ligand of 6MA4 in 6MA5 (O-GlcNAc transferase). (d) the ligand of 6PGA in 6PG4 (WD repeat-containing protein 5). All 3D molecule images were made with the NGL viewer[84]. Material from: [D3].

of these could be demonstrated to be due to protein flexibility previously described in literature[D3]. This feature, too, seemed to be useful in particular situations rather than in the general case.

5.3. Comparison to Docking

To establish FastGrow in the larger context of structure-based tools available for fragment growing, it was compared to a well-known molecular docking system. We chose DOCK[99], an open source and high-profile[100] docking engine that had been used in similar validation scenarios[99][101]. DOCK was used in two configurations: Full flexible docking, which will be referred to as cross-docking or re-docking depending on the task, and anchored docking, which takes an input anchor that stays mostly fixed during the docking process. Full flexible docking is often done when performing fragment growing in practice, although often unnecessary and prone to error. Anchored docking in contrast will ideally only sample the atoms of the fragment instead of the whole molecule and is therefore a direct analogy to FastGrow but using a docking engine.

A statistical comparison of DOCK and FastGrow was performed on the cross-growing set. DOCK was run in both cross-docking and anchored docking configurations. FastGrow was run with and without geometry optimization. Cross-docking with DOCK was outperformed by every other method. This was an expected result because all other configurations used more information, but it does highlight that using all information available can significantly improve pose generation performance. FastGrow also outperformed anchored docking in pose generation performance. This was a somewhat unexpected result. In the RVM publication[D1] we made the prediction that the more granular sampling a docking engine performs would lead to higher performance. The difference between the system may come from a different perception of clash. RVM shape search has a high clash tolerance in comparison to a typical docking engine. DOCK is usually validated on minimized structures, i.e. a validation scenario with less clash. The cross-growing set needs a certain amount of clash tolerance to complete because the validation task consists of growing fragments into non-native and not minimized binding sites.

5.3.1. Runtime

FastGrow significantly outperforms docking in runtime. Cross-docking can perform approximately one cross-docking/fragment growing per minute, which is in line with typical docking runtimes. Anchored docking outperforms that by being able to do around 5 fragment growings per minute, due to the fewer degrees of freedom a fixed anchor affords. In both configurations, with or without geometry optimization, FastGrow outperforms this by several orders of magnitude. Geometry optimization is quite fast at around 100 ms a fragment but is a comparatively high runtime investment. The evaluation and optimization of a scoring function overshadows all other runtime considerations in FastGrow.

	Runtime [h]	Fragments	Runtime per Fragment [ms]
Growing	0:31	425 x 176	24
Optimized Growing	3:17	425 x 176	158
DOCK anchored growing	1:37	425	13 640
DOCK cross-docking	8:20	425	70 464

Table 5.1.: Runtime for FastGrow and DOCK on the cross-growing set. FastGrow always screens the full database of 176 unique fragments. For runtime reasons, DOCK is only given the correct fragment to dock.

5.4. DYRK1A Case Study

DOCK and FastGrow were also compared in a case study. Some of the differences seen in the statistical comparison could be demonstrated in an applied example. The case study system was Dual Specificity Tyrosine-phosphorylation-regulated Kinase 1A (DYRK1A), a kinase involved in Down's Syndrome, neurodegenerative disease, and cancer[102]–[104]. This was facilitated by a pair of recent publications that disclosed novel chemotypes binding to DYRK1A[105], [106]. The first part of the case study involved growing a micromolar fragment into a nanomolar ligand. The second part of the case study analysed how well FastGrow and DOCK could pick out active fragments from a collection of generic fragments in an enrichment validation.

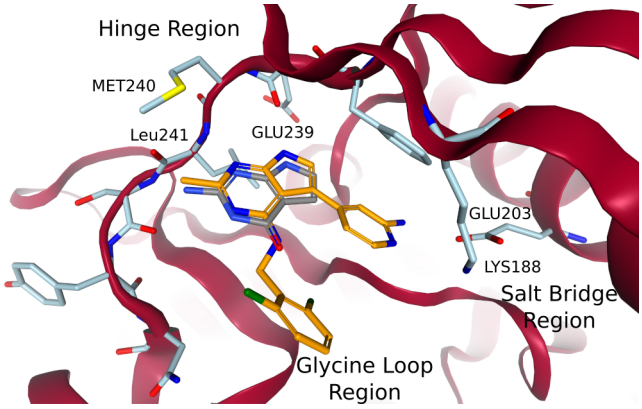


Figure 5.2.: Fragment 1 and compound 34 in a structure of DYRK1A. The structure and fragment 1 coordinates in gray are from PDB code: 7A4R and compound 34 in orange comes from PDB code: 7A5N and is aligned to the binding site using SIENA[89]. The cartoon for the residues GLU160 to ASP178 was removed for clarity as it is in Walmsley et al.[106]. All 3D molecule images were made with the NGL viewer[84]. Material from: [D3].

5.4.1. Growing a Ligand from a Fragment

Walmsley et al. started with a fragment screening and found fragment 1 as a micromolar hit. Subsequent optimization led to compound 34, a nanomolar ligand with favorable properties[106]. Both compounds can be seen in Figure 5.2. Fragment 1 and compound 34 both share a common core and three modifications to fragment 1 are necessary to turn it into compound 34. These 3 modifications are exchanging an amine with a methyl group, growing into the salt-bridge region of the kinase binding site, and growing toward the glycine loop. All three modifications were performed with DOCK and FastGrow.

None of the methods had particular problems exchanging an amine with methyl, but cross-docking with DOCK incurred an unnecessary amount of coordinate drift, which strained some of the interactions between the common core and the hinge region of the kinase. This could be avoided with anchored docking and using the common core as an anchor or by passing the common core as input for FastGrow. Anchored docking encountered an issue when growing into the salt-bridge region and the generated pose was optimized out of the pocket. Cross-docking did manage to generate an acceptable pose, suggesting anchored docking encountered some instability. The JAMDA scoring function integrated into FastGrow and its limited step length optimization is specifically designed to avoid such instability issues[66]. DOCK and FastGrow both flipped an amine at a ring away from an interaction it usually made in most crystal structures. External

information from these other crystal structures could be integrated into the growing to ensure this interaction was made. All methods agreed on the growing toward the glycine loop.

The growing case study could by example demonstrate three aspects of structure-based fragment-growing. First of all, naive docking may perform unnecessarily poorly even in trivial growing. We could show statistically that one should make assumptions about the binding pose when possible and could demonstrate it once again in this case study. Secondly, we could demonstrate the advantage of limited step length optimization as implemented by JAMDA[66] compared to DOCK, which ran into a stability issue and optimized a ligand out of a pocket despite attempts at anchoring the docking. Lastly, we could show how integrating external information can improve a grown pose in practice.

5.4.2. Enrichment

A number of ligands published by Weber et al.[105] and Walmsley et al.[106] had a diaminopyridine moiety in common. A set of highly active fragments could be generated by cutting the variable parts off the diaminopyridine in ligands with low nanomolar activity. These fragments could be used in an enrichment scenario where, after generating a pose, a fragment growing method had to score the fragments such that it highly ranked the highly active fragments within a set of generic fragments. The generic fragments came from a set of fragments generated by the BioSolveIT[107] and were property matched to the highly active fragments[D3]. DOCK and FastGrow were both used to generate poses and score them. The ranking success was determined by the area under the curve of a receiver operating characteristic. 95% confidence intervals were calculated by bootstrap re-sampling similar to Stein et al.[73].

DOCK and FastGrow both performed very well on this data set[D3]. To investigate this we also performed the enrichment screening with just the RVM pose scoring function, which was parameterized for a different purpose. This too performed very well even if all other terms except for clash were set to zero. It seemed the clash term that all scoring functions had in common was very discriminative in this case study. There was, however, still a very clear difference in the runtime of the methods[D3]. Although anchored docking was significantly faster on the comparatively less complex growing steps in the enrichment screening, it was still outperformed by FastGrow with geometry optimization by a factor of 5. The RVM pose scoring function performed comparably in the ranking, but it outperformed both methods by several orders of magnitude in speed.

This highlights the question of how complex a scoring function should be to be useful for computational FBDD.

5.5. Outlook

Integrating the features above into one tool makes FastGrow a fully-fledged modeling tool capable of sophisticated workflows such as water replacement. What the preceding is missing is a prospective application that could not be disclosed in the context of the project. Prospective application not only highlights usability issues but also implicit biases in the retrospective validation. It would be invaluable to apply and evaluate FastGrow in prospective drug design.

An ever-present question in FBDD is what fragments to work with. In an experimental context, fragments are usually constrained by physico-chemical properties. Computational FBDD approaches are not initially constrained in the same way. How one designs fragment sets for computational fragment growing is therefore very open. Some design philosophies, for example, try to optimize for shape diversity[14]. Synthetic accessibility is often considered a constraint in de novo design. One way to achieve this is to only allow the combination of fragments according to synthetic rules[78], [108]. These synthetic rules can be explicit[108], [109] or implicitly encoded in a combinatorial fragment space[110]. By whatever method or philosophy, it would be helpful to provide users of FastGrow with a standard set of fragments to try out.

6. Conclusions

A set overarching conclusions can be drawn across all the individual topics of this project that manifest themselves precisely because they were combined. Torsion angle preferences were used both from a low-level perspective of extracting statistics and from a high-level perspective by using conformations derived from these statistics. Shape-based search was not only analyzed in an isolated evaluation of the algorithm but also in meaningful drug design scenarios. Furthermore, the tool FastGrow was applied in such a context as well and experiences were gained.

6.1. Reliability of Torsion Angle Preferences

Torsion angle preferences as produced in the context of the Torsion Library were somewhat opaque at the beginning of this project. The tool landscape was confusing and the exact behavior of the two matching procedures was nebulous, especially in the prior implementation. The Torsion Library is a long-lived system and a lot of its complexity is warranted. Its main point of differentiation from other systems is its hand-curated nature. That, however, is the most labor intensive and complex part. Expert but nonetheless human curation introduces human error that is very difficult to detect automatically. We saw significant progress on that front with the introduction of the SMARTScompare algorithm into Torsion Library workflows. However, errors in SMARTS that cannot be detected by subset relation detection, must still be found manually. In light of this it is especially important to minimize inconsistencies on all other fronts.

The Torsion Library has seen several different tools since its initial publication in Schäerfer et al.[49]. Changes in the tool landscape have happened in between every publication[49], [81], [D2]. Every tool change confuses not just the users of the Torsion Library but also the experts. Care should be taken to keep at least the function of tools consistent if not the implementation. This is one of the reasons we proposed two tools in [D2]: one tool for expert analysis and one tool for torsion angle classification.

Although the underlying workflow is similar, the two concerns are intuitively separate in practice. A user trying to filter a set of compounds should not be overwhelmed with the differences in matching behavior that are necessary to generate and validate the torsion angle preferences.

It also became clear how important it is to keep the whole process of generating torsion angle preferences and classifying torsion angles transparent. This pre-supposes a completely deterministic process where every single data point in the torsion angle preferences is quickly traceable. Furthermore, it is of utmost importance that the implementation associated with the publications exactly mirrors the rule set defined by these publications. This is easy to take for granted but requires strict discipline in implementation and high legibility of the code. If this is not given, then it is very hard to argue for or against the inclusion of various edge cases, such as those that have arisen multiple times in the course of this project.

Because the Torsion Library is such a long-lived system most core applications have been validated extensively either through formal publications or informal experience. These large-scale evaluations of core functionality have the potential to dwarf more focused efforts. Focused efforts need focused validation because specific improvements may be overwhelmed by more generalized validation. This was largely the case in the improvement of the Torsion Library with SMARTScompare. Although the changes introduced in that effort were chemically meaningful, the generalized performance was largely unchanged[D2], especially if this performance was measured by the heuristic sampling of conformer generation. However, the end-user, the structural biologist or medicinal chemist, is very focused on particular subdivisions of torsional and chemical space. In this space, the user is acutely aware of any inconsistency the Torsion Library may produce. Finding and addressing these specific concerns represents a more realistic and impactful way forward for the Torsion Library than expecting large shifts in generalized performance.

6.2. The Pragmatism of Shape-based Fragment Growing

Shape-based approaches are known for their intuitive visualizations, their speed, and their surprisingly high accuracy. Their accuracy derives from the fact that shape-complementarity is to a degree a hard prerequisite for any small-molecule binding process. A ligand may be smaller than a binding site, though it would probably incur

proportional entropic penalties. It cannot be bigger than a binding site, which would incur exponentially higher energetic penalties. The compromise between these properties is a measure of shape-complementarity of the ligand and the binding site, which in itself is a fairly simple geometric criterion.

The problem of shape-based fragment growing is even simpler than a similar problem in ligand-sized molecules, simply because the molecules are smaller and have generally fewer degrees of freedom in most considered dimensions. This can be leveraged to achieve significantly faster and accurate workflows, only if one does not transfer functionality created for ligands directly into a fragment workflow. Our project demonstrated the measurable difference it can make to consider the nature of fragments instead of simply scaling down a workflow for ligands.

Throughout the project, we exploited the nature of fragments in several ways. We only ever generated a maximum of 10 conformations per fragment, which is relatively few compared to other structure-based applications. The lower rotational complexity of fragments makes this viable as our validation demonstrated. The lower complexity of fragment growing also manifested itself in the parameterization of the RVM shape descriptors[D1]. The design of the descriptor itself already exploits the directionality of a growing process and thereby reduces the degrees of freedom.

There is something particularly pragmatic to be found in shape-based fragment growing when one combines the advantages of shape methodology with the properties of fragments. Both elements synergize to reduce runtime through simple geometric operations on molecules with fewer degrees of freedom and maintain the competitive accuracy and speed of shape methods. The primary difficulty in forging a complete computational FBDD workflow was in integrating a scoring function and geometry optimization, which completely overshadowed all other components in runtime and complexity. It is especially this last part that requires further work to continue the theme of pragmatism.

6.3. User Interaction

The interaction of the user with FastGrow has been a major focus of this project since its inception. A measurable aspect of this is the runtime. Being able to screen more fragments in the same time as established approaches is an advantage in itself. However, screening larger fragment sets leads to more than just larger hit lists, because it can

change a user's perspective from finding single hits to observing trends in the screening. These trends would be invisible in slower systems due to the resources necessary to find them.

Observing these trends can lead to a deeper understanding of a system. For example, a preference for particular functional groups can highlight a geometry necessary to address subpockets of the binding site. As a consequence, a user can screen a more focused set of fragments with this geometry. By exploring a system in this way, a user will quickly build up a branching tree of hypotheses to iteratively test. Drug design is an intrinsically iterative process. Tools built for drug design should therefore support rapid iteration.

Another aspect of interacting with a user is integrating their expert input. Using comparatively simple features such as pharmacophoric constraints a user can quickly bring in their expertise to create complex workflows. An alternative to a model of pharmacophoric constraints would have been to "color" the RVM shape descriptor by a variety of properties. There is precedent to extend shape descriptors in this way[94]. After less than promising initial results when applying coloring to the RVM descriptor, we chose to implement pharmacophoric constraints, which are more intuitive to users than coloring regions of a cylindrical grid. Such constraint models are ubiquitous throughout computational drug design and easy to implement. Pharmacophore models are also a very general description of spatial molecular properties that allow a user to bring their expert knowledge into the searching workflow.

Reliability, pragmatism, and an interactive integration into drug design workflows have defined the course of this project. Continual validation and a focus on intuitive simplicity instead of methodological sophistication have largely been the results of these factors. Further work should be informed by these principles.

7. Outlook

7.1. Data Enables Specific Features

A few features validated in the project suffered from data set sizes that were too small to show statistically significant differences. For example, there seemed to be a difference in performance in the water replacement validation, but the amount of statistical error, which is tied to the sample size, called this into question. Water replacement is a very specific scenario and the corresponding validation data set had the fewest number of entries. More data will result in more statistically significant validation and would make more specific features statistically viable. The automated pipeline of data set generation created in this project meant that the FGVD could be updated with new PDBbind data within a day. This pipeline could be adapted to other larger data sets. However the adaption to other data sets should maintain similar data set quality.

7.2. Advanced Growing Operations

Another feature set that could not be pursued due to a lack of validation data was a variety of different advanced growing operations. FastGrow is currently only validated on single bond fragment growing steps. Not enough growing steps could be generated using the PDBbind refined set to make any statistical statements about other bond types. Even the permissive self-growing found only 112 test cases for double bonded growing. Although as a tool FastGrow supports double bonded growing, no general statement can be made about its performance.

Other advanced growing operations that were considered during development were the growing of small rings and macrocyclization. This was largely not pursued due to time constraints. Additionally, generating a validation data set would probably also have presented a challenge. The thing these operations have in common is that they both

present a user interface challenge. Specifying one exit bond may be enough to grow small rings if one implicitly matches certain functional groups as small-ring-forming reaction vectors, but this would be very restricted by reaction modeling and a more general two exit vector growing should be pursued. Macrocyclization would require the definition of two exit vectors and possibly a complete reassessment of fragment properties to include linker-like structures. Furthermore, these two features would require a rewrite of the FastGrow fragment attachment code to include a more general geometry handling and significantly more geometry corrections, due to valence state changes and cyclization.

7.3. Multi-parameter Optimization

Drug design is a multi-parameter optimization problem[111]. Virtual screening is typically concentrated on the fit of the ligand in the binding site, or in this case the fit of the fragment in the binding site. Often users want to consider properties beyond that. A variety of filters could be explored and integrated into the FastGrow workflow. Hard filters could be incorporated earlier in the workflow and may even result in faster runtimes. Softer filters could be integrated later in the workflow and cut large hit lists down to a manageable size.

When users specify soft filters they may have intuitions about which properties their system may be more or less tolerant to. Creating fitness metrics that express user intuition could be an interesting challenge that would transcend fragment growing. The simplest metric would probably be a combination of normalized absolute differences. This could be extended by allowing fixed ranges of tolerance. A quite complex approach could be to transform absolute differences with a variety of function kernels such as a Gaussian function which would lead to a Gaussian smoothed distance that could be parameterized to a user's intuition. This is both a modeling and a user interface challenge because communicating what differences parameters, kernels, and tolerances make would be essential to the interplay between the model and the user's intuition.

7.4. Automated Metaheuristic Driven Growing

Many fragment growing workflows, and more generally de novo drug design workflows, use established structure-based drug design workflows as their core and focus on guiding a fragment sampling process by various metrics and algorithms[22]. Evolutionary

algorithms are the prototypical example of this[112]. Metaheuristic sampling is a vast field of its own that is constantly growing and may still hold methods worth exploring for fragment growing.

An interesting variation in sampling would be trying to sample fragments for a particular target independently of another. Independently sampled fragments could be combined and prioritized to find particularly optimal products in combinatorial space without having to assess all of its members. This is significantly more difficult in a structure-based context because one has to partition the target space to sample fragments independently. Nonetheless, these efforts could mirror paradigm shifts achieved in combinatorial similarity searches[113] and are therefore worth investing into.

7.5. Fragment Specific Scoring and Affinity Validation

Affinity in FBDD has a somewhat different position than in other forms of drug design. Although it is still the main optimization criterion, it is often considered relative to other parameters such as heavy atom count[114]. This is because the expectation in FBDD is that a fragment can be optimized into a significantly more active ligand even from a comparatively inactive starting point. This leads to differences in how FBDD is performed on an experimental level and should lead to differences in how fragments are treated computationally as well.

After several decades of virtual screening, computational scoring approaches are geared toward full-sized ligands and differentiating between decoys and actives in enrichment focused screening[73]. The perturbation caused by exchanging or growing a single fragment of a molecule may be outside of the focus of scoring functions validated for full-sized ligands.

This, however, can only be measured with a properly curated data set of affinity annotated fragment modifications. Other projects have shown how difficult such an endeavor can be[73], [92], [93]. Without such a data set no general statistical statements about the performance of the scoring function for computational FBDD can be made. Even with a retrospective data set in hand, validating the extrapolatory power of scoring functions must happen in prospective validation.

The paradigm of Fragment-based Drug Design has done much to shape the way we practice and think about drug design. Drug design paradigms are driven by and drive

technology, which leads to constant iteration and improvement. It is by approaching a paradigm in some depth that methods can profit from this cycle. Isolating unique aspects of a paradigm and placing them into a methodological context results in a synergy unparalleled by unfocused generalization. It is in focused efforts that iteration leads to incremental improvement.

Glossary

CADD Computer-Aided Drug Design.

CSD Cambridge Structural Database.

DYRK1A Dual Specificity Tyrosine-phosphorylation-regulated Kinase 1A.

FBDD Fragment-based Drug Design.

FGVD Fragment Growing Validation Data set.

GPU graphics processing unit.

GUI graphical user interface.

HTS High-throughput screening.

ITC Isothermal Titration Calorimetry.

MCS maximum common substructure.

MVC Model-View-Controller.

NMR nuclear magnetic resonance.

PDB Protein Data Bank.

QC quantum chemical.

RMSD root-mean-square deviation.

ROCS Rapid Overlay of Chemical Structures.

RVM Ray Volume Matrix.

Bibliography

Bibliography of External Sources

[1] R. P. Hertzberg and A. J. Popeu, "High-throughput screening: New technology for the 21st century," *Current Opinion in Chemical Biology*, vol. 4, no. 4, pp. 445–451, 2000, ISSN: 1367-5931. DOI: https://doi.org/10.1016/S1367-5931(00)00110-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367593100001101.

[2] B. Berger, J. Peng, and M. Singh, "Computational solutions for omics data," *Nature Reviews Genetics*, vol. 14, no. 5, pp. 333–346, 2013, ISSN: 1471-0064. DOI: 10.1038/nrg3433. [Online]. Available: https://doi.org/10.1038/nrg3433.

[3] S. G. Krimmer and G. Klebe, "Thermodynamics of protein-ligand interactions as a reference for computational analysis: How to assess accuracy, reliability and relevance of experimental data," *Journal of Computer-Aided Molecular Design*, vol. 29, no. 9, pp. 867–883, 2015, ISSN: 1573-4951. DOI: 10.1007/s10822-015-9867-y. [Online]. Available: https://doi.org/10.1007/s10822-015-9867-y.

[4] G. Klebe, "Applying thermodynamic profiling in lead finding and optimization," *Nature Reviews Drug Discovery*, vol. 14, no. 2, pp. 95–110, 2015, ISSN: 1474-1784. DOI: 10.1038/nrd4486. [Online]. Available: https://doi.org/10.1038/nrd4486.

[5] S. Geschwindner, J. Ulander, and P. Johansson, "Ligand binding thermodynamics in drug discovery: Still a hot tip?" *J. Med. Chem.*, vol. 58, no. 16, pp. 6321–6335, Aug. 2015, ISSN: 0022-2623. DOI: 10.1021/jm501511f. [Online]. Available: https://doi.org/10.1021/jm501511f.

[6] A. Sandner, K. Ngo, C. P. Sager, F. Scheer, M. Daude, W. E. Diederich, A. Heine, and G. Klebe, "Which properties allow ligands to open and bind to the transient binding pocket of human aldose reductase?" *Biomolecules*, vol. 11, no. 12, 2021, ISSN: 2218-273X. DOI: 10.3390/biom11121837. [Online]. Available: https://www.mdpi.com/2218-273X/11/12/1837.

[7] S. B. Shuker, P. J. Hajduk, R. P. Meadows, and S. W. Fesik, "Discovering high-affinity ligands for proteins: Sar by nmr," *Science*, vol. 274, no. 5292, pp. 1531–1534, 1996. DOI: 10.1126/science.274.5292.1531. eprint: https://www.science.org/doi/pdf/10.1126/science.274.5292.1531. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.274.5292.1531.

[8] D. A. Erlanson, R. S. McDowell, and T. O'Brien, "Fragment-based drug discovery," *Journal of Medicinal Chemistry*, 2004, ISSN: 0022-2623. DOI: 10.1021/jm040031v.

[9] T. L. Blundell and S. Patel, "High-throughput x-ray crystallography for drug discovery," *Current Opinion in Pharmacology*, vol. 4, no. 5, pp. 490–496, 2004, ISSN: 1471-4892. DOI: https://doi.org/10.1016/j.coph.2004.04.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1471489204001225.

[10] M. Congreve, R. Carr, C. Murray, and H. Jhoti, "A 'rule of three' for fragment-based lead discovery?" *Drug Discov Today*, vol. 8, pp. 876–877, 19 2003, ISSN: 1359-6446. DOI: 10.1016/S1359-6446(03)02831-9.

[11] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings," *Adv Drug Deliver Rev*, vol. 46, pp. 3–26, 1-3 2001, ISSN: 0169-409X. DOI: 10.1016/S0169-409X(00)00129-0.

[12] A. R. Leach, M. M. Hann, J. N. Burrows, and E. J. Griffen, "Fragment screening: An introduction.," eng, *Molecular BioSystems*, vol. 2, pp. 430–46, 9 Sep. 2006. DOI: 10.1039/b610069b. [Online]. Available: https://doi.org/10.1039/b610069b.

[13] A. Vulpetti and C. Dalvit, "Design and generation of highly diverse fluorinated fragment libraries and their efficient screening with improved 19f nmr methodology," *ChemMedChem*, vol. 8, no. 12, pp. 2057–2069, 2013. DOI: https://doi.org/10.1002/cmdc.201300351. eprint: https://chemistry-europe.onlinelibrary.wiley.com/doi/pdf/10.1002/cmdc.201300351. [Online].

Available: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/cmdc.201300351.

[14] R. Zhang, P. J. McIntyre, P. M. Collins, D. J. Foley, C. Arter, F. von Delft, R. Bayliss, S. Warriner, and A. Nelson, "Construction of a shape-diverse fragment set: Design, synthesis and screen against aurora-a kinase," *Chemistry – A European Journal*, vol. 25, pp. 6831–6839, 27 2019. DOI: https://doi.org/10.1002/chem.201900815. [Online]. Available: https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/chem.201900815.

[15] I. J. P. de Esch, D. A. Erlanson, W. Jahnke, C. N. Johnson, and L. Walsh, "Fragment-to-lead medicinal chemistry publications in 2020," *J. Med. Chem.*, Dec. 2021, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.1c01803. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.1c01803.

[16] J. Barker, S. Courtney, T. Hesterkamp, D. Ullmann, and M. Whittaker, "Fragment screening by biochemical assay," *Expert Opinion on Drug Discovery*, vol. 1, no. 3, pp. 225–236, Aug. 2006, ISSN: 1746-0441. DOI: 10.1517/17460441.1.3.225. [Online]. Available: https://doi.org/10.1517/17460441.1.3.225.

[17] C. Dalvit, "Nmr methods in fragment screening: Theory and a comparison with other biophysical techniques," *Drug Discovery Today*, vol. 14, no. 21, pp. 1051–1057, 2009, ISSN: 1359-6446. DOI: 10.1016/j.drudis.2009.07.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359644609002797.

[18] P. Kirsch, A. M. Hartman, A. K. H. Hirsch, and M. Empting, "Concepts and core principles of fragment-based drug design.," eng, *Molecules*, vol. 24, 23 Nov. 2019. DOI: 10.3390/molecules24234309. [Online]. Available: https://doi.org/10.3390/molecules24234309.

[19] D. Joseph-McCarthy, A. J. Campbell, G. Kern, and D. Moustakas, "Fragment-based lead discovery and design," *J. Chem. Inf. Model.*, vol. 54, no. 3, pp. 693–704, 3 Feb. 2014, ISSN: 1520-5142. DOI: 10.1021/ci400731w.

[20] B. Villemagne, M. Flipo, N. Blondiaux, C. Crauste, S. Malaquin, F. Leroux, C. Piveteau, V. Villeret, P. Brodin, B. O. Villoutreix, O. Sperandio, S. H. Soror, A. Wohlkönig, R. Wintjens, B. Deprez, A. R. Baulard, and N. Willand, "Ligand efficiency driven design of new inhibitors of mycobacterium tuberculosis transcriptional repressor ethr using fragment growing, merging, and linking approaches," *Journal of Medicinal Chemistry*, vol. 57, pp. 4876–4888, 11 2014,

PMID: 24818704. DOI: 10.1021/jm500422b. [Online]. Available: https://doi.org/10.1021/jm500422b.

[21] M. Rachman, S. Piticchio, M. Majewski, and X. Barril, "Fragment-to-lead tailored in silico design.," eng, *Drug Discov. Today Technol.*, vol. 40, pp. 44–57, Dec. 2021. DOI: 10.1016/j.ddtec.2021.08.005. [Online]. Available: https://doi.org/10.1016/j.ddtec.2021.08.005.

[22] J. O. Spiegel and J. D. Durrant, "Autogrow4: An open-source genetic algorithm for de novo drug design and lead optimization," *Journal of Cheminformatics*, vol. 12, no. 1, p. 25, 2020, ISSN: 1758-2946. DOI: 10.1186/s13321-020-00429-4. [Online]. Available: https://doi.org/10.1186/s13321-020-00429-4.

[23] N. Chéron, N. Jasty, and E. I. Shakhnovich, "Opengrowth: An automated and rational algorithm for finding new protein ligands," *Journal of Medicinal Chemistry*, vol. 59, no. 9, pp. 4171–4188, 9 Sep. 2015, ISSN: 1520-4804. DOI: 10.1021/acs.jmedchem.5b00886.

[24] M. Hartenfeller, H. Zettl, M. Walter, M. Rupp, F. Reisen, E. Proschak, S. Weggen, H. Stark, and G. Schneider, "Dogs: Reaction-driven de novo design of bioactive compounds," *PLOS Computational Biology*, vol. 8, pp. 1–12, 2 2012. DOI: 10.1371/journal.pcbi.1002380. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1002380.

[25] Y. Bian and X.-Q. Xie, "Computational fragment-based drug design: Current trends, strategies, and applications," *The AAPS Journal*, vol. 20, p. 59, 3 2018, ISSN: 1550-7416. DOI: 10.1208/s12248-018-0216-7. [Online]. Available: https://dx.doi.org/10.1208%2Fs12248-018-0216-7.

[26] G. J. Kleywegt, "Validation of protein crystal structures," *Acta Crystallographica Section D*, vol. 56, no. 3, pp. 249–265, Mar. 2000. DOI: 10.1107/S0907444999016364. [Online]. Available: https://doi.org/10.1107/S0907444999016364.

[27] A. Meyder, E. Nittinger, G. Lange, R. Klein, and M. Rarey, "Estimating electron density support for individual atoms and molecular fragments in x-ray structures," *Journal of Chemical Information and Modeling*, vol. 57, no. 10, pp. 2437–2447, 10 Oct. 2017, ISSN: 1520-5142. DOI: 10.1021/acs.jcim.7b00391.

[28] N.-O. Friedrich, A. Meyder, C. de Bruyn Kops, K. Sommer, F. Flachsenberg, M. Rarey, and J. Kirchmair, "High-quality dataset of protein-bound ligand conformations and its application to benchmarking conformer ensemble generators," *J. Chem. Inf. Model.*, vol. 57, no. 3, pp. 529–539, Mar. 2017, ISSN: 1549-9596. DOI:

10.1021/acs.jcim.6b00613. [Online]. Available: https://doi.org/10.1021/acs.jcim.6b00613.

[29] J. Tong and S. Zhao, "Large-scale analysis of bioactive ligand conformational strain energy by ab initio calculation," *Journal of Chemical Information and Modeling*, vol. 61, pp. 1180–1192, 3 Mar. 2021, doi: 10.1021/acs.jcim.0c01197, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c01197. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c01197.

[30] R. P. Joosten, J. Salzemann, V. Bloch, H. Stockinger, A.-C. Berglund, C. Blanchet, E. Bongcam-Rudloff, C. Combet, A. L. Da Costa, G. Deleage, M. Diarena, R. Fabbretti, G. Fettahi, V. Flegel, A. Gisel, V. Kasam, T. Kervinen, E. Korpelainen, K. Mattila, M. Pagni, M. Reichstadt, V. Breton, I. J. Tickle, and G. Vriend, "PDB_REDO: automated re-refinement of X-ray structure models in the PDB," *Journal of Applied Crystallography*, vol. 42, no. 3, pp. 376–384, Jun. 2009. DOI: 10.1107/S0021889809008784. [Online]. Available: https://doi.org/10.1107/S0021889809008784.

[31] Z. Liu, M. Su, L. Han, J. Liu, Q. Yang, Y. Li, and R. Wang, "Forging the basis for developing protein–ligand interaction scoring functions," *Accounts Chem Res*, vol. 50, pp. 302–309, 2 2017. DOI: 10.1021/acs.accounts.6b00491. [Online]. Available: https://doi.org/10.1021/acs.accounts.6b00491.

[32] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li, and R. Wang, "Comparative assessment of scoring functions: The casf-2016 update," *J. Chem. Inf. Model.*, vol. 59, pp. 895–913, 2 2019. DOI: 10.1021/acs.jcim.8b00545. [Online]. Available: https://doi.org/10.1021/acs.jcim.8b00545.

[33] C. R. Groom, I. J. Bruno, M. P. Lightfoot, and S. C. Ward, "The cambridge structural database," *Acta Crystallographica Section B*, vol. 72, pp. 171–179, 2 Apr. 2016. DOI: 10.1107/S2052520616003954. [Online]. Available: https://doi.org/10.1107/S2052520616003954.

[34] B. Kuhn, E. Gilberg, R. Taylor, J. Cole, and O. Korb, "How significant are unusual protein-ligand interactions? insights from database mining," *J. Med. Chem.*, vol. 62, no. 22, pp. 10 441–10 455, Nov. 2019, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.9b01545. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.9b01545.

[35] D. Hilger, M. Masureel, and B. K. Kobilka, "Structure and dynamics of gpcr signaling complexes," *Nature Structural & Molecular Biology*, vol. 25, no. 1, pp. 4–12,

2018, ISSN: 1545-9985. DOI: 10.1038/s41594-017-0011-7. [Online]. Available: https://doi.org/10.1038/s41594-017-0011-7.

[36] F. H. Allen, S. E. Harris, and R. Taylor, "Comparison of conformer distributions in the crystalline state with conformational energies calculated by ab initio techniques," *Journal of Computer-Aided Molecular Design*, vol. 10, no. 3, pp. 247–254, 1996, ISSN: 1573-4951. DOI: 10.1007/BF00355046. [Online]. Available: https://doi.org/10.1007/BF00355046.

[37] J. Liebeschuetz, J. Hennemann, T. Olsson, and C. R. Groom, "The good, the bad and the twisted: A survey of ligand geometry in protein crystal structures," *Journal of Computer-Aided Molecular Design*, vol. 26, pp. 169–183, 2 2012, ISSN: 1573-4951. DOI: 10.1007/s10822-011-9538-6. [Online]. Available: https://doi.org/10.1007/s10822-011-9538-6.

[38] J. W. Liebeschuetz, "The good, the bad, and the twisted revisited: An analysis of ligand geometry in highly resolved protein–ligand x-ray structures," *Journal of Medicinal Chemistry*, vol. 64, pp. 7533–7543, 11 Jun. 2021, doi: 10.1021/acs.jmedchem.1c00228. ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.1c00228. [Online]. Available: https://doi.org/10.1021/acs.jmedchem.1c00228.

[39] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, pp. 235–242, 1 2000, ISSN: 0305-1048. DOI: 10.1093/nar/28.1.235. [Online]. Available: https://doi.org/10.1093/nar/28.1.235.

[40] N. O. Friedrich, C. D. B. Kops, F. Flachsenberg, K. Sommer, M. Rarey, and J. Kirchmair, "Benchmarking commercial conformer ensemble generators," *Journal of Chemical Information and Modeling*, vol. 57, pp. 2719–2728, 11 2017, ISSN: 1520-5142. DOI: 10.1021/acs.jcim.7b00505.

[41] N. O. Friedrich, F. Flachsenberg, A. Meyder, K. Sommer, J. Kirchmair, and M. Rarey, "Conformator: A novel method for the generation of conformer ensembles," *Journal of Chemical Information and Modeling*, vol. 59, pp. 731–742, 2 2019, ISSN: 1520-5142. DOI: 10.1021/acs.jcim.8b00704.

[42] P. C. D. Hawkins, A. G. Skillman, G. L. Warren, B. A. Ellingson, and M. T. Stahl, "Conformer generation with omega: Algorithm and validation using high quality structures from the protein databank and cambridge structural database," *Journal of Chemical Information and Modeling*, vol. 50, pp. 572–584, 4 2010,

PMID: 20235588. DOI: 10.1021/ci100031x. [Online]. Available: https://doi.org/10.1021/ci100031x.

[43] G. Klebe and T. Mietzner, "A fast and efficient method to generate biologically relevant conformations," *Journal of Computer-Aided Molecular Design*, vol. 8, pp. 583–606, 5 1994, ISSN: 1573-4951. DOI: 10.1007/BF00123667. [Online]. Available: https://doi.org/10.1007/BF00123667.

[44] R. S. Cahn, C. Ingold, and V. Prelog, "Specification of molecular chirality," *Angewandte Chemie International Edition*, vol. 5, no. 4, pp. 385–415, 1966. DOI: https://doi.org/10.1002/anie.196603851. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.196603851. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.196603851.

[45] I. J. Bruno, J. C. Cole, M. Kessler, J. Luo, W. D. S. Motherwell, L. H. Purkis, B. R. Smith, R. Taylor, R. I. Cooper, S. E. Harris, and A. G. Orpen, "Retrieval of crystallographically-derived molecular geometry information," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 6, pp. 2133–2144, Nov. 2004, ISSN: 0095-2338. DOI: 10.1021/ci049780b. [Online]. Available: https://doi.org/10.1021/ci049780b.

[46] S. J. Cottrell, T. S. G. Olsson, R. Taylor, J. C. Cole, and J. W. Liebeschuetz, "Validating and understanding ring conformations using small molecule crystallographic data," *J. Chem. Inf. Model.*, vol. 52, no. 4, pp. 956–962, Apr. 2012, ISSN: 1549-9596. DOI: 10.1021/ci200439d. [Online]. Available: https://doi.org/10.1021/ci200439d.

[47] *Daylight Theory: SMARTS - A Language for Describing Molecular Patterns*, Date accessed: 07.04.2022. [Online]. Available: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html.

[48] J. Sadowski and J. Boström, "Mimumba revisited: Torsion angle rules for conformer generation derived from x-ray structures," *J. Chem. Inf. Model.*, vol. 46, no. 6, pp. 2305–2309, Nov. 2006, ISSN: 1549-9596. DOI: 10.1021/ci060042s. [Online]. Available: https://doi.org/10.1021/ci060042s.

[49] C. Schärfer, T. Schulz-Gasch, H.-C. Ehrlich, W. Guba, M. Rarey, and M. Stahl, "Torsion angle preferences in druglike chemical space: A comprehensive guide," *Journal of Medicinal Chemistry*, vol. 56, pp. 2016–2028, 5 2013, PMID: 23379567. DOI: 10.1021/jm3016816. [Online]. Available: https://doi.org/10.1021/jm3016816.

[50] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, Feb. 1988, ISSN: 0095-2338. DOI: 10.1021/ci00057a005. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

[51] E. Ehmki, R. Schmidt, F. Ohm, and M. Rarey, "Comparing molecular patterns using the example of smarts: Applications and filter collection analysis," *Journal of Chemical Information and Modeling*, vol. 59, pp. 2572–2586, 6 2019, PMID: 31125225. DOI: 10.1021/acs.jcim.9b00249. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00249.

[52] B. Sellers, N. James, and A. Gobbi, "A comparison of quantum and molecular mechanical methods to estimate strain energy in druglike fragments," *Journal of Chemical Information and Modeling*, vol. 57, pp. 1265–1275, 6 2017, PMID: 28485585. DOI: 10.1021/acs.jcim.6b00614. [Online]. Available: https://doi.org/10.1021/acs.jcim.6b00614.

[53] B. K. Rai, V. Sresht, Q. Yang, R. Unwalla, M. Tu, A. M. Mathiowetz, and G. A. Bakken, "Comprehensive assessment of torsional strain in crystal structures of small molecules and protein-ligand complexes using ab initio calculations," *Journal of Chemical Information and Modeling*, vol. 59, pp. 4195–4208, 10 Oct. 2019, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.9b00373. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00373.

[54] B. K. Rai, V. Sresht, Q. Yang, R. Unwalla, M. Tu, A. M. Mathiowetz, and G. A. Bakken, "Torsionnet: A deep neural network to rapidly predict small-molecule torsional energy profiles with the accuracy of quantum mechanics," *J. Chem. Inf. Model.*, vol. 62, no. 4, pp. 785–800, Feb. 2022, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c01346. [Online]. Available: https://doi.org/10.1021/acs.jcim.1c01346.

[55] C. D. Stern, J. Maat, D. L. Dotson, C. I. Bayly, D. G. A. Smith, D. L. Mobley, and J. D. Chodera, "Capturing non-local through-bond effects in molecular mechanics force fields: Ii. using fractional bond orders to fit torsion parameters," *bioRxiv*, 2022. DOI: 10.1101/2022.01.17.476653. eprint: https://www.biorxiv.org/content/early/2022/01/18/2022.01.17.476653.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2022/01/18/2022.01.17.476653.

[56] J. Wahl, J. Freyss, M. von Korff, and T. Sander, "Accuracy evaluation and addition of improved dihedral parameters for the mmff94s," *Journal of Cheminformatics*, vol. 11, no. 1, p. 53, 2019, ISSN: 1758-2946. DOI: 10.1186/s13321-019-0371-6. [Online]. Available: https://doi.org/10.1186/s13321-019-0371-6.

[57] S. Gu, M. S. Smith, Y. Yang, J. J. Irwin, and B. K. Shoichet, "Ligand strain energy in large library docking," *Journal of Chemical Information and Modeling*, vol. 61, pp. 4331–4341, 9 Sep. 2021, doi: 10.1021/acs.jcim.1c00368, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c00368. [Online]. Available: https://doi.org/10.1021/acs.jcim.1c00368.

[58] J. W. Neidigh, R. M. Fesinmeyer, and N. H. Andersen, "Designing a 20-residue protein," *Nature Structural Biology*, vol. 9, no. 6, pp. 425–430, 2002, ISSN: 1545-9985. DOI: 10.1038/nsb798. [Online]. Available: https://doi.org/10.1038/nsb798.

[59] M. Johnson, S. Basak, and G. Maggiora, "A characterization of molecular similarity methods for property prediction," *Mathematical and Computer Modelling*, vol. 11, pp. 630–634, 1988, ISSN: 0895-7177. DOI: https://doi.org/10.1016/0895-7177(88)90569-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0895717788905699.

[60] A. Kumar and K. Y. J. Zhang, "Advances in the development of shape similarity methods and their application in drug discovery," *Frontiers in Chemistry*, vol. 6, p. 315, 2018, ISSN: 2296-2646. DOI: 10.3389/fchem.2018.00315. [Online]. Available: https://www.frontiersin.org/article/10.3389/fchem.2018.00315.

[61] J. A. Grant and B. T. Pickup, "A gaussian description of molecular shape," *The Journal of Physical Chemistry*, vol. 99, pp. 3503–3510, 11 1995. DOI: 10.1021/j100011a016. [Online]. Available: https://doi.org/10.1021/j100011a016.

[62] J. A. Grant, M. A. Gallardo, and B. T. Pickup, "A fast method of molecular shape comparison: A simple application of a gaussian description of molecular shape," *Journal of Computational Chemistry*, vol. 17, no. 14, pp. 1653–1666, 1996. DOI: 10.1002/(SICI)1096-987X(19961115)17:14<1653::AID-JCC7>3.0.CO;2-K.

[63] T. S. Rush, J. A. Grant, L. Mosyak, and A. Nicholls, "A shape-based 3-d scaffold hopping method and its application to a bacterial protein-protein interaction," *Journal of Medicinal Chemistry*, vol. 48, pp. 1489–1495, 5 2005, PMID: 15743191. DOI: 10.1021/jm040163o. [Online]. Available: https://doi.org/10.1021/jm040163o.

[64] *Fastrocs toolkit — real-time shape similarity — lead discovery*, Date accessed: 11.05.2022. [Online]. Available: https://www.eyesopen.com/molecular-modeling-fastrocs.

[65] P. J. Ballester and W. G. Richards, "Ultrafast shape recognition to search compound databases for similar molecular shapes," *Journal of Computational Chemistry*, vol. 28, no. 10, pp. 1711–1723, 10 2007. DOI: 10.1002/jcc.20681. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20681.

[66] F. Flachsenberg, A. Meyder, K. Sommer, P. Penner, and M. Rarey, "A consistent scheme for gradient-based optimization of protein–ligand poses," *J. Chem. Inf. Model.*, vol. 60, pp. 6502–6522, 12 Dec. 2020, PMID: 33258376. DOI: 10.1021/acs.jcim.0c01095. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c01095.

[67] J. O. Ebalunode, Z. Ouyang, J. Liang, and W. Zheng, "Novel approach to structure-based pharmacophore search using computational geometry and shape matching techniques," *J. Chem. Inf. Model.*, vol. 48, no. 4, pp. 889–901, Apr. 2008, ISSN: 1549-9596. DOI: 10.1021/ci700368p. [Online]. Available: https://doi.org/10.1021/ci700368p.

[68] A. N. Jain, "Surflex: Fully automatic flexible molecular docking using a molecular similarity-based search engine," *J. Med. Chem.*, vol. 46, no. 4, pp. 499–511, Feb. 2003, ISSN: 0022-2623. DOI: 10.1021/jm020406h. [Online]. Available: https://doi.org/10.1021/jm020406h.

[69] N. Schneider, G. Lange, S. Hindle, R. Klein, and M. Rarey, "A consistent description of hydrogen bond and dehydration energies in protein–ligand complexes: Methods behind the hyde scoring function," *Journal of Computer-Aided Molecular Design*, vol. 27, pp. 15–29, 1 Jan. 2013, ISSN: 1573-4951. DOI: 10.1007/s10822-012-9626-2. [Online]. Available: https://doi.org/10.1007/s10822-012-9626-2.

[70] L. Mak, S. Grandison, and R. J. Morris, "An extension of spherical harmonics to region-based rotationally invariant descriptors for molecular shape description and comparison," *Journal of Molecular Graphics and Modelling*, vol. 26, pp. 1035–1045, 7 2008, ISSN: 1093-3263. DOI: https://doi.org/10.1016/j.jmgm.2007.08.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S109332630700143X.

[71] J. Zarnecka, I. Lukac, S. J. Messham, A. Hussin, F. Coppola, S. J. Enoch, A. G. Dossetter, E. J. Griffen, and A. G. Leach, "Mapping ligand-shape space for protein-ligand systems: Distinguishing key-in-lock and hand-in-glove proteins," *J. Chem. Inf. Model.*, vol. 61, no. 4, pp. 1859–1874, Apr. 2021, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c00089. [Online]. Available: https://doi.org/10.1021/acs.jcim.1c00089.

[72] R. Wang, Y. Gao, and L. Lai, "Ligbuilder: A multi-purpose program for structure-based drug design," *Journal of Molecular Modeling*, vol. 6, no. 7, pp. 498–516, 2000, ISSN: 0948-5023. DOI: 10.1007/s0089400060498. [Online]. Available: https://doi.org/10.1007/s0089400060498.

[73] R. M. Stein, Y. Yang, T. E. Balius, M. J. O'Meara, J. Lyu, J. Young, K. Tang, B. K. Shoichet, and J. J. Irwin, "Property-unmatched decoys in docking benchmarks," *J. Chem. Inf. Model.*, vol. 61, pp. 699–714, 2 Jan. 2021. DOI: 10.1021/acs.jcim.0c00598.

[74] J.-R. Marchand and A. Caflisch, "In silico fragment-based drug design with seed," *Eur J Med Chem*, vol. 156, pp. 907–917, 2018, ISSN: 0223-5234. DOI: 10.1016/j.ejmech.2018.07.042. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0223523418305981.

[75] Z. Liu, S. B. Singh, Y. Zheng, P. Lindblom, C. Tice, C. Dong, L. Zhuang, Y. Zhao, B. A. Kruk, D. Lala, D. A. Claremon, G. M. McGeehan, R. D. Gregg, and R. Cain, "Discovery of potent inhibitors of 11β-hydroxysteroid dehydrogenase type 1 using a novel growth-based protocol of in silico screening and optimization in contour," *Journal of Chemical Information and Modeling*, vol. 59, pp. 3422–3436, 8 2019, PMID: 31355641. DOI: 10.1021/acs.jcim.9b00198. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00198.

[76] J. Degen and M. Rarey, "Flexnovo: Structure-based searching in large fragment spaces," *ChemMedChem*, vol. 1, pp. 854–868, 8 2006, ISSN: 1860-7179. DOI: 10.1002/cmdc.200500102.

[77] F. Chevillard and P. Kolb, "Scubidoo: A large yet screenable and easily searchable database of computationally created chemical compounds optimized toward high likelihood of synthetic tractability," *J. Chem. Inf. Model.*, vol. 55, pp. 1824–1835, 9 2015, ISSN: 1520-5142. DOI: 10.1021/acs.jcim.5b00203.

[78] K. Sommer, F. Flachsenberg, and M. Rarey, "Naominext – synthetically feasible fragment growing in a structure-based design context," *Eur J Med Chem*, vol. 163,

pp. 747–762, 2019, ISSN: 0223-5234. DOI: 10.1016/j.ejmech.2018.11.075. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0223523418310377.

[79] S. Malhotra and J. Karanicolas, "When does chemical elaboration induce a ligand to change its binding mode?" *Journal of Medicinal Chemistry*, vol. 60, pp. 128–145, 1 2017, ISSN: 1520-4804. DOI: 10.1021/acs.jmedchem.6b00725.

[80] S. Urbaczek, A. Kolodzik, J. R. Fischer, T. Lippert, S. Heuser, I. Groth, T. Schulz-Gasch, and M. Rarey, "Naomi: On the almost trivial task of reading molecules from different file formats," *Journal of Chemical Information and Modeling*, vol. 51, pp. 3199–3207, 12 2011, PMID: 22067015. DOI: 10.1021/ci200324e. [Online]. Available: https://doi.org/10.1021/ci200324e.

[81] W. Guba, A. Meyder, M. Rarey, and J. Hert, "Torsion library reloaded: A new version of expert-derived smarts rules for assessing conformations of small molecules," *Journal of Chemical Information and Modeling*, vol. 56, pp. 1–5, 1 2016, PMID: 26679290. DOI: 10.1021/acs.jcim.5b00522. [Online]. Available: https://doi.org/10.1021/acs.jcim.5b00522.

[82] *SMARTS - Open Babel*, Date accessed: 20.06.2022. [Online]. Available: http://openbabel.org/wiki/SMARTS.

[83] R. J. Gillespie and R. S. Nyholm, "Inorganic stereochemistry," *Quarterly Reviews, Chemical Society*, 1957. DOI: 10.1039/QR9571100339. [Online]. Available: https://doi.org/10.1039%2FQR9571100339.

[84] A. S. Rose and P. W. Hildebrand, "Ngl viewer: A web application for molecular visualization," *Nucleic Acids Res.*, vol. 43, W576–W579, W1 2015, ISSN: 0305-1048. DOI: 10.1093/nar/gkv402. [Online]. Available: https://doi.org/10.1093/nar/gkv402.

[85] R. Schmidt, E. S. R. Ehmki, F. Ohm, H.-C. Ehrlich, A. Mashychev, and M. Rarey, "Comparing molecular patterns using the example of smarts: Theory and algorithms," *Journal of Chemical Information and Modeling*, vol. 59, pp. 2560–2571, 6 2019, PMID: 31120751. DOI: 10.1021/acs.jcim.9b00250. [Online]. Available: https://doi.org/10.1021/acs.jcim.9b00250.

[86] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed. John Wiley & Sons, Inc., 2005, p. 633.

[87] E. Nittinger, T. Inhester, S. Bietz, A. Meyder, K. T. Schomburg, G. Lange, R. Klein, and M. Rarey, "Large-scale analysis of hydrogen bond interaction patterns in protein-ligand interfaces," *J. Med. Chem.*, vol. 60, pp. 4245–4257, 10 2017, ISSN: 1520-4804. DOI: 10.1021/acs.jmedchem.7b00101.

[88] D. E. Koshland, "The key–lock theory and the induced fit theory," *Angewandte Chemie International Edition in English*, vol. 33, pp. 2375–2378, 23-24 1995. DOI: https://doi.org/10.1002/anie.199423751. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.199423751.

[89] S. Bietz and M. Rarey, "Siena: Efficient compilation of selective protein binding site ensembles," *J. Chem. Inf. Model.*, vol. 56, pp. 248–259, 1 2016, PMID: 26759067. DOI: 10.1021/acs.jcim.5b00588. [Online]. Available: https://doi.org/10.1021/acs.jcim.5b00588.

[90] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. P. Magariños, J. P. Overington, G. Papadatos, I. Smit, and A. R. Leach, "The ChEMBL database in 2017," *Nucleic Acids Research*, vol. 45, no. D1, pp. D945–D954, Nov. 2016, ISSN: 0305-1048. DOI: 10.1093/nar/gkw1074. eprint: https://academic.oup.com/nar/article-pdf/45/D1/D945/8846762/gkw1074.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkw1074.

[91] C. Kramer, T. Kalliokoski, P. Gedeck, and A. Vulpetti, "The experimental uncertainty of heterogeneous public ki data," *Journal of Medicinal Chemistry*, vol. 55, pp. 5165–5173, 11 2012, PMID: 22643060. DOI: 10.1021/jm300131x. [Online]. Available: https://doi.org/10.1021/jm300131x.

[92] N. Huang, B. K. Shoichet, and J. J. Irwin, "Benchmarking sets for molecular docking," *J. Med. Chem.*, vol. 49, no. 23, pp. 6789–6801, Nov. 2006, ISSN: 0022-2623. DOI: 10.1021/jm0608356. [Online]. Available: https://doi.org/10.1021/jm0608356.

[93] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet, "Directory of useful decoys, enhanced (dud-e): Better ligands and decoys for better benchmarking," *J. Med. Chem.*, vol. 55, no. 14, pp. 6582–6594, Jul. 2012, ISSN: 0022-2623. DOI: 10.1021/jm300687e. [Online]. Available: https://doi.org/10.1021/jm300687e.

[94] P. C. D. Hawkins, A. G. Skillman, and A. Nicholls, "Comparison of shape-matching and docking as virtual screening tools," *Journal of Medicinal Chem-*

istry, vol. 50, pp. 74–82, 1 2007, PMID: 17201411. DOI: 10.1021/jm0603365. [Online]. Available: https://doi.org/10.1021/jm0603365.

[95] Y. Yuan, J. Pei, and L. Lai, "Ligbuilder v3: A multi-target de novo drug design approach," *Frontiers in Chemistry*, vol. 8, p. 142, 2020, ISSN: 2296-2646. DOI: 10.3389/fchem.2020.00142. [Online]. Available: https://www.frontiersin.org/article/10.3389/fchem.2020.00142.

[96] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee, "Empirical scoring functions: I. the development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes," *J. Comput. Aided Mol. Des.*, vol. 11, pp. 425–445, 5 1997, ISSN: 1573-4951. DOI: 10.1023/A:1007996124545.

[97] O. Korb, T. Stützle, and T. E. Exner, "Empirical scoring functions for advanced protein-ligand docking with plants," *J. Chem. Inf. Model.*, vol. 49, pp. 84–96, 1 Jan. 2009. DOI: 10.1021/ci800298z.

[98] R. E. Amaro, J. Baudry, J. Chodera, Ö. Demir, J. A. McCammon, Y. Miao, and J. C. Smith, "Ensemble docking in drug discovery," *Biophys. J.*, vol. 114, pp. 2271–2278, 10 2018, ISSN: 1542-0086. DOI: 10.1016/j.bpj.2018.02.038. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0006349518303242.

[99] W. J. Allen, T. E. Balius, S. Mukherjee, S. R. Brozell, D. T. Moustakas, P. T. Lang, D. A. Case, I. D. Kuntz, and R. C. Rizzo, "Dock 6: Impact of new features and current docking performance," *J. Comput. Chem.*, vol. 36, no. 15, pp. 1132–1156, 15 2015. DOI: 10.1002/jcc.23905. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.23905.

[100] J. Lyu, S. Wang, T. E. Balius, I. Singh, A. Levit, Y. S. Moroz, M. J. O'Meara, T. Che, E. Algaa, K. Tolmachova, A. A. Tolmachev, B. K. Shoichet, B. L. Roth, and J. J. Irwin, "Ultra-large library docking for discovering new chemotypes," *Nature*, vol. 566, no. 7743, pp. 224–229, 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-0917-9.

[101] W. J. Allen, B. C. Fochtman, T. E. Balius, and R. C. Rizzo, "Customizable de novo design strategies for dock: Application to hivgp41 and other therapeutic targets," *J Comput Chem*, vol. 38, pp. 2641–2663, 30 2017. DOI: https://doi.org/10.1002/jcc.25052. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.25052.

[102] F. K. Wiseman, K. A. Alford, V. L. J. Tybulewicz, and E. M. C. Fisher, "Down syndrome—recent progress and future prospects," *Hum. Mol. Genet.*, vol. 18, no. R1, R75–R83, Apr. 2009, ISSN: 0964-6906. DOI: 10.1093/hmg/ddp010. eprint: https://academic.oup.com/hmg/article-pdf/18/R1/R75/9460787/ddp010.pdf. [Online]. Available: https://doi.org/10.1093/hmg/ddp010.

[103] J. Wegiel, C.-X. Gong, and Y.-W. Hwang, "The role of dyrk1a in neurodegenerative diseases," *FEBS J.*, vol. 278, no. 2, pp. 236–245, 2011. DOI: https://doi.org/10.1111/j.1742-4658.2010.07955.x. eprint: https://febs.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1742-4658.2010.07955.x. [Online]. Available: https://febs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1742-4658.2010.07955.x.

[104] P. Fernández-Martínez, C. Zahonero, and P. Sánchez-Gómez, "Dyrk1a: The double-edged kinase as a protagonist in cell growth and tumorigenesis," *Mol. Cell. Oncol.*, vol. 2, no. 1, e970048, 2015, PMID: 27308401. DOI: 10.4161/23723548.2014.970048. eprint: https://doi.org/10.4161/23723548.2014.970048. [Online]. Available: https://doi.org/10.4161/23723548.2014.970048.

[105] C. Weber, M. Sipos, A. Paczal, B. Balint, V. Kun, N. Foloppe, P. Dokurno, A. J. Massey, D. L. Walmsley, R. E. Hubbard, J. Murray, K. Benwell, T. Edmonds, D. Demarles, A. Bruno, M. Burbridge, F. Cruzalegui, and A. Kotschy, "Structure-guided discovery of potent and selective dyrk1a inhibitors," *J. Med. Chem.*, vol. 64, pp. 6745–6764, 10 May 2021, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.1c00023.

[106] D. L. Walmsley, J. B. Murray, P. Dokurno, A. J. Massey, K. Benwell, A. Fiumana, N. Foloppe, S. Ray, J. Smith, A. E. Surgenor, T. Edmonds, D. Demarles, M. Burbridge, F. Cruzalegui, A. Kotschy, and R. E. Hubbard, "Fragment-derived selective inhibitors of dual-specificity kinases dyrk1a and dyrk1b," *J. Med. Chem.*, vol. 64, pp. 8971–8991, 13 Jul. 2021, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.1c00024.

[107] *Drug discovery with biosolveit apps — expect actives!* Date accessed: 08.03.2022. [Online]. Available: https://www.biosolveit.de/.

[108] M. Hartenfeller, M. Eberle, P. Meier, C. Nieto-Oberhuber, K. H. Altmann, G. Schneider, E. Jacoby, and S. Renner, "A collection of robust organic synthesis reactions for in silico molecule design," *Journal of Chemical Information and Modeling*, vol. 51, pp. 3093–3098, 12 2011, ISSN: 1549-9596. DOI: 10.1021/ci200379p.

[109] J. Degen, C. Wegscheid-Gerlach, A. Zaliani, and M. Rarey, "On the art of compiling and using 'drug-like' chemical fragment spaces," *ChemMedChem*, vol. 3, pp. 1503–1507, 10 2008, ISSN: 1860-7179. DOI: 10.1002/cmdc.200800178.

[110] T. Hoffmann and M. Gastreich, "The next level in chemical space navigation: Going far beyond enumerable compound libraries," *Drug Discovery Today*, vol. 24, pp. 1148–1156, 5 2019, ISSN: 1359-6446. DOI: 10.1016/j.drudis.2019.02.013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359644618304471.

[111] M. Segall, "Advances in multiparameter optimization methods for de novo drug design," *Expert Opinion on Drug Discovery*, vol. 9, no. 7, pp. 803–817, Jul. 2014, ISSN: 1746-0441. DOI: 10.1517/17460441.2014.913565. [Online]. Available: https://doi.org/10.1517/17460441.2014.913565.

[112] G. Schneider, M.-L. Lee, M. Stahl, and P. Schneider, "De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks," *Journal of Computer-Aided Molecular Design*, vol. 14, no. 5, pp. 487–494, 2000, ISSN: 1573-4951. DOI: 10.1023/A:1008184403558. [Online]. Available: https://doi.org/10.1023/A:1008184403558.

[113] L. Bellmann, P. Penner, and M. Rarey, "Topological similarity search in large combinatorial fragment spaces," *Journal of Chemical Information and Modeling*, vol. 61, pp. 238–251, 1 2021, PMID: 33084338, ISSN: 1520-5142. DOI: 10.1021/acs.jcim.0c00850. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00850.

[114] A. L. Hopkins, C. R. Groom, and A. Alex, "Ligand efficiency: A useful metric for lead selection," *Drug Discovery Today*, vol. 9, pp. 430–431, 10 2004, ISSN: 1359-6446. DOI: https://doi.org/10.1016/S1359-6446(04)03069-7. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359644604030697.

[115] *Qt Documentation — Home*, Date accessed: 20.07.2022. [Online]. Available: https://doc.qt.io/.

[116] *SQLite Documentation*, Date accessed: 20.07.2022. [Online]. Available: https://www.sqlite.org/docs.html.

[117] *PostgreSQL: Documentation*, Date accessed: 20.07.2022. [Online]. Available: https://www.postgresql.org/docs/.

[118] *Ruby on Rails Guides*, Date accessed: 22.07.2022. [Online]. Available: https://guides.rubyonrails.org/.

[119] *Ruby Programming Language*, Date accessed: 22.07.2022. [Online]. Available: https://www.ruby-lang.org/en/.

[120] *Django documentation — Django documentation — Django*, Date accessed: 22.07.2022. [Online]. Available: https://docs.djangoproject.com/.

[121] *Our Documentation — Python.org*, Date accessed: 22.07.2022. [Online]. Available: https://www.python.org/doc/.

[122] *Programming, scripting, and markup languages — Stack Overflow Developer Survey 2021*, Date accessed: 22.07.2022. [Online]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-programming-scripting-and-markup-languages.

[123] *SingularityCE User Guide — SingularityCE User Guide main documentation*, Date accessed: 22.07.2022. [Online]. Available: https://docs.sylabs.io/guides/main/user-guide/.

[124] *Docker Documentation — Docker Documentation*, Date accessed: 22.07.2022. [Online]. Available: https://docs.docker.com/.

[125] *Other tools — Stack Overflow Developer Survey 2021*, Date accessed: 22.07.2022. [Online]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-other-tools.

[126] *WebGL Overview - The Khronos Group Inc*, Date accessed: 22.07.2022. [Online]. Available: https://webgl.org/.

[127] *Vue.js - The Progressive JavaScript Framework — Vue.js*, Date accessed: 22.07.2022. [Online]. Available: https://vuejs.org/.

[128] *Components Basics — Vue.js*, Date accessed: 22.07.2022. [Online]. Available: https://vuejs.org/guide/essentials/component-basics.html.

[129] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, and B. A. Leland, "Description of several chemical structure file formats used by computer programs developed at molecular design limited," *Journal of Chemical Information and Computer Sciences*, vol. 32, 1992. DOI: 10.1021/ci00007a012. [Online]. Available: http://dx.doi.org/10.1021/ci00007a012.

Bibliography of the Cumulative Dissertation

[D1] P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, D. Brown, and M. Rarey, "Shape-Based Descriptors for Efficient Structure-Based Fragment Growing," *J. Chem. Inf. Model.*, vol. 60, pp. 6269–6281, 12 2020, PMID: 33196169, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00920. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00920.

[D2] P. Penner, W. Guba, R. Schmidt, A. Meyder, M. Stahl, and M. Rarey, "The Torsion Library: Semiautomated Improvement of Torsion Rules with SMARTScompare," *Journal of Chemical Information and Modeling*, Mar. 2022, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.2c00043. [Online]. Available: https://doi.org/10.1021/acs.jcim.2c00043.

[D3] P. Penner, V. Martiny, L. Bellmann, F. Flachsenberg, M. Gastreich, I. Theret, C. Meyer, and M. Rarey, "FastGrow: on-the-fly growing and its application to DYRK1A," *Journal of Computer-Aided Molecular Design*, 2022, ISSN: 1573-4951. DOI: 10.1007/s10822-022-00469-y. [Online]. Available: https://doi.org/10.1007/s10822-022-00469-y.

Appendices

A. Scientific Contributions

A.1. Contributions to Publications of the Cumulative Dissertation

This selection of articles lists all the author's contributions to scientific journals.

[D1] P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, D. Brown, and M. Rarey, "Shape-Based Descriptors for Efficient Structure-Based Fragment Growing," *J. Chem. Inf. Model.*, vol. 60, pp. 6269–6281, 12 2020, PMID: 33196169, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00920. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00920

P. Penner wrote the main manuscript text and developed the algorithm. V. Martiny and M. Gastreich tested the algorithm and provided feedback on the methodology. P. Ducrot initiated the project. M. Rarey contributed to method development. V. Martiny, A. Gohier, D. Brown coordinated the cooperation with Servier. M. Gastreich coordinated the cooperation with BioSolveIT. All authors reviewed the manuscript.

[D2] P. Penner, W. Guba, R. Schmidt, A. Meyder, M. Stahl, and M. Rarey, "The Torsion Library: Semiautomated Improvement of Torsion Rules with SMARTScompare," *Journal of Chemical Information and Modeling*, Mar. 2022, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.2c00043. [Online]. Available: https://doi.org/10.1021/acs.jcim.2c00043

P. Penner wrote the main manuscript text, implemented the necessary software and generated the analysis data. W. Guba performed the manual analysis and gave expert input on Torsion Library SMARTS. R. Schmidt implemented the modifications to the SMARTScompare algorithm. A. Meyder provided preliminary work. M. Stahl coordinated the cooperation with Roche. M. Rarey provided feedback on the method and validation. All authors reviewed the manuscript.

[D3] P. Penner, V. Martiny, L. Bellmann, F. Flachsenberg, M. Gastreich, I. Theret, C. Meyer, and M. Rarey, "FastGrow: on-the-fly growing and its application to DYRK1A," *Journal of Computer-Aided Molecular Design*, 2022, ISSN: 1573-4951. DOI: 10.1007/s10822-022-00469-y. [Online]. Available: https://doi.org/10.1007/s10822-022-00469-y

P. Penner wrote the main manuscript text and developed the tool. L. Bellmann, F. Flachsenberg and M. Rarey contributed to method development. F. Flachsenberg also added support for restrained optimization to JAMDA. V. Martiny, I. Theret and M. Gastreich tested the tool and provided feedback on the methodology. V. Martiny and C. Meyer coordinated the cooperation with Servier. M. Gastreich coordinated the cooperation with BioSolveIT. All authors reviewed the manuscript.

A.2. Conference Contributions related to the Cumulative Dissertation

A.2.1. Oral Presentations

P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, and M. Rarey, "Next Generation Methods for Structure-Based Fragment Growing," GRS/GRC Computer Aided Drug Design, West Dover VT, USA, 2019

P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, D. Brown, and M. Rarey, "FGVD: Public Structure Data for Fragment Growing Validation," 16th German Conference on Cheminformatics and SAMPL Satellite Workshop, Online, 2020

P. Penner, W. Guba, R. Schmidt, A. Meyder, M. Stahl, and M. Rarey, "Improving Torsion Library Patterns with SMARTScompare," 12th International Conference on Chemical Structures (ICCS), Noordwijkerhout, Netherlands, 2022

A.2.2. Poster Presentations

P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, and M. Rarey, "Next Generation Methods for Structure-Based Fragment Growing," GRS/GRC Computer Aided Drug Design, West Dover VT, USA, 2019

B. Software Architecture and Usage

B.1. Software Architecture

At the beginning of the project, discussions about the software architecture quickly veered toward setting a client-server architecture as the eventual goal. From an organizational perspective, there are several advantages to a client-server architecture. A server is a centrally managed instance and centralization facilitates maintenance. The workload is also centralized to corporate infrastructure. Central server infrastructure can be upscaled and downscaled at will. This may be more efficient than ensuring every user is issued a machine powerful enough to handle their requirements. If the client is a web browser, as it will be in this case, then the organization profits from the virtualization a web browser provides.

An academic software developer can also profit from a client-server model. Setting out to create a client-server architecture in a project separates the development of the scientific application into stages. A client-server model requires at least two components and the core scientific functionality will probably also be a separate component. This leads to 3 stages of development that to some degree mirror the maturity of a scientific application. The starting point is the core functionality, which is then extended by a standardized interface (the backend), and lastly made accessible to non-expert users (the frontend). In this project, the architecture was composed of standalone command-line applications, web server backends, and web server frontends.

B.1.1. Command-line Applications

All scientific models throughout this project were implemented in standalone C++ command-line applications. C++ was chosen as a programming language to ensure high performance, unambiguous control of the data, and because of prior foundations

laid down by the NAOMI C++ library[80]. Besides the scientifically necessary foundations, another core dependency of all software in this project was Qt[115]. Qt is a cross-platform development framework that is typically used to create GUI applications. Qt's capabilities go significantly beyond that of a pure GUI framework. In this project, we primarily used Qt's cross-platform capabilities for file handling, database management, and multithreading. As a widely used framework, Qt continues to be a pragmatic choice to develop cross-platform applications in C++.

Database management was a core technological challenge in this project. The number of compounds to screen in drug design is typically expected to exceed the amount of RAM available, which makes a database necessary. The speed of any screening functionality is therefore predicated on the structure of the database it is screening in. All command-line applications in this project that require a database have support for SQLite[116]. This is usually sufficient for any command-line application expected to run locally. FastGrow also has support for PostgreSQL[117]. FastGrow is expected to screen large fragment libraries that may be deployed as an organization-wide database. PostgreSQL is a common choice for organization-wide deployment. It is important to note how this may affect performance. A local SQLite database will always be faster than an equivalent network mounted PostgreSQL database simply because it does not incur network latency.

There was an expectation in this project that one of the project partners, the BioSolveIT[107], would integrate the developed software into their tool suite and maintain it for continued use. For that reason, another layer of separation between the core functionality and the application code was very useful. The core shape-based algorithm was implemented in a separate library that the BioSolveIT could isolate and modify to fit into their code base. An alternative would have been to integrate the complete command-line-tool, which would have, however, meant that all communication with the core algorithm would have required a call over the command-line. This could have been both a performance concern and security concern, as will be discussed in the web server backend section.

B.1.2. Web Server Backend

The initial web server backends implemented for this project were in Ruby on Rails[118]. Ruby is a multi-paradigm language[119] that is very commonly written in the Rails web framework. Rails provides much of the infrastructure a modern web application needs but tends to be very opinionated in how it is supposed to be used, citing its

"Convention over Configuration" principle. Fully featured implementations of Rails web server backends were generated in this project but eventually abandoned for Django implementations[120]. Django is written in the Python programming language[121], which is more popular than Ruby[122]. The switch to Django was done largely for maintainability reasons. Django and Rails are largely comparable on a technological level, but the pool of Python developers in a scientific setting is deeper than the pool of Ruby developers. It was the author's experience that Django was less restrictive and opinionated as a framework than Rails, possibly making it more appropriate for scientific applications.

Deployment of web servers in the course of this project was envisioned to use containerization. Two technologies were used to achieve this: Singularity[123] and Docker[124]. Singularity is a containerization system devised for high-performance computing clusters. It is fundamentally different from Docker in that a container may not grow in size after it has been built, which means every write process has to be redirected to a file system mounted into the container from the outside. Docker is more popular as a technology[125] and tends to be the container of choice for scalable web systems. Another fundamental difference between Docker and Singularity is that a Docker user is root in their container. This becomes a relevant security concern in containers that manage to break out into the host system. For this reason, many system administrators still regard Docker with suspicion and do not deploy it to their users. This was also the case in this project, which lead to an impasse of either using Singularity for something it wasn't made for and didn't necessarily support or trying to convince the system administrators in question to make exceptions for this project.

In the end, deployment went in several different directions. One of the problems we encountered was that the testers almost exclusively used network-mounted file systems. This introduced system breaking latencies into the backend, components of which quite literally time out. More centralized servers with local hard drives were also not available for our research efforts. The workaround was deploying the frontend described below to the testers without a backend and having the frontend write out queries that could be processed by the command-line tool. This was a far from optimal deployment but the best compromise available to us. We could deploy the full software stack on university servers for the public to use. However, because this was hosted on a university server and not the internal servers of our cooperation partners, they could not use this deployment because company policy forbids communication of confidential work to external servers. Deployment was thus marred by organizational problems.

B.1.3. Web Server Frontend

Web server frontends were developed as single-page Javascript applications. All of our functionality required the visualization of molecules. The NGL viewer[84], a WebGL[126] based molecular viewer, was used to achieve this. There was a significant amount of state the molecular viewer had to persist throughout the intended workflows, which is why they were written as single-page applications. The web GUI of the TorsionAnalyzer was less complicated and was implemented in plain Javascript. The complexity of the FastGrow web GUI was significantly more complicated and required a Javascript frontend framework to implement in a clean and structured way. We used Vue.js[127]. Vue.js also served to standardize the structure of the project so that maintainers may recognize it if they are familiar with the framework.

The general design paradigm in all frontends was typically a variation on the Model-View-Controller (MVC) design pattern and a web component structure. The examples will be for the more complex Vue.js structure. Vue.js is loosely inspired by the Model-View-Viewmodel pattern, but it has a very concrete implementation of a web component structure. The preferred method of building parts of a Vue.js application is through single file components that consist of an HTML template, scoped CSS styles, and the actual Vue.js component definition in Javascript that implements various functions, such as a life cycle function equivalent to a constructor. Outside of the scope of a framework, this can be generalized to a Javascript class that is mounted at a particular point in the DOM, generates its HTML, and communicates with other components. The fundamental departure from other forms of web development here is that Javascript owns the HTML (and in the Vue.js case even the CSS).

The complexity of the FastGrow web GUI required a full implementation of a Model-View-Controller variation. An overview of the architecture can be seen in Figure B.1. In this system, the central Vue.js app that initializes all Vue.js components was used as the controller. The components that represented the views communicated with the app using the usual system of props and events[128]. The app would map all events to model functions, thus acting as a controller. The models were implemented as plain Javascript classes that would receive access to a certain namespace in the data of the central app to store their state and would transform all data passed to them by the central app from the components to generate that state. The state stored in the central app could then be passed back to the components using props.

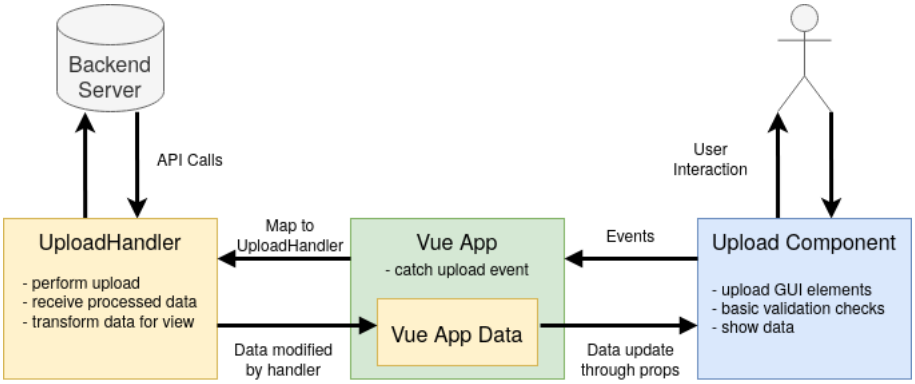


Figure B.1.: Schema of the MVC pattern implemented on top of Vue.js using a generic upload example. The two parts of the model are in yellow. The controller, which is the central Vue.js app, is in green. The view, which is a view component, is in blue.

There are several advantages and disadvantages to such a structure. The immediately obvious disadvantage is the complexity. Components are supported by Vue.js, but the MVC pattern is only present to some degree. Using components as views and the central app as the controller is reasonably intuitive, but splitting the model by having its state owned by the central app and implementing its behavior in a separate plain Javascript class is confusing. This is necessary for implementation reasons. Props are the canonical way to communicate data downstream to components and are implemented as an opaque system of data watchers and listeners. It is the author's opinion that splitting the model is cleaner than trying to hack the framework. The main advantage is that the complexity necessary to implement such an application is structured. Placing all model code in the central app would significantly increase its size and therefore be a detriment to legibility. Separating model code by concerns makes it a lot easier to find functionality and also to test it.

A few principles should be applied when implementing such a structure. First of all: "Fat models, lean controllers". This principle, commonly held in MVC frameworks such as Rails[118], states that as much functionality as possible should be moved into the models and out of the controllers. The controller in this case is the central Vue.js app which quickly grows if this principle isn't applied. Secondly, components should be seen purely as views and should only have functionality that fits in that category. For example, if a component is used to enter data, the component should at most perform basic validation and then send that data to the controller without transforming it significantly. The view is only supposed to render the user interface. Third, all

components should be tested individually. Entirely testing one Model-View-Controller cascade is in the scope of end-to-end testing. However, end-to-end testing is by its nature often very broad and unspecific. Components and especially model classes should be individually unit tested to handle specific edge cases. It is also beneficial to make as many model classes independent from server calls so that they can be unit tested instead of end-to-end tested. Testing the central Vue.js app is often cumbersome, which also supports the principle of moving as much functionality as possible into models.

B.2. Usage

Usability, especially in a scientific context, is often more dependent on the user's use case and their context than on a conventional understanding of user-friendliness. This was very apparent in our project due to infrastructure restrictions that led to the command-line tools being the most used deployment. The following is a description of the various tools built during the project and the FGVD pipeline.

B.2.1. TorsionPatternMiner

The command line interface of the TorsionPatternMiner is the following:

```
$>./TorsionPatternMiner --help

Basic Arguments:
  -h [ --help ]                         Show help and usage
  -m [ --mols ] arg                     input molecule to add to the statistic
  -d [ --database ] arg                 database of molecules to add to the
                                        statistic
  -o [ --outdir ] arg                   output directory containing analysis
                                        files
  -t [ --torlib ] arg                   torsion library to use as basis for
                                        recalculating statistics (default:
                                        internal torlib)
  --analysis                            Only analyze the torsion library, do
                                        not update statistics
```

```
  --curves                          Write out torsion score curves for
                                    every pattern
  --checkhierarchy                  Check that the pattern hierarchy is
                                    correct
  --sortpatterns                    Check that the pattern hierarchy is
                                    correct
  --datasource arg (=0)             data source the molecule data is from:
                                    0 (default: CSD), 1 (PDB)
  -u [ --userinfo ] arg (=1)        set user info level: quiet (0),
                                    warnings (1, default), workflow (2)
  -l [ --license ] arg              provide a new license for the
                                    torsionanalyzer

Matching Arguments:
  --selectivity arg (=0)            selectivity of the torsion matching:
                                    single most selective (2), all most
                                    selective (1), all (0, default)
  --terminalatoms                   score torsions containing terminal
                                    heavy atoms
  --fixconjugated                   do not score torsions of conjugated
                                    bonds
  --terminaldonors                  do not score torsion of terminal
                                    hydrogen bond donors
  --edia                            only score bonds with a sufficient edia
                                    score in the molecule file or database
  --ediacutoff arg (=0.8)           minimum edia score to consider an atom
                                    well resolved
  --minhits arg (=0)                only consider matches with at least
                                    minhits entries in their statistic

Processing Arguments:
  --threads arg (=1)                number of threads to run analysis with
                                    (0 (auto),  1 (default), x (no.
                                    threads)
  --chunksize arg (=100)            number of molecules in a chunk
                                    processed by a single thread (default:
```

```
100)
```

The input can consist of either a set of molecules as an SD file or an already generated database of molecules from a previous run. The output is written into a specified directory and consist of at least 3 files: not_matched_patterns.txt, path_discrepancies.tsv, and results.tsv. The results.tsv contains all matching results against all molecules. The path_discrepancies.tsv file describes the phenomenon that molecules may match torsion rules in different torsion classes. It is expected that a torsion may match a rule in a specific torsion class and in the generic torsion class because the generic torsion class contains many catch-all patterns. Any path discrepancies left after filtering out the generic torsion class matches may be an indication of poorly structured hierarchies. The not_matched_patterns.txt file contains a list of patterns that were not matched by the input. There are two optional outputs: a torsion library file updated with the input molecules and a curves.tsv that contains 360 point values for a von Mises based torsion potential derived from the peaks. The torsion library output was used to generate new Torsion Library versions. The von Mises based torsion potential has been used in several tools[41], [66] but has never been evaluated individually, so its future is unclear.

Besides the input and the output option all other flags are optional and have set defaults. A few of the options deserve some more discussion. The check hierarchy and sort patterns options activate the SMARTScompare based hierarchy checks and the pattern sorting. The selectivity option decides on the matching procedure to be used. The default "single most selective" is selective matching. The setting "all" is the unselective matching. "all most selective" refers to a Phenomenon where a rotatable bond may match the same torsion rule with different :1 and :4 atoms. The torsion rule will be the same as in selective matching, but the :1 and :4 atoms will be different, which may lead to different torsion angles and maybe even to different classifications. "–terminalatoms", "–fixconjugated", and "–terminaldonors" augment what is considered a rotatable bond to include bonds to terminal atoms in torsion rules, to exclude bonds in conjugated substructures or to include bonds to terminal hydrogen bond donors such as amines and hydroxy groups.

Evaluation Scripts

A collection of python evaluation scripts were written or re-written in the course of this project. These can be found in the NAOMI reproducibility folder. The following is an

overview:

- bin_coverage.py: generate statistics about false positives (defined in Section 2.4)
- match_diffs.py: generate match differences between torsion libraries
- mols_for_pattern.py: extract molecules that match a pattern
- multimatches.py: find equivalent matches with quality differences
- plot_torsionlib_patterns.py: plot the histograms of all patterns
- torsionlib_alerts.py: generate and plot strained statistics (such as Figure 2.3)

B.2.2. TorsionAnalyzer

The command line interface of the TorsionAnalyzer is the following:

```
$>./TorsionAnalyzer --help

TorsionAnalyzer analyzes the rotatable bonds of a molecule conformation and
categorizes them into relaxed, tolerable, and strained. These labels are
defined by occurrence frequencies derived from small-molecule crystal
structures. These occurrence frequencies are taken either from the integrated
or a user defined torsion library.

Usage example:

./TorsionAnalyzer -i mols.sdf -o analysis.tsv -u 2


  Arguments:
  -h [ --help ]              Show usage
  -i [ --input ] arg         input molecule(s) to score
  -o [ --output ] arg        output TSV containing scores for every rotatable
                             bond of the input
  -t [ --torlib ] arg        torsion library to use as basis for quality
                             assesment (default: integrated torlib)
  --selectivity arg (=2)     selectivity of the torsion matching: single most
```

```
                              selective (2, default), all most selective (1),
                              all (0)
  --terminalatoms             score torsions containing terminal heavy atoms
  --fixconjugated             do not score torsions of conjugated bonds
  --terminaldonors            score torsions of terminal hydrogen bond donors
  --minhits arg (=0)          only consider matches with at least minhits
                              entries in their statistic
  -u [ --userinfo ] arg (=1) set user info level: quiet (0), warnings (1,
                              default), workflow (2)
  -l [ --license ] arg        provide a new license for the torsionanalyzer
```

The interface of the TorsionAnalyzer is intentionally similar to the interface of the TorsionPatternMiner. It also intentionally has fewer options. The input for a TorsionAnalyzer run is a molecule file and the output is a file containing the matching information for those molecules. One notable difference is that the default of the selectivity option is different. The TorsionAnalyzer is used to classify molecules, not to generate new Torsion Library versions. The selectivity is set to the selective matching equivalent by default. All other options are comparable to the TorsionPatternMiner.

The TorsionAnalyzer can also be deployed as a web application. Detailed information about how to set up the web application can be found in the README files of the backend and frontend repositories. The web application also comes with a built-in help page.

B.2.3. FastGrow

FastGrow can be built in two different command-line packages. One contains just a DatabaseBuilder and a FastGrow executable. The other also contains a Clipper, a Preprocessor, and an InteractionGenerator executable. The first two binaries are intended for end users. The other three binaries are used in the backend server.

The first step to using FastGrow on the command-line is to generate fragment databases to screen. The following is the command-line interface for the DatabaseBuilder:

```
$> ./DatabaseBuilder --help all
The DatabaseBuilder creates shape-based screening databases for the supplied
fragments. Conformations can be generated for the fragments using the
```

```
"--conformations" option and passing the maximum number of
conformations to generate. If "--conformations 0" is passed the input
conformations will be used. The shape descriptors are generated according to the
parameters defined using the options in the "Descriptor arguments" section.
The default parameters are those of the coarse width ranges descriptor variant.

Usage example:

./DatabaseBuilder --database screening_database.db --fragments fragments.smi

A fragment space file (FSF) can be used to generate a screening database of
fragments that follow the compatibility rules defined in the FSF. The names of
the linker types will be exactly as they appear in the FSF. These linker types
can then be used to screen for compatible fragments with FastGrow.

Usage example:

./DatabaseBuilder --database screening_database.db --fsfile fragspace.fsf

All arguments:

Basic arguments:
  -h [ --help ] arg       produce help message, use --help all for all
                          arguments
  -d [ --database ] arg   name and/or file path of the database
  -f [ --fragments ] arg  fragment file to be converted to database (.sdf,
                          .smi)
  --fsfile arg            fsfile to read fragments and linker compatibility
                          from
  --conformations arg     number of conformations to generate (default: 10,
                          pass 0 to use input conformations)
  --stereoisomers arg     enumerate stereoisomers 0 (none), 1 (for unassigned
                          stereo centers), 2 (all, default)
  --interactions arg      generate interactions for the fragments in the
                          database (default: true)
  -o [ --overwrite ]      overwrite existing database
```

```
  -u [ --userinfo ] arg   userinfo level: 0 (quiet), 1 (warnings, default), 2
                          (workflow)
  --license arg           license to activate FastGrow

Descriptor arguments:
  --depth arg             depth of the descriptor cylinder (default: 10Å)
  --width arg             width of the descriptor cylinder (default: 10Å)
  --height arg            how far the cylinder reaches back behind the anchor
                          atom (default: 2Å)
  --raysperdepth arg      rays per depth of descriptor (default: 18)
  --depthinterval arg     distance interval between depth points (default:
                          1.5Å)
  --binsize arg           size of bins along a ray (default: 0.75Å)
  --rvmtype arg           ray volume matrix type (WidthRanges(default): 1,
                          FirstIntersection: 2, IntegerRanges: 3)

Database arguments:
  --databasetype arg      postgres: 0 or sqlite(default): 1
  --username arg          user to access database
  --port arg              port to access database (default: 5432)
  --host arg              host to access database (default: localhost)
  --password arg          password to access database
```

Fragments can be in SMILES or SDF format. These fragments must already have attached linkers. In SMILES these linkers should be written as "[?R?]", which is the NAOMI convention for linker SMILES. In SDFs the linkers should be written as "RGP" records according to the SDF format specification[129]. Fragment databases can also be generated from some versions of FS files such as those by Degen et al.[109].

The user can decide what they want to do about conformations, stereoisomers, and interactions or simply take the defaults. The descriptor arguments are best described by the RVM paper[D1]. These should be consistent between the database building and the growing process. The database arguments interact with the database management system. This can either be SQLite or Postgres. Instead of passing any database passwords on the command-line, a user can alternatively set the "PGPASSWORD" environment variable, which is a little more secure.

Once a fragment database is built, it can be screened using the FastGrow executable, which has the following command-line interface:

```
FastGrow performs a screening of a database using a query consisting of a ligand
and a pocket. This ligand can already have an attached linker atom (defined by an
RGP record in an SDF) or a linker atom can be defined using the atoms' position
in an SDF. The shape descriptor type, parameterization, and weights can be found
in the "Descriptor Arguments" and "Scoring Arguments" sections of the
--help all output.

Usage example:

./FastGrow --pocket pocket.pdb --ligand ligand.sdf --database screening_database.db\
  --results results.sdf

For compatibility with other software silicon atoms are considered linker atoms.
If there is no linker atom in the molecule you can define an exit bond using the
position of two atoms in the file. To define a bond to be cut and a linker
placed both the --anchorposition and the --linkposition need to be defined. The
part of the molecule connected to the anchor anchor will be kept.

Usage example:

./FastGrow --pocket pocket.pdb --ligand ligand.sdf --database screening_database.db\
  --results results.sdf --anchorposition 5 --linkposition 6\
  # implies a bond between atom 5 and 6 in the SDF

To use the chemical compatibility rules of a fragment space in a screening
database you can define the --linktype of the linker atom in the input molecule.
Make sure the linker type is one that is present in your database. By default
all fragments in the database will be screened.

Usage example:

./FastGrow --pocket pocket.pdb --ligand ligand.sdf --database screening_database.db\
  --linktype R3 --results results.sdf  # implies that R3 and linkers compatible to R3\
```

```
  are present in the database

Arguments:

Basic arguments:
  -h [ --help ] arg          produce help message, use --help all for all
                             arguments
  -d [ --database ] arg      name and/or file path of the database
  -p [ --pocket ] arg        path to the PDB file of the pocket
  -e [ --ensemble ] arg      directory containing an ensemble of proteins or a
                             list of protein files to screen
  -l [ --ligand ] arg        path to the SDF file of the ligand/core to grow
                             from
  -i [ --interactions ] arg  use interactions in file as interaction
                             constraints
  -w [ --widthtol ] arg      width (clash) tolerance for rvm comparison
                             (default: 2.0Å)
  -r [ --results ] arg       output file to write results
  --maxresults arg           maxmimum number of results to write (default:
                             1000)
  -u [ --userinfo ] arg      userinfo level: 0 (quiet), 1 (default: warnings),
                             2 (workflow), 3 (debug)
  --license arg              license to activate FastGrow

Linker arguments:
  -t [ --linktype ] arg      linkertype to screen with (default: "R")
  --anchorposition arg       in file ID of the anchor
  --linkposition arg         in file ID of the linker

Descriptor arguments:
  --depth arg                depth of the descriptor cylinder (default: 10Å)
  --width arg                width of the descriptor cylinder (default: 10Å)
  --height arg               how far the cylinder reaches back behind the
                             anchor atom (default: 2Å)
  --raysperdepth arg         rays per depth of descriptor (default: 18)
  --depthinterval arg        distance interval between depth points (default:
```

```
                          1.5Å)
  --binsize arg           size of bins along a ray (default: 0.75Å)
  --rvmtype arg           ray volume matrix type (WidthRanges(default): 1,
                          FirstIntersection: 2, IntegerRanges: 3)

Database arguments:
  --databasetype arg      postgres: 0 or sqlite(default): 1
  --username arg          user to access database
  --port arg              port to access database (default: 5432)
  --host arg              host to access database (default: localhost)
  --password arg          password to access database

Pose Scoring arguments:
  --fillvolumescore arg   weight for filled volume (default: 1)
  --closecontactscore arg weight for close contacts (default: 5)
  --clashscore arg        weight for clash (default: -20)
  --closecontactcutoff arg distance cut off for close contacts (default:
                          5.0Å)

Interaction arguments:
  --distancecutoff arg    absolute cutoff distance for interactions
  --anglecutoff arg       absolute angle deviation cutoff for directed
                          interactions
  --maxrmsd arg           absolute RMSD to allow to the input interactions
  --greedy                only perform greedy interaction matches
  --protoss               run protoss during scoring

Postprocessing arguments:
  --torsioncheck arg      perform torsion check on hits (default: false)
  --uniquecheck arg       perform uniqueness check on hits (default: true)
  --optimization arg      perform optimization of hits (no: 0 (default),
                          restrained: 1, full: 2)
  --resttol arg           distanve tolerance before atom is restrained
  --noranking             do not perform hit ranking, only pose scoring

Output arguments:
```

```
--writemode arg          write mode: multi threaded (0, default), chunked
                         (1), linear (2)
--chunksize arg          number of mols in a chunk when writing chunked
                         output (default: 100)
--poses arg              maximum number of poses to generate for each hit
                         (default: 1)
--threads arg            number of threads to use (default: all cpu cores -
                         1)
```

The input to a growing is the ligand to grow from, the pocket to grow in, and the fragment database. The pocket is a protein structure and is usually in the PDB format. The fragment database should be generated by the DatabaseBuilder. The ligand to grow from may already contain a linker. Alternatively, a bond to cut can be specified on the command-line using the "anchorposition" and "linkposition" options. The type of the linker can also be changed on the command-line with the "linktype" option. The results are written in the SDF format.

The query can be augmented using several options. The "interactions" option accepts a path to a JSON file that contains pharmacophoric constraints. The format of these constraints is demonstrated in unit tests, validation, and the web server. The "widthtol" option sets the clash tolerance of the shape comparison. Pose scoring arguments, interaction arguments, and postprocessing arguments influence various parts of the FastGrow workflow. The output arguments may be relevant for validation and web server deployment. Results can be written out in a multi-threaded, chunked, and linear way. The default multi-threaded approach is for end users and will produce output sorted by score. Linear output writes out all poses as soon as they are scored, thereby minimizing memory usage but not producing a sorted output file. Chunked write-out is used in the web server deployment to quickly transfer chunks of hits to the end user of the web application.

The three other binaries all fulfill specific roles in the web application. The Preprocessor executable normalizes molecule input uploaded to the web application. The Clipper executable cuts bonds in a molecule to produce cores to grow from. The InteractionGenerator executable generates protein-ligand interactions to use in the growing.

The web application wraps all functionality of the command-line tools. Detailed information about how to set up the web application can be found in the README files

of the backend and frontend repositories. The architecture of the FastGrow web application is more complicated than that of the TorsionAnalyzer but follows the concept described in Section B.1.

FGVD Generation

The FGVD generation pipeline is located in the NAOMI reproducibility folder. It consists of a collection of scripts that orchestrate the generation of every data set. Each data set described in Chapter 3 usually has a script named after it that generates this data set. From that script onward the generation splits into various small tools and other scripts. FGVD generation was developed for use with the PDBbind refined set[31] and implicitly assumes its structure. Most of these assumptions are made in the orchestrating shell scripts. Occasionally a tool may also assume certain paths exist.

The first script in the pipeline is the preprocessing script. Preprocessing is necessary because some of the valence states of the SDF ligands in the PDBbind refined set are not localized. The easiest way to solve this is by converting the MOL2 ligand files, which are usually localized, to SDF. Preprocessing also substitutes the original PDB structures for the ones in the PDBbind distribution to achieve maximum consistency in the following steps. Lastly, it generates PDB files with empty binding sites, specifically empty binding sites defined around the PDBbind ligand for a system.

The self-growing script uses a C++ tool to cut random single-bonds in all ligands. This has to happen in one process to avoid any more duplicates in the ligand cutting process than necessary. The tool iterates over all ligands in the PDBbind refined set, reads the corresponding structure files in the course of the checks performed on the fragments, and outputs the performed cuts as well as a list of unique fragments as SMILES.

The cross-growing script first performs an all-against-all SIENA alignment of the members of the PDBbind refined set. SIENA is run with the parameters informally associated with the "Docking" configuration. An exhaustive description of the parameters can be found in the RVM paper[D1]. After the SIENA alignment, a C++ tool is iteratively called on the ensembles generated for every member of the PDBbind refined set. This tool searches for a common core in the ligands of the ensemble and performs all fragment checks. The growing steps are output in a directory next to the SIENA ensembles and gathered into a separate directory at the end.

All other data sets depend on the cross-growing set. Relevant interactions for a cross-growing test case are generated when the cross-growing test case is generated. Water replacements are generated by a C++ tool that is called on all cross-growing test cases. Ensemble growing test cases are generated by first iterating over all members of the PDBbind refined set and performing a clustered all-against-all SIENA alignment with a maximum cluster size of five. The configuration is otherwise the same as in cross-growing generation. Cross-growing test cases for which clustered alignments of at least two PDBs excluding the reference could be generated then use these ensembles in ensemble growing validation.

All data sets are intended to be generated in the same directory that also contains the PDBbind refined set and all necessary tools. All interpretation of chemical information is done in C++ and specifically the NAOMI library[80] to ensure the chemical model stays consistent. Workflow handling is done in Bash and a few slightly more complicated tasks are performed in Python. The actual repository also contains a few scripts to run and evaluate validations. These are, however, very specific to the cluster setup at time of writing and will probably require modification. They are provided as inspiration.

C. Journal Articles

C.1. The Torsion Library: Semiautomated Improvement of Torsion Rules with SMARTScompare

[D2] P. Penner, W. Guba, R. Schmidt, A. Meyder, M. Stahl, and M. Rarey, "The Torsion Library: Semiautomated Improvement of Torsion Rules with SMARTScompare," *Journal of Chemical Information and Modeling*, Mar. 2022, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.2c00043. [Online]. Available: https://doi.org/10.1021/acs.jcim.2c00043. Reprinted with permission from [D2]. Copyright 2022 American Chemical Society.

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

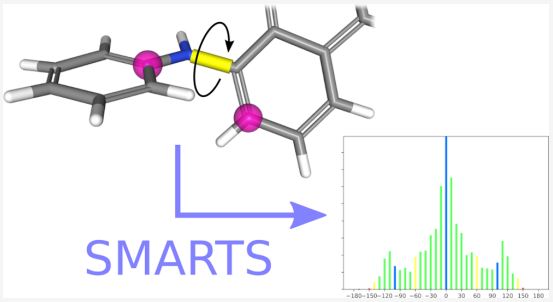
The Torsion Library: Semiautomated Improvement of Torsion Rules with SMARTScompare

Patrick Penner, Wolfgang Guba, Robert Schmidt, Agnes Meyder, Martin Stahl, and Matthias Rarey*

Cite This: J. Chem. Inf. Model. 2022, 62, 1644−1653 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: The Torsion Library is a collection of torsion motifs associated with angle distributions, derived from crystallographic databases. It is used in strain assessment, conformer generation, and geometry optimization. A hierarchical structure of expert curated SMARTS defines the chemical environments of rotatable bonds and associates these with preferred angles. SMARTS can be very complex and full of implications, which make them difficult to maintain manually. Recent developments in automatically comparing SMARTS patterns can be applied to the Torsion Library to ensure its correctness. We specifically discuss the implementation and the limits of such a procedure in the context of torsion motifs and show several examples of how the Torsion Library benefits from this. All automated changes are validated manually and then shown to have an effect on the angle distributions by correcting matching behavior. The corrected Torsion Library itself is available including both PDB as well as CSD histograms in the Supporting Information and can be used to evaluate rotatable bonds at https://torsions.zbh.uni-hamburg.de.



INTRODUCTION

Molecular geometry is described internally by bond lengths, bond angles, and torsion angles. Although bond lengths and bond angles can be parametrized precisely enough for medicinal chemistry applications,[1,2] the extra complexity added by 4-point dihedral angles, resulting in multiple local minima, remains challenging. Conformational strain, which is usually introduced by torsion angles, is vitally important to drug design and may be a decisive factor in bioactivity. Some conformational strain can be compensated by strong intermolecular interactions between a protein and a potential ligand, but in general strain should be avoided.[3]

Conformational strain can be measured ab initio using high level quantum chemical methods, but due to their runtime various empirical methods such as DFT, force-fields, or knowledge-based workflows are used more frequently in practice. Methods closer to ab initio in their level of theory, such as DFT, have the advantage that they specifically consider the molecule at hand. Furthermore, they are able to compare the importance of strain at different torsions in the molecule. Knowledge-based workflows that preprocess large crystallographic databases have the advantage of being very fast at detecting strained torsion angles and comparisons to ab initio methods have shown them to be reliable.[4−6] The speed of these workflows is absolutely necessary in applications such as high-throughput conformer generation[7,8] and various forms of scoring.[9,10] The precision of knowledge-based workflows is, however, highly dependent on the quality of the data used to build them.

The Torsion Library is a knowledge base created by a workflow that uses databases of crystal conformations to generate torsion angles statistics for chemical environments encoded as SMARTS.[11] The basic idea of extracting torsion statistics from the crystal structure database can be dated back to Klebe and Mietzner in 1994, where it was introduced in the context of the conformer generator MIMUMBA.[12] In 2013, Schärfer et al. created the Torsion Library based on this idea and a SMARTS hierarchy to describe progressively more specific torsion motifs.[13] The last major update of the Torsion Library was in 2016, which focused on improving the torsion angle statistics.[14]

In this work, we will perform a general update of the Torsion Library with a focus on improving the SMARTS hierarchy using the SMARTScompare algorithm.[15,16] This is intended as a conservative effort to improve the Torsion Library patterns, without obscuring their chemical environment, and with respect to their statistical impact on torsion angle frequencies.

Received: January 14, 2022
Published: March 23, 2022

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

To that end, both the PDB and the CSD data set have been updated with new data released since the original publication in 2013.[13] All automated changes have been manually evaluated to ensure both the correctness of the method, as well as the correctness of the Torsion Library itself. In an effort to make the Torsion Library more accessible, we have collected all infrastructure functionality in a tool called the TorsionPatternMiner and rewritten the application oriented TorsionAnalyzer, which is now also available as an easy-to-use web service.

METHODS

The Torsion Library. The Torsion Library[13,14] consists of a hierarchical structure of SMARTS patterns[11] that encode the torsion motifs or torsion rules. Figure 1 illustrates a molecule's path through this hierarchy. At the top of the hierarchy are six torsion classes, which only define the elements of the two atoms incident to a rotatable bond. In Figure 1 we see the input torsion being assigned to the "Carbon−Nitrogen" torsion class because the two elements that make up the rotatable bond are a carbon and a nitrogen. Below these are the first torsion rules and torsion subclasses, which encode recognizable substructures important to medicinal chemistry. In Figure 1 the "Carbon−Nitrogen" torsion class does not contain any torsion rules and so the torsion traverses into the "Anilines" subclass. Torsion rules on the same hierarchical level, for example just below the "Carbon−Nitrogen" torsion class, would be matched before any of the torsion subclasses. Torsion subclasses may contain even more specific subclasses and further torsion rules. Both subclasses and torsion rules on the same level are sorted from specific to generic chemical environments.

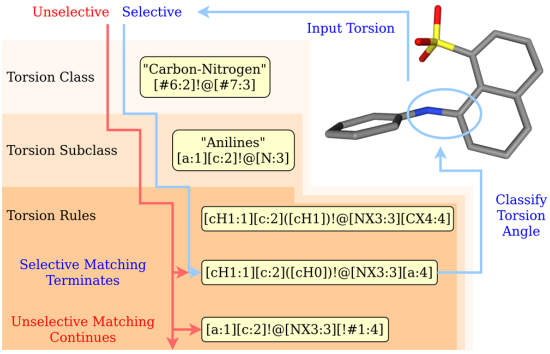


Figure 1. Matching behavior of a torsion in the Torsion Library. The molecule is 8-anilino-1-naphthalenesulfonate (2AN) from the PDB file: 2ANS. All matching procedures pass through the SMARTS hierarchy. Selective matching terminates at the first matching torsion rule, whereas unselective matching continues. Subclasses are subsets of the torsion class or subclass above them in the hierarchy. Torsion subclasses and torsion rules on the same level are sorted from specific to generic. All 3D molecular images were made with the NGL viewer.[17]

In Figure 1 we can see three torsion rules in the "Anilines" subclass. The SMARTS pattern of a torsion rule matches the chemical environment of a torsion and also labels the four torsion atoms. Labels are assigned using SMARTS syntax by labeling the atoms 1 to 4 of a torsion as :1 to :4. These atoms are used to measure the dihedral angle. Each torsion rule has a list of preferred angles, as well as a histogram of how frequently the matching of a torsion rule against a database of crystal structures led to a specific dihedral angle from −180° to 180° sorted into 10° bins. Visualizations of the histograms will be generated such that each 10° bin has its center at the angle it is labeled with and a range of ±5°, for example 30° ± 5°. The entry at 180° will be mirrored for −180°. Preferred angles are manually fitted according to the histogram. The angle is set to a discernible peak in the histogram and two tolerances are defined based on the shape of the peak.[13] Using the preferred angles of a torsion rule, an input torsion can be classified according to a traffic light scheme to show strained torsions as red, tolerable torsions as yellow, and relaxed torsions as green. Torsion angles are labeled as relaxed if they correspond to or are within the first tolerance of a preferred angle. Torsion angles that fall into the second tolerance of a preferred angle are marked as tolerable. If an angle does not fall into any tolerance of any preferred angle, then it is labeled as strained.

It is important to note a difference in how the histograms are filled and how rotatable bonds are classified. The torsion angle of a rotatable bond is classified by traversing down the hierarchy to the most specific torsion rule. In Figure 1 the input torsion passes into the "Carbon−Nitrogen" torsion class and then into the "Anilines" subclass. By ensuring that the subclasses are always in a subset relationship with each other and that the torsion rules are sorted from specific to generic, we know that the first torsion rule SMARTS pattern which a rotatable bond matches is the most specific torsion rule it can match in the entire Torsion Library. This torsion rule can then most specifically describe the likelihood of a dihedral angle for the chemical environment of that rotatable bond. In Figure 1 the first torsion rule SMARTS pattern of the "Anilines" subclass cannot match the input torsion. The second torsion rule does match and so the search procedure terminates. This is the matching behavior most important to a user or compound designer who wishes to classify the quality of the torsion angles in a molecule and will be referred to as *selective matching*.

When the histograms of the Torsion library are filled, all patterns regardless of their position in the hierarchy are matched against all rotatable bonds of the input structures. This will be referred to as *unselective matching*. In Figure 1 the red unselective matching hits the third and very generic torsion rule in the "Anilines" subclass, as well as the second one. It would then continue matching other torsion rules in all other torsion classes and subclasses. Unselective matching is used to fill the histograms so that as many statistically significant torsion angle distributions as possible can be generated, which are then analyzed by an expert.[13]

Both the SMARTS hierarchy and the preferred angle lists are manually generated. Figure 2 gives an overview over how different parts of the Torsion Library ecosystem interact. At the top are the different sources of input in blue, one of these inputs being the expert curation. The Torsion Library is an expert-driven and computationally supported system, which is one of the aspects that distinguishes it from other collections of torsion angle statistics. Every SMARTS pattern is written with a meaningful chemical environment in mind. The preferred angles are based on the histograms of dihedral angle frequencies and written according to expert knowledge as well as careful consideration of confounding factors the statistic may be subject to.

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

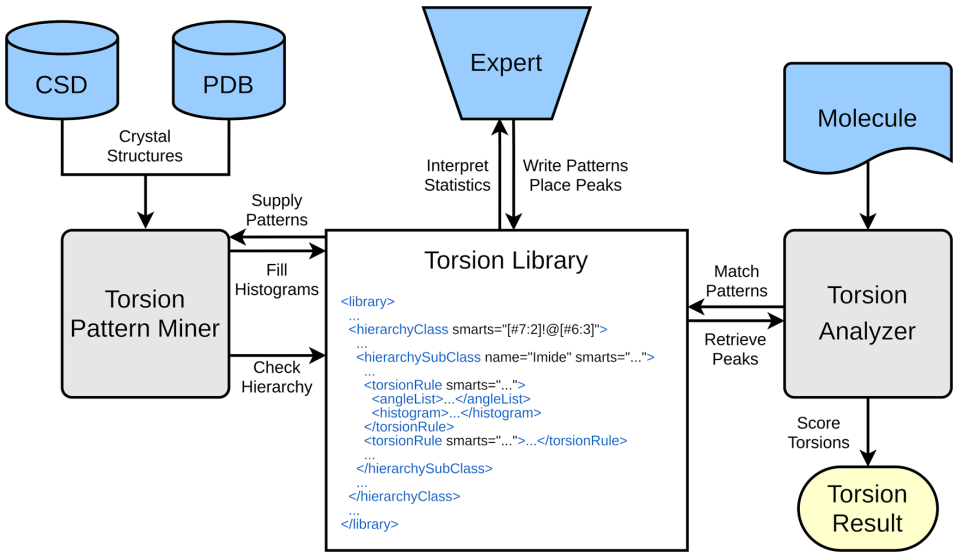


Figure 2. Overview of the Torsion Library ecosystem. All inputs, the crystallographic databases, the symbolic expert, as well as a molecule to classify, are in blue. The two tools that interact with the Torsion Library are in gray. The TorsionPatternMiner is used to process the crystallographic data, as well as check the consistency of the SMARTS hierarchy. The TorsionAnalyzer is the application oriented tool, which uses the Torsion Library to classify molecules. A part of the Torsion Library XML demonstrating the file contents has been placed in the center.

Manually written SMARTS, however, come with their own challenges. SMARTS can be difficult to read and even experts struggle to fully grasp all of their implications. Maintaining the subset relationships between torsion classes and sorting torsion rules by specificity is therefore far from trivial. It is at this point that the SMARTScompare algorithm[15,16] comes into play. SMARTScompare can automatically detect whether SMARTS are in a subset relationship by enumerating what chemical states these SMARTS can match. If the set of chemical states one SMARTS can match is a subset of the set of chemical states the other SMARTS can match, then the SMARTS can be considered to be in a subset relationship. The detection of these relationships can be extended to work in the Torsion Library.

Extensions to SMARTScompare. The Torsion Library Hierarchy is built such that each SMARTS pattern of a torsion rule or subclass is in a subset relationship with all subclasses and the torsion class above it. Furthermore, all elements on the same hierarchy level have to be ordered from specific to generic, which can also be modeled as a subset relationship. All of these subset relationships can be tested with SMARTScompare.[15,16]

Torsion Library patterns use the SMARTS labels :1 to :4 to define which matched atoms will be used to measure the torsion angle. When comparing these labeled SMARTS, it is therefore very important to ensure nodes with the same label are compared to each other. Otherwise patterns with similar chemistry but describing very different torsion angles may incorrectly be considered as in a subset relationship or even equal. To prevent this, all labeled atoms are mapped before the actual maximum common substructure search performed by SMARTScompare. This had not been implemented as of the original publications[15,16] and was added in the course of this work.

Not all Torsion Library SMARTS patterns contain all labels. The top-level torsion classes that only describe the elements of the two atoms making up the rotatable bond can only contain the labels :2 and :3. The torsion subclasses may contain all labels, but many do not. This is often due to the fact that the :1 and :4 nodes can describe very different chemistry connected to the core chemical substructure of the subclass pattern, such as sulfonamides bound to either rings or aliphatic substituents. In these cases, all labels that are present in both patterns are mapped and all others ignored. Torsion angles are the same when measured from atoms :1 to :4 or from :4 to :1, and several patterns have been written in one way or the other. Therefore, labels are also mapped and matched both ways.

Checking the Hierarchy. SMARTS hierarchy checks were performed by recursively traversing the entire Torsion Library and checking that all SMARTS of torsion rules or subclasses below the current class or subclass SMARTS are in a subset relationship with that SMARTS. Labeled subset relationships can be calculated with our extension of the SMARTScompare algorithm and any inconsistencies in the hierarchy are returned as warnings. Corrections can then be performed by the expert, who understands the chemical environment the pattern is supposed to encode.

Following this, torsion rules were compared to subclasses on the same level to find subset relationships. If a torsion rule was a subset of a subclass, then this implied that the torsion rule encoded a more specific version of the subclass' pattern and should be in this subclass. Torsion rules could then be moved after careful consideration.

Torsion rules and subclasses on the same level are sorted from most specific to most generic. This relationship can also be described as a subset relationship and so all patterns were sorted accordingly. If a pattern was in a subset relationship with another pattern, then it was sorted before that superset pattern in the Torsion Library. All of these sorting operations

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

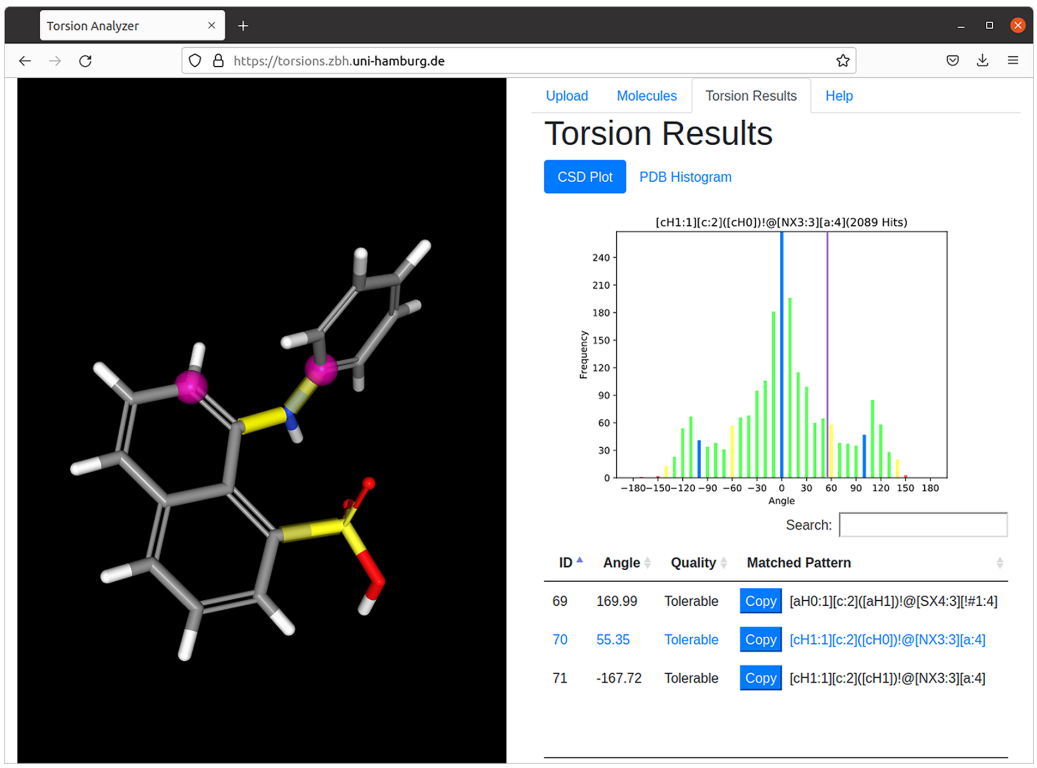


Figure 3. Screenshot of the Torsion Analyzer web application. The molecule loaded is 2AN from PDB code: 2ANS. The left-hand side shows the molecule and its rotatable bonds, which are colored according to the traffic light scheme. Purple spheres mark the :1 and :4 atoms. The right-hand side provides information about the matched torsion rule, such as its torsion angle distribution in the CSD.

were subsequently verified manually to ensure the algorithm worked as intended.

CSD and PDB Data. New crystallographic data sets were constructed using slightly modified protocols taken from the original Torsion Library publication.[13] One significant change was moving away from a proprietary subset of the PDB[18] in favor of automatically extracting high-quality ligands directly from the PDB.[19] The StructureProfiler[20] was used to perform common PDB structure quality filtering and the EDIA tool[21] employed to estimate the electron density fit of torsion atoms. CSD molecules were extracted using the CSD Python API[22] and closely following the rules defined by the original publication.[13] Detailed parameters and filters can be found in Section S2 of the Supporting Information (SI).

Torsion Tools. Torsion angle statistics were mined from the structure data using the new TorsionPatternMiner. The TorsionPatternMiner is a commandline application that performs all statistical and hierarchical analysis, including the application of the SMARTScompare algorithm, to create a revised Torsion Library. It is intended as an expert tool for individuals seeking to modify the Torsion Library with their own patterns or data. Its sibling tool, the TorsionAnalyzer, is the application-oriented tool to classify torsion angles of input conformations. The TorsionAnalyzer was reimplemented as a standalone commandline application. Both tools are mainly implemented using the Naomi C++ library.[23]

The TorsionAnalyzer Web server and local GUI (Graphical User Interface) application are both wrappers around the TorsionAnalyzer commandline application. An overview of the interface can be seen in Figure 3. The web version is based on the PVP (Python, Vanilla Javascript, Postgres) stack. The local GUI application is based on Electron.js,[24] which allows a developer to deploy a web-based frontend as a desktop application, with a Node.js environment emulating a backend server. The local application is therefore completely independent from the Web server and the Internet in general.

RESULTS

CSD and PDB Data. In general, updating the CSD and PDB data led to more data support of the existing patterns. The angle distributions have remained very stable, despite the fact that the new CSD data set is more than double the size of the original one created in 2013. This is discussed in more detail in Section S3 of the SI. The original data set contained around 140 000 molecules, whereas the new data set contains around 360 000, which represents almost a decade of growth of the CSD.[13] The new PDB data set is, however, smaller than the previous one. The previous data set was reported to contain 77 065 ligands.[18] The current PDB data set contains a little under 20 000 ligands. In part, this may be due to an overly strict filtering of the StructureProfiler. It may however also be due to a more permissive manual screening procedure that produced the original data set. It is in the spirit of this publication that we have replaced a manual, repetitive, and possibly error prone procedure with an automated one. It has remained problematic to base torsion statistics on the PDB

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

alone, due to the comparatively few highly resolved and electron density supported ligands in the PDB.[13] The main data set therefore continues to be the CSD data set.

Checking the Hierarchy. As of the 2016 publication,[14] the Torsion Library contained 552 SMARTS patterns. Automatically checking the Torsion Library hierarchy revealed 291 patterns that were not in a subset relationship with the pattern immediately above them in the hierarchy. The most common, as well as the most trivial, incongruency between parent and child patterns was a missing "!@" (nonring bond) specifier in various subclass patterns. All patterns in the Torsion library are supposed to describe acyclic bonds, so "!@" was added everywhere it was missing.[13] This never had a functional impact, seeing as all torsion rules did include the "!@" specifier.

A more impactful example of a hierarchy incongruency can be found in Figure 4. A torsion rule in the benzamidine

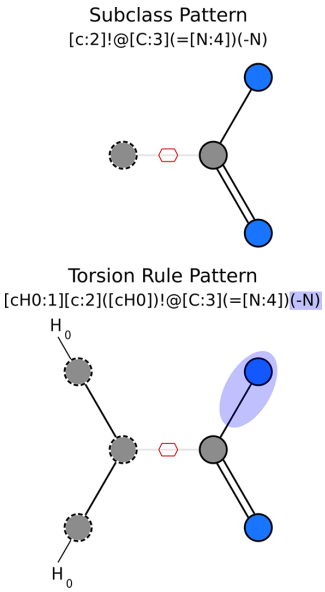


Figure 4. Pattern of the benzamidine subclass and the pattern of a torsion rule in it. The parts of the torsion rule pattern underlaid in blue were missing before hierarchy checking. This violated the subset relationship of subclass and torsion rule, which was detected by a hierarchy check. The pattern was subsequently corrected. A full legend for all pattern visualizations can be found in Figure S1 of the SI.

subclass was missing the part underlaid in blue, leaving an incomplete amidine. The difference in matching is visualized with two molecules in Figure 5, one of which does not contain a benzamidine substructure. The unselective matching, which does not match the subclass pattern beforehand, therefore produced a severely distorted angle distribution. Figure 6 shows two histograms of CSD torsion angles unselectively matching this benzamidine torsion rule before and after fully writing out the amidine part. Before the benzamidine correction, this pattern would match any double bond to a nitrogen one carbon away from the specified aromatic structure, such as a hydrazine or simply a nitrogen in a conjugated chain. The far rarer benzamidine angles were thus overshadowed. This gave the expert a very distorted image for

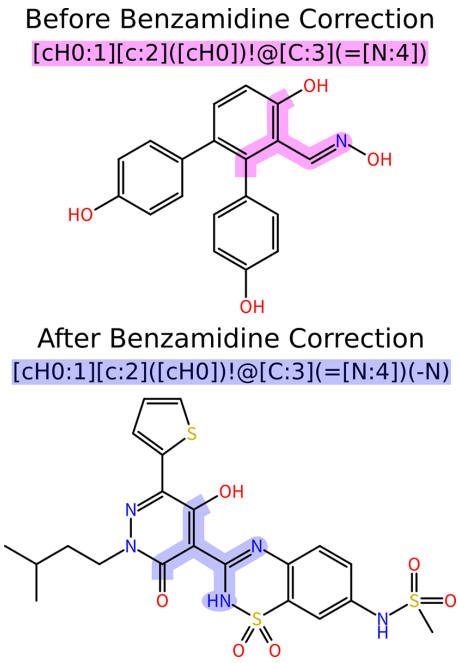


Figure 5. Matching behavior of the benzamidine pattern before and after correction on two molecules. The top molecule is 7EG from PDB code: 5TLG. The bottom molecule is N3H from PDB code: 3CDE. Atoms matched by each pattern are underlaid in the corresponding color.

this pattern and led to incorrect preferred angles, which in turn influenced the actual classification of molecular torsions.

Specifically, the pattern in question is of an amidine bound to a bis-ortho substituted aromatic ring. The bis-ortho substitution is encoded implicitly by forbidding hydrogens at the ortho positions relative to the amidine. The bis-ortho substitution intuitively supports the torsion angle distribution after the correction in Figure 6, because the substituents force the amidine into a torsion angle around ±90°.

It is important to note that the benzamidine correction also led to far fewer matches, which amplified the noise. The spurious peaks at −30° and 150° in the angle distribution after the benzamidine correction are mostly made up of one series of dithiadiazole compounds. Nonetheless, the histogram after benzamidine correction in Figure 6 is clearly more representative of bis-ortho substituted benzamidines.

A few similar cases were encountered and corrected throughout the Torsion Library, the benzamidine example being one of the more human readable ones. In general, pattern corrections were kept as conservative as possible. As the benzamidine example demonstrates, small changes to a torsion pattern can lead to significant differences in the torsion angle distribution. Checking the hierarchy takes 1.6 s on a single core (machine specifications can be found in Table S1 of the SI), so the hierarchy was checked regularly in this process. All changes can be found in Section S5 of the SI, which gives an overview of how and where specific changes are recorded.

Sorting Torsion Rules into Subclasses. After having corrected the hierarchical relationships, 21 of the 514 total torsion rules were sorted into a lower subclass. Three

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

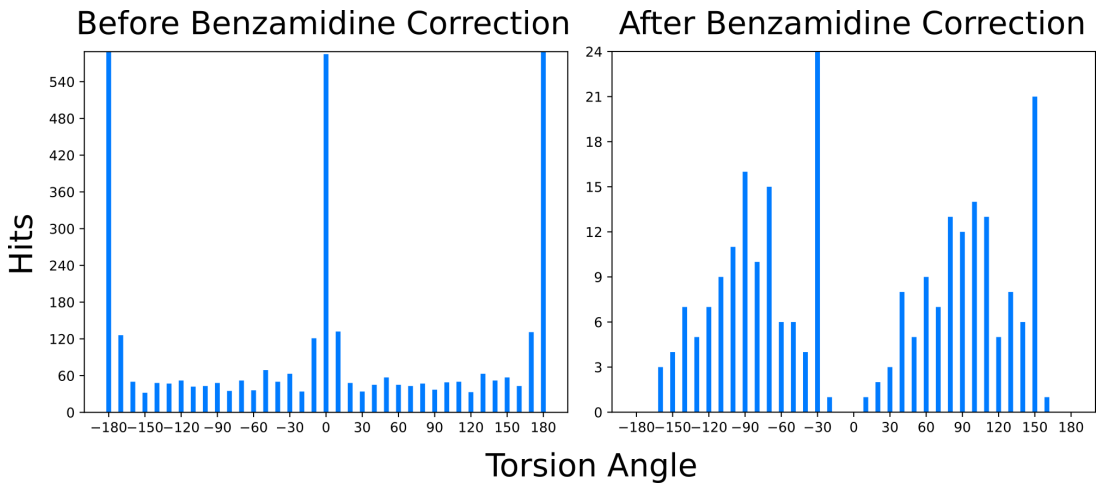


Figure 6. Distribution of torsion angles of molecules in the CSD for the torsion rule pattern in Figure 4 before and after it was corrected. "Hits" refers to how many torsion angles hit a specific bin. The torsion angle bin around 180° is duplicated for −180°. The histogram contained 3088 hits before correction and 256 hits after correction.

subclasses were involved in this step: "aro-aro", "Aliphatic Amides", and "[CX3][CX4]". Subclass "aro-aro" is in the generic top-level torsion class and describes acyclic bonds between two aromatic systems. "Aliphatic Amides" deals with bonds between carbons and an amide nitrogen. Figure 7 is an example of a pattern sorted into "[CX3][CX4]", which is a subclass that handles bonds between aliphatic sp^2 hybridized carbons and sp^3 carbons.

The torsion rule found in Figure 7 can be sorted into the subclass, because the :3 node, although not explicitly set to "X4" (four valences), implicitly has four bonding partners: two hydrogens, a bond to the :2 node and a bond to the :4 node. Because this is an implicit condition, it can be easily overlooked. In fact, 11 of the 21 torsion rules sorted into subclasses were such cases. Figure 8 visualizes the matching of both the subclass and the torsion rule. The :3 node is in both cases satisfied by a carbon with two bonds to hydrogens and two bonds to other carbons.

The SMARTScompare subset matching is capable of detecting implicit subset conditions, but does have limitations. Schmidt et al. discuss these limitations in some depth.[15] Some examples of problematic patterns are generically defined tautomeric structures and complex recursive SMARTS. Especially tautomerism is still a challenge in the Torsion Library.

In the case of Figure 7 SMARTScompare significantly corrects the Torsion Library's matching behavior. Patterns on the same level as a subclass that they can be sorted into, are matched before that subclass. This means a pattern may stop the selective matching process from recursing into a subclass and prevent any torsion rules in it from being matched. Selective matching behavior directly influences how well the Torsion Library classifies a users' input molecules. Correct sorting of a pattern into a subclass is also a prerequisite for sorting patterns by specificity within subclasses. All of this is now handled by an automated and fast check instead of painstaking manual SMARTS comparisons.

Sorting Torsion Rules by Specificity. Sorting torsion rules by specificity yielded 17 position changes of 14 torsion

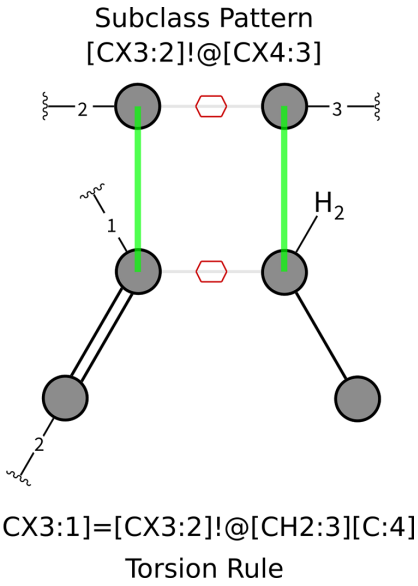


Figure 7. A torsion rule that can be sorted into a subclass. The green lines connect the nodes that are equivalent or more specific to their subclass pattern counterparts. Although the node labeled :3 in the torsion rule is not explicitly set to "X4" (four valences), the "H2" (two bonds to hydrogens) as well as its other bonds implicitly make it more specific than the "[CX4:3]" node in the subclass pattern. A full legend for all pattern visualizations can be found in Figure S1 of the SI.

rules and 3 subclasses. The sorting algorithm had to perform a few thousand SMARTS comparisons and did so in 2.1 s on a single core (machine specifications can be found in Table S1 of the SI).

Figure 9 shows an example of two patterns being sorted against each other. The top pattern is more specific than the lower pattern, or in other words follows a subset relationship, due to the elements underlaid in blue. It specifies more parts of

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

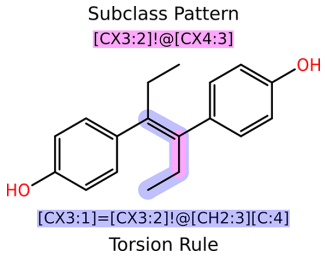


Figure 8. Visualized matching of a subclass and a torsion rule that can be sorted into that subclass. The molecule is DES from PDB code: 1S9P. Atoms matched by each pattern are underlaid in the corresponding color. The torsion rule matches the same atoms as the subclass pattern as well as a larger and more specific substructure. Although the :3 node is written differently both patterns match their :3 node to a carbon with two bonds to hydrogens and two bonds to other carbons.

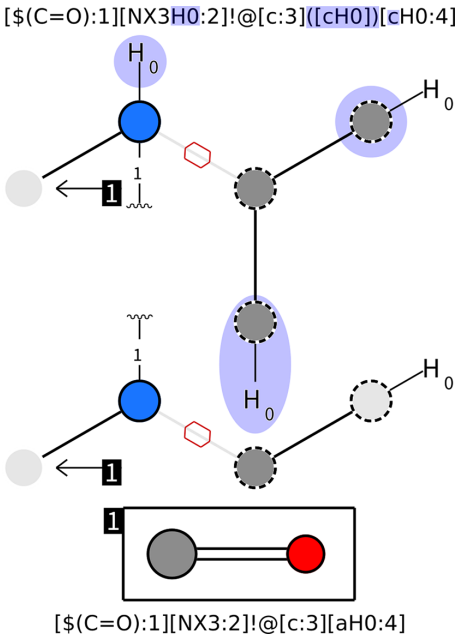


Figure 9. Comparison of two patterns performed during pattern sorting. Both patterns match amide nitrogens bound to aromatic structures. The top pattern also specifies that this aromatic structure is bis-ortho substituted to the amide. The SMARTS pattern elements, and their corresponding visualizations, that make the top pattern more specific than the bottom pattern are underlaid in blue. The top pattern will be sorted before the bottom pattern. A full legend for all pattern visualizations can be found in Figure S1 of the SI.

the chemical environment and is therefore more specific. Figure 10 visualizes both patterns matching a molecule. The more specific pattern matches a larger and more specific substructure of the molecule. In this case the SMARTS expressions are comparatively simple. Other sorting operations are performed on significantly longer and more complex patterns.

Figure 11 gives an example of three patterns being sorted before the pattern *a* highlighted in blue. Table 1 shows the

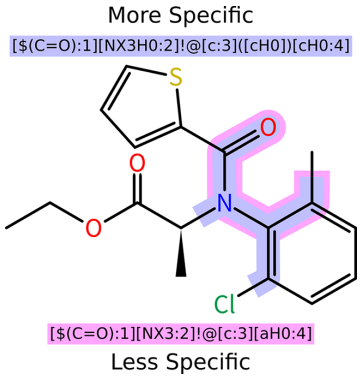


Figure 10. Visualization of two patterns to be sorted matching a molecule. The molecule is HJW from PDB code: 6Q6M. Atoms matched by each pattern are underlaid in the corresponding color. The more specific pattern matches a larger and more specific substructure of the molecule and must therefore be sorted before the less specific pattern.

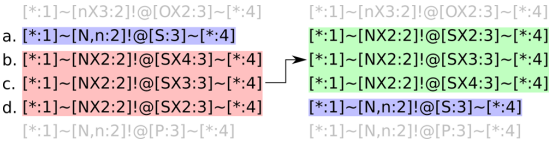


Figure 11. Pattern *a* is a superset of patterns *b*, *c*, and *d*. Patterns *b*, *c*, and *d* are therefore never matched. Sorting by subset relationships moves patterns *b*, *c*, and *d* from their incorrect positions in red to a correct position in green, before pattern *a* in blue. Table 1 shows the change in the number of selective matches these patterns received before and after sorting.

number of selective matches of CSD molecules these patterns received before and after sorting. Selective matching terminates at the first pattern it matches and is the main method to report the quality of torsion angles in a molecule to a user. In the case of the four patterns in Figure 11 selective matching would always terminate at pattern *a* for any torsion that could have hit patterns *b*, *c*, or *d*, because pattern *a* is a superset of all these patterns. All patterns *a* to *d* deal with a torsion between nitrogen and sulfur. Patterns *b*, *c*, and *d* specify how many bonds the sulfur has to other atoms, thereby distinguishing between sulfones, sulfoxides, and sulfides, respectively. Pattern *a* makes no distinction and will match all aliphatic sulfur oxidation states. For this reason, patterns *b*, *c*, and *d* had no matches before sorting in Table 1. Sorting these patterns by subset relationships leads to a distribution of the selective matches onto all of the patterns. This means instead of all torsions matching the more generic pattern *a*, many are now distributed to the more specific patterns *b*, *c*, and *d*.

Making subset relationships the only measure of specificity is limiting. A pure subset check results in a lot of patterns being in neither a subset nor a superset relationship to any other pattern, which is why only a handful of sorting operations were performed. Some relationships between SMARTS patterns not detected by SMARTScompare can however have an important effect.

Figure 12 gives an example of two SMARTS matching either substituted or unsubstituted positions ortho to the rotatable bond. "[cH0]" forbids hydrogen at an aromatic carbon, which

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

Table 1. Number of Selective Matches Four Patterns Received before and after Sorting by Subset Relationships in the CSD Dataset[a]

pattern	matches before sorting	matches after sorting
a. [*:1]~[N,n:2]!@[S:3]~[*:4]	2253	983
b. [*:1]~[NX2:2]!@[SX4:3]~[*:4]	0	1004
c. [*:1]~[NX2:2]!@[SX3:3]~[*:4]	0	135
d. [*:1]~[NX2:2]!@[SX2:3]~[*:4]	0	131

[a]Pattern *a* gets all possible matches before sorting. Patterns *b*, *c*, and *d* are sorted in front of pattern *a* and can then be matched. Pattern *a* is still matched, but now every pattern can address the specific chemical environment it encodes.

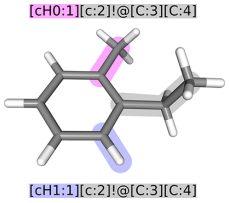


Figure 12. Two patterns including a position ortho to the torsion that cannot be correctly sorted. The pattern requiring ortho substitution can be seen as more impactful and specific for the rotatable bond. If substitution is written by forbidding hydrogens then patterns with or without substitution are mutually exclusive and cannot be detected as a subset relationship, which makes correct sorting impossible. All 3D molecule images were made with the NGL viewer[17]

implies substitution. "[cH1]" requires one hydrogen at an aromatic carbon, which implicitly forbids substitution. The patterns are not in a subset relationship because the number of hydrogens is mutually exclusive (not at least zero/one hydrogen, but exactly zero/one hydrogen). If we place the pattern without substitution before the other pattern in a selective matching, then any torsion with a single ortho substitution, such as the one in Figure 12, will match with its unsubstituted ortho position first and terminate the matching process. Ortho substitution can have a large impact on a torsions preferred angles and can be seen as the more specific chemistry. This means that by writing substitution in this way, more specific chemistry cannot be detected by a subset relationship and the patterns cannot be automatically sorted by specificity, leading to a poor description of the chemical environment of a rotatable bond. The Torsion Library solves this issue by specifying both ortho positions, thus explicitly forcing the matching in the substituted or unsubstituted direction. This approach does work but highlights the need for a consistent set of rules to write patterns that are both optimal for SMARTScompare as well as human readable.

CONCLUSIONS

The Torsion Library is a computationally supported but expert-driven system. Most manual processes execute their intention inconsistently. In fact, most computational processes tend to inconsistently execute a programmer's intention as well. By utilizing both we can immediately spot inconsistencies and so improve the correctness of the Torsion Library's hierarchy, as well as its matching behavior. The manual validation of automated hierarchy checking and pattern sorting has furthermore validated the process itself.

The Torsion Library is not the most accessible topic due to the inherent complexity of SMARTS patterns and the heavy focus on detail. To alleviate this, we have reworked large parts of the Torsion Library ecosystem. Statistical and hierarchical analysis is now performed by the TorsionPatternMiner, which is a purpose-built tool for experts to create and maintain the Torsion Library. The automation of SMARTS checking and sorting was able to correct several issues that would have been difficult to spot manually. User facing functionality, such as detecting strained bonds, is now handled in a significantly simplified TorsionAnalyzer, available as a command line tool, a local GUI application and a web application. With this we hope to lower the barriers to entry into the Torsion Library ecosystem.

Clearly, there are still limitations to both the SMARTScompare algorithm[15] and the Torsion Library. Pattern sorting can only follow a subset relationship as defined by SMARTScompare. This leads to many incomparable patterns that may nonetheless be in a subset relationship when considered on a semantic level. Semantic problems, such as the ortho substitution problem above, are the most difficult to detect due to the challenges of making SMARTS human readable. A canonical way to write the most human readable torsion SMARTS patterns could possibly avoid both the limitations of the SMARTScompare implementation, as well as shed light on semantic errors that may be overshadowed by the SMARTS syntax.

In some of the authors' work on scoring[9] the most problematic limitation of the Torsion Library was the fact that the relative energetic effect of one torsion rule compared to another is very difficult to elucidate. This is a general limitation of database derived statistical distributions of torsion angles, but some initial work has been done to facilitate such comparisons.[10] Drawing physical conclusions from torsion angle distributions is subject to many very complex biases arising from the nature of the data set as well as the patterns used to generate statistics. For example, crystal structures undersample high-energy conformations compared to room-temperature, gas-phase Boltzmann populations.[25] Any attempt to calculate Boltzmann distributions, and thereby relative energies, using crystal structure data will be skewed by this effect. Such complex phenomena require that an expert remains in the loop.

Database derived torsion libraries continue to be a pillar to most 3D operations in drug design. Their reliability and speed ensures this will continue to be the case for years to come. It is therefore imperative that they be of the highest quality possible.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.2c00043.

Legend for all SMARTS visualizations, discussion of more detail oriented topics, explanation of the files in the collection of changes (PDF)

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

Zipped version of the corrected Torsion Library XML file including both PDB and CSD histograms (ZIP)

Collection of files detailing exactly the changes to the Torsion Library and their effects on matching behavior (ZIP)

AUTHOR INFORMATION

Corresponding Author

Matthias Rarey − *Universität Hamburg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0002-9553-6531; Email: matthias.rarey@uni-hamburg.de

Authors

Patrick Penner − *Universität Hamburg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0003-4988-6183

Wolfgang Guba − *Roche Pharma Research & Early Development, Roche Innovation Center Basel, F. Hoffmann-La Roche Ltd., CH-4070 Basel, Switzerland*

Robert Schmidt − *Universität Hamburg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* Present Address: BioSolveIT GmbH, An der Ziegelei 79, 53757 Sankt Augustin, Germany; orcid.org/0000-0002-7089-4842

Agnes Meyder − *Universität Hamburg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* Present Address: Roche Pharma Research & Early Development, Roche Innovation Center Basel, F. Hoffmann-La Roche Ltd., CH-4070 Basel, Switzerland.; orcid.org/0000-0001-8519-5780

Martin Stahl − *Roche Pharma Research & Early Development, Roche Innovation Center Basel, F. Hoffmann-La Roche Ltd., CH-4070 Basel, Switzerland*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.2c00043

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that the TorsionPatternMiner or TorsionAnalyzer is licensed for a fee to non-academic institutions in the future. The TorsionPatternMiner and the TorsionAnalyzer are part of the NAOMI ChemBio Software Suite available at https://uhh.de/naomi. They are available for academic use without a fee and can be licensed for nonacademic use. The code for the TorsionAnalyzer web server and local GUI application are available open source at: https://github.com/rareylab/torsions. A freely accessible instance can be found at https://torsions.zbh.uni-hamburg.de. The Torsion Library in different versions is available at https://www.zbh.uni-hamburg.de/forschung/amd/datasets/torsion-library.html.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the Cambridge Crystallographic Data Center (CCDC) for their generous permission to publish data derived from the CSD. We highly appreciate the vision of CCDC to advance structural science and to foster the sharing of knowledge gained from analyzing crystal structures.

REFERENCES

(1) Gillespie, R. J.; Nyholm, R. S. Inorganic stereochemistry. *Chem. Soc. Rev.* **1957**, *11*, 339.

(2) Haynes, W. M., Lide, D. R., Bruno, T. J., Eds.; *CRC Handbook of Chemistry and Physics*,, 97th ed.; Taylor & Francis: Oxford, 2016.

(3) Liebeschuetz, J. W. The Good, the Bad, and the Twisted Revisited: An Analysis of Ligand Geometry in Highly Resolved Protein−Ligand X-ray Structures. *J. Med. Chem.* **2021**, *64*, 7533−7543.

(4) Hao, M.-H.; Haq, O.; Muegge, I. Torsion Angle Preference and Energetics of Small-Molecule Ligands Bound to Proteins. *J. Chem. Inf. Model.* **2007**, *47*, 2242−2252.

(5) Liebeschuetz, J.; Hennemann, J.; Olsson, T.; Groom, C. R. The good, the bad and the twisted: a survey of ligand geometry in protein crystal structures. *J. Comput.-Aided Mol. Des* **2012**, *26*, 169−183.

(6) Tong, J.; Zhao, S. Large-Scale Analysis of Bioactive Ligand Conformational Strain Energy by Ab Initio Calculation. *J. Chem. Inf. Model.* **2021**, *61*, 1180−1192.

(7) Hawkins, P. C. D.; Skillman, A. G.; Warren, G. L.; Ellingson, B. A.; Stahl, M. T. Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database. *J. Chem. Inf. Model.* **2010**, *50*, 572−584.

(8) Friedrich, N.-O.; Flachsenberg, F.; Meyder, A.; Sommer, K.; Kirchmair, J.; Rarey, M. Conformator: A Novel Method for the Generation of Conformer Ensembles. *J. Chem. Inf. Model.* **2019**, *59*, 731−742.

(9) Flachsenberg, F.; Meyder, A.; Sommer, K.; Penner, P.; Rarey, M. A Consistent Scheme for Gradient-Based Optimization of Protein−Ligand Poses. *J. Chem. Inf. Model.* **2020**, *60*, 6502−6522.

(10) Gu, S.; Smith, M. S.; Yang, Y.; Irwin, J. J.; Shoichet, B. K. Ligand Strain Energy in Large Library Docking. *J. Chem. Inf. Model.* **2021**, *61*, 4331−4341.

(11) Daylight Theory: SMARTS—A Language for Describing Molecular Patterns. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html, Date accessed: 18.08.2021.

(12) Klebe, G.; Mietzner, T. A fast and efficient method to generate biologically relevant conformations. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 583−606.

(13) Schärfer, C.; Schulz-Gasch, T.; Ehrlich, H.-C.; Guba, W.; Rarey, M.; Stahl, M. Torsion Angle Preferences in Druglike Chemical Space: A Comprehensive Guide. *J. Med. Chem.* **2013**, *56*, 2016−2028.

(14) Guba, W.; Meyder, A.; Rarey, M.; Hert, J. Torsion Library Reloaded: A New Version of Expert-Derived SMARTS Rules for Assessing Conformations of Small Molecules. *J. Chem. Inf. Model.* **2016**, *56*, 1−5.

(15) Schmidt, R.; Ehmki, E. S. R.; Ohm, F.; Ehrlich, H.-C.; Mashychev, A.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms. *J. Chem. Inf. Model.* **2019**, *59*, 2560−2571.

(16) Ehmki, E. S. R.; Schmidt, R.; Ohm, F.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis. *J. Chem. Inf. Model.* **2019**, *59*, 2572−2586.

(17) Rose, A. S.; Hildebrand, P. W. NGL Viewer: a web application for molecular visualization. *Nucleic Acids Res.* **2015**, *43*, W576−W579.

(18) Proasis2. http://www.desertsci.com, Date accessed: 29.09.2021.

(19) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res.* **2000**, *28*, 235−242.

(20) Meyder, A.; Flachsenberg, F.; Sieg, J.; Friedrich, N.-O.; Fährrolfes, R.; Kampen, S.; Rarey, M. StructureProfiler: an all-in-one tool for 3D protein structure profiling. *Bioinformatics* **2019**, *35*, 874−876.

(21) Meyder, A.; Nittinger, E.; Lange, G.; Klein, R.; Rarey, M. Estimating Electron Density Support for Individual Atoms and Molecular Fragments in X-ray Structures. *J. Chem. Inf. Model.* **2017**, *57*, 2437−2447.

(22) Groom, C. R.; Bruno, I. J.; Lightfoot, M. P.; Ward, S. C. The Cambridge Structural Database. *Acta Crystallogr. B* **2016**, *72*, 171−179.
(23) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the Almost Trivial Task of Reading Molecules from Different File formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(24) Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS. https://www.electronjs.org, Date accessed: 28.09.2021.
(25) Allen, F. H.; Harris, S. E.; Taylor, R. Comparison of conformer distributions in the crystalline state with conformational energies calculated by ab initio techniques. *J. Comput.-Aided Mol. Des.* **1996**, *10*, 247−254.

https://doi.org/10.1021/acs.jcim.2c00043
J. Chem. Inf. Model. 2022, 62, 1644−1653

C.2. Shape-Based Descriptors for Efficient Structure-Based Fragment Growing

[D1] P. Penner, V. Martiny, A. Gohier, M. Gastreich, P. Ducrot, D. Brown, and M. Rarey, "Shape-Based Descriptors for Efficient Structure-Based Fragment Growing," *J. Chem. Inf. Model.*, vol. 60, pp. 6269–6281, 12 2020, PMID: 33196169, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00920. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00920.

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

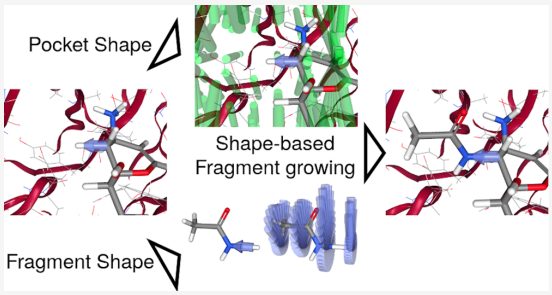
Shape-Based Descriptors for Efficient Structure-Based Fragment Growing

Patrick Penner, Virginie Martiny, Arnaud Gohier, Marcus Gastreich, Pierre Ducrot, David Brown, and Matthias Rarey*

Cite This: J. Chem. Inf. Model. 2020, 60, 6269–6281 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Structure-based fragment growing is one of the key techniques in fragment-based drug design. Fragment growing is commonly practiced based on structural and biophysical data. Computational workflows are employed to predict which fragment elaborations could lead to high-affinity binders. Several such workflows exist but many are designed to be long running noninteractive systems. Shape-based descriptors have been proven to be fast and perform well at virtual-screening tasks. They could, therefore, be applied to the fragment-growing problem to enable an interactive fragment-growing workflow. In this work, we describe and analyze the use of specific shape-based directional descriptors for the task of fragment growing. The performance of these descriptors that we call ray volume matrices (RVMs) is evaluated on two data sets containing protein−ligand complexes. While the first set focuses on self-growing, the second measures practical performance in a cross-growing scenario. The runtime of screenings using RVMs as well as their robustness to three dimensional perturbations is also investigated. Overall, it can be shown that RVMs are useful to prefilter fragment candidates. For up to 84% of the 3299 generated self-growing cases and for up to 66% of the 326 generated cross-growing cases, RVMs could create poses with less than 2 Å root-mean-square deviation to the crystal structure with average query speeds of around 30,000 conformations per second. This opens the door for fast explorative screenings of fragment libraries.



INTRODUCTION

Fragment-based drug design (FBDD) is by now an established practice in drug design.[1] Beginning in the early 1990s, it has since been applied in several successful projects and even led to approved drugs. FBDD's advantages are reported to be a more efficient exploration of chemical space in comparison to high-throughput screening (HTS) campaigns and high complementarity to the hits of HTS.[2] An approach using fragments is said to sample chemical space more efficiently because fewer molecules are required to describe the respective chemical space.[3] If desirable chemical functionalities are discovered in smaller fragments, these can be combined to create larger molecules, without having to sample the larger chemical space of lead-like molecules. Fragment screening hits, although less potent, have lower false positive rates than HTS hits. This can be attributed to fragment solubility, less risk of aggregation, and more robust detection methods.[4] Fragment screening is closely linked to biophysical detection methods and structure-based design (SBD). It is these methods that contribute most to the difficult task of achieving the potency necessary to convert a fragment hit into a viable lead. Specifically, computational methods of SBD can be used to reduce the dependence on costly experimental methods.[2]

Computational methods that use fragments as building blocks or chemical probes actually predate the experimental methods.[5] These methods were applied to various tasks such as probing for possible interactions in a crystal structure[6] but also the task of designing ligands de novo.[7] Many modern systems work toward extending fragments into potent leads. Some systems are purely ligand based. These usually work by sampling the chemical space around a given active compound or set of active compounds, generating ligands de novo as opposed to performing a similarity search in large databases of existing or accessible compounds. Structure-based systems emulate established FBDD practices and are often categorized into fragment linking, fragment merging, and fragment growing.[5] Yet, the methodologies that implement these ideas can be very different.

Computational fragment elaboration methods utilize several more or less established schools of thought. Many methods

Received: August 10, 2020
Published: November 16, 2020

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

build on the wealth of experience found in the development of ligand-docking software by either applying ligand-docking software in FBDD scenarios[8] or developing docking software with fragments explicitly considered.[9−11] Some workflows apply concepts used in docking but create fragment-specific algorithms.[12] Genetic algorithms are also a mainstay of fragment-growing workflows,[13,14] but sampling combinations of fragments can also be driven by other probabilistic approaches.[15] Reacting to the fact that many de novo workflows produce unsynthesizable ligands, some workflows focus on synthesis rules in their development.[16−18] These use reaction rules to determine whether fragments can be attached to others. A popular computational paradigm is the concept of shape complementarity. Shape-based methods have been shown to be very fast and to perform well in virtual-screening tasks.[19] Shape complementarity has also found its way into computational FBDD workflows.[20] Shape-based methods can profit from the limitations a fragment-binding pose imposes on the fragment extension. The directionality of the modifiable bonds of a fragment and the binding pocket naturally limit the directions a fragment can be extended into. Leveraging this directional information, as well as the shape defined by the pocket, can improve a fragment-growing workflow.

Liu et al. use a directional shape-based descriptor in their larger fragment elaboration workflow as an initial prefilter for fragment candidates.[21] A probe of polyacetylene is used to define the depth of a binding site. This probe extends up to a maximum of 10 Å in the growth direction. For each carbon atom, 24 sampling lines, or more specifically rays with the carbon atom as their origin, are generated at 15° angles to each other. Rays generated this way are used to describe the shape of the pocket. The efficiency and accuracy of this step is not separately described in their work. The larger workflow is characterized in the context of a search for inhibitors of 11β-hydroxysteroid dehydrogenase type 1, which highlights the fact that measuring the performance of fragment-growing workflows can be difficult.

Validation of fragment-growing workflows is often done on a phenomenological basis. Either a selection of ligands coming out of a workflow is synthesized and tested against an assay[8,14,18,21] or a similarity metric is computed between the output ligands and known actives.[12,17] A few attempts have been made to leverage known structural information in statistically significant samples for the development and validation of structural fragment growing[16,22] or fragment docking.[11,23] However, an established gold standard does not exist. An aspect that is sometimes neglected in validation is the runtime. Although some developers do report runtimes,[8,12,13,16,21] the systems themselves are often designed to be long running noninteractive systems, which is not conducive to the iterative nature of drug design projects and prone to reduce the scientist's control over the process.

This work will investigate the usefulness of directional shape-based descriptors called ray volume matrices (RVMs) as a prefilter in structural fragment-growing applications. The runtime of the descriptor will receive special attention, seeing as it is an important aspect of a responsive workflow. A data set of protein−ligand complexes will be created for validation purposes. The data set will consist of self-growing scenarios in which the descriptor will be used to reconstruct a ligand in its cocrystallized pocket. The major aim of this data set is to characterize the shape-based descriptions' correctness, robustness, and runtime. Another data set will be created to simulate a growing scenario closer to real scenarios in hit-to-lead optimization. This data set, which we call the cross-growing set, will contain a number of growing pairs that are made up of functionally equivalent pockets and ligands with a common core, so that ligands can be grown into pockets they are not cocrystallized in, mimicking a cross-docking scenario.

METHODS

Descriptor Generation. The starting point for descriptor generation will be a fragment bound in its target, a quite common scenario in fragment-based lead generation.[5] The shape-based description illustrated below assumes that the binding mode of the fragment does not change significantly as a result of extending it. Although this is a common assumption, it is not always correct.[22]

The first choice to be made before extending the fragment is at which bond the extension is to take place. This bond will be referred to as the exit bond. The exit bond can be chosen based on possible reaction centers,[16] using heuristics,[21] or be left up to the user's discernment. Assuming such an exit bond has been chosen, the bond's direction defines the direction of growth. This growing direction can be used to limit the volume a descriptor needs to describe.

A cylindrical shape descriptor's volume is defined by two parameters: the cylinder's depth and the cylinder's radius (see Figure 1). The cylinder's depth begins at the atom of the chosen exit bond that will be kept when a building-block is attached. This will be referred to as the anchor atom. In our implementation, the depth and radius of the cylinder are set to a value that depends on the use case. In later sections, we will set the depth and radius to 10 Å, similar to the depth set by Liu et al. The RVM is also extended behind the anchor atom to more accurately describe clash between an attached building-block and the fragment already bound in the pocket. The volume is sampled in regular distance increments along the cylinder axis, which corresponds to the growing direction. At each such distance increment, rays at regular angle intervals to each other are created in a circular fashion perpendicular to the growing direction. Analogous to the implementation by Liu et al.[21] these rays are used to describe the position of atoms in the pocket.

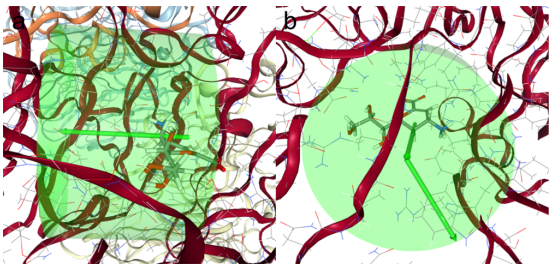


Figure 1. Volume described by the descriptor is shown as a green cylinder in a structure of the neuraminidase (2QWD). The cylinder is shown from the side in (a) and with its axis pointing toward the viewer in (b). The cylinder's depth is longer than its radius because of the cylinder being extended behind the anchor atom so that the descriptor can detect clashes with the fragment bound in the pocket.

First Intersection Widths. The sampling rays are intersected with the van der Waals spheres of the pocket atoms to find the first point of intersection between the sampling ray and the pocket. The resulting intersection segments are shown in Figure 2. The distance between the first point of

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

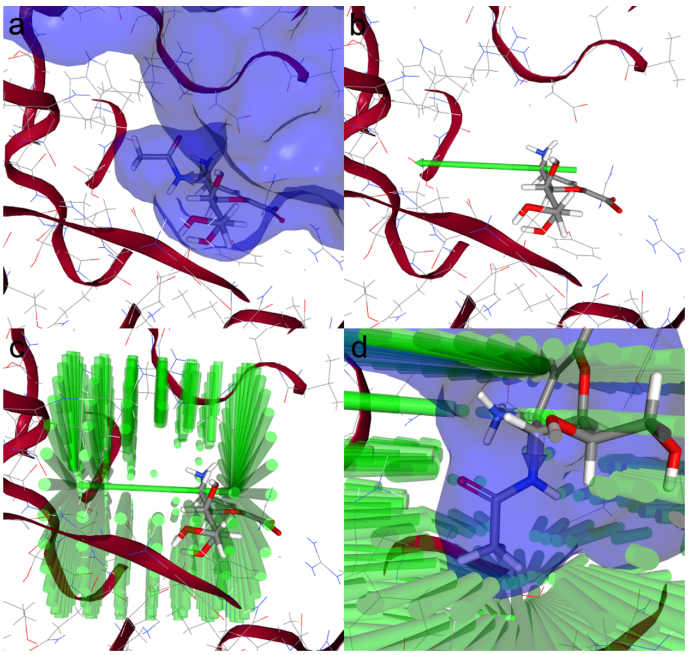


Figure 2. Descriptor generation for a part of a neuraminidase inhibitor in its binding pocket (2QWD). The full inhibitor is shown in (a). The acetylamide fragment that the blue arrow is pointing at is cut off and a query is generated for the remaining part of the inhibitor. The green arrow in (b) represents the descriptor depth. The green cylinders shown in (c) start at the first intersection point between a sampling ray and a pocket atom. The points of first intersection are complementary to a protein surface created by NGL Viewer[25] as shown in (d).

intersection and the axis of the cylinder will be referred to as the first intersection width. All atoms of the pocket and the bound fragment are used in the intersection, excluding polar hydrogens, which may be involved in hydrogen bonds and thus closer to atoms than the sum of their respective van der Waals radii.[24] Apolar hydrogens are included to maintain the pockets' selectivity. Furthermore, atoms of the bound fragment that clash with the linker or anchor atom are not considered. These are atoms either directly bound to the anchor atom, which means their van der Waals spheres are not representative of their interaction, or only a few bonds apart from the anchor atom. The widths at every depth and every sampling ray are saved in the RVM. The rows in this matrix represent a descriptor depth and the columns in this matrix represent the angle of the sampling ray. An RVM generated this way represents the geometrical query that will be used to search for building-blocks to extend the bound fragment at the anchor atom.

A very similar descriptor can be generated for the building-blocks to be used in a fragment elaboration. In this case, the RVM's cylindrical volume must encapsulate the building-block. The growing direction will point at the building-block, the same way the growing direction pointed at the empty pocket that the building-block should fill before. The depth of the cylinder is the distance between the building-blocks' linker atom and the building-blocks' furthest atom from the linker atom in the growing direction. The sampling rays are generated analogously to the pocket descriptor. To describe the width of the fragment, the sampling rays are intersected with the van der Waals spheres of all atoms of the building-block, excluding polar hydrogens and the linker atom. The distance of the furthest intersection point from the cylinder axis is the width of the fragment at that point and represents an intersection of a sampling ray with the outermost atom (Figure 3). These widths form the RVM of the

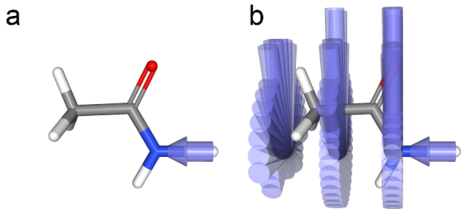


Figure 3. Descriptor for an acetylamide building-block with the linker at the nitrogen. The blue arrow in (a) represents the growing direction when the building-block is placed into the pocket. The linker atom is the only atom behind the arrow. The blue cylinders in (b) represent the first intersection widths. The cylinders start at the cylinder axis and continue to the outermost intersection point between a sampling ray and an atom.

building block. Note that measuring from the cylinder axis to the first intersection point for pockets or the outermost intersection for fragments can misrepresent the shapes of these objects in some cases. For example, a fragment may deviate from the central cylinder axis and point toward a corner of the cylinder, leaving the central axis free of steric hindrance.

Width Ranges. To improve the correctness of the descriptor, the first intersection widths can be converted into width ranges. Saving first intersection widths in a matrix results in a two-dimensional matrix. The distance from the cylinder axis to the first intersection is saved as a number resulting in a number for each ray without the need for a third dimension in

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

the matrix. Width ranges use bins instead of scalar values, making the resulting matrix three-dimensional (3D). A width range consists of a number of regular-sized bins that add up to the length of the cylinder radius. The bin values are generated by moving along the sampling ray up to the cylinder radius and checking whether the bin at this point clashes with an atom. If any part of the bin clashes with the van der Waals radius of the atom at all, the bin is set to occupied (one). If it does not, the bin is set to empty (zero). Bins are able to describe ranges along the sampling ray that intersect with pocket atoms without having to stop at the first intersection point. Width ranges were implemented to deal with irregular pocket shapes and building-blocks that deviate from the cylinder axis. Figure 4 gives examples of how irregular pocket shapes may influence the descriptor. A poor description of building-blocks and pocket shapes may result in false negatives that are difficult to correct.

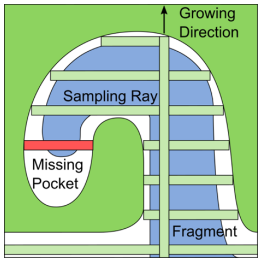


Figure 4. Some pockets cannot be completely sampled using first intersection widths. The light green rays represent the sampling rays of the pocket descriptor. A part of the pocket is hidden by steric bulk. Without this part of the pocket present in the pocket descriptor, the fragment or building-block that is supposed to bind in that subpocket will be considered as clashing in that area.

Database Generation. Seeing as the RVM descriptor relies on 3D coordinates, conformational sampling has to be performed beforehand. An unbiased conformational sampling would yield several conformations that cannot fit into the volume covered by the descriptor. To ensure more useable conformations, a cylindrical constraint is imposed upon candidate conformations of a building-block. The cylinder is placed along the exit bond as in the Descriptor Generation. The van der Waals spheres of all building-block atoms must be within the cylinder for the conformation to be accepted. The conformation generation procedure was adapted from the Conformator[26] workflow.

A screening database of RVMs is created by conformationally sampling the building-blocks and generating a descriptor for each of these conformations. The database is screened using a descriptor generated for a pocket as the query.

Descriptor Comparison. The building-block is eventually attached to the fragment by superimposing the linker atom of the building-block with the anchor atom of the bound fragment. These two atoms are the origin of the two cylindrical descriptors, thus no alignment is required for descriptor comparison. For efficiency reasons, the descriptor parameters such as depth, radius, and bin sizes are assumed to be set equally for pocket and building block descriptors.

Two descriptors using first intersection widths are compared by iterating over all the widths of the pocket and the fragment descriptor and calculating the difference of these widths. If the pocket width is substantially larger than the fragment width, the fragment does not fill out the pocket. If the fragment width is larger than the pocket width, the fragment clashes with the pocket at this point. Width ranges are compared on a bitwise level. A bitwise AND is performed on the bitstrings making up the descriptor and the building-block widths. The resulting bitstring contains all the bins both widths have in common, in other words all the bins of the building-block that clash with the pocket.

The cylindrical nature of the descriptor has an important advantage that can be exploited during comparison. As remarked by Liu et al.[21] shifting the columns of a descriptor implicitly rotates the shape the descriptor represents. If the building-block descriptor is shifted relative to the pocket descriptor, the building-block's shape is effectively rotated inside the shape of the pocket without the need for any geometrical transformation (Figure 5). In our implementation, the pocket descriptor is rotated once into every possible orientation before the building-blocks are queried. Every building-block descriptor is then compared to every possible rotation of the pocket descriptor.

Clash Filtering. All comparisons using RVMs are performed with a clash tolerance. This clash tolerance is defined by the user

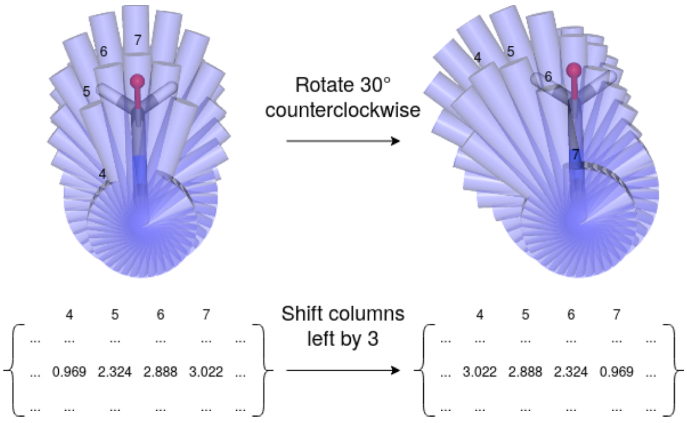


Figure 5. Rotating a descriptor by shifting columns. The rays in the images are 10° apart from each other. Each ray also corresponds to a column in the matrix. Shifting the matrix to the left by three columns results in a 30° counterclockwise rotation after projecting the descriptor back into 3D space.

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

and limits the largest possible clash a descriptor pose can generate without being discarded. In the latter sections, we will define the clash tolerance to be 2 Å. First intersection widths use this clash limit directly by checking whether a fragment ray that goes beyond the first intersection of the pocket (thus generating clash) extends beyond the first intersection point by more than the defined clash tolerance. In other words and using our 2 Å example, if a fragment ray is longer than the pocket ray it is being compared to by more than 2 Å, the descriptor containing that fragment ray is not considered further. Width ranges convert the clash tolerance into bins to use them. Using our example: if we have 0.5 Å bins, have defined the clash tolerance to be 2 Å, and more than four consecutive bins that are occupied (set to one) in the fragment ray are also occupied (set to one) in the pocket ray, then that descriptor is filtered out. The four consecutive bins along their respective rays add up to a length of 2 Å and any clash beyond that would violate the clash tolerance. Because of the nature of width ranges consecutively clashing bins can be interrupted, as shown in Figure 6. Only consecutively clashing bins can exceed the clash tolerance, as interrupted clashes suggest the clash is superficial. Further discussion of clash perception can be found in the Supporting Information.

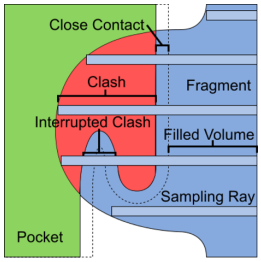


Figure 6. Comparison of a fragment descriptor with a pocket. The light blue cylinders are the sampling rays of the descriptor. All three elements used for scoring (clash, close contact, and filled volume) are shown along sampling rays. The dashed line represents the distance that is considered close enough for something to be a close contact.

Pose Scoring. A simple shape-based scoring scheme was implemented to compare generated poses to each other. Scoring is based on the assumptions that filling the volume of the pocket is good, that having close contacts with the pocket surface is better, and that clash should be avoided. Four parameters, which are filled volume, close contacts, clash, as well as a cutoff distance for close contacts, are needed to formulate this heuristic into a scoring function. All four scoring components are visualized in Figure 6. The weights are parameterized based on the validation following this section.

Using width ranges, all bins that are set to full in both the pocket and the building-block descriptor are counted as clashes. All bins that are within the close contact distance rounded up to the nearest bin are counted as close contacts. All other bins are counted as filled volume.

Using first intersection widths, the difference of the fragment width and the pocket width is considered the clash range. Subtracting the close contact distance from the pocket width yields the close contact distance cutoff. The difference between the fragment width and the close contact distance cutoff is the close contact range. The rest of the fragment width is considered the filled volume range.

In theory, each fragment can generate as many poses as it has conformations multiplied by the number of sampling rays used. Only the best pose of each fragment is included in the hit list. The simple scoring heuristic is used to perform this pose-scoring step efficiently.

Self-Growing Validation. Two data sets were generated based on the PDBbind refined set v.2019[27] to validate the performance of both the RVM using first intersection widths and the RVM using width ranges. The first data set performs self-growing validation, that is, it measures to what extent the shape descriptor is able to replace parts of a crystallized ligand in its own pocket. In docking terms, this is similar to self-docking and it will be referred to as self-growing in the following.

An overview of the generation procedure is given in Figure 7. All ligands in the PDBbind refined set were cut at random single bonds to produce fragments. These fragments were then filtered using the "Rule of Three".[28] Fragments also had to have at least three heavy atoms, not counting the linker atom, to be considered as such. Furthermore, fragments were filtered according to structural criteria found in the Supporting Information to ensure that they were, for example, not completely solvent-exposed.

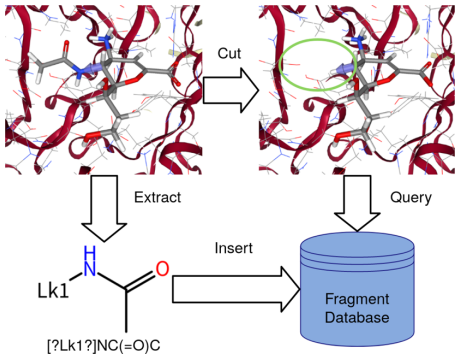


Figure 7. Overview of the self-growing validation. A bond is chosen, according to filter criteria, shown in the top left image as a blue arrow. Ligands are cut at this bond in their pockets and used as queries to see if they can be reconstructed. The fragments cut off of the ligands are extracted and inserted into the fragment database to be queried. "Lk1" represents the linker resulting from the cutting procedure.

If a fragment had been extracted before from another ligand, the iteration continued until a new unique fragment was found. If no unique fragments could be extracted, the last extracted duplicate fragment was used. The crystal conformations of the fragments were saved for later comparisons. The screening databases for validation were created using SMILES of the fragments according to the procedure outlined in the RVM generation, making the conformer sampling and subsequent descriptor generation independent of the initial crystal structure coordinates.

The self-growing validation is run by using the descriptor comparison procedure outlined in the Descriptor Generation section. A screening is performed and the best scored pose of every fragment that generated less clash than a given tolerance is written to a hit list. For all sets based on the PDBbind refined set, the clash tolerance is 2 Å, which is derived from the fact that only complexes with less than 2 Å of clash are accepted into the

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

Table 1. Grid Search Parameters for the Discretization of Descriptors[a]

parameter						
depth interval	0.5 Å	1.0 Å	1.5 Å	2.0 Å		
rays per depth	72(5°)	36(10°)	24(15°)	18(20°)	12(30°)	6(60°)
bin size	0.25 Å	0.5 Å	0.75 Å	1.0 Å	1.25 Å	1.5 Å

[a]All combinations of all parameters were used. The values given in the round brackets denote the angle between the rays if the number before the brackets is used as the number of rays per depth.

refined set.[27] Furthermore, the depth and radius of the descriptor cylinder were fixed at 10 Å.

The screening hit list is evaluated in the following ways. The hit list is scanned for the fragment initially extracted from the query complex to determine whether the ligand could be reconstructed using the shape descriptor. If the reconstructed ligand is found in the hit list, the pose of the reconstructed part is compared to the crystal conformation. A pose is considered acceptable if the root-mean-square deviation (RMSD) to the crystal conformation is less than 2 Å. In the absence of a convention specific to fragment growing, the 2 Å RMSD threshold is taken from ligand docking.[29] It is important to note that all RMSDs are calculated only for the atoms that are actually grown and not for the whole ligand. The self-growing set can also be used to measure the runtime of queries using the shape descriptor. To this end, the time it takes to process each query is recorded.

Parameterization. Several components of the shape description require parameterization, first and foremost the discretization of the cylindrical descriptor itself. The RVM has at least two parameters, besides the depth and radius of the cylinder, that determine its shape: the depth interval between the generation of sampling rays and the number of sampling rays generated at each depth interval. Width ranges are also influenced by the size of their bins. Parameterization was performed using a grid search across the parameter space. The grid search was performed on the self-growing set. The parameters can be found in Table 1.

The discretization is parameterized independently of the scoring and the optimum was determined by how many acceptable poses were generated. One discretization that is not an optimum will be chosen to highlight the effect that finer and coarser discretizations have on runtime. For the sake of better comparison, it will be a combination of optimal discretization parameters from both descriptor variants. Not all discretizations react equally to different parametrizations, which is briefly discussed in the Supporting Information. The scoring heuristic is left at an initial guess during parametrization of the discretization. The weights for the initial guess were 1 for filled volume, 2 for close contacts, and −4 for clashes and a close contact distance of 2.0 Å.

The scoring heuristic also requires proper parameterization of its three scoring weights for filled volume, close contacts, and clashes as well as the close contact distance cutoff. The numeric values of the scoring weights are not necessarily meaningful but their relation to each other is, which is why mainly the relation is explored. The self-growing set was used in a grid search of parameters. The filled volume weight was used as a baseline parameter and it was assigned to be 1. The rest of the parameters were sampled starting from a value and proceeding toward an end value in regular step sizes. The grid parameters are given in Table 2.

Robustness. Structural data are always subject to experimental error,[30] which is why robustness experiments were performed to measure the capacity of the RVM to compensate for perturbations in the 3D coordinates of the query. The self-growing set was used to create perturbed queries by translating or rotating the ligand core relative to its pocket. The axes of rotation and translation were defined by setting the x axis to the exit bond, the y axis to an arbitrary orthogonal of the exit bond, and the z axis to the cross product of the previous two axes. Poses were generated by rotating and translating ligand cores along and around these axes, starting at a minimum perturbation up to a maximum perturbation in regular step sizes. All perturbations were applied in a negative and a positive direction along or around the axes. The transformation parameters are given in Table 3.

Table 2. Grid-Search Parameters for the Simple Scoring Heuristic[a]

parameter	start	end	step size
clash weight	−1	−50	−5
close contact weight	1	10	1
close contact cutoff distance	1.0 Å	3.0 Å	0.5 Å

[a]All combinations of all parameters were used. The clash weight started at −1 but continued in multiples of −5, so the first three parameters were −1, −5, and −10.

Table 3. Transformations for the Robustness Validation[a]

transformation	start	end	step size
translation	0.5 Å	3.0 Å	0.5 Å
rotation	15°	60°	15°

[a]Translations and rotations were performed separate of one another.

Figure 8 shows an example of a ligand core translated by 0 or 0.5 Å along the three axes as well as that same ligand core rotated along these 3 axes by 0 or 30°. Some poses of the ligand core generated with the sampling parameters above already

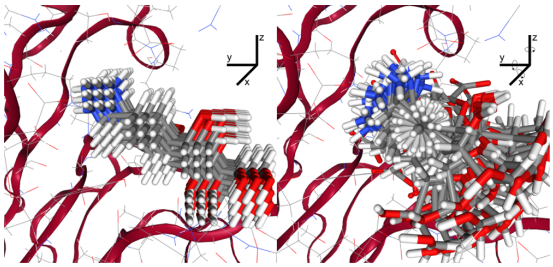


Figure 8. Translational and rotational sampling of a ligand core. Both transformations are based around three axes where the x axis is the direction of the exit bond, the y axis is an arbitrary orthogonal to x, and the z axis is the cross product of x and y. The translational parameters are either 0 or 0.5 Å for each axis. The rotational parameters are either 0 or 30° for each axis.

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

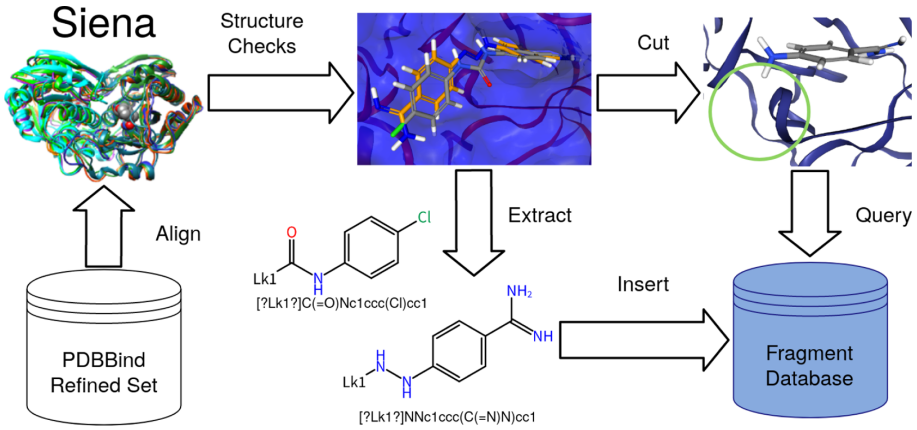


Figure 9. Generation of the cross-growing set. Functionally equivalent pockets from the PDBbind refined set are aligned to binding site ensembles. These ensembles pass through structural checks to ensure the ligands have a common core and the binding mode of that core has not changed significantly. The variable parts attached to the common core are extracted as fragments and the common core as well as the pocket can be used as a query to find them again. The examples are from the PDB codes 1BJU and 3GY2.

contained a substantial amount of clash with the pocket. If an atom pair of a pocket atom and an atom of the transformed ligand core caused more than 2 Å of clash, the pose in question was discarded, which mimics the quality criteria of the PDBbind refined set.[27]

Cross-Growing Validation. The cross-growing set was created to simulate a growing scenario closer to real hit-to-lead optimization workflows than the self-growing set described previously. The idea of the data set was to mine pairs of ligands bound to the same pocket but in different PDB structures that could conceptually be grown from one another by extending or modifying one part of one of the ligands.

A data set by Malhotra and Karanicolas that used shape overlap as a measure of conserved binding mode was previously used in the validation of fragment growing.[22] Measuring conserved binding mode by shape overlap does not take into account changes in growing direction and is generally very permissive of geometrical drift. For this reason, we have chosen to create our own data set.

The tool SIENA[31] was used to determine functionally equivalent ensembles of pockets within the PDBbind refined set v.2019. Detailed parameters for SIENA can be found in Table S1 of the Supporting Information. The ligands of all aligned pockets were compared to each other in a pairwise maximum common substructure search to find a common core. The bond connecting the variable parts to the common core was used as the exit bond. The variable parts, in other words the potential fragments, were filtered according to the same rules as in the self-growing set. Furthermore, to ensure the binding mode of the two ligands was equivalent, the position and direction of the exit bonds were compared. An overview of the procedure is shown in Figure 9.

RESULTS AND DISCUSSION

Self-Growing Set Generation. The splitting and filtering procedure was run on all 4852 ligands of the PDBbind refined set v.2019. The filtering resulted in 3299 test cases and 1189 unique fragments. Table 4 contains a summary of the size of both datasets. Over 1600 ligands did not generate any fragments that were compatible with the "Rule of Three" and the structural filters. Visual inspection reveals a few of the reasons for this at a glance. One reason is that many of these ligands consist of a scaffold with very little decoration. An example of this is the ligand of 187L, which is a *para*-xylene. A methyl fragment has fewer than three heavy atoms and is, therefore, not extracted as a growing fragment. Other examples of sparsely decorated scaffolds and thus few chances at extracting suitable fragments include steroidal ligands and various annealed heteroaromatic ring scaffolds. A significant number of fragments are extracted multiple times, which results in the difference between the number of test cases and unique fragments. The property space of the fragments and how it is affected by duplicates can be found in the Supporting Information.

Table 4. Size of Both Validation Data Sets in Test Cases and Unique Fragments

	test cases	unique fragments
self-growing set	3299	1189
cross-growing set	326	155

Parameterization. Optima for the discretization and scoring parameter grid-search were those that maximized the number of acceptable poses generated. The optima for first intersection and width ranges variants of the RVM can be found in Table 5. An alternative discretization was chosen for the width ranges variant to illustrate the influence of discretization on runtime. The "coarse width ranges" discretization was chosen as a combination of the coarser optimal first intersection discretization and the optimal width range discretization. The depth interval and rays per depth parameter were taken from the optimal first intersection discretization and the bin size parameter from the optimal width range discretization, which resulted in a coarser width range discretization more comparable

Table 5. Final Discretizations for All Three RVM Variants

descriptor variant	depth interval (Å)	rays per depth	bin size (Å)
first intersection widths	1.5	18	
coarse width ranges	1.5	18	0.75
width ranges	0.5	72	0.75

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

to the optimal first intersection discretization. Scoring parameters were determined for all three of these discretizations (Table 6) and although the absolute values of the parameters vary, the relation of the scoring parameters, especially clash and close contact, stay similar. The close contact distance for coarse width ranges is somewhat of an outlier and highlights that the close contact distance also fulfills the role of keeping the fragment in the pocket and not only describing shape complementarity.

Table 6. Final Scoring Parameters for Both Descriptor Variants

descriptor variant	clash	close contact	close contact distance (Å)
first intersection widths	−30	5	3.0
coarse width ranges	−20	5	5.0
width ranges	−50	10	3.0

In general, the optimization surfaces of both parameterizations were very flat. All grid points of the first intersection discretization grid search were within 10% of their optimum. Only 3.8% of the grid points in the scoring grid-search for the first intersection variant were further than 10% away from their optimum. Only at the extremes, the granularity of the discretization can conceptually be associated with functional differences of the descriptor. An example of this is the increase of RMSD to crystal structure observed when the number of rays is decreased (Figure S6). Lowering the number of rays decreases the number of rotational comparisons performed and thus the resolution at which rotational poses are sampled.

Performance on the Self-Growing Set. All three descriptor variants were run against the self-growing set. The results can be found in Figure 10a. 96% (3158) of the ligands in the self-growing set could be reconstructed in any pose. 82% (2716) of the ligands could be reconstructed in an acceptable pose (i.e., with less than 2 Å RMSD to the crystal structure). 99% (3287) of the ligands could be reconstructed in any pose using width ranges and 86% (2822) of the ligands were reconstructed with a pose with less than 2 Å RMSD to the crystal structure. The coarse width range discretization managed to reconstruct 99% (3292) as well and 85% (2793) of the ligands were reconstructed in an acceptable pose.

First intersection widths fail to reconstruct a little over 4% (141) of the test cases in the self-growing set. Width ranges and coarse width ranges fail to reconstruct a little under 1% (12) of the test cases. The test cases that fail using width ranges are almost a subset of the test cases that fail using first intersection widths. The coarse width range discretization performs slightly better at reconstructing ligands than the optimal width range discretization and is only unable to reconstruct seven ligands. This is probably because of its more unspecific discretization. It suffers for that slightly when it comes to the number of acceptable poses. In general, both width range variants are better at generating acceptable poses. All results only consider the top pose as scored by the pose scoring heuristic. Considering more poses does increase the quality of the results as shown in Figure S13 of the Supporting Information but becomes untenable in a scenario where thousands of fragments are to be grown into a binding site.

First intersection widths and width ranges both fail to reconstruct 5T9W. Figure 11 illustrates this. Although the attached fragment and the ligand core are only clashing superficially, the clash is sampled in a way that makes it seem

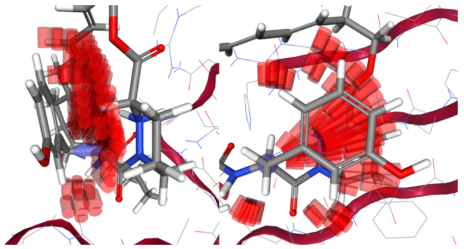


Figure 11. Overestimation of clash in the 5T9W test case. The blue arrow denotes the exit bond and the growing direction. The ligand core and the fragment needed to reconstruct the 5T9W ligand clash superficially but because of the way the clash is sampled, it is estimated to be more severe than it is. The fragment is in the closest pose to the crystal structure that the descriptor using width ranges can generate.

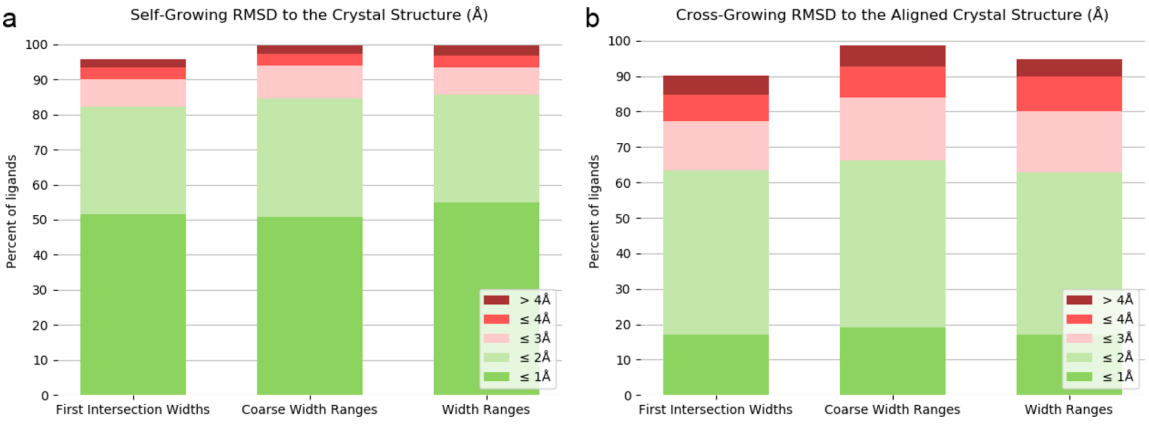


Figure 10. Results of performing the self-growing (a) and cross-growing (b) validation. The size of the bars represent which percentage of the ligands of the test cases could be reconstructed. The color of the stacked bars denote the RMSD of the poses according to the provided legend. The RMSD is always measured only between the atoms actually grown and the crystal structure or aligned structure.

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

far more severe. It is refound using coarse width ranges at a clash tolerance of 2 Å because of the difference in discretization between the descriptor variants. The first intersection widths severely oversample the clash because of everything after the first point of clash, up to the width of the fragment descriptor, being considered as clash, without taking any interruptions of the clash into account. This is one example of the functional difference between width ranges and first intersection widths.

The test case generated from 1B1H is a better example demonstrating the functional difference between width ranges and first intersection widths. Figure 12 shows that the ligand

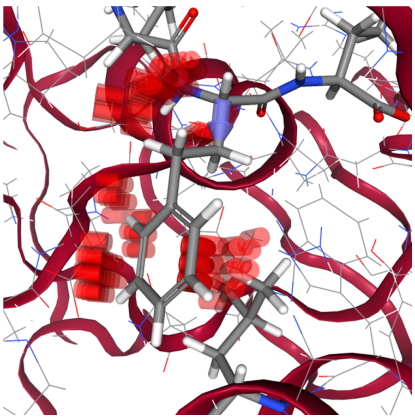


Figure 12. 1B1H test case demonstrates the difference between width ranges and first intersection widths. The blue arrow denotes the exit bond and the growing direction. Both descriptors clash with leucine 401 shown at the bottom in licorice. Only width ranges sample the pocket occupied by the fragment necessary to reconstruct the ligand of 1B1H. First intersection widths cannot sample the occupied subpocket.

reconstructed in the 1B1H test case requires a subpocket to be sampled that will not be sampled using first intersection widths. The growing direction in the 1B1H test case enters leucine 401. For intersection widths, this results in a completely closed pocket beyond that point. The point of first intersection with the pocket will be the origin of the sampling rays because the rays originate in an atom. Width ranges also generate filled bins for the leucine, as seen in Figure 12, where some hydrogen clash is generated between the leucine and the fragment, but enough of the subpocket is sampled so that the phenyl ring can enter and generate a pose with a low RMSD to its crystal structure.

Width ranges perform better at the task of reconstructing a ligand in its own pocket, or self-growing, than first intersection widths. The main advantage of first intersection widths is their runtime. Runtimes for all 3 RVM variations can be found in Table 7. A query in the self-growing set takes on average 6 s using width ranges and querying a database that contains all fragments of the self-growing set. In terms of conformations screened, width ranges screen 1445 conformations per second. First intersection widths on the other hand take on average 119 ms for a query in the self-growing set. In terms of conformations screened that means first intersection widths screen 77,221 conformations per second. This difference in runtime is heavily dependent on the discretization of the descriptor. The coarse width range discretization, which is much closer in discretization to first intersection widths than width ranges, takes on average 283 ms for a query in the self-growing set, meaning 32,366

Table 7. Runtimes of the Three Descriptor Variants on the Self-Growing Set[a]

descriptor variant	average query runtime (ms)	conformations per second
first intersection widths	119	77,221
coarse width ranges	283	32,366
width ranges	6355	1445

[a]The screening databases are made up of 9188 conformations except for the screening database for width ranges, which is made up of 9185 conformations.

conformations per second. Machine specifications for the benchmarking run are given in Table S3 of the Supporting Information.

On a technical level, any difference in speed independent of discretization is a difference in the comparison of fragment sampling rays and pocket sampling rays. As mentioned in the Descriptor Generation subsection of Methods, first intersection widths employ comparisons of numbers to each other, whereas width ranges perform multiple bitwise operations and are, therefore, more complex. The first intersection width comparison only involves subtracting fragment and pocket widths from one another. Both of these operations, bytewise AND followed by bit tests, as well as subtracting numerical types, are quite low level and it makes sense that they perform comparably, meaning their runtimes are within 1 order of magnitude of each other.

Robustness. Test cases for the robustness experiment were generated as outlined in the Robustness section in Methods. A little over half of all translational poses and a little over 80% of all rotational poses were filtered out because of clash the transformation introduced between the ligand core and its pocket. It appeared that rotation within the pocket was more restrained than translation.

Figures S8 and S9 show the results of running the robustness experiment with the fully parameterized variants of the RVM. At 2 Å of translational perturbation, there is a clear difference of about 10% between width ranges and first intersection widths of whether the ligand could be reconstructed. At 45° of rotational perturbation, the difference between the descriptors is similar. Width ranges and first intersection widths seem to diverge in how much performance they lose with respect to 3D perturbations. To a degree, this is visible in the RMSDs of ligand poses to the crystal structures as well. The number of poses with an RMSD of below 2 Å drops a lot faster with respect to perturbation than the number of ligands reconstructed. At the extremes of the translational and rotational perturbations, all RVM variants converge to a similar amount of ligands reconstructed with acceptable poses.

The robustness advantage of the width ranges RVM may be because of its better description of a pocket. First intersection widths may be more sensitive to 3D perturbations that occlude subpockets because of unfavorable angles of the descriptor to the pocket. These occlusions can change the first intersection widths so drastically that a ligand cannot be reconstructed in its own pocket. Width range descriptors will also change but are not dependent on the first intersection of a sampling ray with the pocket and, therefore, not as sensitive to occlusions.

Cross-Growing Set Generation. The cross-growing set procedure outlined in Methods was run on all 4852 structures in the PDBbind refined set v.2019. For 4572 of these structures, SIENA was able to find at least one functionally equivalent pocket. The aligned ligands were checked for a common core

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

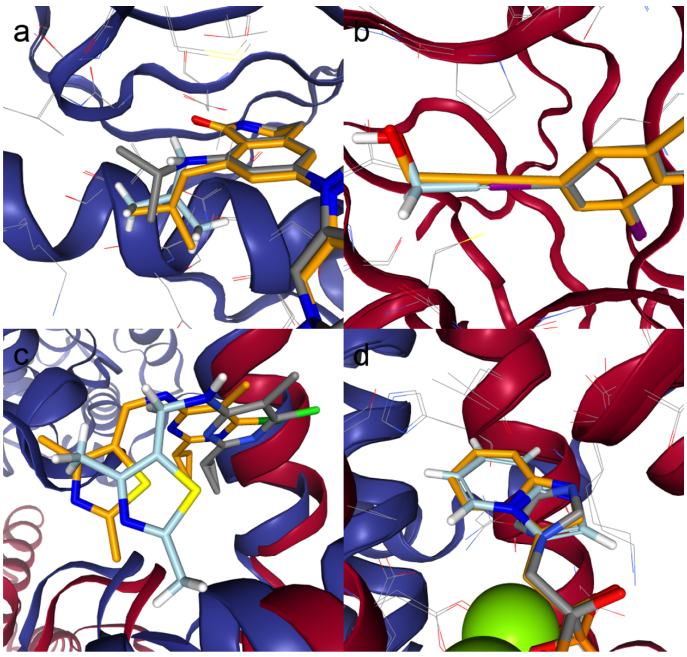


Figure 13. Collection of the results from the cross-growing validation. The gray structures are the ligands of the crystal structure that was grown in. The orange structures represent the ligands that were aligned to this binding site and that the cross-growing validation attempts to grow. The blue structures are the parts actually grown.

with a conserved binding mode and 3714 such cases were found. After filtering by the "Rule of Three" and the structural filters, 326 cross-growing cases using 155 unique fragments were left. This is an order of magnitude fewer test cases than in the self-growing set as can be seen in Table 4. An account of what filters were most aggressive in reducing the data set size be found in the Supporting Information.

In theory, a cross-growing case could be seen as symmetrical: both ligands are bound to the same functionally equivalent pocket and so it should be possible to grow both ligands into both pockets. In practice, over half of the complementary fragments that could be generated for each symmetric application of a cross-growing test cases are filtered out by molecular properties or structural filters. Many of these fragments have fewer than three heavy atoms or, in the case of a pure growing without substituent replacement, consist only of a hydrogen.

Performance on the Cross-Growing Set. The results showing the performance of the cross-growing data set can be found in Figure 10b. 90% of the ligands could be reconstructed using first intersections widths. Width ranges and coarse width ranges could reconstruct 95 and 99% of the ligands, respectively. As discussed in Methods, the RMSD was calculated with respect to the pose of the aligned ligand generated by SIENA. The percentage of ligands reconstructed with acceptable poses (meaning an RMSD below 2 Å) was around 63−66% for all descriptor variants with the coarse width ranges discretization at the upper bound.

Figure 13 shows a few example results of the cross-growing set taken from the run using coarse width ranges. Figure 13a shows an isopropylamine being replaced by an isobutyl in a lipophilic side chain replacement (4O07 grown in 4O05, HSP90). Figure 13b features a very constrained growing in which a narrow grove separates two subpockets. In this case, an Iodine at an aromatic ring is substituted for an alkyne with a terminal hydroxy group (4AGN grown in 4AGM, p53). In Figure 13c, two binding sites of a phosphodiesterase are aligned with a visible translational error between the ligands. Nonetheless, the growing manages to recreate the conformation of the larger ligand, and achieve an acceptable RMSD (5C2A grown in 5C28, PDE10A). The alignment error does not affect the growing directly, seeing as the reference structure is only used for comparison and explicitly excluded from the growing workflow, but this is an example of an alignment error contributing to the RMSD. Minodronate is grown from zoledronate in Figure 13d (2E92 grown in 2E91, geranyl−geranyl pyrophosphate synthase).

Because of protein flexibility, the fragments that are grown into the pockets of the cross-growing set may not always fit into the amino acid side chain conformations of their complementary pocket. An example of this is the test case generated for 2AX9 and 3B67 (androgen receptor) shown in Figure 14. Because of the change in the position of the tryptophan, the ligand of 3B67 cannot be grown into the pocket of 2AX9 by any of the descriptor variants. A few such cases are present in the cross-growing set. Simply increasing the clash tolerance in these cases would result in an unnecessarily high false positive rate. A better solution would be to address this problem with a description of protein flexibility.

A problem that is exacerbated in the cross-growing set is the fact that a purely shape-based method has significant problems recreating directed interactions between the protein and the ligand. In the self-growing set, each fragment had a more or less fitting subpocket that pushed the fragment into its correct orientation, thereby conserving directed interactions. In the

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

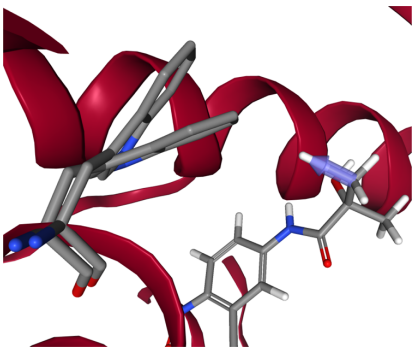


Figure 14. Test case generated for 2AX9 and 3B67. Tryptophan 741 changes its position, which opens up a part of the pocket in 3B67. This is not the case in 2AX9 and means that the ligand of 3B67 cannot be grown into the binding site of 2AX9 without significant clash.

cross-growing set, coordinates of the residues in the binding site drift somewhat between the pair of aligned structures. This can result in poses shown in Figure 15. The descriptor using first

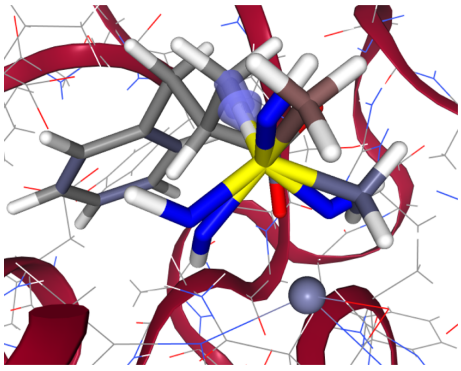


Figure 15. Test case generated for 1CBX and 1CPS (carboxypeptidase A) and two poses of the reconstructed ligand generated by first intersection widths and coarse width ranges. Carbons of the fragment are colored red and blue for coarse width ranges and first intersection widths, respectively. The blue pose generated by first intersection widths flips the metal coordinating sulfodiimine of the ligand away from its interaction partner, the zinc atom.

intersection widths has flipped the metal coordinating sulfodiimine of the ligand away from its interaction partner. The RVM only sees the sterical limitations of the subpocket it is growing into and cannot properly differentiate between the orientations of the methyl sulfodiimine functional group. The coarse width range discretization does find a discriminating pattern in the sterics of the subpocket but it, too, cannot properly grasp the electrostatics of the situation; this would require a direct description of protein−ligand interactions.

CONCLUSIONS

It is encouraging to see such a simplistic descriptor perform well on statistical sample sizes of real data. Both the simpler first intersection width approaches, as well as the only slightly more complex width range approach, show very reasonable performances of around 80% of test cases reconstructed at an RMSD of less than 2 Å in self-growing evaluations and around 60% of test cases in cross-growing evaluations. While a comparison can be drawn between self-docking and self-growing, as well as cross-docking and cross-growing, both growing problems are easier than their corresponding docking problems. This is because of the fragments being smaller and less complex, thus not necessarily requiring an overly complex docking workflow, but also because of the constraint of directionality imposed on the system through the exit bond of the ligand core. By leveraging these circumstances of the fragment-growing scenario, it is possible to create very fast descriptors that can screen over 10,000 conformations per second, with good accuracy.

There is still significant room for improvement of the runtimes of both comparison algorithms. A few optimizations have already been applied, such as using integers instead of floats to store the first intersection widths. All optimizations have been high level to ensure cross-platform compatibility and with maintainability as well as flexibility of the workflow in mind. The method as described above leaves a lot of creative freedom for implementation specific optimization on a molecular, descriptor or database level. At the moment, the method already runs very quickly even on weaker laptops and user machines.

In general, a full docking workflow will perform better than this simplistic shape-descriptor, mainly because of its more complex description of the protein and the ligand. A few limitations of the shape-based descriptor have been discussed, which will need to be addressed to elevate it from a coarse and fast prefilter to a full fragment-growing tool. First and foremost, we have seen that the descriptor lacks a comprehensive description of protein−ligand interactions. Other authors have seen marked improvements in the performance of shape-based descriptors after adding interaction descriptions.[32] In the cross-growing set, we encountered expected problems related to protein flexibility, whereby subpockets would be blocked by side chain movements. If the binding site was given as an ensemble of representative side chain orientations, a query could be generated for each orientation and the result of all queries combined when generating the eventual hit list. Similar approaches are used in docking.[33] A significant difference between our workflow and a docking workflow is that a bond is formed. To ensure this bond obeys the rules of structural chemistry, a few considerations have to be made about bond lengths and angles. A further improvement of that could be a filtering step based on torsion angle likelihood, so that the generated poses are not subject to torsional strain. A method that complements the speed of searches using the shape-based descriptor, is a lookup using SMARTS and pregenerated knowledge-based torsional statistics.[34]

The performance of docking workflows can be split into several categories. Two of these categories are pose prediction, which is the ability of a docking workflow to correctly identify the native ligand pose, and ranking, which is the ability of a docking workflow to correctly rank known ligands of a target according to their binding affinities.[29] If we apply these categories to our workflow, then this work has been exclusively dealing with the pose prediction problem of correctly identifying the native fragment pose. A correct ranking of fragments relative to each other is necessary for our workflow to be useful in real fragment-growing scenarios. Previous work exists where classical docking workflows, and therefore their scoring functions and optimization schemes, were applied to the problem of fragment elaboration.[8]

In summary, we have described and analyzed variants of shape-based directional descriptors for the purpose of fragment

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

growing. These have proven to perform well on statistical sample sizes of real world data. Their simplicity results in fast runtimes that permit interactive screenings of large fragment collections. The width range variation performed slightly better on most tasks at the cost of a more complex implementation. The first intersection width implementation was generally faster but had a few functional problems associated with it that were of minor statistical importance. In general, shape-based descriptors are suited as coarse and fast prefilters in fragment growing but require some more sophisticated components, such as a comprehensive protein−ligand interaction description, to achieve a higher performance. These more sophisticated and computationally intensive components would only have to be run on a subset of the input fragments leading to an efficient fragment-growing workflow to add to the toolset of computational FBDD.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.0c00920.

Results of all experiments (parameterization, self-growing, robustness, and cross-growing) (ZIP)

Discussion of more detail-oriented topics and further validation of dataset characterization (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Matthias Rarey − *ZBH−Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0002-9553-6531; Email: rarey@zbh.uni-hamburg.de

Authors

Patrick Penner − *ZBH−Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0003-4988-6183

Virginie Martiny − *Institut Recherches de Servier, 78290 Croissy, France*

Arnaud Gohier − *Institut Recherches de Servier, 78290 Croissy, France*

Marcus Gastreich − *BioSolveIT GmbH, 53757 Sankt Augustin, Germany*

Pierre Ducrot − *Institut Recherches de Servier, 78290 Croissy, France*

David Brown − *Institut Recherches de Servier, 78290 Croissy, France*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c00920

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that the RVM screening software is licensed for a fee to nonacademic institutions in the future.

■ ACKNOWLEDGMENTS

The authors would like to thank Christian Lemmen, Louis Bellmann, and Florian Flachsenberg for fruitful and less fruitful discussions.

■ REFERENCES

(1) Erlanson, D. A.; Fesik, S. W.; Hubbard, R. E.; Jahnke, W.; Jhoti, H. Twenty years on: the impact of fragments on drug discovery. *Nat. Rev. Drug Discovery* **2016**, *15*, 605−619.

(2) Hajduk, P. J.; Greer, J. A decade of fragment-based drug design: Strategic advances and lessons learned. *Nat. Rev. Drug Discovery* **2007**, *6*, 211−219.

(3) Fink, T.; Bruggesser, H.; Reymond, J.-L. Virtual Exploration of the Small-Molecule Chemical Universe below 160 Daltons. *Angew. Chem., Int. Ed.* **2005**, *44*, 1504−1508.

(4) Schuffenhauer, A.; Ruedisser, S.; Marzinzik, A.; Jahnke, W.; Selzer, P.; Jacoby, E. Library design for fragment based screening. *Curr. Top. Med. Chem.* **2005**, *5*, 751−762.

(5) Joseph-McCarthy, D.; Campbell, A. J.; Kern, G.; Moustakas, D. Fragment-based lead discovery and design. *J. Chem. Inf. Model.* **2014**, *54*, 693−704.

(6) Miranker, A.; Karplus, M. Functionality maps of binding sites: A multiple copy simultaneous search method. *Proteins: Struct., Funct., Genet.* **1991**, *11*, 29−34.

(7) Schneider, G.; Lee, M.-L.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 487−494.

(8) Marchand, J.-R.; Caflisch, A. In silico fragment-based drug design with SEED. *Eur. J. Med. Chem.* **2018**, *156*, 907−917.

(9) Böhm, H.-J. The computer program LUDI: A new method for the de novo design of enzyme inhibitors. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 61−78.

(10) Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A Fast Flexible Docking Method using an Incremental Construction Algorithm. *J. Mol. Biol.* **1996**, *261*, 470−489.

(11) Hoffer, L.; Horvath, D. S4MPLE—Sampler For Multiple Protein-Ligand Entities: Simultaneous Docking of Several Entities. *J. Chem. Inf. Model.* **2013**, *53*, 88−102.

(12) Degen, J.; Rarey, M. FLEXNOVO: Structure-based searching in large fragment spaces. *ChemMedChem* **2006**, *1*, 854−868.

(13) Fechner, U.; Schneider, G. Flux (2): Comparison of molecular mutation and crossover operators for ligand-based de novo design. *J. Chem. Inf. Model.* **2007**, *47*, 656−667.

(14) Foscato, M.; Venkatraman, V.; Jensen, V. R. DENOPTIM: Software for Computational de Novo Design of Organic and Inorganic Molecules. *J. Chem. Inf. Model.* **2019**, *59*, 4077−4082.

(15) Chéron, N.; Jasty, N.; Shakhnovich, E. I. OpenGrowth: An Automated and Rational Algorithm for Finding New Protein Ligands. *J. Med. Chem.* **2016**, *59*, 4171−4188.

(16) Sommer, K.; Flachsenberg, F.; Rarey, M. NAOMInext—Synthetically feasible fragment growing in a structure-based design context. *Eur. J. Med. Chem.* **2019**, *163*, 747−762.

(17) Chevillard, F.; Kolb, P. SCUBIDOO: A Large yet Screenable and Easily Searchable Database of Computationally Created Chemical Compounds Optimized toward High Likelihood of Synthetic Tractability. *J. Chem. Inf. Model.* **2015**, *55*, 1824−1835.

(18) Hartenfeller, M.; Zettl, H.; Walter, M.; Rupp, M.; Reisen, F.; Proschak, E.; Weggen, S.; Stark, H.; Schneider, G. DOGS: Reaction-Driven de novo Design of Bioactive Compounds. *PLoS Comput. Biol.* **2012**, *8*, No. e1002380.

(19) Kumar, A.; Zhang, K. Y. J. Advances in the Development of Shape Similarity Methods and Their Application in Drug Discovery. *Front. Chem.* **2018**, *6*, 315.

(20) Yuan, Y.; Pei, J.; Lai, L. LigBuilder 2: A practical de novo drug design approach. *J. Chem. Inf. Model.* **2011**, *51*, 1083−1091.

(21) Liu, Z.; Singh, S. B.; Zheng, Y.; Lindblom, P.; Tice, C.; Dong, C.; Zhuang, L.; Zhao, Y.; Kruk, B. A.; Lala, D.; Claremon, D. A.; McGeehan, G. M.; Gregg, R. D.; Cain, R. Discovery of Potent Inhibitors of 11β-Hydroxysteroid Dehydrogenase Type 1 Using a Novel Growth-Based Protocol of in Silico Screening and Optimization in CONTOUR. *J. Chem. Inf. Model.* **2019**, *59*, 3422−3436.

(22) Malhotra, S.; Karanicolas, J. When Does Chemical Elaboration Induce a Ligand To Change Its Binding Mode? *J. Med. Chem.* **2017**, *60*, 128−145.

(23) Favia, A. D.; Bottegoni, G.; Nobeli, I.; Bisignano, P.; Cavalli, A. SERAPhiC: A Benchmark for in Silico Fragment-Based Drug Design. *J. Chem. Inf. Model.* **2011**, *51*, 2882−2896.
(24) Nittinger, E.; Inhester, T.; Bietz, S.; Meyder, A.; Schomburg, K. T.; Lange, G.; Klein, R.; Rarey, M. Large-Scale Analysis of Hydrogen Bond Interaction Patterns in Protein-Ligand Interfaces. *J. Med. Chem.* **2017**, *60*, 4245−4257.
(25) Rose, A. S.; Hildebrand, P. W. NGL Viewer: a web application for molecular visualization. *Nucleic Acids Res.* **2015**, *43*, W576−W579.
(26) Friedrich, N.-O.; Flachsenberg, F.; Meyder, A.; Sommer, K.; Kirchmair, J.; Rarey, M. Conformator: A Novel Method for the Generation of Conformer Ensembles. *J. Chem. Inf. Model.* **2019**, *59*, 731−742.
(27) Liu, Z.; Su, M.; Han, L.; Liu, J.; Yang, Q.; Li, Y.; Wang, R. Forging the Basis for Developing Protein−Ligand Interaction Scoring Functions. *Acc. Chem. Res.* **2017**, *50*, 302−309.
(28) Congreve, M.; Carr, R.; Murray, C.; Jhoti, H. A 'Rule of Three' for fragment-based lead discovery? *Drug Discovery Today* **2003**, *8*, 876−877.
(29) Su, M.; Yang, Q.; Du, Y.; Feng, G.; Liu, Z.; Li, Y.; Wang, R. Comparative Assessment of Scoring Functions: The CASF-2016 Update. *J. Chem. Inf. Model.* **2019**, *59*, 895−913.
(30) Zheng, H.; Hou, J.; Zimmerman, M. D.; Wlodawer, A.; Minor, W. The future of crystallography in drug discovery. *Expert Opin. Drug Discovery* **2014**, *9*, 125−137.
(31) Bietz, S.; Rarey, M. SIENA: Efficient Compilation of Selective Protein Binding Site Ensembles. *J. Chem. Inf. Model.* **2016**, *56*, 248−259.
(32) Hawkins, P. C. D.; Skillman, A. G.; Nicholls, A. Comparison of Shape-Matching and Docking as Virtual Screening Tools. *J. Med. Chem.* **2007**, *50*, 74−82.
(33) Amaro, R. E.; Baudry, J.; Chodera, J.; Demir, Ö.; McCammon, J. A.; Miao, Y.; Smith, J. C. Ensemble Docking in Drug Discovery. *Biophys. J.* **2018**, *114*, 2271−2278.
(34) Guba, W.; Meyder, A.; Rarey, M.; Hert, J. Torsion Library Reloaded: A New Version of Expert-Derived SMARTS Rules for Assessing Conformations of Small Molecules. *J. Chem. Inf. Model.* **2016**, *56*, 1−5.

https://dx.doi.org/10.1021/acs.jcim.0c00920
J. Chem. Inf. Model. 2020, 60, 6269−6281

C.3. FastGrow: On-the-Fly Growing and its Application to DYRK1A

[D3] P. Penner, V. Martiny, L. Bellmann, F. Flachsenberg, M. Gastreich, I. Theret, C. Meyer, and M. Rarey, "FastGrow: on-the-fly growing and its application to DYRK1A," *Journal of Computer-Aided Molecular Design*, 2022, ISSN: 1573-4951. DOI: 10.1007/s10822-022-00469-y. [Online]. Available: https://doi.org/10.1007/s10822-022-00469-y. Material from [D3].

Journal of Computer-Aided Molecular Design
https://doi.org/10.1007/s10822-022-00469-y

FastGrow: on-the-fly growing and its application to DYRK1A

Patrick Penner[1] · Virginie Martiny[2] · Louis Bellmann[1] · Florian Flachsenberg[1,4] · Marcus Gastreich[3] · Isabelle Theret[2] · Christophe Meyer[2] · Matthias Rarey[1]

Received: 8 June 2022 / Accepted: 26 July 2022

Abstract
Fragment-based drug design is an established routine approach in both experimental and computational spheres. Growing fragment hits into viable ligands has increasingly shifted into the spotlight. FastGrow is an application based on a shape search algorithm that addresses this challenge at high speeds of a few milliseconds per fragment. It further features a pharmacophoric interaction description, ensemble flexibility, as well as geometry optimization to become a fully fledged structure-based modeling tool. All features were evaluated in detail on a previously reported collection of fragment growing scenarios extracted from crystallographic data. FastGrow was also shown to perform competitively versus established docking software. A case study on the DYRK1A kinase, using recently reported new chemotypes, illustrates FastGrow's features in practice and its ability to identify active fragments. FastGrow is freely available to the public as a web server at https://fastgrow.plus/ and is part of the SeeSAR 3D software package.

Keywords Fragment-based drug design · Molecular shape · Fragment growing · Fragment evolution · Structure-based drug design · Molecular docking

Introduction

Fragment-based drug design or discovery (FBDD) has become a mature paradigm in both the hit generation, as well as the lead optimization parts of early phase pharmaceutical research [1]. Three distinct pillars underpin FBDD techniques: fragment library design, fragment screening, and optimizing fragments into lead compounds, typically using linking, merging, and growing [2]. Computational methods have emerged to support each of these three areas specifically [3–6]. There is a special focus on the optimization of fragments to leads [7, 8], due to its methodological overlap with the general hit-to-lead optimization problem. A more general overview of FBDD and computational methods supporting FBDD can be found in dedicated reviews [2, 9, 10].

FBDD is frequently, if not almost exclusively, a structure-driven approach, meaning it relies on experimentally resolved or modeled structures of fragments that are bound to the target of interest. One popular method to progress from a low affinity fragment hit to a more traditional hit or lead is fragment growing [2, 11]. In fragment growing a molecule bound to a target is extended by attaching a suitable additional fragment. This is a common scenario in drug design that has inspired both academic developers as well as software suppliers to create specialized fragment growing software [12–15].

Structure-based fragment growing software is often based on docking methodology [8, 10, 13], which was initially developed with full-sized ligands in mind. Furthermore, there is very little consensus on how fragment growing should be validated. A few attempts have been made to standardize the validation of docking fragments into empty pockets [16, 17], but fragment growing validation is performed heterogeneously. This leads to a situation where it is unclear how appropriate and successful individual methodologies used for fragment growing actually are.

✉ Matthias Rarey
matthias.rarey@uni-hamburg.de

[1] ZBH - Center for Bioinformatics, Universität Hamburg, Bundesstr. 43, 20146 Hamburg, Germany

[2] Institut de Recherches Servier, 125 Chemin de Ronde, 78290 Croissy-sur-Seine, France

[3] BioSolveIT GmbH, An der Ziegelei 79, 53757 Sankt Augustin, Germany

[4] BioSolveIT GmbH, An der Ziegelei 79, 53757 Sankt Augustin, Germany

Published online: 22 August 2022

In this work we will describe our fragment growing workflow FastGrow based on the Ray Volume Matrix (RVM) shape descriptor [18], a pharmacophoric interaction description, and JAMDA geometry optimization [19]. Its features will be statistically evaluated on a previously reported data set of fragment growing steps extracted from crystallographic data [18] and compared to DOCK, a well-known, open source docking suite [20].

Furthermore, we will demonstrate FastGrow's capabilities in the context of an FBDD campaign on the target DYRK1A (Dual Specificity Tyrosine-phosphorylation-regulated Kinase 1A). DYRK1A is a kinase implicated in various forms of cancer, neurodegenerative disease, and Down's Syndrome. We will focus on the publications by Walmsley et al. [21] and Weber et al. [22].

Methods

FastGrow workflow

The FastGrow workflow is a combination of several recognizable or previously described features that in combination facilitate efficient structure-based fragment growing. Beginning at pose generation and scoring those poses with an empirical scoring function, FastGrow is also capable of searching with interaction constraints and built-in ensemble flexibility.

Ray volume matrix pose generation

FastGrow is primarily based on the Ray Volume Matrix (RVM) shape descriptor [18]. RVM shape screening is a fast way to generate accurate poses for thousands of fragments in a few seconds. In short, it uses a shape description symmetric to both pockets and fragments, to perform rapid comparisons and orient fragment conformations in a binding site. The input is a pre-calculated fragment database and a fragment bound to its target. The RVM is very fast and generates accurate poses, but it is limited to shape comparison.

JAMDA scoring and optimization

A remedy to this limitation is the inclusion of an interaction aware scoring function. Here we chose the JAMDA scoring function [19], an empirical scoring function that is modeled after well-known scoring functions such as PLANTS [23], ChemScore [24], and the original Böhm scoring function [25]. Thus, the JAMDA scoring function contains many similar score contribution terms. Its main novelty is its limited step length gradient-based optimization, which results in stable and consistent geometry optimizations.

In FastGrow JAMDA can perform the role of ranking fragments in the final output hit list and optimizing the poses that the RVM search produces, especially with respect to intermolecular interactions. JAMDA has several common interaction terms and can compensate for minor orientation errors in interacting groups with geometry optimization. FastGrow mostly performs restrained JAMDA geometry optimization. A restrained JAMDA geometry optimization tries to keep the position of the input core more or less the same as it optimizes the extensions. This is achieved with a quadratic penalty term that is applied if a core moves more than 0.5Å away from its input position.

Interaction constraints

A set of optional interaction constraints, which are modeled after generic pharmacophore features, can be used to guide the pose generation and filter fragments that cannot fulfill all interactions. The pharmacophore features can encode a number of types, most prominent of which are hydrogen bond donors, hydrogen bond acceptors, and hydrophobic points. Hydrophobic points refer to geometric points that abstract hydrophobic complementarity in a pharmacophoric way. Interaction constraints are represented by a point with a type and a tolerance radius. They will be referred to as search points. This is one of the most effective ways through which the user can directly interact with the workflow.

In the FastGrow web application, hydrogen bond acceptor and donor search points are generated according to interaction geometries defined by Nittinger et al. [26]. Hydrophobic search points are generated based on the definition in JAMDA [19], which is itself based on the ChemScore definition [24]. Search points are built from the predicted protein-ligand interactions and are placed on the ligand side of the interaction.

Ensemble flexibility

An input fragment can be positioned in multiple aligned binding sites so that it can be simultaneously screened against a database of fragments. This means multiple conformations of amino acid side chains and even backbone movements can be mimicked when generating the growing hits. When a fragment is scored against an ensemble only the best score of a fragment with respect to a member of the ensemble is used to position it on the hit list. This means that if a fragment conformation clashes with an amino acid side chain in three out of four conformations of the binding site, then the score of the fourth non-clashing, and by implication highest scoring, binding site will be used.

Feature validation

All statistical validation was done on either the cross-growing data set that was established in the previous paper on the RVM [18] or subsets of this data set selected for specific properties of the protein-ligand complexes.

The test cases of the cross-growing set simulate growing one ligand in a PDB structure using only the structural information of another, related PDB structure and its ligand. Both of these ligands are crystallized in the same binding sites, measured by sequence identity, and have a common core structure. The difference between them is one substituent/fragment with one single bond to the common core. The PDB structure of one ligand is used to create the test ligand by cutting it down to the common core and attaching the necessary fragment. The pose of the test ligand can then be compared to the reference crystal structure of that ligand, which has remained unused until this point. Around 300 such cases were generated for the PDBbind refined set v.2019 in the original publication, which contains more detail about the generation procedure [18].

The main evaluation metric used was the atom RMSD and whether it was above or below the conventional threshold of 2Å. Only the RMSD of the fragment atoms was measured. Confidence intervals were estimated by exploiting the binomial nature of a binary less than 2Å RMSD classifier, which can be approximated by a normal distribution. Further information on statistical methods can be found in Sect. 2 of the Supplementary Information. All feature specific validation test cases were compared to corresponding test cases in the cross-growing set.

Maintaining interactions

The real-world use case that was simulated by the interaction test cases is that a FastGrow user aims to maintain an important interaction when replacing or extending a substituent due to previous experience or external information. To this end, interactions were generated using the model discussed in the Interaction Constraints Sect. [19, 26] for the test cases of the cross-growing set. These were used as input and then regenerated later for the resulting pose. Interactions generated for the reference structure and the resulting pose were compared to see whether the input interactions could be maintained.

To ensure these interactions were stable across both binding site structures included in the cross-growing, search points were generated from interactions in the growing binding site and the reference binding site. The JAMDA/ChemScore [19, 24] definition of hydrophobic interactions is quite permissive and may lead to many hydrophobic search points. In validation we only consider fully hydrophobic rings and terminal hydrophobic groups. A search point that was generated in one binding site structure was considered stable if a search point of the same type could be generated in the other structure within 2Å of it. The generated search points were then available as an input to FastGrow. A comparison was then performed of growings of FastGrow without search point information, with search point information, and with additional restrained optimization. The resulting poses were compared with respect to how well they maintained the interactions by also generating search points for the grown poses. If the grown poses regenerated the search point within 2Å of the input search point, they had succeeded in maintaining the interaction.

Water replacement

In the second case that was used to validate interaction constraints, a user utilizes water molecules visible in crystallographic structures to either create new hydrogen bond interactions or to simply "push" a water out of a binding site. A subset of cross-growing test cases was extracted by checking whether a water was replaced in the course of the growing. This was detected by calculating van der Waals (vdW) radii overlaps between waters in the binding site that was used for growing and the ligand to be grown. If the ligand to be grown and a water exceeded a 60% vdW overlap threshold, the water was considered to have been replaced by the ligand to be grown. Search points were generated for replaced waters and the ligand. If a search point that was generated by a water molecule was within 2Å of a search point of the same type being generated by the ligand, then the search point of the water was used as a query for the water replacement growing. For the purposes of steric/hydrophobic water replacement, waters generated dummy hydrophobic interactions, in addition to the more physical hydrogen donor and acceptor interactions. The input for a water replacement was therefore the typical cross-growing input of a core in a binding site, as well as the search points generated by the replaced waters.

Handling binding site flexibility

To evaluate FastGrow's ensemble flexibility implementation, we simulated a scenario where a user generates a set of representative binding site conformations and uses these to perform a growing. RMSD clustered ensembles of binding sites from the PDB were generated for all test cases of the cross-growing set using SIENA [27]. SIENA was run in the "docking" configuration, which implies, for example, binding site sequence identity. SIENA output was limited to five binding site conformations using the built-in all-atom clustering. The SIENA query binding site was the input binding site of the cross-growing test case in question, not the reference binding site. A minimum of two binding site

conformations was necessary for a test case to be included in the ensemble flexibility subset. Note that the reference binding site containing the ligand to be grown is excluded from the binding site ensemble.

Comparison to docking

To establish FastGrow in the larger context of structure-based tools, it was compared to the well-known docking program DOCK [20]. DOCK is a high-profile [28] open-source suite of tools that has been validated in similar scenarios [20, 29]. The pose re-prediction capabilities of DOCK version 6.9 were compared to FastGrow using the cross-growing set in two configurations: a full flexible cross/re-docking of the ligand to be grown and a fixed anchor docking that receives the common core of the two ligands as an input, just as FastGrow does.

The protein-ligand complexes for docking were prepared in the same way as in the internal FastGrow workflow, which involves removing all crystal waters as well as molecules that clash with the input core from the binding site. Binding sites were re-protonated using protoss [30], which replaces the protonation scheme that is used by the PDBbind refined set. The active site for docking was defined as all atoms within 15Å of the native ligand. The binding site was chosen to be rather large so as to avoid any stability issues at its edges for the sphere generation. Spheres were generated by either the sphgen version included in the source code distribution or sphgen_cpp [31] when necessary. Those spheres within 10Å of the native ligand were selected for docking. Docking grids were generated using the GRID implementation that was provided with DOCK. Unless otherwise specified, all parameters were set to defaults originating either from the software package, DOCK publications [20, 29], and/or the DOCK fans mailing list [32]. The scripts that were used to automate this process are available at https://github.com/rareylab/dock_scripts. Only the actual call to DOCK after all input data had been pre-calculated was included in runtime measurements.

Anchored docking as implemented in DOCK handles input coordinates differently than FastGrow. FastGrow either freezes input coordinates or allows a minimal amount of movement in restrained JAMDA optimization. DOCK finds the largest rigid structure that is connected to a specified atom in the anchor. Unfortunately, as of DOCK 6.9 [33], it is not possible to rigidify structures manually, which means that in the worst case DOCK will still sample degrees of freedom in the input core. The atom specifying the anchor in our validation scenarios with anchored docking was always chosen to be the atom neighboring the linker atom. This was done to ensure the degrees of freedom of the growth vector were as comparable as possible to FastGrow. In the fragment growing enrichment case study the complete anchor was rigid, which meant that DOCK was not expected to sample any more degrees of freedom than FastGrow, and runtime comparison could be performed fairly.

DYRK1A case study

The first part of the DYRK1A case study simulated a fragment hit optimization from a micromolar fragment (PDB code: 7A4R) to a nanomolar ligand (PDB code: 7A5N). Three fragment growings were performed that roughly corresponded to three areas of optimization in the publication by Walmsley et al. [21]. The three fragment growings were performed with FastGrow and with DOCK by full re-docking, as well as anchored docking. The generated poses were compared and discussed.

The second part of the DYRK1A case study described screening libraries of fragments with FastGrow. The idea of this enrichment study was to measure how well FastGrow could pick out fragments known to be very active from a collection of generic fragments. Multiple high affinity ligands produced by collaborators at Vernalis and Servier contain a 2,6-diaminopyridine moiety. The diaminopyridine fragment from PDB code: 7AJW was extended by a collection of known active fragments generated from Servier/Vernalis ligands, as well as a collection of property matched generic fragments created by BioSolveIT [34].

Three workflows were used for growing: FastGrow's pose scoring, FastGrow with restrained JAMDA optimization, and anchored docking with DOCK. All methods had to generate a pose for all fragments before these poses could then be scored. The workflow was designed to measure both the quality of the pose generation, as well as the screening power. Only fragments for which all methods could generate a pose were included in the statistics. After all fragments had been scored, the enrichment of all workflows was first compared to the enrichment achievable by sorting with "Rule of Three" properties [35], which served as the baseline, and then to each other. Confidence intervals were calculated using bootstrap re-sampling similar to the procedure presented in Stein et al. [36]. Further information on statistical methods can be found in Sect. 2 of the Supplementary Information.

Results

The cross-growing set was regenerated on the PDBbind refined set v.2020 [37]. The procedure was the same as previously described [18], but resulted in more test cases due to the growth of the PDBbind. The previous cross-growing set contained 326 test cases and 155 unique fragments, whereas the new cross-growing set contained 425 test cases and 176 unique fragments.

The shape-based growing has improved slightly since the original publication [18] and recreated 66.8 ± 4.5% (95% confidence interval) of ligand poses at an RMSD less than 2Å. Restrained JAMDA optimization at 70.8 ± 4.3% showed minor improvements in RMSD (Fig. S1). As shown in the section below, the changes it performed seemed to be more beneficial to individual interaction geometries than to the pose as a whole.

The small position changes a restrained JAMDA optimization performed at the core atoms occasionally led to slightly higher RMSD. JAMDA was used in a restrained configuration that applied a quadratic penalty term to movement of the core atom. These are assumed to already be in the correct position, but a certain amount of movement was necessary for correct optimization. A full run of the cross-growing set took around half an hour without JAMDA optimization and three and a half hours with optimization, as can be seen in Table 1.

Maintaining interactions

Search points were generated for all 425 test cases of the cross-growing set. 252 of these test cases generated at least one interaction that was stable across both binding site conformations. 85 acceptor search points and 21 donor search points were found in the cross-growing set. There were almost five times as many hydrophobic search points (a total of 532), than there were donor and acceptor search points, even though only terminal hydrophobic groups and "fully hydrophobic" rings were included. This was largely due to the very permissive definition according to ChemScore [24] in comparison to the geometrically constrained hydrogen bonds. The largest contributors to the number of hydrophobic search points were hydrophobic ring systems.

In general, the pose generation performance of growings with search points was better than for those without (77.0 ± 5.2% vs. 64.3 ± 5.9%, see Sect. 4 of the Supplementary Information). It is, however, more meaningful to evaluate the interaction geometries themselves that growings with search points produce. Figure 1 shows what percentage of interactions could be maintained by the grown poses. The purely shape-based growing conserved 62.4 ± 12.0% of the hydrophobic interactions, but had significant difficulties maintaining hydrogen bond acceptor and donor interactions at 30.6 ± 9.8% and 28.6 ± 19.4%, respectively. Shape-based growing had no way of telling where to place hydrogen bond interactions because it did not receive any search point information and the RVM descriptor does not contain any electrostatic information [18].

FastGrow using search points and FastGrow using restrained JAMDA optimization as well as search points both outperformed purely shape-based growing in conserving acceptors. FastGrow using search points had a success rate of 55.3 ± 10.6% and FastGrow using search points with optimization had a success rate of 68.2 ± 9.9%. Furthermore, hydrogen bond donor functionalities, as the most geometrically constrained interactions in this set, seemed to profit the most from restrained JAMDA optimization after a search point query with 76.2 ± 18.2% of them being conserved. The upper bound to maintaining any interactions was the performance of the pose generation. Search points with JAMDA optimization showed a very similar ability to maintain interactions as the pose generation performance.

Water replacement

Water replacement test cases could be extracted for 162 cross-growing test cases. 81 cases generated one search point for the water replacement query. 58 cases generated

Table 1 Overview of the performance and runtime of FastGrow on the cross-growing set

	Success rate	Runtime [h]	Runtime per fragment [ms]
Growing	66.8 ± 4.5%	0:31	24
Optimized growing	70.8 ± 4.3%	3:16	158

FastGrow screens every unique fragment in the cross-growing set against every test case, which means the runtime per fragment is the full runtime first divided by 425 test cases and second divided by 176 unique fragments. Machine specifications can be found in Sect. 1 of the Supplementary Information

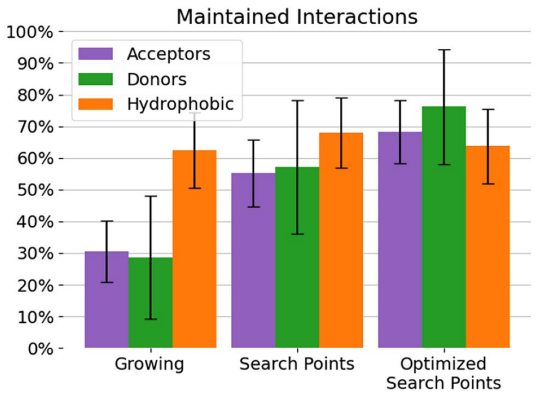


Fig. 1 Percentage of stable interactions that could be maintained in the cross-growing set by different methods. "Growing" represents purely shape-based growing. "Search Points" and "Optimized Search Points" represent growing with search points with or without subsequent restrained JAMDA optimization. The y-axis denotes what percentage of input interactions could be maintained during growing. The three most prominent interaction types are color coded according to the legend. The error bars are 95% confidence intervals

two search points, which implied that in these test cases two waters were replaced by the ligand to be grown. The highest number of waters replaced by a ligand was four. Figure 2 shows a few examples of the ligand to be grown overlapped with water in the binding site used for growing. While some of these waters were replaced by hydrophobic substructures, some were in positions that the ligand subsequently occupied to make directed interactions with the binding site.

Although there was a difference in pose generation between purely shape-based growing with a $64.2 \pm 7.4\%$ success rate and water replacement with subsequent optimization at $72.8 \pm 6.8\%$, it was not significant. The somewhat small effect may in part be a symptom of the comparatively few test cases. Water replacement does not seem to confer a significant general advantage and may be more useful in specific systems. Figure 3 shows a side chain of a quinolinone-6-sulfonamide derivative crystallized in PDB code: 6MA4 being grown into 6MA5. The purely shape-based growing without water replacement did not know about any of the interactions the carboxylic acid could make with the backbone and turned it away. Using the waters in the binding site led to the side chain fully stretching out and interacting with the backbone.

Ensemble flexibility

Ensemble test cases with at least two binding site conformations, not including the reference PDB, could be generated for 246 cross-growing test cases. Almost half of the ensemble test cases (120) contained five binding site conformations. 29 ensemble cases had only two binding site conformations. Except for five test cases, the average all atom RMSD of the ensemble binding sites to the SIENA

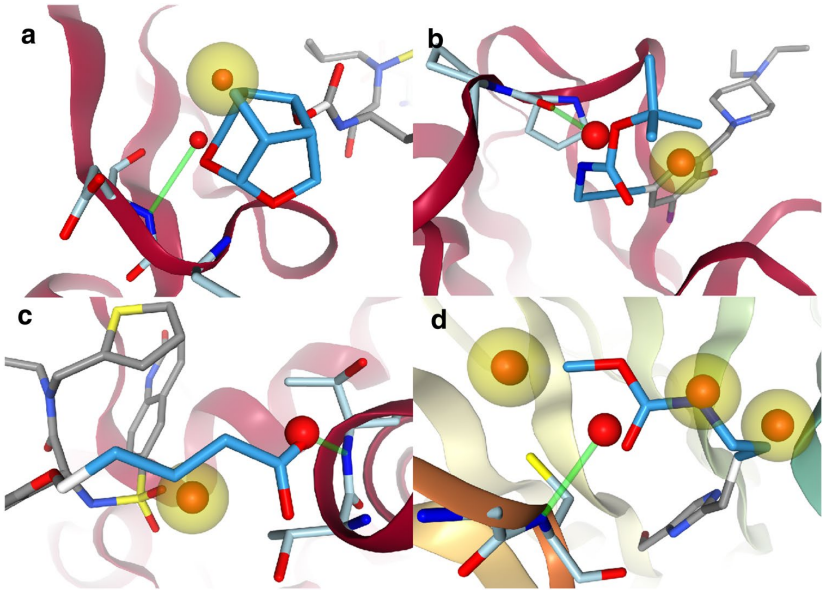


Fig. 2 Water replacement test cases that were extracted from the cross-growing set. The replaced waters are shown with the interactions that were used in the query to replace them. Yellow spheres are hydrophobic interactions and green cylinders hydrogen bond interactions. Binding site residues are light blue. The ligands are non-native to the binding site. The darker blue part of a ligand will be grown in a water replacement test case. **a** The ligand of 5ULT in 3GI6 (Gag-Pol polyprotein). **b** The ligand of 4AGO in 4AGM (Cellular tumor antigen p53). **c** The ligand of 6MA4 in 6MA5 (O-GlcNAc transferase). **d** The ligand of 6PGA in 6PG4 (WD repeat-containing protein 5). All 3D molecule images were made with the NGL viewer[38]

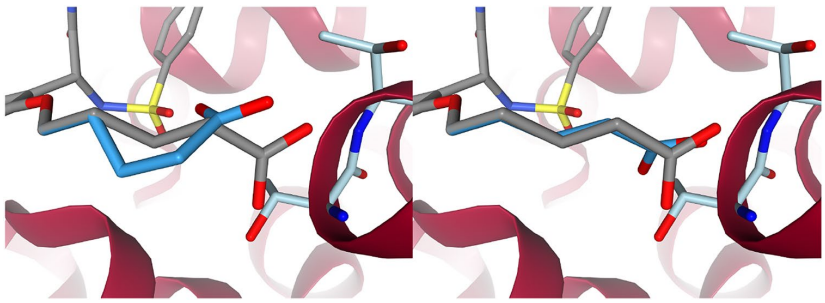


Fig. 3 Growing with and without water replacement. The purely shape-based growing is to the left and the one using water replacement is to the right. The binding site is from PDB code: 6MA5. The ligand of 6MA4 in gray is aligned to the binding site and used as a reference for RMSD calculation. The grown ligands are in a darker blue

query was between 0 and 1.5 Å distributed around a mean RMSD of 0.74 Å.

Automatically generated binding site ensembles did not show a strong statistical improvement of the pose generation performance (69.3 ± 5.7% vs. 75.2 ± 5.4%, see Sect. 6 of the Supplementary Information). It is well-known that ensembles tend to cause false positive hits in docking [39] and while many systems profited from ensembles, a similar amount generated poses with higher RMSDs.

When used appropriately, ensemble flexibility can make a significant difference [40]. Conformational changes of binding site residues may obstruct growing of a ligand bound in a different structure of that binding site. Figure 4 shows a shikimate precursor mimicking ligand from PDB code: 3N76 that was grown into 3N7A and an ensemble of 4B6O and 4KIU. FastGrow could not grow the phenolic substituent of 3N76 into the groove that was defined by 3N7A. It needed information about the flexibility of the loop and the movement of a conserved Tyrosine to calculate a successful pose with less than 2 Å RMSD.

Docking comparison

The pose re-prediction performance of FastGrow was compared to DOCK [20] on the cross-growing set. Only test cases where both methods could generate at least one pose were included in the statistics and only the top poses compared. Anchored docking could not generate poses for 12 test cases and FastGrow could not generate poses for 5 test cases. One of these test cases overlapped, so 16 (4%) of the 425 test cases were excluded. A discussion of why molecules failed can be found in Sect. 7 of the Supplementary Information.

Figure 5 shows the performance of DOCK cross-docking and anchored docking versus FastGrow growing and growing with subsequent restrained JAMDA optimization. Anchored docking with a success rate of 54.3 ± 4.8% outperformed cross-docking at 33.3 ± 4.6%, which is to be expected. Anchored docking received more input information, i.e., the core of the cross-growing test case, which significantly reduced the degrees of freedom in the system and therefore the potential for error. It is nonetheless an important point to make that using all the input information available can significantly impact the correctness of a prediction.

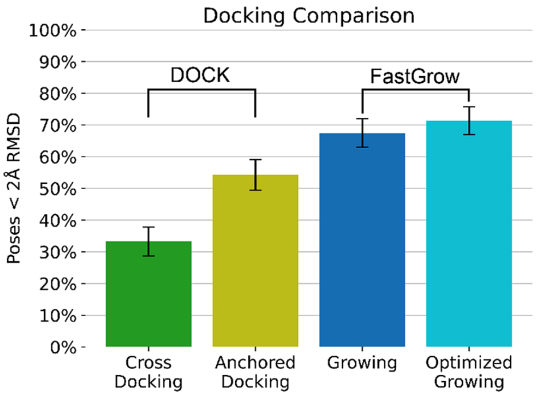


Fig. 5 Performance comparison of DOCK in a "Cross Docking" and an "Anchored Docking" to FastGrow with or without subsequent restrained JAMDA optimization. The error bars are 95% confidence intervals

Both growing at 67.5 ± 4.5% and growing with optimization at 71.4 ± 4.4% significantly outperformed anchored docking. It was unexpected to see FastGrow outperforming anchored docking. Both systems received the same input information and had been validated for this task. The difference probably arose in how sensitive both systems were to steric clashes with the binding site. FastGrow has a comparatively high clash tolerance in the initial pose generation, which produced many poses a typical docking workflow

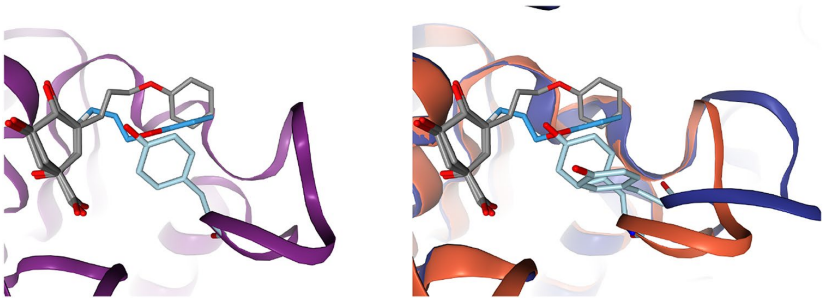


Fig. 4 Growing in a binding site with a highly flexible loop [41]. To the left in purple is PDB code: 3N7A. To the right are 4KIU in orange and 4B6O in blue. All are structures of the 3-dehydroquinate dehydratase. The conserved TYR24 is in light blue. The ligand atoms in darker blue were grown by FastGrow in the 4KIU/4B6O ensemble, which was not possible in 3N7A alone. The reference ligand from 3N76 is in gray

would have rejected [18]. Restrained JAMDA optimization then resolved these clashes with minor geometry adjustments. The permissiveness of the pose generation may have been able to sample the sometimes uncomfortable fit of a non-native ligand better than a more clash sensitive system.

DYRK1A case study

DYRK1A is one member of the Dual Specificity Tyrosine-phosphorylation-regulated Kinases. Its expression pattern suggests a role in the central nervous system [42], which supports its implication in neurodegenerative disease [43] and Down's Syndrome [44]. Furthermore, DYRK1A is suspected to be involved in some of the pathways that lead to an increased cancer risk for individuals with Down's Syndrome [45]. Servier and Vernalis recently published a collection of novel and highly active inhibitors for DYRK1A that were produced in a collaborative FBDD campaign in Walmsley et al. [21] and Weber et al. [22].

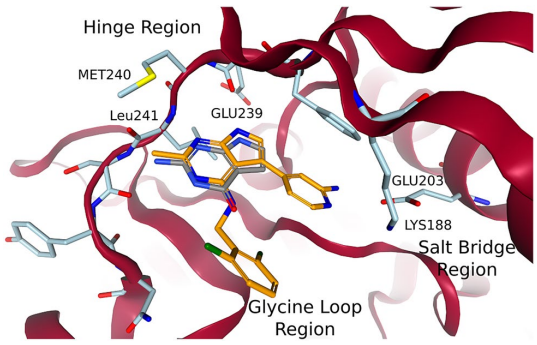


Fig. 6 Fragment 1 and compound 34 in a structure of DYRK1A. The structure and fragment 1 coordinates in gray are from PDB code: 7A4R and compound 34 in orange comes from PDB code: 7A5N and is aligned to the binding site using SIENA [27]. The cartoon for the residues GLU160 to ASP178 was removed for clarity as it is in Walmsley et al. [21]. All 3D molecule images were made with the NGL viewer [38]

Growing a ligand from a fragment

Walmsley et al. discovered fragment 1 (PDB code: 7A4R) as one of their micromolar hits (DYRK1A cKi 1.5 μM) from a fragment screening that was performed on DYRK1A using the Vernalis fragment library [21, 46, 47]. Toward the end of the publication they focused on compound 34 (PDB code: 7A5N), a nanomolar ligand (DYRK1A IC_{50} 7 nM) that they described as *"[...] a potent, in vivo-tolerated, selective inhibitor of DYRK1A kinase"* [21]. Both fragment 1 and compound 34 can be seen in Fig. 6. Both fragment 1 and compound 34 exhibited canonical hinge binding [48] and compound 34 retained the central ring core of fragment 1. Compound 34 could therefore be grown from fragment 1 by exchanging the amine close to the hinge to a methyl, growing into the salt-bridge region, and using the vector defined by the carbonyl at the ring of fragment 1 to address the glycine loop. These three steps were performed with FastGrow and DOCK.

Exchanging the amine of fragment 1 to a methyl was an important step towards the selectivity of compound 34. The uncommon position of the LEU241 carbonyl opened up space near the hinge, which was specific to DYRK1A [21]. Computationally, the switch from amine to methyl should be trivial. Figure 7 shows the poses that were generated by DOCK and FastGrow. Both the anchored docking approach using DOCK and FastGrow generated a realistic pose for fragment 1 with a methyl. Cross-docking with DOCK, which was not constrained by the position of the core atoms, generated a pose that drifted out of the pocket. Clearly, not using the available information of the core atoms led to a poorer quality pose. Although the change in RMSD is minor, several interaction geometries shifted into the unphysical range.

A hydrogen position at the pyrrole substructure of the pyrrolopyrimidine could be used to grow into the salt-bridge

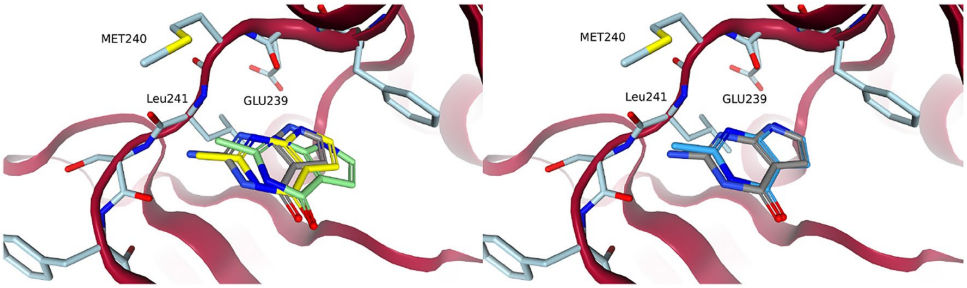


Fig. 7 Amine to methyl exchange of fragment 1. Fragment 1 from PDB code: 7A4R is in grey. To the left are poses that were generated by DOCK cross-docking in light green and anchored docking in yellow. To the right is the pose that was generated by FastGrow in a darker blue

region. The pyridine nitrogen of the aminopyridine to be placed there was expected to interact with LYS188, while the amine was expected to interact with GLU203. Figure 8 shows the poses generated by the three methods. Using an anchored docking or a cross-docking configuration led to significantly different poses in this case. The cross-docking generated a pose very close to the eventual orientation the aminopyridine has in compound 34 (i.e., PDB code: 7A5N). Despite receiving the coordinates of the full methylated fragment 1 as an input, the anchored docking optimized its pose out of the pocket. This could have been an overreaction of the underlying scoring function and optimization to minor clashes. Cross-docking did find a good pose, proving that both the scoring function and optimization *could* support one, however some strong effect perceived by the optimization led to a poorer pose for anchored docking.

All methods initially placed the amine of the aminopyridine away from GLU203. There are structures that support this as at least a reasonable position. Weber et al. modeled the ligand of PDB code: 7AJ2 with multiple conformations one of which has an aminopyridine that points away from GLU203. Most structures reported by Walmsley et al. and Weber et al., however, were modeled so that the aminopyridine pointed toward GLU203. We could achieve a pose more consistent with the other structures by using the position of the water HOH603 to estimate a reasonable position for the amine to interact with GLU203. HOH603 was then removed as usual in the growing process. We could also use the position of the amine in a different structure to achieve the same result. A search point with the correct type at that position guided FastGrow to place the amine near GLU203.

The last growing step involved replacing the carbonyl in the now modified fragment 1 with a difluoro-benzylamine, which interacts with the glycine loop. Figure 9 shows that all methods agreed on this step. The anchored docking, cross-docking, and FastGrow all reproduced the conformation of the difluoro-benzylamine of compound 34 in 7A5N.

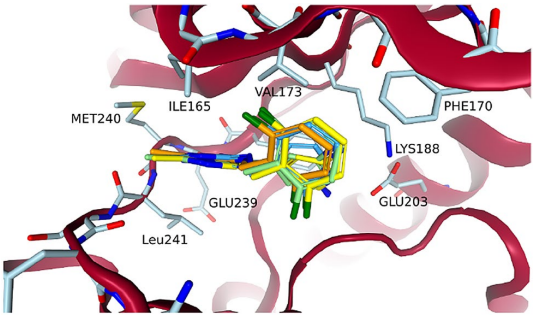


Fig. 9 Poses of compound 34 that were grown from fragment 1 in PDB code: 7A4R. In green and yellow are the DOCK cross-docking and anchored docking poses, respectively. The FastGrow pose is in a darker blue. The compound 34 reference structure aligned to 7A4R is in orange

Each of the three growing vectors considered above resulted in at least one series of compounds in the publication by Walmsley et al. [21]. While this case study was therefore reductive, it demonstrated the three methods: DOCK cross-docking and anchored docking, as well as FastGrow, in a practical application. Cross-docking generated good poses for the two larger modifications but exhibited unnecessary pose drift in the methylation. Anchored docking incorporated template information but seemed to have stability issues when confronted with clashes. FastGrow generated realistic poses for all three parts of the incremental growing. In one case the pose generation could be improved by using a crystallized water or external information to place a search point as guidance.

Fragment growing enrichment

A number of ligands from Walmsley et al. [21] and Weber et al. [22] had common cores that could be used to grow

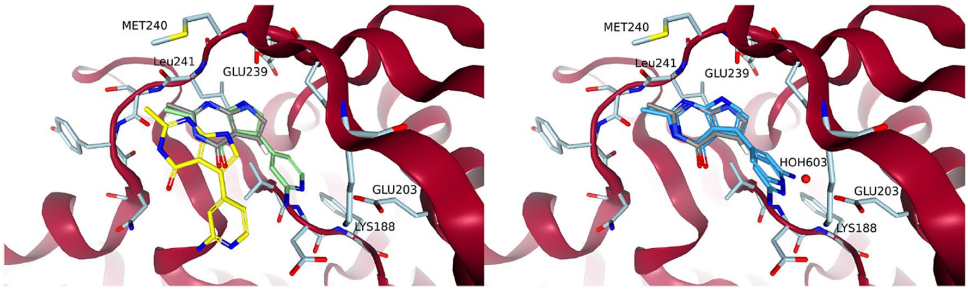


Fig. 8 Growing the aminopyridine into the salt-bridge region. The core of methylated fragment 1 that was used as an input to FastGrow and anchored docking is in grey. To the left are the poses that were generated by DOCK cross-docking in green and anchored docking in yellow. To the right are two poses produced by FastGrow in darker blue. The pose with the amine towards GLU203 was guided by placing a search point at the position of HOH603

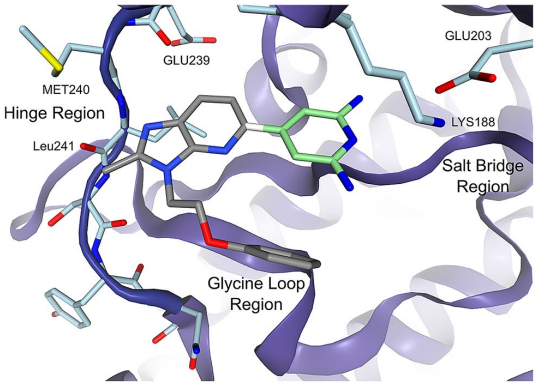


Fig. 10 The fragment screening enrichment core generated from PDB code: 7AJW. The diaminopyridine many of the ligands in the data set have in common is in light green. The bond between the gray and the green substructures is cut and the green structure used for growing. The growing direction will therefore be towards the hinge region

other ligands. The diaminopyridine moiety of the ligand in PDB code: 7AJW was used as the growing seed or core for the enrichment case study. Figure 10 shows the ligand of 7AJW with the diaminopyridine highlighted in green. Any fragment growing from this core would initially point directly to the hinge region.

Many ligands of DYRK1A, especially those from Weber et al. had a conserved diaminopyridine moiety. Filtering all diaminopyridine-containing ligands down to only the ones with affinities of less than 10 nM resulted in 77 ligands. These 77 ligands were fragmented at the bond to the diaminopyridine. The Rule of Three [35] properties of these fragments were inspected to detect outliers far outside of the property distribution. Nine fragments were discarded as outliers. There was a large group of fragments with exactly the same TPSA, which had to be downsampled to avoid biasing the screening. This lead to a collection of 40 highly active fragments. A comparison of properties before and after the filtering can be found in the Supplementary Information, Sect. 8.

The set of 40 known highly active single-digit nanomolar fragments was combined with a set of generic fragments assumed to be not as active. 2653 property-matched generic fragments were generated using the BioSolveIT fragment set [34]. The fragments were property matched using "Rule of Three" properties [35]. After property matching, sorting by molecular weight was chosen as a comparison baseline. Molecular weight retained some residual discriminative power despite property matching and was therefore chosen as a comparison baseline instead of an idealized null baseline. This is discussed further in Sect. 8 of the Supplementary Information. For evaluation we used the receiver operating characteristic (ROC) and the area under the curve (AUC). The ROC curves and AUCs of the three methods (FastGrow's pose scoring, FastGrow with restrained JAMDA optimization and anchored docking with DOCK) can be seen in Fig. 11.

All methods outperform the molecular weight baseline. The methods themselves all performed similarly and very well. Most surprising is that the FastGrow pose scoring function, which was never parametrized or evaluated for this purpose, performed almost on par with the other more sophisticated scoring functions. The FastGrow pose scoring function is made up of three terms: filled volume, number of close contacts, and clash [18]. Clash is the only term it shares in common with the other two scoring functions and therefore must be the driving force behind the general high performance. To substantiate this we repeated the experiment with all terms of the pose scoring function set to zero except for clash. Figure S7 shows that the pose scoring function with just clash performed comparably to the other scoring functions. 3D clash, and by extension shape complementarity, seemed to be a very discriminative property in this particular system, which leads to the high performance of all three of these methods.

Table 2 shows the time it takes for each of the three methods to generate poses for all fragments in the enrichment screening and score them. There was a factor five difference in speed between DOCK anchored docking and FastGrow with restrained JAMDA optimization. There was a factor

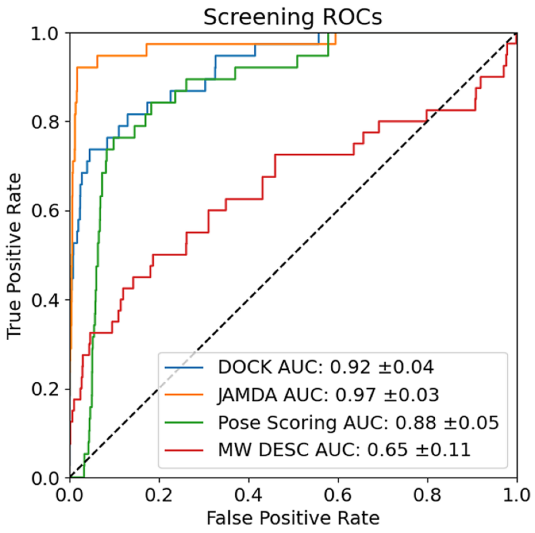


Fig. 11 ROC curves and AUCs of DOCK anchored docking, FastGrow with JAMDA, the FastGrow pose scoring function and sorting by descending molecular weight. The annotated errors on the AUCs describe a 95% confidence interval

Table 2 Runtimes of fragment growing enrichment for the three methods

Method	Screening time [s]	Runtime per fragment [ms]
FastGrow + Pose scoring	2	0.70
FastGrow + JAMDA	226	84.04
Anchored docking	1179	437.75

Screening time denotes a full screening with all 2693 fragments in seconds. Runtime per fragment is the arithmetic mean over all fragment runtimes in milliseconds. Machine specifications can be found in Sect. 1 of the Supplementary Information

500 difference in speed between anchored docking and using FastGrow's pose scoring function. Of the 2 s FastGrow and its pose scoring function spent screening, half was spent in the input-output operations of extracting 2693 fragments from the screening database and writing the hits. The hundred-fold increase in runtime between the simple FastGrow pose scoring function, which performed competitively in this case study, and the more sophisticated scoring functions was disproportional to the apparent gain in performance.

Conclusions

FastGrow is a novel and very fast approach for structure-based fragment growing. It achieved competitive pose re-prediction performance and enrichment compared to other well-known docking workflows but significantly outperformed these in speed. FastGrow can be used not only to screen larger collections of fragments, but also to reveal trends in fragment types and poses that are not visible in shorter hit lists. Besides the purely shape-based growing, FastGrow has been equipped with optional pharmacophore-like constraints, which can also be used to displace water molecules. Moreover, it may read in and work with an ensemble of protein structures to describe the flexibility of a target.

Our validation showed that we could maintain important interactions during growing. Success varied depending on the type of interaction, but modeling interactions as pharmacophore-like search points generally led to an improvement in pose re-prediction. Especially the more geometrically constrained interactions profited from restrained JAMDA optimization. We validated displacing waters by using those visible in crystal structures as hints for potential interactions, which improved pose generation in some scenarios. The ubiquitous tendency towards false positives in ensemble flexibility approaches meant that we could not find a pronounced statistical improvement when using ensemble flexibility. However, it was shown that some systems profited from or even required multiple structures to describe binding properly.

A growing case study on a DYRK1A FBDD campaign demonstrated the advantages and shortcomings of both DOCK and FastGrow in iterative growing. Using the position of a fixed fragment as input had clear advantages in both the statistical evaluation on the cross-growing set, as well as the case study. Anchored docking with DOCK encountered stability issues in the case study, which could not happen with FastGrow, due to its restrained optimization and the inherent stability of the JAMDA scoring function [19]. We could also demonstrate FastGrow's pose generation being guided by including external interaction information in the query.

The enrichment case study on DYRK1A demonstrated a scenario that could largely be solved by clash or in other words shape complementarity. A general statistical evaluation should however also include examples that require electrostatic complementarity in addition to shape complementarity. Building up a balanced dataset of such cases without biasing it by the methods to be evaluated is a significant challenge and beyond the scope of this work. A more general analysis will be necessary to address the lingering questions of how to incorporate shape and interactions into a scoring function of appropriate complexity.

As is often the case, the inclusion of external and inferred information is generally more successful than building a generalized model that encodes this information. It is for this reason we have focused on interactive, intuitive and quickly iterable approaches in FastGrow. Its ability to generate new ligands is comparable to established approaches and it would be interesting to see its sampling improved and compared to newer machine learning based generative models. FastGrow is already in use for current projects at Servier, as well as other organizations, and we hope it will enable further interesting results in the near future.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10822-022-00469-y.

Acknowledgements The authors would like to dearly acknowledge Pierre Ducrot for his enthusiasm in starting the project and Christian Lemmen for helpful discussions and support. We would also like to thank all users at Servier for early testing of the software.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability All data generated or analysed during this study is included in the supplementary information files.

Software availability The FastGrow web server is available at https://fastgrow.plus/. The code of the FastGrow web server is available at https://github.com/rareylab/fast_grow_server. Automation scripts for

DOCK execution are available at https://github.com/rareylab/dock_scripts. The core functionality is a part of the fragment-based "Inspirator Mode" in BioSolveIT's SeeSAR modeling package [49].

Declarations

Competing interests M.R. is a co-founder and shareholder of BioSolveIT. P.P. received funding from Servier.

References

1. Jahnke W, Erlanson DA, de Esch IJP, Johnson CN, Mortenson PN, Ochi Y, Urushima T (2020) Fragment-to-lead medicinal chemistry publications in 2019. J Med Chem 63:15494–15507. https://doi.org/10.1021/acs.jmedchem.0c01608
2. Erlanson DA, Fesik SW, Hubbard RE, Jahnke W, Jhoti H (2016) Twenty years on: the impact of fragments on drug discovery. Nat Rev Drug Discov 15:605–619. https://doi.org/10.1038/nrd.2016.109
3. Degen J, Wegscheid-Gerlach C, Zaliani A, Rarey M (2008) On the art of compiling and using 'drug-like' chemical fragment spaces. ChemMedChem 3:1503–1507. https://doi.org/10.1002/cmdc.200800178
4. Bajusz D, Wade WS, Satała G, Bojarski AJ, Ilaš J, Ebner J, Grebien F, Papp H, Jakab F, Douangamath A, Fearon D, von Delft F, Schuller M, Ahel I, Wakefield A, Vajda S, Gerencsér J, Pallai P, Keserő GM (2021) Exploring protein hotspots by optimized fragment pharmacophores. Nat Commun 12:3201. https://doi.org/10.1038/s41467-021-23443-y
5. Miranker A, Karplus M (1991) Functionality maps of binding sites: a multiple copy simultaneous search method. Proteins Struct Funct Bioinf 11:29–34. https://doi.org/10.1002/prot.340110104
6. Dennis S, Kortvelyesi T, Vajda S (2002) Computational mapping identifies the binding sites of organic solvents on proteins. Proc Natl Acad Sci 99:4290–4295. https://doi.org/10.1073/PNAS.062398499
7. Chevillard F, Kolb P (2015) Scubidoo: a large yet screenable and easily searchable database of computationally created chemical compounds optimized toward high likelihood of synthetic tractability. J Chem Inf Model 55:1824–1835. https://doi.org/10.1021/acs.jcim.5b00203
8. Sommer K, Flachsenberg F, Rarey M (2019) NAOMInext: synthetically feasible fragment growing in a structure-based design context. Eur J Med Chem 163:747–762. https://doi.org/10.1016/j.ejmech.2018.11.075
9. Bian Y, Xie X-Q (2018) Computational fragment-based drug design: current trends, strategies, and applications. AAPS J 20:59. https://doi.org/10.1208/s12248-018-0216-7
10. Rachman M, Piticchio S, Majewski M, Barril X (2021) Fragment-to-lead tailored in silico design. Drug Discov Today Technol 40:44–57. https://doi.org/10.1016/j.ddtec.2021.08.005
11. Joseph-McCarthy D, Campbell AJ, Kern G, Moustakas D (2014) Fragment-based lead discovery and design. J Chem Inf Model 54(3):693–704. https://doi.org/10.1021/ci400731w
12. Fragment Growing BioSolveIT. https://www.biosolveit.de/application-academy/fragment-growing/. Accessed 18 May 2022
13. Ligand Designer Schrödinger. https://www.schrodinger.com/products/ligand-designer. Accessed 22 July 2021
14. Grow and link fragments. https://www.cresset-group.com/discovery-services/specific-services/grow-and-link-fragments/. Accessed 22 July 2021
15. Molecular Operating Environment (MOE) MOEsaic PSILO. https://www.chemcomp.com/Products.htm. Accessed 22 July 2021
16. Chachulski L, Windshügel B (2020) LEADS-FRAG: a benchmark data set for assessment of fragment docking performance. J Chem Inf Model 60:6544–6554. https://doi.org/10.1021/acs.jcim.0c00693
17. Favia AD, Bottegoni G, Nobeli I, Bisignano P, Cavalli A (2011) SERAPhiC: a benchmark for in silico fragment-based drug design. J Chem Inf Model 51:2882–2896. https://doi.org/10.1021/ci2003363
18. Penner P, Martiny V, Gohier A, Gastreich M, Ducrot P, Brown D, Rarey M (2020) Shape-based descriptors for efficient structure-based fragment growing. J Chem Inf Model 60:6269–6281. https://doi.org/10.1021/acs.jcim.0c00920
19. Flachsenberg F, Meyder A, Sommer K, Penner P, Rarey M (2020) A consistent scheme for gradient-based optimization of protein-ligand poses. J Chem Inf Model 60:6502–6522. https://doi.org/10.1021/acs.jcim.0c01095
20. Allen WJ, Balius TE, Mukherjee S, Brozell SR, Moustakas DT, Lang PT, Case DA, Kuntz ID, Rizzo RC (2015) DOCK 6: impact of new features and current docking performance. J Comput Chem 36(15):1132–1156. https://doi.org/10.1002/jcc.23905
21. Walmsley DL, Murray JB, Dokurno P, Massey AJ, Benwell K, Fiumana A, Foloppe N, Ray S, Smith J, Surgenor AE, Edmonds T, Demarles D, Burbridge M, Cruzalegui F, Kotschy A, Hubbard RE (2021) Fragment-derived selective inhibitors of dual-specificity kinases DYRK1A and DYRK1B. J Med Chem 64:8971–8991. https://doi.org/10.1021/acs.jmedchem.1c00024
22. Weber C, Sipos M, Paczal A, Balint B, Kun V, Foloppe N, Dokurno P, Massey AJ, Walmsley DL, Hubbard RE, Murray J, Benwell K, Edmonds T, Demarles D, Bruno A, Burbridge M, Cruzalegui F, Kotschy A (2021) Structure-guided discovery of potent and selective DYRK1A inhibitors. J Med Chem 64:6745–6764. https://doi.org/10.1021/acs.jmedchem.1c00023
23. Korb O, Stützle T, Exner TE (2009) Empirical scoring functions for advanced protein-ligand docking with plants. J Chem Inf Model 49:84–96. https://doi.org/10.1021/ci800298z
24. Eldridge MD, Murray CW, Auton TR, Paolini GV, Mee RP (1997) Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. J Comput Aided Mol Des 11:425–445. https://doi.org/10.1023/A:1007996124545
25. Böhm H-J (1994) The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. J Comput Aided Mol Des 8:243–256. https://doi.org/10.1007/BF00126743
26. Nittinger E, Inhester T, Bietz S, Meyder A, Schomburg KT, Lange G, Klein R, Rarey M (2017) Large-scale analysis of hydrogen bond interaction patterns in protein-ligand interfaces. J Med Chem 60:4245–4257. https://doi.org/10.1021/acs.jmedchem.7b00101

27. Bietz S, Rarey M (2016) SIENA: efficient compilation of selective protein binding site ensembles. J Chem Inf Model 56:248–259. https://doi.org/10.1021/acs.jcim.5b00588
28. Lyu J, Wang S, Balius TE, Singh I, Levit A, Moroz YS, O'Meara MJ, Che T, Algaa E, Tolmachova K, Tolmachev AA, Shoichet BK, Roth BL, Irwin JJ (2019) Ultra-large library docking for discovering new chemotypes. Nature 566(7743):224–229. https://doi.org/10.1038/s41586-019-0917-9
29. Allen WJ, Fochtman BC, Balius TE, Rizzo RC (2017) Customizable de novo design strategies for DOCK: application to HIVgp41 and other therapeutic targets. J Comput Chem 38:2641–2663. https://doi.org/10.1002/jcc.25052
30. Bietz S, Urbaczek S, Schulz B, Rarey M (2014) Protoss: a holistic approach to predict tautomers and protonation states in protein-ligand complexes. J Cheminf 6:12. https://doi.org/10.1186/1758-2946-6-12
31. Andrew Magis, Peter Sayeski: sphgen_cpp. http://dock.compbio.ucsf.edu/Contributed_Code/sphgen_cpp.htm
32. dock_fans. https://groups.google.com/g/dock_fans. Accessed 7 Mar 2022
33. DOCK 6.9 User Manual. https://dock.compbio.ucsf.edu/DOCK_6/dock6_manual.html. Accessed 7 Mar 2022
34. Drug discovery with BioSolveIT apps–expect actives!. https://www.biosolveit.de/. Accessed 8 Mar 2022
35. Congreve M, Carr R, Murray C, Jhoti H (2003) A 'Rule of Three' for fragment-based lead discovery? Drug Discov Today 8:876–877. https://doi.org/10.1016/S1359-6446(03)02831-9
36. Stein RM, Yang Y, Balius TE, O'Meara MJ, Lyu J, Young J, Tang K, Shoichet BK, Irwin JJ (2021) Property-unmatched decoys in docking benchmarks. J Chem Inf Model 61:699–714. https://doi.org/10.1021/acs.jcim.0c00598
37. Liu Z, Su M, Han L, Liu J, Yang Q, Li Y, Wang R (2017) Forging the basis for developing protein-ligand interaction scoring functions. Accounts Chem Res 50:302–309. https://doi.org/10.1021/acs.accounts.6b00491
38. Rose AS, Hildebrand PW (2015) NGL Viewer: a web application for molecular visualization. Nucleic Acids Res 43:576–579. https://doi.org/10.1093/nar/gkv402
39. Amaro RE, Baudry J, Chodera J, Demir Özlem, McCammon JA, Miao Y, Smith JC (2018) Ensemble docking in drug discovery. Biophys J 114:2271–2278. https://doi.org/10.1016/j.bpj.2018.02.038
40. Claußen H, Buning C, Rarey M, Lengauer T (2001) FlexE: efficient molecular docking considering protein structure variations. J Mol Biol 308:377–395. https://doi.org/10.1006/jmbi.2001.4551
41. Dias MVB, Snee WC, Bromfield KM, Payne RJ, Palaninathan SK, Ciulli A, Howard NI, Abell C, Sacchettini JC, Blundell TL (2011) Structural investigation of inhibitor designs targeting 3-dehydroquinate dehydratase from the shikimate pathway of Mycobacterium tuberculosis. Biochem J 436(3):729–739. https://doi.org/10.1042/BJ20110002
42. Marti E, Altafaj X, Dierssen M, de la Luna S, Fotaki V, Alvarez M, Pérez-Riba M, Ferrer I, Estivill X (2003) Dyrk1A expression pattern supports specific roles of this kinase in the adult central nervous system. Brain Res 964(2):250–263. https://doi.org/10.1016/S0006-8993(02)04069-6
43. Wegiel J, Gong C-X, Hwang Y-W (2011) The role of DYRK1A in neurodegenerative diseases. FEBS J 278(2):236–245. https://doi.org/10.1111/j.1742-4658.2010.07955.x
44. Wiseman FK, Alford KA, Tybulewicz VLJ, Fisher EMC (2009) Down syndrome-recent progress and future prospects. Hum Mol Genet 18(R1):75–83. https://doi.org/10.1093/hmg/ddp010
45. Fernández-Martínez P, Zahonero C, Sánchez-Gómez P (2015) DYRK1A: the double-edged kinase as a protagonist in cell growth and tumorigenesis. Mol Cell Oncol 2(1):970048. https://doi.org/10.4161/23723548.2014.970048
46. Baurin N, Aboul-Ela F, Barril X, Davis B, Drysdale M, Dymock B, Finch H, Fromont C, Richardson C, Simmonite H, Hubbard RE (2004) Design and characterization of libraries of molecular fragments for use in nmr screening against protein targets. J Chem Inf Comput Sci 44(6):2157–2166. https://doi.org/10.1021/ci049806z
47. Hubbard RE, Murray JB (2011) Chapter twenty - experiences in fragment-based lead discovery. Academic Press, Cambridge, pp 509–531
48. Xing L, Klug-Mcleod J, Rai B, Lunney EA (2015) Kinase hinge binding scaffolds and their hydrogen bond patterns. Bioorgan Med Chem 23(19):6520–6527. https://doi.org/10.1016/j.bmc.2015.08.006
49. Fragment Growing BioSolveIT. https://www.biosolveit.de/SeeSAR/. Accessed 1 June 2022

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keinen anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Hamburg, den August 23, 2022

Patrick Penner