Development and Improvement of Tools and Algorithms for the Problem of Atom Type Perception and for the Assessment of Protein-Ligand-Complex Geometries

Dissertation

zur Erlangung des Doktorgrades
der Naturwissenschaften
(Dr. rer. nat.)

dem
Fachbereich Pharmazie der
Philipps-Universität Marburg
vorgelegt

von
Diplom-Pharmazeut
Gerd Neudert
aus
Hennigsdorf

Marburg/Lahn 2011

Erstgutachter	Prof. Dr. Gerhard Klebe, Institut für Pharmazeutische Chemie, Philipps-Universität Marburg
Zweitgutachter	Prof. Dr. Eyke Hüllermeier, Institut für Mathematik und Informatik, Philipps-Universität Marburg

Eingereicht am 16.12.2011

Tag der mündlichen Prüfung am 27.01.2012

Hochschulkennziffer: 1180

Die Untersuchungen zur vorliegenden Arbeit wurden auf Anregung von Herrn Prof. Dr. Gerhard Klebe am Institut für Pharmazeutische Chemie des Fachbereichs Pharmazie der Philipps-Universität Marburg in der Zeit von Juli 2005 bis Juli 2011 durchgeführt.

Meinen Eltern

Deutsche Kurzfassung

Ein wichtiger Grundstein der modernen Arzneistoffforschung ist das so genannte *High Throughput Screening*, bei dem Moleküle riesiger Substanz-Bibliotheken automatisiert in einfachen *in vitro*-Systemen auf ihre biologische Wirkung getestet werden. Ziel dieses Prozesses ist das Auffinden neuer Leitstrukturen, die zu neuen Arzneistoffkandidaten weiterentwickelt werden können.

Diese experimentelle Methode hat drei nennenswerte Nachteile. (i) Trotz der Möglichkeiten, welche die kombinatorische Chemie bietet, ist die Diversität der getesteten Verbindungen relativ beschränkt. (ii) Leitstrukturen mit einem hohen Optimierungspotential werden unter Umständen nicht erkannt, weil ihre Wirkung unterhalb der Empfindlichkeit des Testsystems liegt. (iii) Die finanziellen Kosten skalieren linear mit der Menge getesteter Substanzen.

Aus diesen Gründen hat das sogenannte *Virtual Screening* in den vergangenen Jahren zunehmend an Bedeutung gewonnen. Hierbei werden Substanz-Datenbanken beliebiger Größe und Diversität im Computer generiert. Ist die dreidimensionale Struktur eines Zielproteins bekannt, können die Moleküle einer erstellten Datenbank durch so genanntes *Docking* in die Bindetasche eingepasst werden.

Beim Docking wird für jede einzelne Substanz eine Vielzahl verschiedener Protein-Ligand-Konformationen erzeugt, unter denen im Falle aktiver Liganden oftmals auch ein nah-nativer Bindemodus ist. Voraussetzung für die Auswahl der "korrekten" Konformation sowie für das anschließende Ranking der verschiedenen Moleküle ist eine möglichst genaue Affinitätsvorhersage, die jedoch nur mit extrem hohem Rechenaufwand möglich ist. Deshalb wurden Bewertungsfunktionen auf Basis einfacherer Modelle entwickelt, die eine schnelle jedoch ungenauere Affinitätsabschätzung ermöglichen. Meist wird durch Re-

gressionsverfahren an einem Referenzdatensatz eine empirische Korrelation zwischen Affinität und leicht bestimmbaren physiko-chemischen Eigenschaften hergestellt. Aus theoretischen Überlegungen zu den thermodynamischen Grundlagen ist eine exakte Bestimmungen der Bindungsenergie mit solchen Modellen prinzipiell unmöglich.

Im Rahmen der vorliegenden Arbeit wurde deshalb eine neue Bewertungsfunktion entwickelt die nicht auf eine Affinitätsvorhersage abzielt, sondern auf die Erkennung nah-nativer Protein-Ligand-Geometrien, welche dann rechenintensiveren Methoden zur Energiebestimmung zugeführt werden können. Das entwickelte Programm *DSX* beruht auf dem Formalismus der wissensbasierten Funktion DrugScore und nutzt somit das vorhandene Wissen aus kristallografischen Datenbanken, insbesondere die daraus abgeleiteten, Atomtyp-spezifischen Distanzverteilungsfunktionen. Es basiert jedoch auf neu definierten Atomtypen und wurde um zwei neue Arten von Potentialen zur Bewertung von Torsionswinkeln und (De-)Solvatationseffekten erweitert. Die Validierung von *DSX* wurde mit einem umfangreichen, Literatur-bekannten Datensatz durchgeführt, der den Vergleich mit einer Vielzahl anderer Bewertungsfunktionen erlaubt. *DSX* nimmt hierbei die Spitzenposition bei der angestrebten Erkennung nahnativer Bindungsmodi ein und gehört auch beim Ranking unterschiedlicher Verbindungen zu den besten Bewertungsfunktionen.

Neben der Definition neuer Atomtypen war ein entscheidender Schritt zur Entwicklung von *DSX* die automatische Zuweisung von Atomtypen. Zu diesem Zweck wurde ein leistungsfähiges Programmier-Framework entwickelt und anhand eines Literatur-bekannten Datensatzes validiert. Effizienz und Qualität der Atomtyp-Zuweisung übertreffen die Programme für die Vergleichsdaten zur Verfügung stehen. Um Wissenschaftler ohne Programmierkenntnisse von der Atomtyp-Erkennung profitieren zu lassen, wurde das Frontend *fconv* entwickelt. Anfänglich nur zur Konvertierung verschiedener Molekül-Dateiformate gedacht, wurde dieses Programm um eine Vielzahl im Wirkstoffdesign häufig benötigter Funktionen erweitert und der wissenschaftlichen Gemeinschaft als Open Source Projekt zur Verfügung gestellt.

Basierend auf den für *DSX* entwickelten Potentialen wurde eine Reihe wei-

terer Anwendungen entwickelt und implementiert: Das Programm *HotspotsX* ermöglicht die Berechnung günstiger Interaktionsfelder in Protein-Bindetaschen, welche als Startpunkt für Pharmakophormodelle sowie als Ideengeber für die Leitstrukturoptimierung genutzt werden können. *HotspotsX* wurde in verschiedenen Studien erfolgreich eingesetzt.

Das Programm *DSFP* ist eine Bewertungsfunktion, die keinen einfachen Gesamtscore für einen gegebenen Komplex berechnet, sondern bindetaschenabhängige Fingerprints erstellt. Diese können dann anhand der aus bekannten Kristallstrukturen abgeleiteten Referenzfingerprints bewertet werden.

In Kooperation mit einer anderen Arbeitsgruppe wurde das Programm *DSX_rota* entwickelt, das zur Auswahl geeigneter Aminosäure-Rotamere im Enzymdesignprogramm TransCent dient.

Zur Vorhersage stabiler Wassernetzwerke in Bindetaschen wurde das Programm *DSX_wat* entwickelt. Hiefür wurde ein heuristischer Ansatz gewählt, um aus DSX-Interaktionsfeldern exklusive Sets möglicher Wasserpositionen zu bestimmen. Gute Übereinstimmungen mit kristallografisch ermittelten Wasserpositionen konnten festgestellt werden und die Berücksichtigung der generierten Wassernetzwerke bildet einen von mehreren Ansatzpunkten für die zukünftige Verbesserung von *DSX*.

Contents

Abbreviations xvii

List of Figures xix

List of Tables xxiii

List of Algorithms xxv

List of Publications xxvii

I. The Program fconv 1

1. Introduction and Motivation 3

1.1. Availability 6

2. Method of Operation 9

3. Parsing of Important File Formats 13

3.1. MOL2 Files 13

3.2. PDB Files 14

4. Atom Type Perception 17

4.1. Known Approaches 17

4.2. Workflow 19

4.3. Assignment of Element Types 20

4.4. Assignment of Connectivity 22
4.4.1. Distance Dependent Probabilities from the CSD 22
4.4.2. Probabilistic Connecting and Disconnecting 26
4.5. Assignment of Initial Hybridization States 28
4.5.1. Ring Perception 28
4.5.2. Ring Planarity 37
4.5.3. Geometry Check 38
4.6. Optimization of Conjugated Systems 39
4.6.1. Non-bipartite Maximum Weighted Matching 40
4.6.2. Assigning Weights for the Matching 53
4.6.3. Matching for Triple Bonds 60
4.7. Aromaticity Detection 60
4.8. Assignment of Internal Atom Types 62
4.8.1. Changing Protonation States 64
4.9. Assignment of Bond Types 65
4.10. The Problem of General Ambiguity 66
4.11. Performance of the Atom Type Perception 67
4.11.1. Quality of Assignments 67
4.11.2. Computational Speed 69

5. Additional Exemplary Features of fconv **71**
5.1. RMSD Values, Alignments, and Substructure Search 71
5.1.1. Definitions 72
5.1.2. Method for Spatial Alignments 73
5.1.3. Functional Alignment and Derived Features 74
5.2. Triangulation-Based Binding Site Detection 94
5.2.1. Regular Triangulations 95
5.2.2. The Concept of Alpha Shapes 97
5.2.3. Determining the Cavities 100
5.2.4. Application 100
5.2.5. Further Development 100

6. Summary and Outlook **103**
6.1. Summary 103
6.2. Outlook 105

II. The Program DSX and Derived Applications **107**

7. Introduction and Motivation **109**
7.1. Virtual Screening 109
7.2. The Use of Scoring Functions 112
7.3. Classification of Scoring Functions 113

8. Theory **117**
8.1. Theoretical Background 117
8.2. Distance-Dependent Pair Potentials 120
8.3. DSX Pair Potentials 123
8.4. DSX Torsion Potentials 126
8.5. DrugScore SAS- and DSX SR-potentials 128

9. Methods **131**
9.1. Pair Potentials 133
9.2. Torsion Angle Potentials 137
9.3. SR Potentials 138
9.4. Ligand Relaxation 139
9.5. Volume Correction 139
9.6. Implementation Details 140
9.7. Test Sets and Validation 141

10. Results and Discussion **145**
10.1. Docking Power 149
10.2. Ranking Power 152
10.3. Scoring Power 155
10.4. Influence of Local Minimization 158

10.5. Runtime Performance 159

11. Exemplary Applications of DSX Potentials 161
11.1. The Program HotspotsX 161
11.2. The Programs TransCent and DSX_rota 167
11.3. The Program DSFP 168
11.3.1. Fingerprint Generation 169
11.3.2. Results 170
11.4. The Program DSX_wat 171
11.4.1. Introduction 171
11.4.2. Motivation 173
11.4.3. Method 173
11.4.4. Results and Implications 179

12. Summary and Outlook 183
12.1. Summary 183
12.2. Outlook 186
12.2.1. DSX 186
12.2.2. HotspotsX 187
12.2.3. DSFP 187
12.2.4. DSX_wat 188

III. Appendix 189

A. Appendix 191
A.1. Modified Octree Structure for Neighborhood Partition 191
A.1.1. Implementation 193
A.2. Atom Type Definitions 196

Glossary 207

Bibliography 213

Acknowledgements **229**

Declaration **231**

Abbreviations

Å	Ångström ($1\,\text{Å} = 10^{-10}\,\text{m} = 100\,\text{pm}$).
ASCII	American Standard Code for Information Interchange.
BFS	Breadth-First Search.
CIF	Crystallographic Information File format, defined by the International Union of Crystallography (IUCr).
CSD	Cambridge Structural Database.
DFS	Depth-First Search.
DLG	Docking LoG file; the output format of the program AutoDock.
GPL	GNU General Public License.
HETATM	Entries for atoms in a PDB file that are non-amino acid atoms.
LIE	Linear interaction energy.
MC	Monte Carlo.

MCES	Maximum common induced edge subgraph.
MCIS	Maximum common induced subgraph.
MD	Molecular Dynamics.
MO	Molecular orbitals.
MOL2	Tripos® MOL2 file format.
NMR	Nuclear Magnetic Resonance.
PDB	Protein Data Bank; Also used for the file format of the PDB.
PDBQT	The input file format for AutoDock 4 and higher.
RMSD	Root-mean-square-deviation.
RTI	Record Type Indicator.
SDF	Structure Data File format, developed by MDL(Molecular Design Limited).
SSSR	Smallest Set of Smallest Rings.
UML	Unified Modeling Language™, version 2 (specified by the Object Management Group).
vdW	van der Waals (a Dutch physicist).

List of Figures

1.1. Different interpretation of Sybyl types 4

2.1. General activity diagram for fconv 10
2.2. Class diagram for molecule structures 11

4.1. Work flow atom type perception 20
4.2. Assignment of element types 21
4.3. C-C bond lengths 25
4.4. C-N bond lengths 26
4.5. C-N bond lengths with amide type 27
4.6. Minimum cycle bases 29
4.7. Non-fundamental cycle basis 30
4.8. Ambiguity of minimum cycle bases 31
4.9. Example for ring perception algorithm 32
4.10. A false ring 34
4.11. Rings in the CSD sample 37
4.12. Simple system of double bonds 40
4.13. Naive example for the matching algorithm 44
4.14. Naive sequence of processed edges 46
4.15. Improved sequence of processed edges 47
4.16. Maximum number of terminal edges 50
4.17. Isolated cycle 51
4.18. Label and edge numbers for CSD structures 52
4.19. LOCGEP03 53
4.20. Double bonds in pyrrole 54

4.21. Double bonds in triazole 55
4.22. Double bonds in N-methylimidazole 55
4.23. Double bonds in uracil 56
4.24. Double bonds in guanine 56
4.25. methazolamide 59
4.26. azulene 61
4.27. Exemplary fconv types 63
4.28. Priority for fconv types 64

5.1. Functional and spatial alignment 73
5.2. Difference between MCIS/MCES 75
5.3. Connected and disconnected subgraphs 79
5.4. Stereochemistry 81
5.5. Valid and Invalid alignments 82
5.6. 1D vs. 3D alignment 85
5.7. Geometric representation of an amino acid 87
5.8. Worst case combinations 91
5.9. Similarity matrix 93
5.10. Voronoi cells 96
5.11. Alpha complex in 2D 98
5.12. Alpha shapes in 2D 98
5.13. Alpha shapes in 3D 99

8.1. Similar and dissimilar densities 124
8.2. Different torsion types 128

9.1. Threshold for clustering of density functions 136

10.1. Volume fractions for protein and ligand atoms 146
10.2. Comparison of symmetric and asymmetric pair potentials 147
10.3. Pair potentials for hydroxyl and ether oxygen 148
10.4. Example for torsion potentials 148
10.5. Example for SR potentials 149

10.6. Influence of local minimization 158

11.1. PKA complexed with fragment 163
11.2. PKA complexed grown fragment 164
11.3. PKA hotspots for nitrogen 165
11.4. PKA SAS 165
11.5. Fragment Hits 171
11.6. Clique problem 175
11.7. Possible water clusters 176
11.8. Predicted water 179
11.9. Scores of X-ray waters 180
11.10. Scores of predicted waters 181
11.11. Artificial water score 181

A.1. Principle of Quadtrees 193
A.2. Modified Quadtree 195

List of Tables

4.1. Elements considered for covalent bonds 23
4.2. Comparison with literature bond lengths 24
4.3. Default protonation states 65
4.4. Success rate for atom type perception 68
4.5. Computational speed for atom type perception 69

7.1. Gap between *best pose* and *top pose* for 7 docking programs 111

9.1. Setups for *DSX* validation 133

10.1. Success rates for *docking power* 151
10.2. Success rates for *ranking power* 153
10.3. Spearman correlations 154
10.4. Pearson correlations 156
10.5. *DSX* runtime performance 160

A.1. DrugScore atom types 196
A.2. fconv atom type definitions 197
A.3. Mappings of fconv atom types 202

List of Algorithms

4.1. Assign connectivities 28
4.2. Ring Perception 33
4.3. Maximum weighted matching 42
4.4. Reduce Labels function for matching 45
4.5. Assigning double bond weights 58
4.6. Aromaticity detection 62
4.7. Assign aromaticity 62
5.1. Atom compatibility hashes 78
5.2. Shape-based alignment 88
5.3. COMPATIBLE function 89
5.4. MERGE_COMPATIBLE function 90
11.1. Find water clusters 177
11.2. Merge water clusters 178

List of Publications

Articles

- Neudert G., Klebe G., (2011) *DSX: A Knowledge-Based Scoring Function for the Assessment of Protein-Ligand Complexes*, J. Chem. Inf. Model., 51(10): 2731-2745

- Neudert G., Klebe G., (2011) *fconv: Format Conversion, Manipulation and Feature Computation of Molecular Data*, Bioinformatics, 27: 1021-1022

- Behnen J., Köster H., Neudert G., Craan T., Heine A., Klebe G., (2011) *Experimental and Computational Active Site Mapping as a Starting Point to Fragment-based Lead Discovery*, ChemMedChem, 7(2): 248-261.

- Fischer A., Enkler N., Neudert G., Bocola M., Sterner R., Merkl R. (2009) *TransCent: computational enzyme design by transferring active sites and considering constraints relevant for catalysis*, BMC Bioinformatics., 10(1): 54.

- Ritschel T., Kohler P.C., Neudert G., Heine A., Diederich F., Klebe G. (2009) *How to replace the residual solvation shell of polar active site residues to achieve nanomolar inhibition of tRNA-guanine transglycosylase.*, ChemMedChem, 4(12): 2012-2023.

- Neudert G., Koch O., Pfeffer P., Englert L., Klebe G., *DrugScoreFP: Profiling Protein-Ligand Interactions using Fingerprint Simplicity paired with Knowledge-Based Potential Fields*, manuscript in preparation

Posters

- Neudert G., Klebe G., (2007) *An Enhanced Version of DrugScore: Improved performance using Pair Potentials derived for Novel Sets of Atom Types*, 4th Joint Sheffield Conference on Chemoinformatics; poster contribution

- Betz M., Neudert G., Klebe G. (2011) *Prediction and Scoring of Water Molecule Networks in the Protein-Ligand Interface*, Gordon Research Conference on Computer-Aided Drug Design; poster contribution

Part I.

The Program fconv

Chapter 1.

Introduction and Motivation

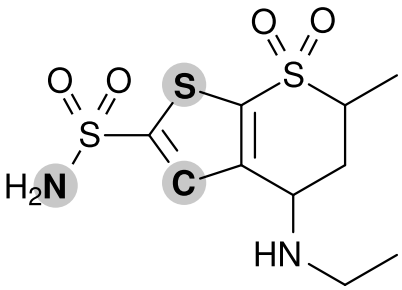
Applications dealing with molecule data, that are biomolecules and small molecules, need a consistent internal representation of these data and usually rely on a specific set of atom types. These atom types are used to calculate various model-based properties like partial charges, bond geometries or the potential within a force field for molecular mechanics. Furthermore, there are several specifications for the exchange of molecular data and a program must handle the most prevailing file formats commonly used in a particular research field. In case of the programs presented in this thesis, the most usual formats are PDB, MOL2, SDF, DLG and CIF.

Concerning the atom types, two different approaches are possible when designing a new application:

1. The program relies on the atom types as they are specified in its input files.

2. The program applies its own atom type perception to all input data.

A prerequisite for the first option is that the input file format makes use of exactly the same atom types that are required for the application. This is generally limiting the number of possible input formats and it forces users to utilize additional programs for conversion. For example, an application could rely on Sybyl atom types and therefore be limited to MOL2 file input.



Atom	Sybyl	OpenBabel	I-interpret
C	C.2	C.ar	C.ar
N	N.am	N.3	N.ar
S	S.3	S.2	S.2

Figure 1.1.: Ligand from PDB entry *'1cil'* on the left side with three atoms highlighted. On the right side, the Sybyl atom types of the highlighted atoms are shown after assignment by three different programs.

If input data are only available in PDB format, a PDB to MOL2 converter must be used. There are different such converters available, free software like OpenBabel[1], proprietary software like MN.CONVERT[2] and even complete modeling packages like MOE[3] or Sybyl®[4] that could accomplish this task. Atom type perception (e.g. to determine Sybyl atom types from PDB input) is a non-trivial task and in many cases it is just ambiguous. This ambiguity may be due to missing hydrogens or unusual geometries in the input files (see section 4.10), but even due to different interpretations of the atom type definitions. For example, Figure 1.1 shows Sybyl atom types of three distinct atoms, determined by Sybyl itself, OpenBabel and I-interpret[5] (Zhao et al., 2007). One could think that only the types set by Sybyl (which is maintained by Tripos®, the company who defined these types) are correct, but the problem simply is that the definitions[6] are rather imprecise. The official definition for the type 'C.2' is "carbon sp2" and for 'C.ar' it is "carbon aromatic". So OpenBabel and I-interpret are not wrong with 'C.ar', because the highlighted carbon is part of an aromatic system. However, the carbon is also sp2-hybridized and Sybyl uses 'C.ar' only for six-membered aromatic rings. In case of the

[1] `http://sourceforge.net/projects/openbabel`
[2] `http://www.molecular-networks.com`
[3] `http://www.chemcomp.com`
[4] `http://tripos.com`
[5] `http://www.sioc-ccbg.ac.cn/software/I-interpret`
[6] `http://tripos.com/mol2/atom_types.html`

nitrogen, the official definition for 'N.am' is "nitrogen amide", but does not specify whether this type should be used for carbon amides only or also for sulfonamides (as Sybyl does it). OpenBabel uses 'N.am' only for carbon amides and applies 'N.3',which is defined as "nitrogen sp3", to sulfonamides. I-interpret assumes the sulfonamide to be deprotonated by default and therefore assigns 'N.ar' due to the delocalized system (in contrast Sybyl assigns 'N.ar' in six-membered systems only and uses 'N.2' or 'N.pl3' in other aromatic systems). In consequence, there are many cases where different programs assign different atom types, which is a major drawback with respect to the consistency when relying on input files that were generated by other, arbitrary programs.

The second option, to not rely on the types as specified in the input files, is much better with respect to both robustness of an application's results and user friendliness. To realize the atom type perception as an integral part of a program, individual routines can be newly designed or existing software libraries can be used. Examples for the latter are the OpenBabel[1] developers framework (under GPL license) or the proprietary OEChem TK[7].

In addition to the atom type assignment problem, also the parsing of some input files is problematic. Especially in case of PDB files, many examples exist where the format specifications are not fulfilled, e.g. where mandatory entries are missing or used for other information (see section 3.2). For automatized large scale database processing, a program must be tolerant with respect to such errors in a file format and either fix them or handle them in a consistent way.

When starting the work that is presented in this thesis, a primary goal was to improve the scoring function DrugScore (Gohlke et al., 2000; Velec et al., 2005), which uses statistical potentials that are based on Sybyl atom types. Deriving new potentials that are based on an extended set of atom types was expected to yield better results (see section 8.3). DrugScore belongs to the first category mentioned in the beginning, hence it relies on the atom types given by the input files. Furthermore, it is very error-prone regarding misassignments in

[7] `http://www.eyesopen.com`

these input files and it is even limited with respect to the possible input formats. Considering these facts and the plans to extend DrugScore, the decision was taken to start with a completely new scoring function based on the DrugScore formalism, but implemented on entirely new code (which lead to the program *DSX* presented in Part II).

The next decision was to implement an own framework for atom type perception, file parsing and general handling of molecule data, because OpenBabel had a low quality with respect to correct atom type assignment and was not flexible enough for the intended extensions of atom types (meanwhile OpenBabel has improved with respect to both aspects). An additional motivation was the pursuit of being independent from third party libraries and licensing.

After the basic versions of the new atom type perception and file parsing libraries were realized, also a command line front-end was implemented with the aim to make the file conversion features accessible to users without programming skills. This front-end was called *fconv*, which stands for "**f**ile **conv**ersion". Driven by user demands and own needs, *fconv* was rapidly extended by many new functions. Examples are conversion and error correction of formats such as PDB/PDBQT, MOL2, SDF, DLG and CIF, extracting ligands from PDB as MOL2, automatic or ligand based cavity detection, RMSD calculation and clustering, substructure searches, alignment and structural superimposition, building of crystal packings, adding hydrogens and the calculation of various properties like the number of rotatable bonds, molecular weights or vdW volumes. The number of possible options (that is higher than the number of different tasks the program can solve) is over 90 today.

1.1. Availability

fconv is maintained as open source program under GPL license since its publication (Neudert and Klebe, 2011b) and is used at other universities and even by scientists in the pharmaceutical industry. It can be obtained from `http://www.agklebe.de/drugscore/fconv_download.php`, from where also

tutorials are available. Please note that *fconv* is a tool under constant development and thus, some details described in the above-mentioned paper may have changed in this thesis. By far not all functions and details of *fconv* will be described, hence the available source code is the only exhaustive reference.

Chapter 2.
2 Method of Operation

A generalized work flow of the *fconv* command line front-end is shown in Figure 2.1. The program accepts multiple files as input which are usually processed sequentially, hence one file is read in, then processed and after writing the output the next file is processed. In case of input data like multi-MOL2 or SDF, where several individual molecules are concatenated in one comprehensive file, these structures are handled in blocks of n molecules (default: n =5000), hence in the diagram, the term "next input file" also means "next block of molecules". This sequential approach is the favored way of processing, because it allows for files that are limited in their size only by the computer's file system. However, at the moment there are also some tasks that require to read all files at once before processing is possible. An example is the hierarchical clustering of input structures with respect to their RMSD values (option -clust), which requires an initial all-against-all calculation. An appropriate algorithm with sequential file parsing would require multiple read access on some of the molecules. In addition, an unlimited number of molecules would make it impossible to keep the calculated similarity matrix in memory and hence further slow down the computation.

Regardless of the task to perform, *fconv* generally translates the input molecules into objects of an adequate class hierarchy. In consequence, even for a trivial task like splitting a multi-MOL2 file into individual MOL2 files (option -s),

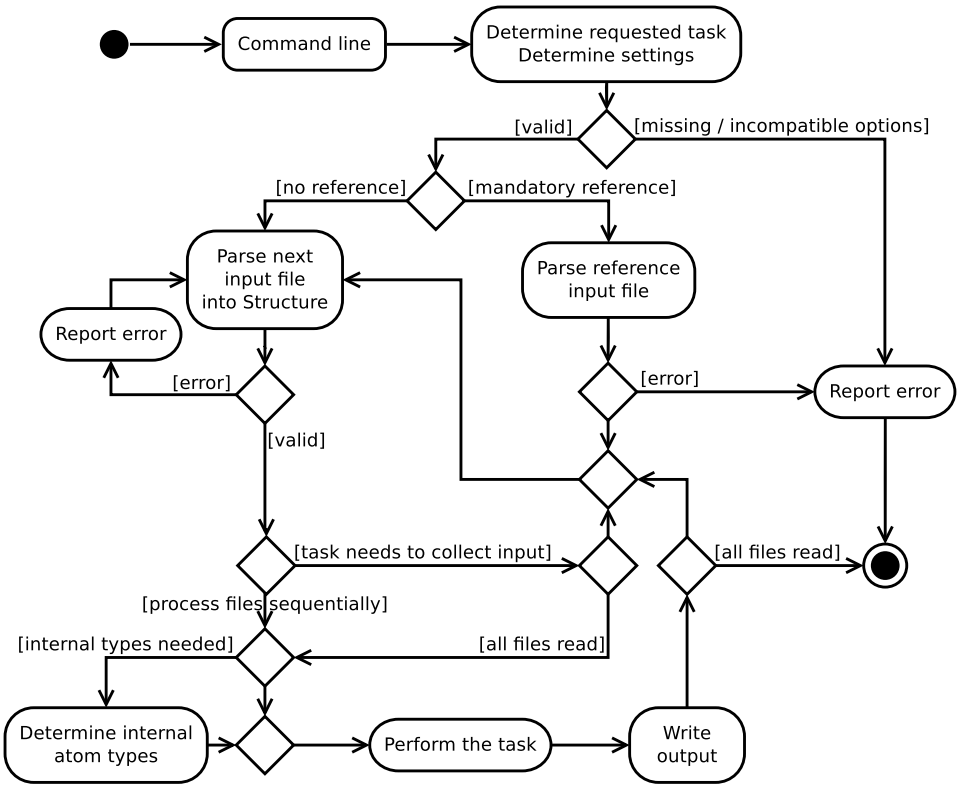


Figure 2.1.: UML activity diagram of a typical *fconv* work flow.

the program not just searches for the molecule delimiters (`@<TRIPOS>MOLECULE`) and cuts at these positions, but instead logically interprets all molecules. Although being slower in this case, the advantage is a consistent output of the program, which also includes an error correction. For example, if a user deleted an atom within one of the molecule entries, but forgot to also delete all bonds this atom took part of, the file would be corrupt for many programs. Also the removal of a bond entry without correcting all following bond indexes would result in a refuse to load the structure for many programs. *fconv* fixes such errors automatically when rewriting. Therefore, the mandatory translation into objects of an appropriate class hierarchy and back-translation when writing output files was a major design principle in the development of *fconv*.

The mentioned class hierarchy is illustrated by the corresponding class diagram in Figure 2.2. A **Parser** object is responsible for the translation “file

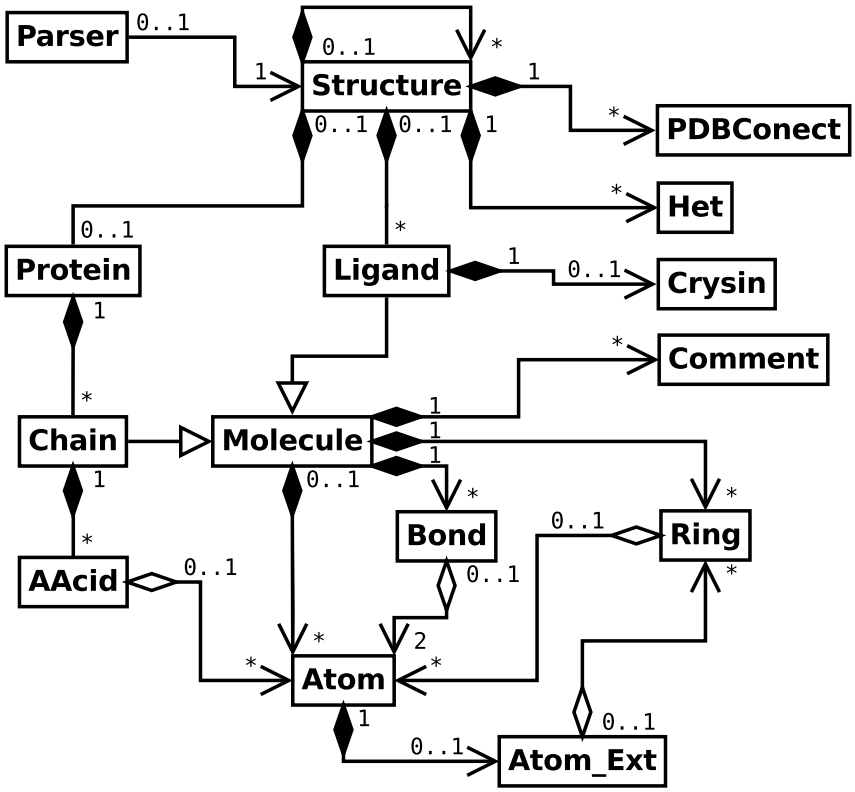


Figure 2.2.: UML class diagram of the hierarchical structure used to represent molecules. Several utility classes are left for simplification. Compositions with unspecified directionality are bidirectional.

to objects" and vice versa. It feeds a **Structure** object with the necessary data. The next level in the hierarchy are **Protein** and **Ligand** objects. A **Protein** is only generated in case of PDB input. Also additional objects, representing `HET` or `CONECT` entries, are generated for PDB files. For all other input formats (and also for the ligands within a PDB), **Ligand** objects are created. The **Ligand** class is derived from **Molecule**, as is the **Chain** class that represents a protein chain. A **Molecule** object consequently holds all essential informations about the molecular structure, that is **Atom**s, **Bond**s and **Ring**s. **Structure** objects can also hold a list with other **Structure** objects. This design was implemented to handle multiple `MODEL` entries within a PDB (e.g. in case of NMR structures), but can also be used to e.g. model substructures within a

structure or to model the life cycle of a changing structure.

The methods to apply the various tasks in *fconv* are implemented on **Structure**, **Protein** or **Molecule** level, where the latter performs the most relevant operations like atom type perception.

Chapter 3.

3 Parsing of Important File Formats

The *fconv* parser is able to read PDB/PDBQT, MOL2, SDF, DLG and CIF files, whereas PDB and MOL2 are the most important in context of this thesis and requests of *fconv* users. Thus only the latter two formats can be written out currently.

The complete specifications of the PDB format can be found following the "File Formats" link on the PDB website[1] and specification of the MOL2 format is available on the Tripos website[2]. In the following sections only some relevant aspects of these formats will be discussed.

3.1. MOL2 Files

The ASCII-based free MOL2 file format consists of an unlimited number of data records that are separated by a so called Record Type Indicator (RTI). Each record consists of a number of data lines that are following the RTI. Information within a data line is separated by whitespaces.

Molecules start with the RTI `@<TRIPOS>MOLECULE` followed by data lines

[1] `http://www.wwpdb.org/docs.html`
[2] `http://www.tripos.com/support`

that specify the name of the molecule, number of atoms and number of bonds. Other records that are interpreted by the *fconv* parser are `@<TRIPOS>ATOM`, `@<TRIPOS>BOND` and `@<TRIPOS>CRYSIN`. Also `@<TRIPOS>COMMENT` records can be parsed (option -V) being rewritten without modification. In addition, when writing out MOL2 files, *fconv* can also generate `@<TRIPOS>SUBSTRUCTURE` and `@<TRIPOS>DICT` records. Several other RTIs exist and there is no limitation to extend the format by new record types.

The most essential data is found in the `@<TRIPOS>ATOM` record. Each line defines an atom by ascending id, name, Cartesian coordinates and (Sybyl) atom type; optionally followed by a substructure id, substructure name and charge. Some programs refuse to accept MOL2 files where the optional data is missing. Therefore, even if optional information is lacking in the input, *fconv* generates default values when writing out.

Another major design principle for *fconv* is to rely only on essential data. For example, it ignores the number of atoms as specified in the `@<TRIPOS>MOLECULE` record. If this number is correct, then it is also redundant as it equals the number of lines in the `@<TRIPOS>ATOM` record. If it is incorrect, e.g. because a program deleted hydrogens without adjusting this information, then some programs would refuse the file due to this inconsistency. However, only the actual number of atoms in the file is of relevance, hence it is fine to ignore the discussed field.

When writing MOL2 files, all (except leading) whitespaces in atom names are replaced by '_', due to the whitespace separated data lines.

3.2. PDB Files

The ASCII-based PDB file format also consists of different record types that start with an identifier. In contrast to MOL2, a PDB record always consist of only one data line. These data lines are not separated by delimiters, but each data field has a fixed column or number of columns. From July 1998 until July 2007 the format version 2.3 was used. From 2007 on, versions 3.x are in use.

Up to now, there were three remediations of the PDB. The first was in August 2007, where changes with respect to the new format version 3.0 were applied. The most important change with respect to parsing and interpretation was the correction of many atom names in `HETATM` records. These names consist of 4 characters and in case of single character element (like C or N), the element must be at second position, while two character elements (like Cl or Mg) must be at position one and two in the name. Many entries violated this rule. For example the entry *'11ba'* had an atom name string '`C5*U`' for a carbon, which was corrected to '␣`C5U`'. Other entries had whitespaces inside the atom names, for example *'1aij'*, where the name '␣`N`␣`A`' was corrected to '␣`NA`␣'. The format also specifies a column for the element symbol only, but there were many entries that used the corresponding columns for other information. An example is *'1d49'*, where IDs were put into these columns.

The development of *fconv* file parsers was started before these changes and still many versions of the old, uncorrected files exist, because users processing large numbers of PDB entries usually have their own local (and possibly dated) copy of the PDB. Furthermore not all entries were corrected directly at release time of the new format, but rather in the following years. At least, invalid PDB files may be generated by other programs used in a particular work flow. In consequence, *fconv* tries to handle the described problems at the best (see section 4.3).

The second remediation of the PDB was in March 2009, where mainly new types of data records were introduced. It was also officially noticed that some ligands have invalid geometries, but could not be corrected as no agreement with available electron densities was possible. Examples for such invalid geometries, that are commonly found especially in older PDB entries, will be discussed in chapter 4. An obvious reason for their existence is the lack of an appropriate check of small molecules in the structure deposition process, which is in contrast to very detailed evaluation of the macromolecules and the corresponding crystallographic data.

The third remediation was in July 2011. Besides correcting some wrong occupancy values that were introduced in the 2009 changes and replacing

some incomplete coordinates a major change concerned the representation of oligopeptides and antibiotics. Up to that point they were represented inconsistently as polymer, as a single molecule or as a collection of molecules. Now, all oligopeptides that consist of only standard amino acids are to be represented as polymers and all others as single molecules. Although being an improvement, this decision emphasizes a general problem of the PDB format with respect to its usage in cheminformatics. It is aimed at crystallographic and biological semantics, but from a chemist's point of view the different representation in dependence whether an amino acid is proteinogenic or not is just an inconsistency that complicates automatized handling of these data. A related example is the usage of `ATOM` records instead of `HETATM` records, if a part of a ligand matches an amino acid. In addition, such `ATOM` records get individual residue identifiers, which makes it necessary to check for interatomic distances to find out whether `ATOM` records are connected to `HETATM` records and thus a part of a ligand.

Following the "use only essential data" design principle, *fconv* evaluates only a small number of different record types. Especially redundant data like the number of atoms is ignored (as in case of MOL2 files). `CONECT` records are evaluated with respect to validity and if so they are generated when rewriting a PDB. However, with respect to atom type perception *fconv* does not rely on these `CONECT` records, but only on element types and coordinates.

Chapter 4.
4 Atom Type Perception

The automatic assignment of atom types is the heart of *fconv* and basis for the majority of subsequently applied calculations. This chapter starts with a short overview of existing approaches to this problem followed by a more detailed description of the atom type perception algorithm used by *fconv*. Whenever chemical structures are shown, no hydrogens will be depicted. For example, $R-C-N$ instead of $R-CH_2-NH_2$ will be used. This corresponds to the usual case, where hydrogens are not present in the input files. Therefore, also for molecules where the true structure is known, the hydrogens will be neglected, simply because many of the following discussions would become pointless, if hydrogen information is available. To give an example, in case of RO as input, it is not trivial to decide whether it is $R-OH$ and $R=O$. But if the input is $R-OH$, there is no ambiguity.

4.1. Known Approaches

Several approaches for atom type perception have been reported in the last 20 years. Meng and Lewis (1991) developed the program *IDATM* which was aimed at the assignment of hybridizations and connectivities for structures as found in the Cambridge Structural Database (CSD) (Allen, 2002). They did not assign any bond orders and therefore, Baber and Hodgkin (1992) implemented

another program that was aimed at this task.

The algorithm used by *IDATM* was later implemented to the already mentioned OpenBabel. To assign atom types based on the initial connectivities and hybridizations, OpenBabel applies the PATTY algorithm that was developed by Bush and Sheridan (1993). This algorithm sets atom types based on pattern matching. Arbitrary patterns can be specified in a particularly designed language, making this approach very flexible with respect to possible differentiations in atom types. However, the quality of the assignment depends on the initially determined hybridizations.

Later, Hendlich et al. (1997a) developed *BALI*, which was designed to perceive atom types and bond orders for ligands from PDB files. *BALI* relies on `CONECT` records, which implies two drawbacks of this program. First, `CONECT`s are not present in all PDB files, especially when these are generated by other programs, and even if they are contained, they may be erroneous. Second, the binding to a file format specific data hinders an application of the algorithms to structures supplied via other input formats. The latter was not considered problematic, because *BALI* was developed as an integral part for what became later on the database Relibase(Hendlich et al., 2003), which was limited to PDB input.

In 2001, Sayle[1] reported a new algorithm that was again aimed at the atom type assignment for PDB ligands. On a very small test set he achieved much better success rates of correct assignments in comparison to all above mentioned programs.

Froeyen and Herdewijn (2005) translated the problem of assigning correct bond orders into an integer linear program to search for the Lewis formula that is in agreement with the Octet Rule (Langmuir, 1919) and has the lowest formal charge. A necessary part of their algorithm is the assignment of missing hydrogen topology for all permutations to calculate the formal charges. This approach is especially valuable for the generation of valid 2D-Lewis drawings.

Labute (2005) formulated the problem of bond order assignment as a maximum weighted matching for non-biparite graphs. The weight for each edge

[1] `http://www.daylight.com/meetings/mug01/Sayle/m4xbondage.html` (accessed 09/2011)

(bond) is calculated from bond length and a likelihood of both participating atoms to be part of a single/double/triple bond. These likelihoods are computed from analysis of 200 000 formulas from vendor catalogs, which means not from three dimensional coordinates but only from 2D bond information. In contrast to all other mentioned approaches, this algorithm works without the necessity of a ring detection. The author states that his algorithm has problems with unusual geometries, which are often found in the PDB.

Zhao et al. (2007), who published their work after the *fconv* atom type perception was implemented, use an approach very similar to that of Sayle and achieved the best results among all algorithms mentioned so far.

More details about these programs/algorithms, especially similarities and differences with respect to *fconv*, will be given in the following subsections.

4.2. Workflow

According to the previously referred "use only essential data" design principle, the atom type perception in *fconv* only requires element types and coordinates. The algorithm is implemented as a method of **Molecule** and after parsing an input file each **Molecule** object has a list of **Atom** objects. Figure 4.1 shows the general work flow and all steps will be explained in the following.

Instead of applying the complete assignment algorithm, it is also possible to take the atom types from a supplied reference molecule (`ref_mol`), and transfer them to all other input molecules. Optionally, also this reference can be atom typed initially (not shown in the diagram). This can be useful in case of multiple docking solutions of a compound, if it is necessary to assign identical atom types to all generated conformers. Working with FlexX (Rarey et al., 1996) and GOLD (Jones et al., 1995, 1997) it sometimes happened that these docking programs generated intramolecular distances between non-bonded atoms that were short enough to assign a bond between these atoms and thereby generating a new ring which in consequence lead to different atom types. Therefore, it is advisable to supply a compound with valid geometry

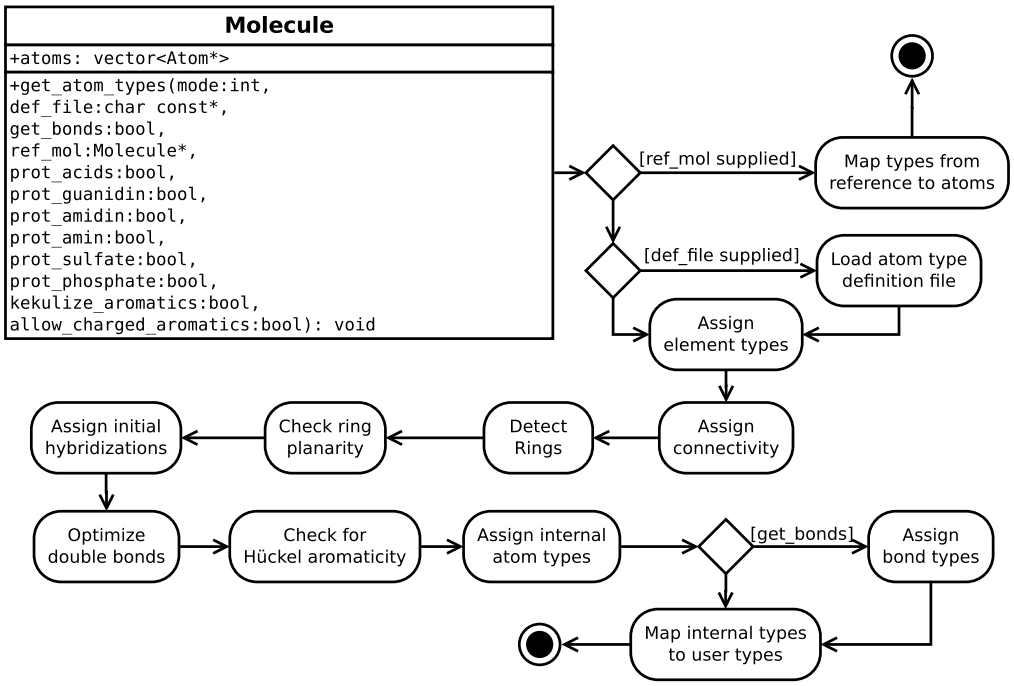


Figure 4.1.: UML activity diagram for the *fconv* atom type perception. Some details are left for simplification.

(usually the input geometry for the docking program) as reference, assign atom types to this reference and transfer these to all other docking solutions.

The option to supply an atom type definition file (`def_file`) will be discussed in section 4.8, so the first step of the atom type assignment is to set the element types.

4.3. Assignment of Element Types

In case of DLG or SDF input, the **Parser** has already set the element types for each **Atom** Object. SDF files do not explicitly contain this information, but types can be easily deduced from the recorded Sybyl types that were put in the corresponding attribute by the **Parser**. As mentioned in section 3.2, things are more complicated in case of PDB input. Thus, it is checked whether initial Sybyl types are assigned to the **Atom** objects; and if so, they are used

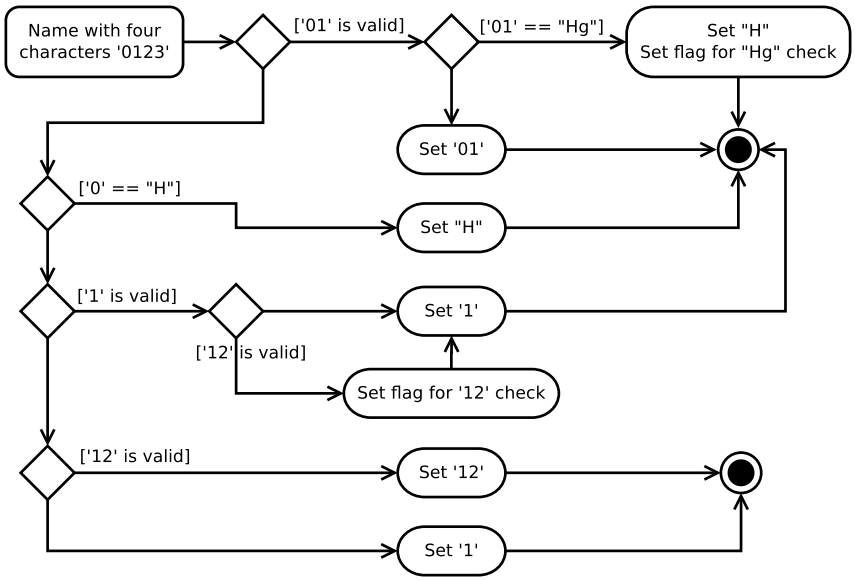


Figure 4.2.: UML activity diagram for the deduction of element types from PDB atom names. '01' represents the string that is formed by the first and the second character of the four character name. The term "valid" means that the formed string represents a valid element type.

to determine the element types. If they are not present, it is checked whether the element attribute holds an identifier for a valid element type. If this is not the case, the element type has to be deduced from the atom name. Figure 4.2 shows the algorithm that is responsible for this. Thereby, *fconv* uses a list of known element types that does not represent all chemically known elements but only the usual ones that are also found in PDB files. In case the "check flag" for mercurial or other '12'-two letter elements (calcium, iron, nickel, sodium or selenium) is set, only the first letter is used as (more often occurring) element and when assigning hybridizations (see section 4.5), it is checked whether the corresponding two letter element is more probable.

Unfortunately, it is still possible to fail when relying on the element type as specified in the input file. An example is PDB entry *'1dlf'* in the version before the remediation in 2007. The ligand had a sulfur with the correct name '␣S␣␣', but in the element column there was an 'O'. As 'O' is a valid element, *fconv* relies on it and does not use the atom's name to determine the element type.

In addition to the procedure in Figure 4.2, there are also some special rules (not shown) for the deduction from names of the often occurring molecules nicotinamide-adenine-dinucleotide, dihydro-nicotinamide-adenine-dinucleotide phosphate, and 2'-fluoroguanyl-(3'-5')-phosphocytidine that must be applied for older versions of PDB files. These molecules are identified by their three letter residue names ('NAD', 'NDP', and 'GPC') as specified in the input.

4.4. Assignment of Connectivity

To detect intramolecular connectivity, all of the approaches described in section 4.1 make use of elements' covalent radii as, e.g., tabulated by Allen et al. (1979). They assign a covalent bond between two atoms a and b, if

$$d(a,b) < r_c(a) + r_c(b) + t \tag{4.1}$$

where d is the Euclidean distance between the atoms, r_c the covalent radii and t a certain tolerance. Either 0.4 Å or 0.45 Å are used as tolerance within the different programs. Although it was early reported by Baber and Hodgkin (1992) that some covalent bonds are not recognized using the 0.4 Å threshold, most of the more recent approaches (except for Sayle) use this narrower limit. Applying a higher limit aggravates the problem of wrong connections, more precisely the problem to decide which bonds are wrong when the maximum valence of an atom is exceeded.

fconv employs a different approach using binned distance-dependent probabilities, derived from the CSD.

4.4.1. Distance Dependent Probabilities from the CSD

The CSD contains a huge number of high quality X-ray structures. A subset of 165 222 CSD compounds was used to derive statistics for the different bond types that are specified for the MOL2 format. The program ConQuest (Bruno et al., 2002) was used to assemble this set with the following search criteria:

Table 4.1.: Elements that are considered for covalent bonds in *fconv* and their maximal allowed valence.

Element types	max. allowed valence
S	6
P	5
C, N, Si, Se, As	4
B	3
O	2
H, F, Cl, Br, I	1

(i) only organic structures, (ii) complete coordinates determined and (iii) no error flags associated. The bond and atom types in the exported sample file are thus set by ConQuest.

Possible bond types are single-, double-, triple-, aromatic-, and amide bonds ('1', '2', '3', 'ar', and 'am' respectively; the format also specifies 'du' and 'un' for dummy and unknown bonds). The first three types are well defined, simply corresponding to the definition of single-, double-, and triple bonds in organic chemistry. The aromatic bond type however is assigned differently by different programs (examples will be discussed in section 4.9). The amide bond can be used for the bond between carbonyl carbon and the nitrogen in amides, but ConQuest assigns the single bond type instead. However, it is no problem to also retrieve statistics for amide bonds, because the amide pattern is easy to detect.

As *fconv* is designed for usage with pharmaceutical and biological structures it considers bonds only between the elements given in Table 4.1. In earlier versions, also covalent bonds to metals were possible. However, organometallic compounds are of minor importance in pharmaceutical chemistry and a differentiation between ionic and covalent metal bonds is difficult in many cases. Therefore, metals are consistently treated as ions and no bonds are assigned to them.

Throughout this thesis, the term "density function" is used rather often. Strictly speaking this term describes continuous functions and the discrete

Table 4.2.: Some bond length ranges as specified in the literature (Lide, 2009) in comparison to the observed bond lengths in the chosen CSD data sample. For the observed values the range corresponds to values > 0 in Equation 4.2. The value given in parentheses corresponds to the distance of $P_{\max}$.

	Bond length in pm	
Bond type	Literature	Observed
CC, triple	117-119	96-134 (119)
CC, double	130-135	102-165 (134)
CC, aromatic	136-142	121-156 (138)
CC, single	143-159	117-175 (151)
CN, triple	114-116	85-147 (114)
CN, double	128-133	104-152 (130)
CN, aromatic	133-137	119-151 (134)
CN, single	134-154	118-170 (146)

functions used here and in Part II are only estimates for the density functions. Thus, whenever speaking about density functions, estimates for such functions are meant. Here, such an estimate for the probability of a bond type t with bond length d between two atoms of element types i and j is calculated for a bin size of 1 pm.

$$P(i,j,t,d) = \begin{cases} \frac{N(i,j,t,d)}{\sum_{d'} N(i,j,t,d')} & N(i,j,t,d) \geq 0.0005 \cdot N_{\max}(i,j,t) \\ 0 & N(i,j,t,d) < 0.0005 \cdot N_{\max}(i,j,t) \end{cases} \tag{4.2}$$

In addition to the definition in Equation 4.2 a small minimum value is added to the functions in case of zero values that can occur in between the maxima of two bond types. Even for the rather narrow binning, the observed occurrences N result in very smooth curves. However, for usage in *fconv* the data was additionally smoothed using a Gaussian function with a sigma value of 1.5 pm.

Figure 4.3 shows an example for density functions as one would expect them from a theoretic point of view. For comparison, Table 4.2 lists some typical bond lengths as found in the literature, together with the distances sampled from the CSD data set.

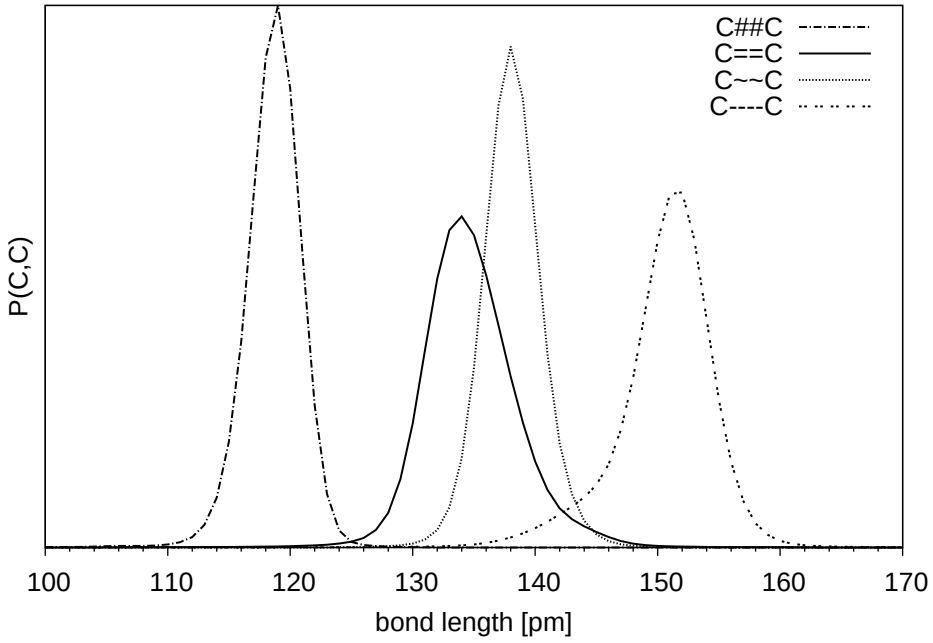


Figure 4.3.: Carbon-carbon bond lengths. From left to right: Density functions for triple bond (C##C), double bond (C==C), aromatic bond (C~~C) and, single bond (C----C).

The curve for C-C double bonds shows a tailing in direction to longer distances. Most likely, this is due to a number of double bonds that are part of a conjugated double-bond network. Depending on the particular composition of such a network, the bond lengths can be elongated to the range of aromatic bonds, because they have only partial double-bond character in delocalized electron systems. For the same reason, the curve for single bonds shows a slight tailing in direction to shorter distances, because in conjugated systems they gain partial double-bond character.

Another example is given in Figure 4.4, showing the derived density functions for carbon-nitrogen bonds. In case of the single bonds, there are two different maxima, indicating that at least two significantly different classes of bond types are merged into the single bond type. As mentioned above, ConQuest assigns the single bond type also to amide bonds. Using, the amide type instead results

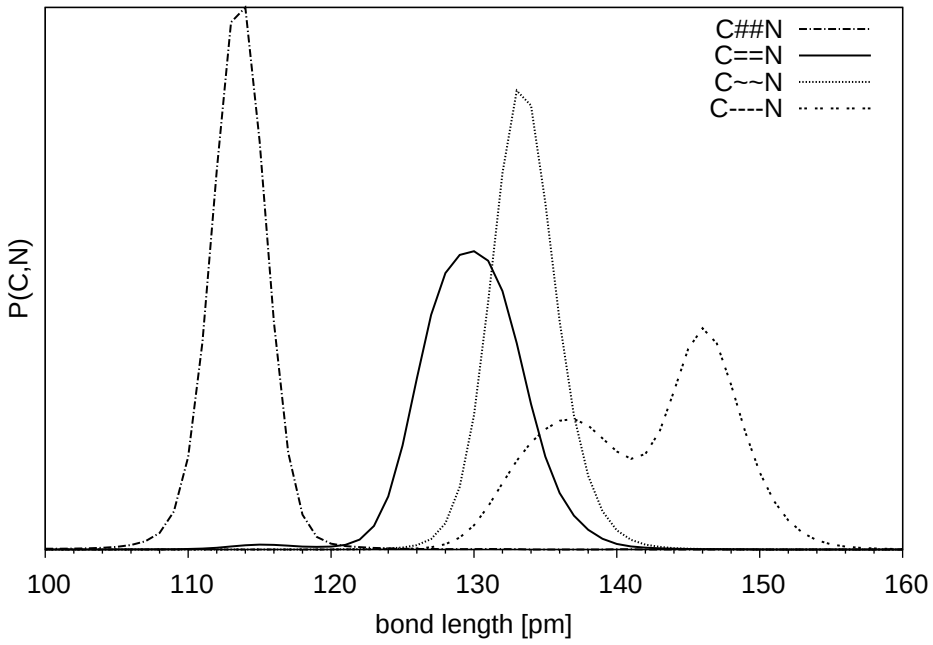


Figure 4.4.: Carbon-nitrogen bond lengths. From left to right: Density functions for triple bond (C##N), double bond (C==N), aromatic bond (C~~N), and single bond (C----N).

in Figure 4.5. Obviously, these bonds are most similar to the aromatic bond type and thus, it is advisable to separate them from the single bond type. When searching for the probability of a nitrogen to be sp2-hybridized, *fconv* uses the highest value from either the double, aromatic or amide type.

4.4.2. Probabilistic Connecting and Disconnecting

Alg. 4.1 shows how *fconv* assigns connectivities. The advantage of using probabilities as defined in Equation 4.2 is in the second part of the algorithm (lines 8-10). Determining the connection with the lowest probability is straightforward; in contrast programs using Equation 4.1 need to check for additional geometric features to determine the most probable bonded atoms.

The maximum valances that are used by *fconv* are given in Table 4.1. Involvement of d-orbitals in bonding is only considered for sulfur and phosphorus,

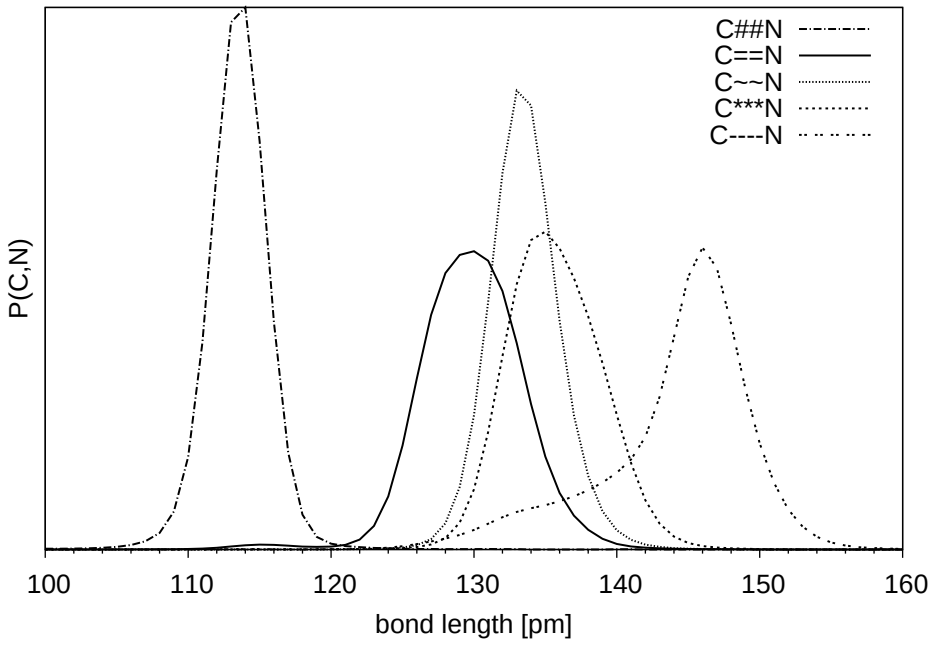


Figure 4.5.: Carbon-nitrogen bond lengths with amide bond type. From left to right: Density functions for triple bond (C##N), double bond (C==N), aromatic bond (C~~N), amide bond (C***N), and single bond (C----N).

leading to six (e.g. in SF_6) and five (e.g. in PCl_5) instead of only four valences, respectively. Of course, higher valences are also possible for silicon, arsenic, selenium, chlorine, bromine, and iodine, but not considered by *fconv*. They are of low relevance in biological structures and (in contrast to sulfur and phosphorus) would require additional geometric evaluations, e.g. not to assign bonds between chlorine in CCl_3 groups.

In line 3 of Alg. 4.1, iterating only over all neighboring atoms is an important speedup compared to the naive iteration over all atoms in A. The efficient data structure that is used to determine neighborhoods is presented in section A.1.

Algorithm 4.1: The algorithm used to assign connectivity data to the **Atom** objects. The maximum distance that is considered in the connectivity test is 2.6 Å. $P_{\max}$ denotes the maximum probabilities among all possible bond types.

Input : Set of indexed **Atom**s $A = \{a_1, \ldots, a_k\}$
Result: A with connectivity data

```
1  Let N_a := {b ∈ A|DIST(b, a) < max_dist} be the neighborhood of a ∈ A
2  forall a ∈ A do
3      forall b ∈ N_a do
4          if a.index ≤ b.index then next b
5          if P_max(a, b,DIST(a, b)) > 0 then
6              a.bonded_atoms := a.bonded_atoms ∪ {b}
7              b.bonded_atoms := b.bonded_atoms ∪ {a}
8  forall a ∈ A do
9      while |a.bonded_atoms| > a.max_valence do
10         delete connection (a, b) with lowest P_max(a, b, DIST(a, b))
```

4.5. Assignment of Initial Hybridization States

To assign initial hybridization states, *fconv* uses geometric constraints and information whether atoms are part of ring systems considering also the geometry of these rings. Using the connectivity information, the next step is therefore the detection of rings.

4.5.1. Ring Perception

The question of how to detect rings (synonymously cycles) in a molecular structure is intimately connected with the question of what type of rings should be detected by the application. The latter is not as easy to answer as one might think of it initially. An overview of different sets of rings, together with a solid graph theoretical definition can be found in the excellent paper by Berger et al. (2004). In that work, also drawbacks in the definitions and the correlated algorithms are discussed. The initial statement of this paragraph shall be underlined with a citation from the mentioned paper: “The different

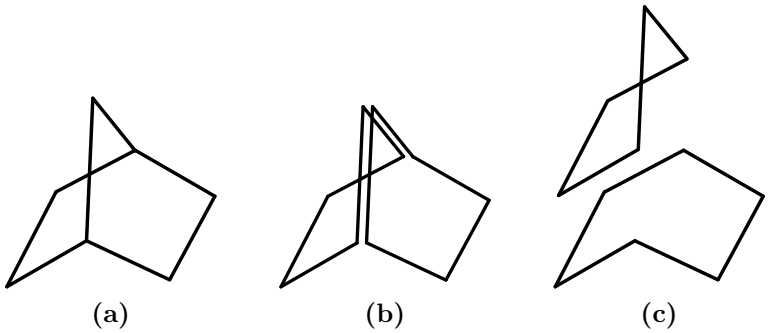


Figure 4.6.: (a) The molecule norbornan. (b) The minimum cycle basis (sum of lengths= 10). (c) A cycle basis that is not minimum (sum of lengths= 11).

cycle sets arose from the attempt to ensure that all chemically meaningful rings are included. Of course, what is chemically meaningful depends on the application."

Many cheminformatics tools try to determine a so called Smallest Set of Smallest Rings (SSSR). The original graph theoretical definition of an SSSR was that it represents a "minimum fundamental cycle basis". The problem to find such a set was shown to be NP-complete by Deo et al. (1982). Thus, many approaches to the problem use heuristics and are therefore not correct in all cases. Furthermore, Berger et al. (2004) mention that several authors use the term SSSR synonymously for "minimum cycle basis", whereas such a minimum cycle basis is only in some cases also fundamental. A minimum cycle basis can be defined as follows:

(i) Each cycle of the set is a relevant cycle, that is a cycle that cannot be described as the sum of smaller cycles, whereas the sum is defined as the symmetric difference of the sets of edges of the cycles.

(ii) The combination of all cycles from the set can be used to describe the complete ring systems, that is each edge and each vertex that is part of any cycle is contained in the minimum cycle basis.

(iii) The sum of the lengths of all cycles in the basis set is minimal.

Figure 4.6 shows an example to illustrate the definition of minimum cycle bases. The cycle basis shown in Figure 4.6c is no minimum basis. Additionally, it

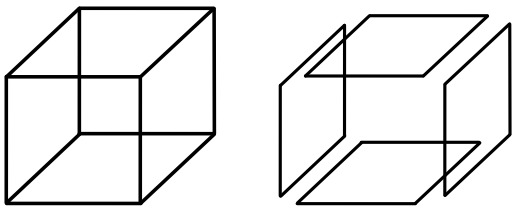


Figure 4.7.: (a) The molecule cubane. (b) A minimum cycle basis that is not minimum fundamental.

comprehends a non-relevant cycle, because the 6-membered ring can also be expressed as the symmetric difference of the two 5-membered rings.

As already stated, a minimum cycle basis is not necessarily a minimum fundamental cycle basis. The latter is derived from a minimum spanning tree. The other way around, it must be possible to remove a specific edge from each cycle in a minimum fundamental set such that the result is a minimum spanning tree. To illustrate the difference between minimum fundamental and non-fundamental bases consider the example given in Figure 4.7. The four cycles in (b) are sufficient to describe the complete ring system as they contain all edges and vertices. However it is not possible to remove an edge from each of them in such a way that the result is a minimum spanning tree. Instead, a minimum fundamental set would consist of five cycles in this example.

A general problem with the SSSR, either defined as fundamental set or not, is that it is not unique. Figure 4.8 shows a simple example, where three different minimum cycle bases exist.

Among the already mentioned programs for atom type perception, OpenBabel and I-interpret are examples for applications that determine an SSSR, while it was not published what kind of ring perception is used by the others.

To compute SSSRs many different algorithms exist (recall that also different definitions exist). Exemplary given, an $O(|E|^{2.376} \cdot |V|)$ algorithm, where $|E|$ is the number of edges and $|V|$ the number of vertices, can be found in Horton (1987).

The smallest set of relevant cycles that is unique for a graph G, is the

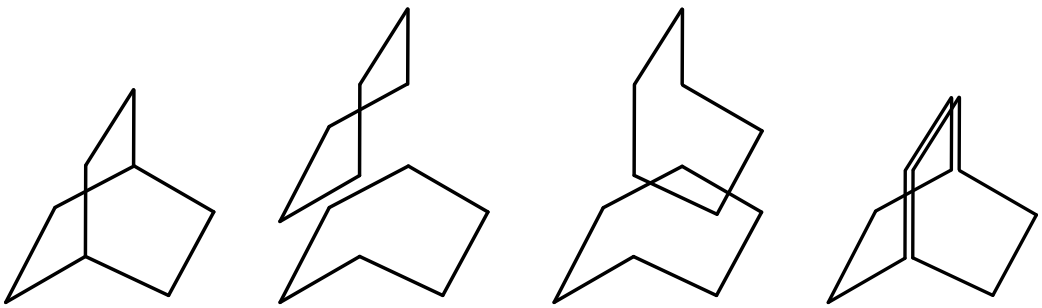


Figure 4.8.: (a) The molecule bicyclo[2.2.2]octane. (b,c,d) The three possible minimum cycle bases.

complete set of relevant cycles $R(G)$, hence the union of all minimum cycle bases (Vismara, 1997). For example, in Figure 4.6, $R(G)$ would consist of two 5-membered rings, in Figure 4.7 of six 4-membered rings, and in Figure 4.8 of three 6-membered rings.

fconv makes use of this definition and determines $R(G)$. With respect to the assignment of hybridizations it would be sufficient to compute whether an atom is part of a planar ring or not. However, there are tasks where more information is needed, the most trivial example being the function to report the number of relevant rings in a molecule. Furthermore, *fconv* tries to avoid ambiguous results whenever it is possible. In contradiction, using SSSRs instead of $R(G)$ would lead to different results for identical molecules, if the atoms are ordered differently in the used data structure.

An $O((|E| - |V| + 1) \cdot |E|^3)$ algorithm to compute $R(G(V, E))$ was given by Vismara (1997). More precisely this algorithm computes a polynomial number of ring prototypes, because $|R(G)|$ itself may have exponential size with respect to the number of vertices. Retrieving $R(G)$ from this prototype representation takes $O(|V| \cdot |R(G)|)$ time.

For *fconv* another algorithm was developed that computes $R(G)$ directly. This is acceptable, because the number of rings in biological molecules usually scales linearly with the number of atoms in the molecule.

The detailed algorithm is shown in Alg. 4.2. In essence, a Breadth-First

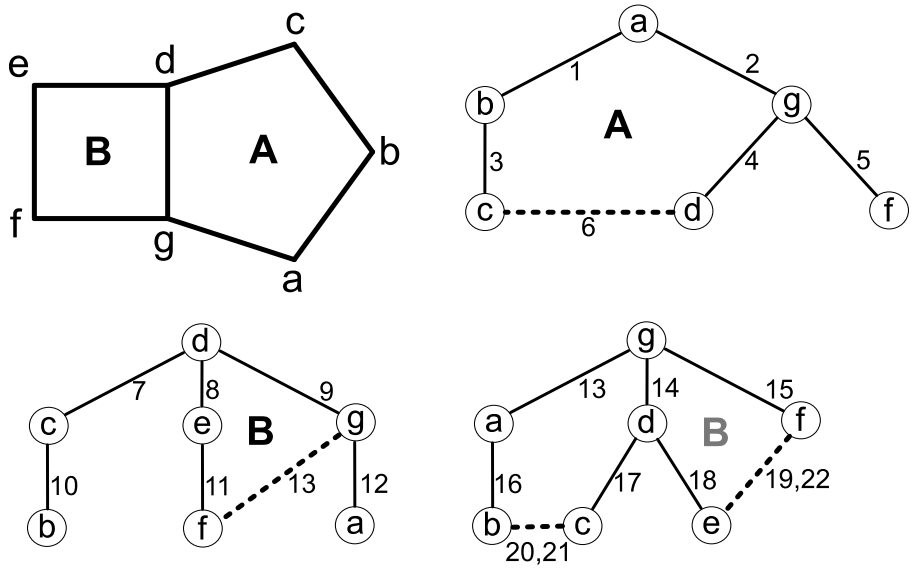


Figure 4.9.: An example that demonstrates the function of Alg. 4.2. The numbers represent steps in the loop from lines 10 to 26 in the algorithm. Ring A is found after step 6, Ring B after step 13 and a duplicate of B after step 19.

Search (BFS) for each atom that can be origin of an undiscovered relevant ring is used. Two heuristics are applied, the first being a limit for the maximum ring size. This restriction is used also by many SSSR algorithms in the literature. The *fconv* default for this limit is $max\_members = 10$, to ensure that aromatic rings up to this size can be detected in the aromaticity perception (see section 4.7), but the user can freely choose another limit (option `--m` in *fconv*). The second heuristic is correlated with the definition in line 3 of Alg. 4.2 and will be clarified in the context of the following explanation.

To illustrate the algorithm, a simple example is shown in Figure 4.9. Starting with atom a as root, the first ring (a,b,c,d,g) is found in step 6, when discovering the connection between c and d (line 20). At this point, the algorithm continues with the next possible root atom (line 5).

The BFS guarantees that the first ring that is found is also a smallest ring with respect to the root atom. It is not possible that the ring can be expressed as the sum of smaller rings, because in that case the root atom must be

Algorithm 4.2: The algorithm to determine the set of all relevant cycles.

Input : Set of **Atom**s A and max_members $\in \mathbb{N}$
Result: Set of **Ring**s R

Let v.prev be the predecessor of vertex v in a tree structure and P_v the set of all its predecessors excluding the root
Let $R_a \subseteq R$ be the set of rings containing atom a
Let max_rings(a) be the maximum number of relevant rings that can originate from a
$R := \{\}$
forall $a \in A$ **do**
    if $|R_a| =$ max_rings(a) **then** next a
    a.level $:= 1$
    $V := \{a\}$ `// queue root for BFS`
    max_local = max_members
    while $V \neq \{\}$ **do**
        $v = V[1]$; $V := V \setminus \{v\}$
        forall $u \in v$.bonded_atoms **do**
            if $u = v$.prev **then** next u
            if v.level + u.level $\geq$ max_local **then** next u
            if u.prev $= \{\}$ **then**
                if $2v$.level $- 1 \geq$ max_local **then** next b
                u.level $:= v$.level $+ 1$
                u.prev $:= v$
                $V := V \cup \{u\}$ `// last in the queue`
            else
                if $P_v \cap P_u \neq \{\}$ **then** next u
                $r := P_v \cup P_u \cup \{a\} \cup \{v\} \cup \{u\}$ `// potential new ring`
                if $r \notin R$ **then**
                    $R := R \cup \{r\}$
                    next a
                else max_local $= |r|$

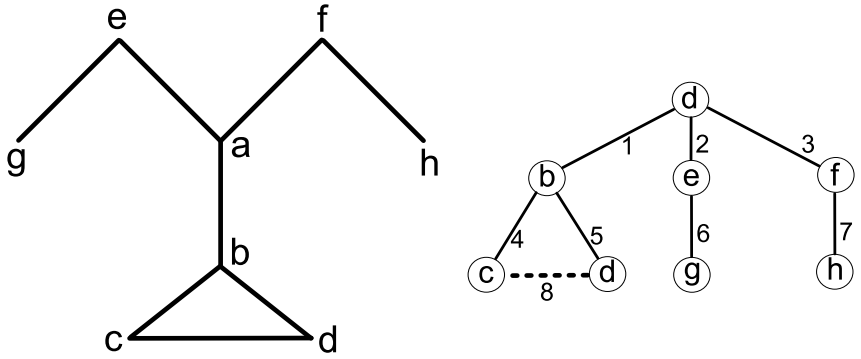


Figure 4.10.: Without the check in line 21 of Alg. 4.2, a false ring a,b,c,d,b would be detected.

contained in at least one of these smaller rings. Thus, each ring that is found first originating from a particular root atom must be a relevant ring. It must however be checked whether the found ring really is a smallest ring containing the root. This is done in line 21 of the algorithm by a check whether the root atom is the first branching vertex that is shared by both atoms which would connect the ring. A simple example where this is not the case is given in Figure 4.10. Here, a false ring that contains atom b twice would be detected without the check.

Back to the example in Figure 4.9, it would also be possible not to use a new root atom at this point, but to continue with the branch created in step 5. If, a was part of two relevant rings, the second would be discovered continuing with the other branches. However, that way it would also be possible to detect non-relevant rings. In other words, for each ring that is found after the first and that consists of more atoms, it must be proven that it is really a relevant cycle. This would nullify the advantage of not starting with a new root atom after each new ring.

Proceeding with each individual atom in this way would safely generate all relevant rings (within the specified maximum ring size), but it is also a waste of time since many duplicates of already existing rings would be found (but not added to R, see line 23). Instead, this is prevented by the use of the second heuristic. It defines the maximum number of rings $|R_a|$ an atom a can be part

of, if used as root (line 6). The values used are 0 for atoms with less than two adjacent atoms, 1 for atoms with less than three adjacent atoms, 3 for atoms with less than four adjacent atoms, and 6 for all with more adjacent atoms. Note that from the elements considered by *fconv* only sulfur and phosphorous can have more than four adjacent atoms (see Table 4.1). Though, also for these two elements, exceeding 4 valences is possible only in non-standard biological molecules, so 4 will be considered the maximum valence in all subsequent considerations.

In the example, after the first ring was found, atoms b to c exceed the given limit, because they are already part of one ring. Thus, the algorithm proceeds with d, which has three connected atoms and is therefore allowed to originate two rings. The next ring is found after step 13 and g is taken as the next root. The result is a duplicate of the second ring after step 19 and therefore, the maximum ring size for the current root atom is set to the size of this duplicate (line 26). This is important, because after duplicates are found the algorithm proceeds traversing the tree, but rings with a higher size than the duplicate can be non-relevant rings and are therefore discarded at this point. In the example, after steps 20 to 22 no duplicates are found, because the loop just continues in line 14 of the algorithm.

A counterexample for the mentioned limits of $|R_a|$ was already shown with norbornane (Figure 4.6). There, the bridging atom has only two connected atoms, but is part of two rings. If one of the latter is found, the bridging atom can no longer be a root atom for the loop in line 10 of Alg. 4.2. However, all rings will be found using other atoms of the shown molecule as root atoms, because the limitation is not applied to the u in the loop in line 12. To give a counterexample that leads to an undiscovered relevant ring, one had to construct a graph where $|R_v|$ for vertex v exceeds its limit and at the same time all other atoms of one $r \in R_v$ also exceed their limit. The author is not aware of any organic compound that would match such a counterexample.

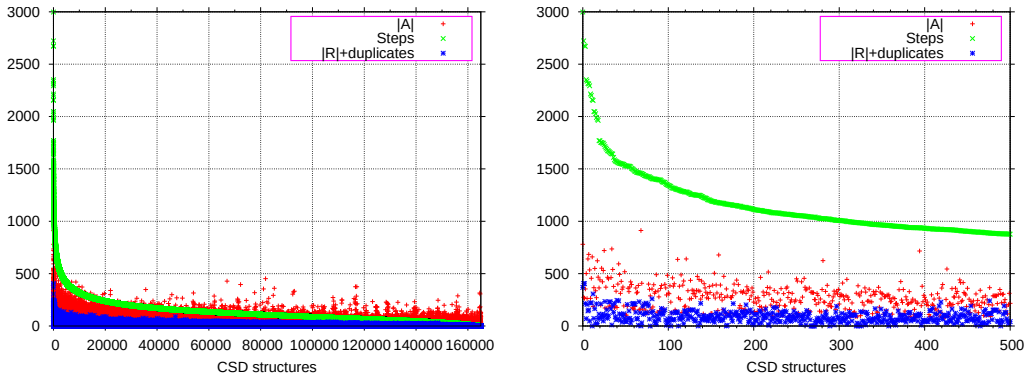
4.5.1.1. Time Complexity

Next, an estimation of time complexity of the proposed algorithm shall be discussed. First consider a huge acyclic graph $G(V, E)$, where each vertex, except of the terminal ones, has the highest possible degree of four. Now connect the terminal edges such that all vertices have degree four and all cycles (due to the connection) have a size higher than the maximum size to be considered ($max\_members$). In consequence the main loop in line 5 of Alg. 4.2 is executed $|V|$ times.

In the BFS (lines 10 to 26) there will always be four branches at each level in the tree. The algorithm enqueues a new vertex only if $max\_members < 2 \cdot current\_level - 1$ (line 16). Thus, the BFS-loop is executed $4^{max\_members/2}$ times. The limitation of the ring size therefore guarantees $O(|V|)$ instead of highly exponential runtime in case of acyclic molecules or molecules where all cycles have more members than the limit size. In case of huge biological molecules, hence proteins, there is only a relatively low number of rings and more acyclic parts

Of course, the worst case runtime for arbitrary graphs is not polynomial. As initially mentioned the number of relevant rings can scale exponentially with $|V|$ (Vismara, 1997). However, with the restriction to a maximum degree of four for all vertices this is reduced to a linear dependency. If a potentially new relevant cycle is encountered in line 20 of Alg. 4.2, it must be checked to be a true shortest cycle (line 21). This check is in constant worst case time due to the limitation of $max\_members$. Furthermore, it must be checked whether the new cycle is just a duplicate of an already existing cycle. In the *fconv* implementation a hash value is therefore calculated for each ring and the check for a duplicate is done in constant time lookup in a hash table. Thus, both checks scale linearly with $|R(G)|$ and therefore the worst case runtime is $o(|V| \cdot |R(G)|)$ using the mentioned heuristics.

Figure 4.11a shows the number of atoms, the number of steps in the BFS, and the sum of rings and duplicates for the CSD structures that were used for Equation 4.2 in section 4.4, sorted by decreasing number of BFS steps. The



(a) Complete CSD sample **(b)** First section of the sample

Figure 4.11.: The numbers $|A|$ (red), number of BFS steps in Alg. 4.2 (green), and number of perceived rings plus the number of encountered duplicates (blue) for the CSD structures used in section 4.4.

section of the first 500 structures is shown in Figure 4.11b. The highest number of steps in the BFS was needed for CSD entry *'LENYEJ'*, a 780 atom molecule with 88 rings (98 without limiting the ring size).

The interested reader may try "`fconv -nr input_file ---d`" (recall section 1.1), to get reported the number of relevant rings, the number of fused rings and their composition, and also the atoms of each individual ring. The output will be given per molecule.

4.5.2. Ring Planarity

As a next step, all perceived rings are checked for planarity. This information is essential for the subsequent geometry assessment, double bond assignment, and aromaticity detection. All of the already mentioned approaches for atom typing apply very similar thresholds for the averaged dihedral angle in a ring to determine whether it is planar or not. Sayle uses 7.5° for 5-membered rings and 12.0° for larger ones. Zhao and Hendlich both applied the same threshold for 5-membered rings, but use 15.0° for larger rings. Interestingly, it seems as if Zhao handles fused ring systems as a single ring. This is concluded by the fact that the PDB entry *'1tmn'* was reported as success in his test set (Zhao

et al., 2007), although the corresponding ligand contains a tryptophan where the 5-membered ring has an averaged angle of 9.3°.

fconv also applies the 15.0° threshold for 6-membered and larger rings, and uses 10.0° for 5-membered rings. In addition, it is also evaluated whether all ring atoms with more than 3 connected atoms have a trigonal planar geometry. Therefore, a limit of 0.9 is applied for the scalar triple product defined by the three bond vectors. If all atoms of a planar ring are also trigonal planar, then the ring is labeled as fully sp2-hybridized.

4.5.3. Geometry Check

Especially in smaller rings, higher deviations from ideal geometries are possible. For that reason, all atoms that are part of fully sp2-hybridized rings are subjected to the double bond optimization and aromaticity detection, even if they exhibit bond lengths or angles that usually do not allow for this hybridization type. Apart from non-ideal geometries due to ring constraints, the PDB contains a fair number of ligands where aromatic rings exhibit rather strange geometries. Unfortunately, some crystallographers put much effort into the refinement of protein structures, but take less care about the ligands. An example is the PDB entry *'1cps'*, where the bond lengths in the ligand's phenyl ring vary from 1.2 Å to 1.6 Å.

Regarding the atoms that are not part of a planar ring, there are cases where the hybridization is clearly defined and cases where it is ambiguous. For example, if a carbon has more than three bonded atoms or an oxygen has more than one bonded atom, they must be sp3-hybridized. Also if a carbon has only three bonded atoms but no planar geometry, it must be sp3-hybridized. If there are only two atoms bonded to a carbon and it has linear geometry, then it must be sp-hybridized. But if it is not linear, the hybridization is not definite. In this case, not the hybridization with the highest bond length probability (see subsection 4.4.1) is assigned, but the lowest hybridization with a probability > 0. Hence, as long as the sp2-probability for any of the two bonds is not zero, this hybridization is assigned initially. If this assignment is erroneous, it will

be corrected in the steps described in the next sections.

4.6. Optimization of Conjugated Systems

At this point, the hybridizations of several atoms are definite and imply also some definite bond types. However, there may still be many atoms which can be either sp-, sp2- or sp3-hybridized. Furthermore, even if a number of atoms have a definite sp2-hybridization, it is not clear how they are connected by double and single bonds, as long as they are not pairwise isolated. For example, a carbon chain is shown in Figure 4.12. If no hydrogen atoms are available in the input, the possibility of being sp2-hybridized cannot be excluded for any of these atoms, because the maximal valence for sp2-hybridized carbon is never exceeded and a check for trigonal planar geometry is not possible as it would require three bonded atoms. (In contrast existence of sp-hybridization can be excluded, because none of the atoms has linear geometry.) Although some of the bond lengths shown in (a) are more likely corresponding to single bonds, Table 4.2 proofs that there is still a probability > 0 for double bonds. An sp2-hybridized carbon cannot be part of more than one double bond (although this bond might be delocalized like in aromatic systems). A naive solution to the problem of assigning these bonds might be to maximize the number of double bonds. This would lead to two possible chemical formulas shown in Figure 4.12 (b) and (c). However, regarding the probability functions in Figure 4.3, it becomes obvious that in both solutions at least one very likely double bond is missing, while other, more unlikely double bonds are set. For the most likely solution the probabilities must be considered which would lead to the correct formula given in (d), which maximizes the sum of individual double-bond probabilities.

Hendlich et al. (1997a) generate all possible assignments of alternating single/double bonds and rank each assignment by a cost function. This cost function is based on deviations from ideal bond lengths as found in the literature and penalties for specific arrangements like sp2-hybridized carbons

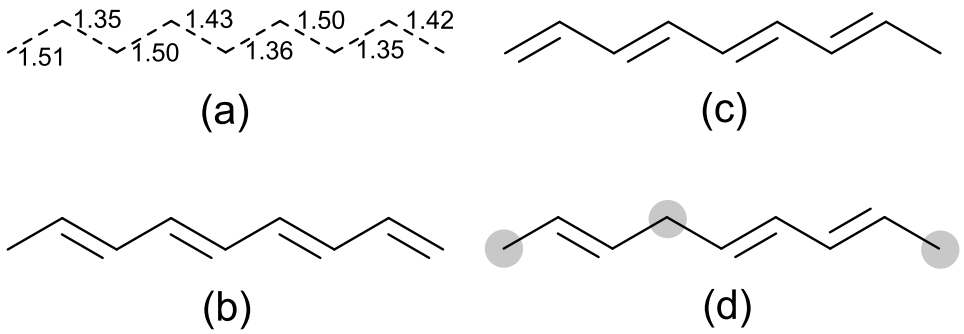


Figure 4.12.: (a) A set of atoms, their connectivities and corresponding distances in Å. (b) and (c) Possible solutions, maximizing the number of double bonds. (d) The correct solution, maximizing the probabilities for double bonds. In consequence, the atoms marked with gray background cannot be sp2-hybridized and must be reset to sp3.

with no assigned double bond. With such penalties and enumerating only alternating systems, the algorithm would be able to find the best solution from Figure 4.12 (b) and (c), but it would not be able to retrieve (d) as best solution. Furthermore, even if only alternating systems are considered for enumeration, the combinatorial number of possible assignments can be high in case of more complicated systems.

fconv applies an approach similar to Labute (2005), treating the double-bond assignment as a maximum weighted matching problem for a non-bipartite graph.

4.6.1. Non-bipartite Maximum Weighted Matching

This kind of matching is the most challenging among the four basic matching problems (maximum cardinatlity/maximum weighted, for bipartite/non-bipartite graphs). The first efficient algorithm is due to Edmonds (1965) and is also the algorithm that was implemented by Labute. It has a time complexity of $O(|V|^4)$ (where $|V|$ is the number of edges in the graph) and was improved by Gabow (1976) and also Lawler (1976) to run in $O(|V|^3)$ time. Galil et al. (1986) presented an $O(|V| \cdot |E| \cdot log|V|)$ algorithm that is faster than $O(|V|^3)$ in case of sparse graphs. Finally, Gabow et al. (1989) developed an

$O(|V| \cdot (|E| \cdot loglog log_{(|E|/|V|+1)}|V| + |V| \cdot log|V|))$ algorithm that also bounds to $O(|V|^3)$ in case of dense graphs with quadratic number of edges (with respect to the number of vertices) and to $O(|V| \cdot |E| \cdot log|V|)$ in case of sparse graphs with linear number of edges. All algorithms mentioned so far are extensions of Edmonds' algorithm, which is not trivial to implement, hence they are all rather complicated. The improvements for special cases of graphs are mostly based on special data structures like Fibonacci heaps (Fredman and Tarjan, 1987).

There also exists a number of approximating algorithms, e.g. the Path Growing Algorithm (Drake and Hougardy, 2003), that are both easy to implement and fast, but they are not useful for the application to double-bond optimization as they can produce wrong results in even simple cases.

For this thesis, a specialized algorithm was developed that is exact, but much easier to implement in comparison to exact algorithms mentioned above. Its time complexity is worse, but in case of graphs corresponding to molecular structures, it is efficient enough to find the solution very quickly and not being the bottleneck in the complete process of atom type perception. However, its appropriateness must be discussed in the following.

Alg. 4.3 shows the developed maximum weighted matching algorithm that takes a set of weighted edges as input. In the initialization (line 1), an array with all incident edges is assigned to each edge of the input. Special functional groups like $R - SO_2 - R'$ are not subjected to the algorithm, but only carbons, nitrogens, oxygens, sulfur and phosphorus that can have a maximum of one double bond and a maximum valence of three. Consequentially, a vertex in the corresponding graph cannot have more than three incident edges and therefore an edge cannot have more than four incident edges. The algorithm could also be applied to dense graphs with much more edges, but it is efficient only for a restricted number as will be shown later.

The algorithm starts with an arbitrary edge (line 6) that becomes the current edge c in the main loop (line 8). A set of possible labels L is initialized with only one element (line 5). A label is always associated with a weight, which is zero by initialization.

Algorithm 4.3: A simple maximum weighted matching algorithm for non-bipartite graphs.

Input : Set of weighted edges $E(G)$ of an undirected graph $G(V, E)$
Result: The maximum weighted match $M \subseteq E$

Let $A_E(e(u,v)) := \{a \in E | (u \in a) \veebar (v \in a)\}$ be the set of edges incident to $e \in E$
forall $e \in E$ **do**
  if e.is_labeled **then** next e
  $P := \{\}$ `// set of processed edges`
  $L := \{(1, 0)\}$ `// set of labels (1) and associated weights (0)`
  $S := \{e\}$
  while $S \neq \{\}$ **do**
    $c := s[1]$ `// first element in` S
    $S := S \setminus \{c\}$
    $N := \{\}$ `// set of new labels`
    forall $l \in L$ **do**
      forall $a \in A_E(c)$ **do**
        if $(\neg a$.is_labeled$) \wedge (a \notin S)$ **then** $S := S \cup \{a\}$
        else if $l \notin a$.labels **then** next a
        if c.weight $\geq a$.weight **then**
          $nl := (|L| + |N| + 1, c.\text{weight})$ `// new label`
          $N := N \cup \{nl\}$
          c.labels $:= c$.labels $\cup \{nl\}$
          forall $p \in (P \setminus A_E(c))$ **do**
            if $l \in p$.labels **then**
              p.labels $:= p$.labels $\cup \{nl\}$
              nl.weight $:= nl$.weight $+ p$.weight
      if $|N|$ increased **then** next l
      c.labels $:= c$.labels $\cup \{l\}$
      l.weight $:= l$.weight $+ c$.weight
    if $\neg c$.is_labeled **then**
      $nl := (|L| + 1, c.\text{weight})$
      c.labels $:= c$.labels $\cup \{nl\}$
      $N := N \cup \{nl\}$
    $L := L \cup N$
    $P := P \cup \{c\}$
    REDUCE-LABELS(P, S, L, E) `// see Alg. 4.4`
  $l_{\max} := \{l \in L | max(l.\text{weight})\}$
  $M := M \cup \{p \in P | l_{\max} \in p.\text{labels}\}$

Now the current edge c passes through a loop for each possible label l (line 11, executed only once in the first cycle of the main loop). If no incident edge already has the label l, then l is assigned to c and the weight of c is added to the weight of the label (lines 24,25). Note that an edge may have an unlimited set of different labels. All edges sharing a particular label correspond to one possible matching, hence two incident edges are not allowed to have the same label. If one of the edges incident to c is already labeled with l, there are two possibilities:

(i) The incident edge a with the label l has a higher weight compared to c. In that case it is not possible that c, together with all other edges that are already l-labeled and that will be l-labeled in subsequent loop cycles, would finally result in a higher weight for l than it is the case with a. Of course, it can still happen that c together with all already l-labeled edges will be part of the best match in the end, but then with respect to another label and not to l. Therefore, nothing happens at this point and the loop in line 11 continues with the next cycle.

(ii) The incident edge a with the label l has a lower or equal weight compared to c (line 15). If only already labeled edges are considered, removing l from a and setting it to c would result in a higher (or equal) total weight for l. However, considering also subsequent loop cycles this may lead to a lower weight than it would have been possible if l was left to a. In consequence, l is not removed from a, but a new label nl is created and added to c (lines 16,18). Then, the new label is also added to all already l-labeled edges that are not incident to c, updating also the weight for the new label (lines 19 to 22).

Furthermore, each incident edge that is completely unlabeled is added to the set of edges to be processed S, if not already part of this set (line 13). After the loop over all possible labels has finished, there is the possibility that c is still completely unlabeled. In that case a new label is generated and assigned to c (lines 26 to 29). Now all newly generated labels are added to the set of possible labels (line 30) and c is added to the set of processed edges (line 31). Finally, a function to reduce the number of labels and to also setup the next element

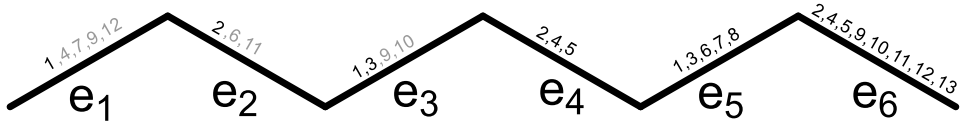


Figure 4.13.: Example of Alg. 4.3 without the REDUCE-LABELS function. The algorithm starts with edge e_1 and for all edges weight(e_n) > weight(e_{n-1}). The numbers on the edges are the labels, whereas black labels are directly assigned when processing an edge and gray labels are assigned when processing another edge (line 19 to 22 in the algorithm).

in S is executed (line 32), before the main loop proceeds with the next edge. When all edges are processed, the label with the highest weight is determined and all edges with that label are reported as the maximum match (lines 33,34).

Without the REDUCE-LABELS function, the algorithm would also yield the correct maximum weighted matching, but with unacceptable runtime and an extreme demand of memory. Consider the simple example given in Figure 4.13 where the labeling starts with edge e_1 and the weights of the edges are increasing, such that the current edge c in the algorithm always has a higher weight than its incident predecessor. The number of possible labels $|L|$ is one after processing the first edge, two after the second, three after the third, then five, eight, and finally 13 after edge e_6. It can be easily seen that the number follows the Fibonacci sequence, hence increases rather quickly. In the example the maximum number of incident edges was two, but as stated above the maximum possible number of incident edges a vertex can have is three and thus in a worst case scenario $|L|$ would grow even faster.

Before explaining the REDUCE-LABELS function that prevents the explosion of the number of labels, the following consideration shall illustrate the idea behind that function. Consider the penultimate step in the example from Figure 4.13 after edge e_5 was processed. At this point the number of labels is eight. From all already processed edges, e_5 is the only one that is incident to a not processed edge (in the following algorithm, such edges will be called terminal edges). Regardless of how many edges are still to be processed, there are only two possibilities with respect to the maximum match. (i) e_5 is part of

Algorithm 4.4: The REDUCE-LABELS function being part of Alg. 4.3.

Input : P, S, LN, E `// L is called LN here`
Result: reduced LN and modified S

1 Let $T := \{t \in P | A_E(t) \cap S \neq \{\}\}$ be the set of terminal edges
2 Let $s_{\max} \in S$ be one of the edges with the maximum number of incident, labeled edges `//` $|T \cap A_E(s_{\max})| = max(|T \cap A_E(s \in S)|)$
3 Move $s_{\max}$ to position 1 in S
4 $B := \{\}$ `// best labels`
5 **for** $i := 1$ **to** $2^{|T|}$ **do** $B := B \cup \{(0,0)\}$ `// fill with zero weight labels`
6 **forall** $l \in LN$ **do**
7 $\quad i := 1$
8 $\quad k := 1$
9 $\quad$ **forall** $t \in T$ **do**
10 $\quad\quad$ **if** $l \in t$.labels **then** $i := i + k$
11 $\quad\quad k := k \cdot 2$
12 $\quad$ **if** l.weight $> b[i]$.weight **then** $b[i] := l$
13 $LN := \{b \in B | b \neq (0,0)\}$

the maximum match or (ii) e_5 is not part of the maximum match. Therefore, only the best (highest weighted) label that is assigned to e_5 and the best label that is not assigned to e_5 must be regarded for further processing (which would be label 1 and label 2 in the example). Thus $|L|$ reduces from eight to two and factually, when applying this reduction after each cycle of the main loop the number is never greater than two in the given example.

Alg. 4.4 shows the detailed algorithm of the REDUCE-LABELS function. Please note, that the set of labels L is called LN in this algorithm. This is important to differentiate the situation before reduction, when it was extended by N, from the situation after reduction (and in the parts described so far). In other words: $L = (LN)_{\text{reduced}}$. In line 1 the set of terminal edges T is determined (all processed edges that are incident to unprocessed edges, which can be only edges that are in S). The rationale for line 2 and 3 will be given later. In the aforementioned example (Figure 4.13) there was only one terminal edge, leading to two possible cases. In case of two terminal edges, there would be four possible cases (or only three cases if the terminal edges are incident,

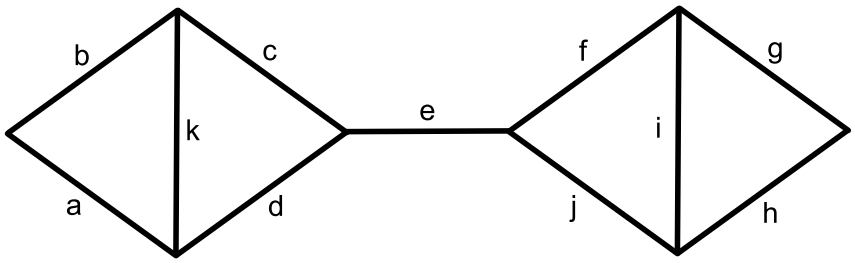


Figure 4.14.: Without the steps in line 2 and 3 of Alg. 4.4, a worst case sequence of processed edges could be in the order a,b,c,d,e,f,g,h,i,j,k. The corresponding numbers of terminal edges after each step are 1,2,3,4,5,6,7,8,8,4,0 respectively.

because they cannot both be a part of the maximum match). The maximum number of possible cases is $2^{|T|}$ and the maximum weight label for each of these cases shall be determined. Fortunately, not all of these cases must be evaluated explicitly. Instead, for the existing labels the corresponding case is determined (lines 7 to 11) and if its weight is better than the weight of the currently best label for this case it replaces this label (line 12).

Obviously, $2^{|T|}$ is the upper limit for $|L|$ ($= |LN|$ after the reduction). Now consider the example given in Figure 4.14, which shows a worst case for the sequence in which the edges are processed (from 'a' to 'k'). This worst case is only possible, if the set S is implemented as a LIFO (last in first out) structure (hence a stack). After step 'h', the number of terminal edges is eight (each edge from 'a' to 'h' is incident to either 'i', 'j' or 'k').

One possibility to decrease the number of terminal edges would be the usage of a FIFO (first in first out) structure (hence a queue). However, in the implementation, a modified LIFO is used. This has the same performance as a FIFO in the shown example and performs even better in the general case. The latter will be demonstrated in subsubsection 4.6.1.1.

To decrease the possible number of terminal edges, the LIFO is modified in the REDUCE-LABELS function. There is a simple rule that determines the next edge to be processed. It must be one of those edges that have the highest number of incident, terminal edges. Therefore, in line 2 of Alg. 4.4 such an edge

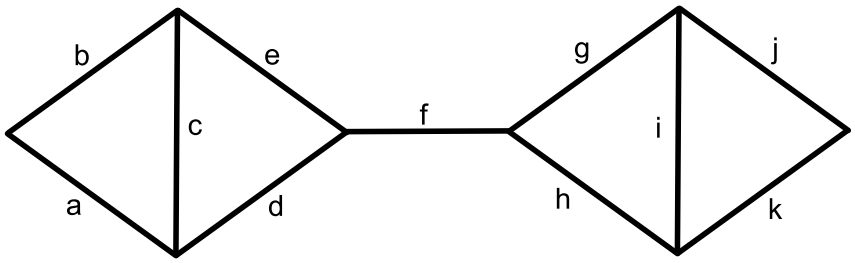


Figure 4.15.: Using the steps in line 2 and 3 of Alg. 4.4, a worst case sequence of processed edges could be in the order a,b,c,d,e,f,g,h,i,j,k. The corresponding numbers of terminal edges after each step are 1,2,3,3,2,1,2,2,3,3,0 respectively.

is determined and it is moved to the first position in S (line 3), ensuring that it will be processed next. In a real case implementation this is just done within the step in line 1 of the algorithm. Figure 4.15 shows again a worst case sequence for the previous example, but this time with the described additional rule. Now, the maximum number of terminal edges is reduced significantly from eight to three. However, $2^{|T|}$ is only the worst case number for $|L|$ when no two terminal edges are incident, which is not the case in the given example. It is therefore interesting to also have a look at $|L|$ after each call to the REDUCE-LABELS function. For example after processing edge 'd' in Figure 4.14, there are four terminal edges ('a','b','c','d'). But instead of $2^{|T|} = 16$ possible cases (labels) there are only seven, because cases where two incident edges share the same label are not possible (e.g. the case "'a' and 'c' share a best label" is possible, but the case "'a' and 'b' share a best label" is not possible). Also, seven is still only an upper limit for $|L|$, because two or more different cases may have the same best label. In detail the worst case numbers for $|L|$ after each step in Figure 4.14 are 2,3,5,7,10,17,27,44,44,7,0 leading to a mean value of 15. In case of Figure 4.15, the numbers after each step are 2,3,4,5,3,2,3,3,4,5,0 leading to a mean value of 3. However, the shown example does not correspond to a worst case graph, which has to be considered when determining the asymptotic time complexity of the algorithm. This will be the subject of the next paragraphs.

4.6.1.1. Time Complexity

For time complexity analysis, the inspection starts with Alg. 4.3. In case of a connected graph, the loop in line 2 is executed only once for the first edge in E, because all other edges are processed in the loop in line 7, which will be called the main loop in the following. Generally, both loops together are executed exactly $|E|$ times, so up to that point the runtime scales linearly with the number of edges. The loop in line 11 is executed $|E| \cdot \langle |L| \rangle$ (number of edges times the averaged number of labels) times. Line 12 iterates over all incident edges, which equals four at maximum in the cases the algorithm is used for, hence it does not influence the asymptotic time complexity. Checking an edge for a particular label (like in line 14) and setting a label (like in line 18) is done in almost constant time $O(1)$, because a simple array structure (with a set or unset bit for each possible label) is used that reserves an amount of memory initially and is only rarely extended (when the number of labels increases significantly). Extending sets like S, N and L is also done in constant time, because data are just appended to the used data structures. Finally the innermost loop in line 19 is executed $|E| \cdot \langle |L| \rangle \cdot \frac{|E|-1}{2}$ times in a connected graph (which is the worst case). This leads to an asymptotic time complexity of $O(|E|^2 \cdot \langle |L| \rangle)$. That would be also the worst case time bound for the complete algorithm, if the REDUCE-LABELS function in line 32 has a lower complexity compared to the part from line 11 to 25. Thus two questions remain: (i) What is the complexity of the REDUCE-LABELS function and (ii) how does $\langle |L| \rangle$ depend on the number of edges $|E|$?

Generating the set T in line 1 of Alg. 4.4 scales linearly with $|S|$, because only edges incident to $s \in S$ must be checked. It is executed only once in the function, so it will play no role for the final complexity. As already mentioned, determining $s_{\max}$ is done together with the generation of T in a real case implementation. Moving $s_{\max}$ to the beginning of S is done in constant time, because S is implemented as a doubly linked list. The loop in line 5 is only present in the pseudocode. In the implementation this is a single memory allocation with zero initialization, which is usually faster than linear time.

The time critical part starts in line 6, when iterating over all currently possible labels and then in line 9 over all terminal edges. This leads to a complexity of $O(\langle|LN|\rangle \cdot |T|) = O(\langle|L| + |N|\rangle \cdot |T|)$ (Line 13 is realized by removing each label that is not in B, which is done in constant time per label, because L/LN is implemented as doubly linked list).

Now a worst case dependency of $\langle|L|\rangle$ and $\langle|N|\rangle$ from $|E|$ is needed. In case the graph has no cycles, the maximum number of terminal edges is

$$|T|_{\max} = \frac{|E| + 5}{3} \quad \text{for} |E| > 5 \tag{4.3}$$

The rationale for this equation is illustrated in Figure 4.16, where a situation with $|E| = 7$ is shown. After step 'd', there are 4 terminal edges. Following the given rule to determine the next edge to be processed, the next step must be 'e', even if there would be more edges at other positions. After 'e' there are only 2 terminal edges left and at least 3 edges (examples are the dotted lines in the figure) must be added to get 5 terminal edges. It can be easily seen that (without cycles) there are always at least three more edges necessary to get a higher number for $|T|$, which leads to the above mentioned equation. In case of a lower number of edges, it is possible to get a higher $|T|$ using cycles. For example, if only the five edges 'a' to 'e' in Figure 4.15 were present, there would be a maximum $|T| = 3$. However, the cycles must be extended and at that point (edge 'f' in Figure 4.15) the maximum number is always limited. Therefore, the maximum number remains three for all eleven edges in the example, whereas in a graph with eleven edges and without cycles the maximum number is five. An extreme example is shown in Figure 4.17, where only 6 edges lead to 4 terminal edges. However, the shown graph is isolated and cannot be extended (recall that a maximum of three incident edges for each vertex is allowed). So Equation 4.3 is the general worst case.

What can also be seen in Figure 4.16 is that in case of a maximum of terminal edges, for $|T|_{\max} > 3$ there must be always at least two pairs of incident edges. Without this minimum of incident edges there are $2^{|T|-4}$ cases. Due to the adjacency of the two pairs there are not 16 possible combinations for them, but

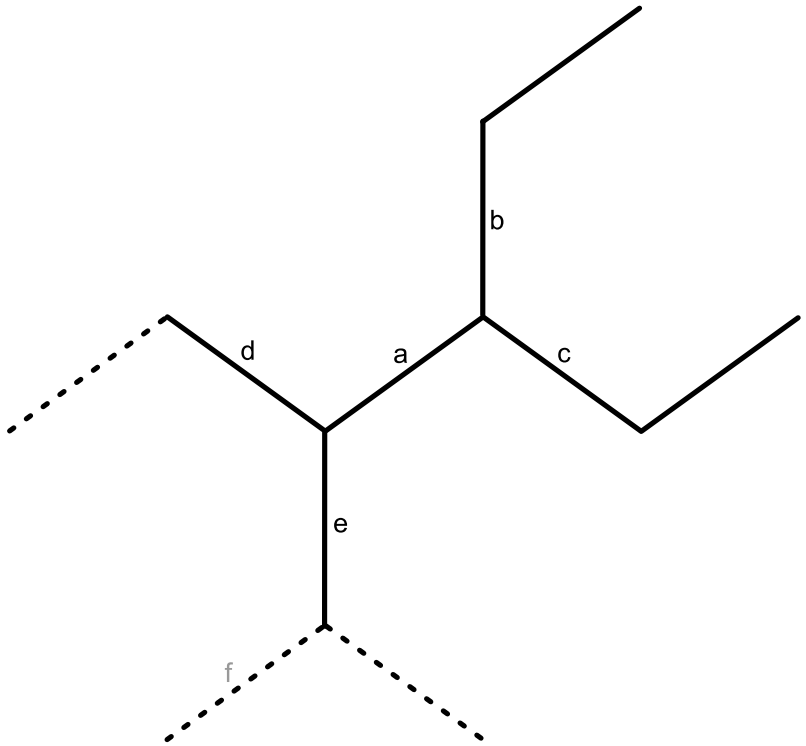


Figure 4.16.: A graph with seven edges (solid lines) following the described rules for labeling. The number of terminal edges after step a is 1, 2 after b, 3 after c, 4 after d and 2 after e. The dotted lines are possible extensions that lead to a higher number of terminal edges (after step f).

only 9, which leads to

$$|L|_{\max} = 9 \cdot 2^{|T|-4} \tag{4.4}$$

and combining this with Equation 4.3, the worst case dependency of $|L|_{\max}$ from $|E|$ is

$$|L|_{\max} = 9 \cdot 2^{\frac{|E|+5}{3}-4} \tag{4.5}$$

Of course, $\langle|L|\rangle < |L|_{\max}$ is always true, but it does not change the asymptotic complexity, which now is $O(|E|^2 \cdot 2^{\frac{|E|}{3}})$ for lines 1 to 31 in Alg. 4.3 and $O((2^{\frac{|E|}{3}} + \langle|N|\rangle) \cdot |E|)$ for Alg. 4.4. As $|N|$ scales also linear with $|L|$ in the worst case, this equals $O(2^{\frac{|E|}{3}} \cdot |E|)$ and thus, in combination with the main loop of Alg. 4.3 this is the same complexity as for the part from line 1 to 31 in this algorithm.

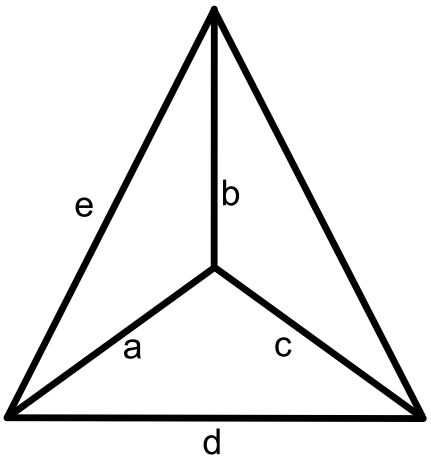


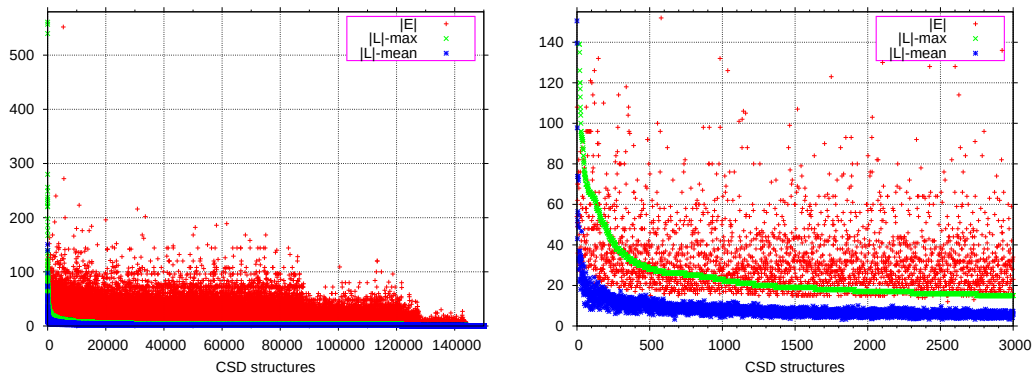
Figure 4.17.: An example where only 6 edges can lead to 4 terminal edges. However, this graph cannot be extended due to the limit of maximal three incident edges.

Therefore

$$O(|E|^2 \cdot 2^{\frac{|E|}{3}})$$

is the worst case time bound for the complete algorithm. This is much worse in comparison to the known algorithms, but first, the presented matching is much easier to implement and second, it will be shown that the real case application is always far away from the worst case. Actually, the algorithm performs very well for normal cases, as it only uses data structures with almost constant time access, while other algorithms are based on structures like priority queues, which enhances the complexity even in trivial cases.

There is still one open question from the last section, where it was stated that the modified LIFO S performs better compared to a FIFO structure for S. Processing the example shown in Figure 4.16 with FIFO rules leads to the same relation as shown in Equation 4.3 (actually more easy to see compared to the modified LIFO). The LIFO however has the advantage that it proceeds faster in direction of the end of a branch (the terminal edges reduce, when the end is reached), hence it works like a Depth-First Search (DFS). In contrast, the FIFO works more like a breadth first search, which results in more “open ends” in average.



(a) Complete CSD sample **(b)** First section of the sample

Figure 4.18.: The numbers $|E|$ (red), $|L|_{\max}$ (green), and $\langle|L|\rangle$ (blue) for the CSD structures used in section 4.4.

4.6.1.2. Evaluation

Figure 4.18a shows the number of edges, the maximum number of labels and the averaged number of labels for the CSD structures that were used for Equation 4.2 in section 4.4, sorted by decreasing $|L|_{\max}$. The section of only the 3000 first structures is shown in Figure 4.18b. The maximum number of possible labels is found for the CSD entry *'LOCGEP03'* (Zhai et al., 2009) with $|L|_{\max} = 561$ and $\langle|L|\rangle = 150$ (which is also the highest value in the data sample) for an input of 54 edges. The corresponding structure is shown in Figure 4.19. There are several equivalent maximum weighted matchings, which in this case are also maximum cardinality matchings.

It can be seen that in almost all cases the averaged number of labels is below the number of edges in the input, even for the more complicated CSD structures. The same statistic for PDB structures would result in generally lower numbers, the most complicated structures being hemes (e.g. heme B in PDB entry *'1PHE'* with $\langle|L|\rangle = 5$). It shall be noted that the algorithm executes in milliseconds even for $\langle|L|\rangle$ in a range of a few thousands. Therefore, it is absolutely sufficient for the use in *fconv*.

Finally, it is worth mentioning that the presented algorithm still has potential for improvements. For example, it happens that after a call of the REDUCE-

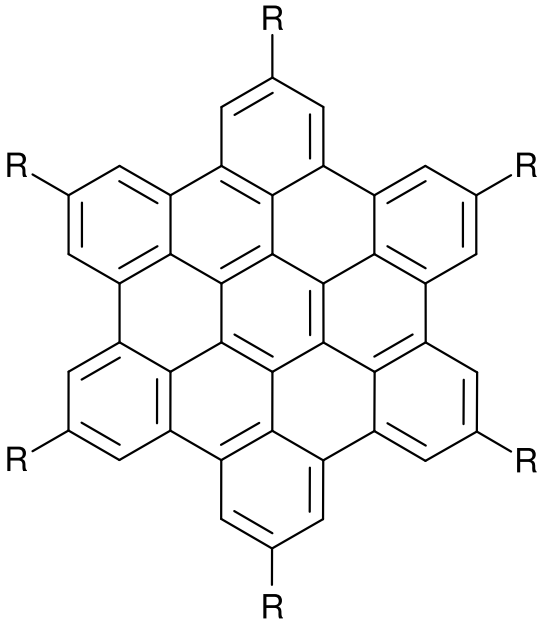


Figure 4.19.: The CSD structure *'LOCGEP03'*, shown with one of many equivalent double-bond assignments.

LABELS function some of the already processed edges are completely unlabeled. A simple example is Figure 4.13, where this happens twice. Such edges could be safely removed from the set of processed edges P and must not be checked for any labeling in subsequent loop cycles. Another possible improvement might be a more canny choice of $s_{\max}$, e.g. to prefer edges that lead faster to end points, that is to edges with no more unlabeled incident edges.

4.6.2. Assigning Weights for the Matching

In the previous section it was shown that the developed matching algorithm works appropriately. However, this would be meaningless as long as no reasonable weights are assigned to the putative double bonds.

Bonds that are the central bond of a non-planar torsion can be directly excluded. Also bonds that correspond to special functional groups with more than one double bond at one atom (e.g. $R - SO_2 - R$) are excluded from the optimization process. To all remaining sp2-sp2 candidate bonds the probabilities

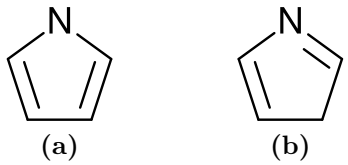


Figure 4.20.: (a) The correct formula for pyrrole. (b) This formula would be correct if a negative charge was assigned to the not doubly bonded carbon. However, aromatics should always be typed in a neutral form when possible.

as defined in Equation 4.2 are assigned as initial weights, that is values between zero and one. Here, the higher value from either the double bond or the aromatic bond type is used.

Furthermore, if the bond is not part of a planar ring, probabilities based on bond angles are evaluated. These bond angle-dependent probabilities are derived analogously to the distance-dependent probabilities. In addition to the angle γ, they depend on the element type i and the hybridization type h.

$$P(i,h,\gamma) = \begin{cases} \frac{N(i,h,\gamma)}{\sum_{\gamma'} N(i,h,\gamma')} & N(i,h,\gamma) \geq 0.0005 \cdot N_{\max}(i,h) \\ 0 & N(i,h,\gamma) < 0.0005 \cdot N_{\max}(i,h) \end{cases} \tag{4.6}$$

The same data set as for the bond lengths was used and a bin size of 2° with Gaussian smoothing with $\sigma = 2°$ was applied. From the two atoms being part of a putative double bond, the highest probability for an sp2-bond angle is chosen and compared to the already assigned bond length probability. If it is higher, then it replaces the distance probability, otherwise it is discarded.

Special weights must be assigned in case of planar, fully sp2-hybridized rings, because these are candidates for aromatic systems. In most cases, it is more favorable to form an aromatic system by formally assigning double bonds to the ring than to form e.g. exocyclic double bonds. The reason is the high chemical stability of aromatic compounds (see section 4.7).

Some examples shall be given to deduce the basic rules for the described ring structures. Figure 4.20 shows two different formulas for pyrrole. In the context of this work, the formula in (b) shall be considered incorrect, although

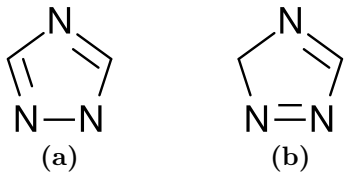


Figure 4.21.: (a) One correct formula for 1,2,4-triazole. (b) Similar to the case of pyrrole in Figure 4.20b this assignment will be considered wrong in the context of *fconv*.

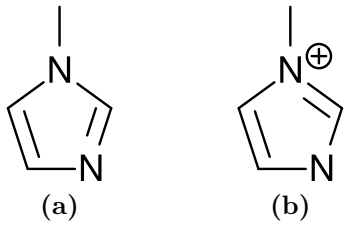


Figure 4.22.: (a) One correct formula for N-methylimidazole. (b) A charged resonance structure shall be unfavored again.

it would be a correct resonance structure, if a negative charge was assigned to the not doubly bonded carbon. Whenever it is possible to type aromatics in an uncharged form, it should be done so. *fconv* considers cases where aromatics have charged nitrogens, but no charged carbons. Thus, a structure like shown in Figure 4.20b would not be typed aromatic by the procedure described in the next section. Therefore, the first rule is that the weight of $C = C$ must be higher than that of $C = N$.

The second example, shown in Figure 4.21, is 1,2,4-triazole. Again, a structure with a not doubly bonded carbon should be avoided, hence the second rule is that the weight of $C = N$ must be higher than that of $N = N$.

The third example in Figure 4.22 shows N-methylimidazole. Although in this case the charged resonance structure would be considered aromatic in the aromaticity detection, it should still be avoided as long as an uncharged formula is possible. The third rule is therefore $weight((NR) = X) > weight((N^{+}R_2) = X)$.

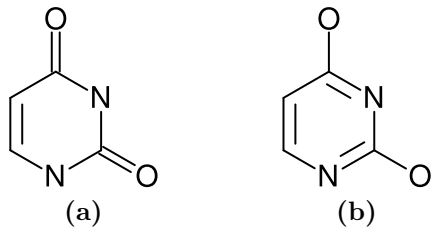


Figure 4.23.: (a) The preferred tautomer of uracil. (b) The aromatic tautomer.

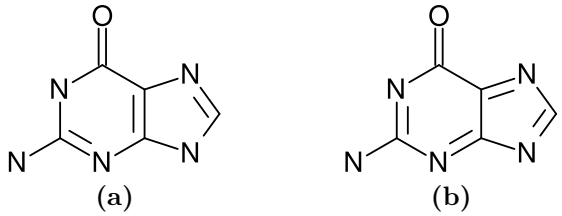


Figure 4.24.: (a) One possible tautomer of guanine. (b) A wrong assignment for guanine.

Figure 4.23 shows uracil in two tautomeric forms. It is known that in the crystal form only the lactam form occurs (Stewart and Jensen, 1967). The reason is that the delocalization of the amide bonds compensates the lost delocalization of the aromatic system shown in Figure 4.23b. Thus, rule number four is to decrease the weight of $C = N$ bonds, if they are part of a possible amide system.

In many biologically relevant structures there are also guanidino or amidino substructures that are part of a ring and also compensate a part of the loss of aromatic delocalization. An example is guanine as shown in Figure 4.24a. The assignment of double bonds as shown in Figure 4.24b is wrong, because it breaks the aromaticity of the imidazole ring without compensation. A possible solution would be to treat each ring individually in a first assignment of double bonds and then check it for aromaticity. However, for all cases *fconv* was evaluated on, the extension of rule one to $weight(C = C) > 2 \cdot weight(C = N)$ was sufficient.

The last rule does not concern atoms that are directly part of small planar

rings, but that connect these rings. An example are the biologically relevant heme structures. Whenever a planar carbon or nitrogen connects to planar sp2-hybridized rings, the same value is added to these exocyclic bonds as if they were part of such a ring. Of course, in case of heme, these connecting atoms already are parts of a 16-membered planar ring. But as mentioned in subsection 4.5.1 the maximum ring size considered by *fconv* is limited to ten atoms by default. If this number is increased to 16 it would have the same impact on the result as this last rule.

Furthermore, terminal double bonds (not to confuse with terminal edges in the previous subsections) are less likely, because the corresponding compounds would be highly reactive in most cases. Therefore, their weight is downscaled.

In addition, there is a threshold of a minimum weight that must be reached to assign a double bond. The detailed weighting as it is currently implemented into *fconv* is shown in Alg. 4.5.

After the maximum weighted matching is retrieved, all unmatched sp2-hybridized atoms are reset to sp3-hybridization. Finally, a check for isolated double bonds, which are not conjugated to any other double bond, is performed. For such isolated double bonds, the weight is compared to a higher threshold of 0.06. If it is lower, the involved atoms are also reset to sp3-hybridization. The rationale behind this is that in conjugated systems, higher deviations from ideal double bond geometries are possible compared to isolated cases.

Actually, the *fconv* user has the choice whether he wants to allow for aromatic rings with charged nitrogens (which is the default) or not (see option `-NCA` in *fconv*). In this regard, an example from the introduction of Labute (2005) shall be discussed. It concerns the ligand methazolamide from the PDB entry *'1bzm'* that is shown in Figure 4.25. Labute offensively states that "careless application of patterns for peptide recognition and heuristics such as aromaticity maximization" as applied by other approaches like that of Sayle may lead to incorrect structures. He gave the configuration shown in Figure 4.25a as an example for a correct assignment by his program and the configuration shown in Figure 4.25b as an example for an incorrect assignment by other programs. Actually, there are two errors concerning the example as given in Labute (2005).

Algorithm 4.5: The detailed assignment of weights for putative double bonds.

Input : Set E of possible double bonds with $e \in E = (u, v)$ with atoms u and v
Result: E with assigned weights

```
1  forall e ∈ E do
2      if (e.u and e.v are in the same planar sp2-ring) or (only one of them is in a
       planar sp2-ring and the other connects such rings) then
3          if e≙'C=C' then
4              if e.u or e.v is part of a possible amide then
                   e.weight := 8 + P(e.u, e.v, t, d)
5              else e.weight := 12.5 + P(e.u, e.v, t, d)
6          else if e≙'C=N' then
7              if possible amide then e.weight := 1 + P(e.u, e.v, t, d)
8              else if 'N' would be charged then e.weight := 5 + P(e.u, e.v, t, d)
9              else
10                 e.weight := 6 + P(e.u, e.v, t, d)
11         else if e≙'N=N' then
12             if e.u or e.v is part of a possible amide then
                   e.weight := P(e.u, e.v, t, d)
13             else if one 'N' would be charged then
                   e.weight := 2 + P(e.u, e.v, t, d)
14             else
15                 e.weight := 4 + P(e.u, e.v, t, d)
16     else if e is a possible exocyclic carbonyl bond then
17         e.weight := 1 + P(e.u, e.v, t, d)
18     else
19         if |e.u.bonded_atoms| = 1 or |e.v.bonded_atoms| = 1 then
20             e.weight := 0.8 · max(P(e.u, e.v, t, d), max(P(e.u, h, γ), P(e.v, h, γ)))
21         else
22             e.weight := max(P(e.u, e.v, t, d), max(P(e.u, h, γ), P(e.v, h, γ)))
23     if e.weight < 0.03 then e.weight := 0
```

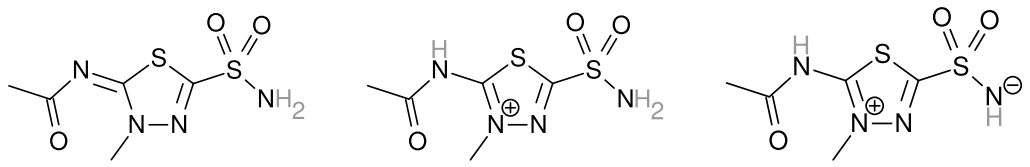


Figure 4.25.: Methazolamid from PDB entry *'1bzm'*. Implicit hydrogens are shown in gray color. (a) The formula that is claimed to be correct by Labute. (b) The formula that is claimed to be incorrect by Labute. (c) The most probable formula in neutral aqueous solution.

The first is that one nitrogen in the thiadiazole ring is missing, which has no influence on the following arguments and was corrected in the shown figures. The second error is that Labute assumes the assignment in (a) to be correct, just because it matches the formula given in the original publication for the X-ray structure (Chakravarty and Kannan, 1994). Actually, it is not incorrect, but the formula in (b) is also not incorrect. The authors of the structure might have used the compound shown in (a), but a simple protonation at the exocyclic nitrogen would have transformed it to (b). The structure has a resolution of 2 Å, so there is very low chance that the hydrogens of the compound could have been observed explicitly. Furthermore, the authors did not even try to discuss the protonation state in the complex. In aqueous solution, there will be an equilibrium between both forms, so it is generally wrong to speak of correct and incorrect assignments in such cases. However, one should try to pick out the most probable form, that is the form that dominates the equilibrium.

First consider Figure 4.25a. Here the exocyclic group on the left-hand side is no amide, because the p-orbital of the nitrogen is involved in a double bond with a ring carbon. Thus, in contrast to an amide, the nitrogen's lonepair is better described by an sp2-orbital and can therefore hardly overlap with the carbonyl p-orbital, hence there is no stabilizing delocalization as found in amides. Therefore, the lonepair can be freely accessed by a proton. Furthermore, together with the methylated nitrogen in the ring, there is an amidine substructure which is likely rather basic, because after protonation the

positive charge is delocalized over both nitrogens. In the example however, the protonation would lead to a structure as shown in Figure 4.25b, where first a stable amide bond is created and second the positive charge is delocalized in a then aromatized ring. It is therefore highly probable that the structure (b) is dominating in aqueous solution. Even more probable is that the sulfonamide group is deprotonated, which leads to the zwitterion shown in Figure 4.25c.

Using default parameters, *fconv* assigns Figure 4.25c. If the user decides to not allow for charged aromatic moieties, then a zero weight would be assigned in lines 8 and 13 of Alg. 4.5. Together with the decision to not deprotonate sulfonamides (see subsection 4.8.1), this would lead to Figure 4.25a. (Not changing the protonation state default for sulfonamides would lead to a combination of (a) and (c) that is not shown in the figure.)

4.6.3. Matching for Triple Bonds

A system of possible triple bonds is evaluated similarly to the case of double bonds, although the problem is very simple in comparison to the latter. First, much more candidates for triple bonds can be eliminated by checking for a linear geometry. Second, even in alternating triple/single-bond systems, the difference in the bond lengths usually allows for a clear differentiation between these types. And third, triple bonds cannot be involved in aromatic ring systems, so no extended weighting scheme is necessary.

The matching for triple bonds is executed before the double bond matching. Each unmatched sp-hybridized atom is reset to sp2-hybridization and is subjected to the double bond matching with one exception. If an isolated sp-hybridized atom is bonded to two sp2-hybridized atoms, then a double bond is assigned in both direction, hence it is typed as an cumulene structure.

4.7. Aromaticity Detection

Aromatic rings are highly stable, and therefore often occurring systems. In his pioneering work, Erich Hückel gave a quantum chemical explanation for

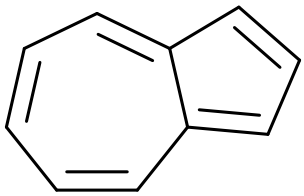


Figure 4.26.: The molecule azulene as an example for an aromatic system consisting of two fused rings.

the stability of such structures (Hückel, 1931a,b, 1932), using an early form of the MO-theory. A prerequisite for aromaticity is a cyclic system where electrons are delocalized. For delocalization, each ring atom must contribute at least one electron in a p-orbital. Furthermore, all p-orbitals must overlap which leads to the prerequisite of planarity. Hückel discovered that there must be $4n + 2 \quad (n \in \mathbb{N})$ electrons within the π-orbitals (the result of overlapping p-orbitals), because otherwise there would be unpaired electrons in at least two of these orbitals. In contrast to aromatic compounds, such open-shell configurations are highly reactive and therefore also called anti-aromatic.

fconv uses the Hückel rules to determine aromaticity. Therefore, after the double bonds are assigned to all planar and fully sp2-hybridized rings, the π-electrons must be counted and checked for the $4n + 2$ condition. First each ring is checked separately, though, if it does not have $4n + 2$ π-electrons, it could still be aromatic. An example is shown in Figure 4.26, where neither the five-membered, nor the seven-membered ring is aromatic, if examined individually. For the latter, there are $4n + 2$ electrons in double bonds, but the ring is not fully conjugated, that is one of the p-orbitals does not overlap with its ring neighbors. This is also the case for the small ring and furthermore, there are only $4n$ electrons. However, regarding both rings as one fused π-system, all Hückel criteria are fulfilled, and actually azulene is an aromatic compound. For this reason, non-aromatic individual rings are checked whether they are aromatic as fused ring systems.

The algorithm is defined in Alg. 4.6.

The ASSIGN_AROMATICITY function that is defined in Alg. 4.7 tests whether

Algorithm 4.6: Aromaticity detection.

Input : A set R of planar fully sp2-hybridized rings
Result: Assigned aromaticity attribute for all $r \in R$.

```
1  N := {}                                   // individual non-aromatic rings
2  forall r ∈ R do
3  |  ASSIGN_AROMATICITY(r)                  // see Alg. 4.7
4  |  if ¬r.is_aromatic then N := N ∪ {r}
5  Let F be the set of all possible combinations from N that lead to a fused ring
   system
6  forall r ∈ F do ASSIGN_AROMATICITY(r)
```

Algorithm 4.7: The ASSIGN_AROMATICITY function from Alg. 4.6.

Input : A ring r
Result: Assigned aromaticity attribute for r.

```
1  Let M be the match from Alg. 4.3
2  Let e(u, v) ∈ M = e(u) = e(v) be a double bond with atoms u and v
3  π_count := 0
4  forall a ∈ r do
5  |  if a ∈ M then
6  |  |  if e(a) ∈ r then π_count := π_count + 1
7  |  |  else Return
8  |  else if a.element ∈ {'N','O','S'} then π_count := π_count + 2
9  |  else Return
10 if π_count − 2  mod 4 = 0 then r.is_aromatic = true
```

each ring atom is either (i) part of a double bond in the ring (lines 5,6) or (ii) has no double bond but contributes two π electrons. The latter is possible only for nitrogen, oxygen or sulfur. Note that also aromatic rings with selenium are possible (e.g. selenophen), but currently not considered by *fconv*.

4.8. Assignment of Internal Atom Types

The definition of an extended set of atom types was an important step with respect to the development of the *DSX* scoring function (see section 8.3). A major requisite with respect to this application was the possible differentiation

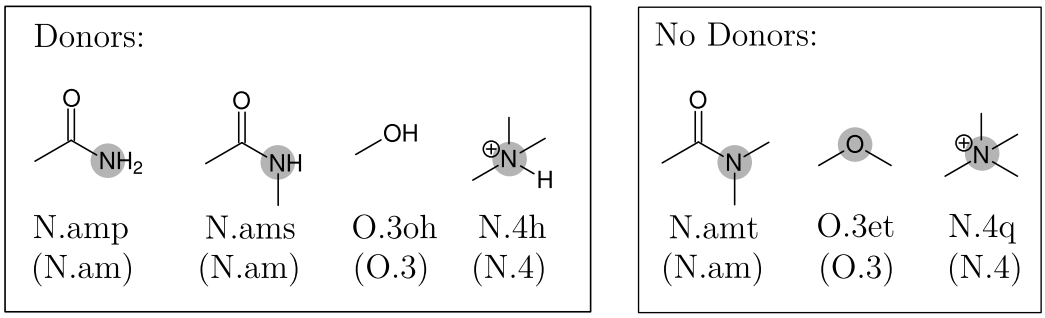


Figure 4.27.: Some exemplary *fconv* atom types. Their corresponding Sybyl types are shown in parentheses. The atoms in the left box can serve as donor, the atoms in the right box not.

between atoms that can serve as a donor and those that cannot. An example is shown in Figure 4.27. It can be seen that identical Sybyl types are used sometimes for donors and atoms without donor function.

A next demand was that a unique mapping from *fconv* atom types onto Sybyl types must be possible. Other types were introduced because chemically different behavior can be expected. Examples are types for 3-membered rings or types that differentiate a functionality depending on whether it is bonded to an aromatic ring or not (e.g. aliphatic or phenolic hydroxyl groups).

When writing this thesis there were 160 different atom types (*fconv* version 1.21). Their definitions can be found in Table A.2 and the mapping onto Sybyl types in Table A.3.

With the *fconv* option `'--m'`, users can choose between internal atom types or Sybyl types. However, they can also define their own mappings using an *fconv* definition file. A template definition file for Sybyl atom typing is generated with `'fconv --M=1'`. Now the mapping in this file can be changed. For example, the Sybyl specification allows for Sybyl type 'C.ar' only in 6-membered rings and uses 'C.2' in other cases. If the user is interested to set also other aromatic carbons to 'C.ar', he can change the original mapping from the corresponding internal types to 'C.ar' and then use the definition file for *fconv* with `'fconv options files --d=definition_file'`. Other parameters that can be changed in the definition file are the maximum ring size

N.ims N.ams

Figure 4.28.: The imide type on the left has a higher priority than the amide type on the right.

(see subsection 4.5.1), the allowance of charged aromatics (see subsection 4.6.2), Kekulization (see section 4.9), and various default protonation states (see subsection 4.8.1).

Currently, the detection of the different functional groups is based on a non-elegant chain of conditional statements. A major drawback of this approach is that it is not trivial to extend the set of internal atom types by new definitions. Furthermore, there is the risk of breaking some existing rules by the introduction of new types. An advantage however is the computational speed, because in contrast to conditional constructs, a more flexible and maintainable approach would require a generic pattern matching algorithm. The latter will be possibly implemented for future versions of *fconv*.

The order of internal atom types within the definition files is not arbitrary. Instead, it corresponds to the priority of types which is important in cases where more than one type would be possible. An example is given in Figure 4.28, where the amide type would also match for the imide case, but the imide has a higher priority and is thus set if possible.

4.8.1. Changing Protonation States

In the previous sections it was already stated that in most cases the input molecules have no attached hydrogens. In case of carbons this can result in ambiguous cases where a differentiation between sp2- and sp3-hybridization is not definitely possible, but only the most probable configuration with respect to geometric properties can be chosen.

The situation is even worse in case of functional groups that can be protonated

Table 4.3.: The groups that are considered for different protonation states in *fconv*, together with the applied default.

Functional group	Default
carbon acids	unprotonated (charged)
H_xSO_y and SO_2NH_x	unprotonated (charged)
H_xPO_y	unprotonated (charged)
guanidino groups	protonated (charged)
amidino groups	protonated (charged)
amino groups	unprotonated (uncharged)

or deprotonated in dependence of the environmental conditions, hence solvent, pH-value and intermolecular interactions. In theory, it is possible to differentiate between, e.g., protonated and deprotonated carboxyl groups, due to the different bond lengths to the oxygens in the protonated case. Unfortunately, this difference is not really reliable. Even in the high quality CSD structures, there are cases where the bond lengths are nearly equal in the protonated case. At least for structures as found in the PDB, the true protonation state is often unknown, because the resolution of these structures does not allow for a definite observation of hydrogens.

fconv does not intend to make a case-dependent estimation of protonation states. Instead, it uses defaults for different functional groups. These defaults can also be changed via the definition files that were introduced in the last section, and also via the option `'--p=xxxxxx'`.

Table 4.3 gives an overview of the considered groups and their defaults. Of course these defaults are only applied, if the input is ambiguous. If a hydrogen is present in the input file, the group will never be typed unprotonated.

4.9. Assignment of Bond Types

The major part for the task to assign bond types was already done in section 4.6. Using the information of the obtained matching, double bonds can be assigned correspondingly. In case of aromatic rings, the bond type 'ar' is assigned by

default. This can be changed via the flag 'Kekulize_aromatics' in the definition file or via the *fconv* option `'-KA'`.

Furthermore, the type 'ar' is also assigned to other partially delocalized systems like carboxylate groups or charged amidines. Currently, this cannot be changed by the user, but will be optional in future versions of *fconv*. What remains are bond types within distinct functional groups, e.g. 'am' for amide bonds (also used in imides) or double bonds in case of carbonyl groups.

An *fconv* feature that is utilized for local relaxation in *DSX* (see section 9.4) or by the application MiniMuDS (Spitzmüller et al., 2011), is the determination of freely rotatable bonds in a molecule. This is of course based on the bond types. Double bonds and also amide bonds in carbon amides are strictly not rotatable. Sulfonamides prefer particular dihedral angles, but in contrast to carbon amides can be considered rotatable. Not trivial is the case of single bonds, if they are part of a conjugated, hence partially delocalized systems. If the option `'-gfr'` is used, *fconv* reports single bonds in conjugated systems as rotatable only if they are not part of a planar torsion. In contrast, using the option `'-gfr2'`, all single bonds are reported as freely rotatable bonds.

4.10. The Problem of General Ambiguity

Before a validation of the atom type perception quality is presented, the problem to decide what is correct and what is incorrect must be discussed. In the introduction, a first problem was explained, namely that the Sybyl definitions can be interpreted differently (see Figure 1.1). With respect to this, an assessment of assigned atom types is straightforward for a particular program. They are correct, if the results conform to the program's own definitions.

Also the problem of ambiguous protonation states was discussed (see subsection 4.8.1 and Figure 4.25). In contrast to the first problem, this remains a general problem and for a particular complicated example, ten chemists might give ten different answers of what is correct.

The same holds true for the third and fourth problem of general ambiguity,

namely tautomerism and different oxidization states. Unfortunately, these problems concern many biologically relevant molecules. If hydrogen information is missing, a differentiation between a keto and an enol form is not possible, as the bond lengths are not different with respect to the experimental accuracy. Instead of relying on marginal bond length differences, *fconv* tries to be as robust as possible. Therefore, molecules with possible keto-enol tautomerism are always assigned to the keto-form.

Examples for ambiguous oxidization states are the quinone/hydroquinone equilibrium or the $NADPH/NADP^+$ equilibrium. In the first case *fconv* always assumes the aromatic type (the hydroquinone). In the second case also the aromatic state ($NADP^+$) is assigned by default, but here the user can change the behavior, as a charged state is involved (see subsection 4.6.2). Note that there are numerous examples in the PDB, where the authors claim to have $NADPH$ in their structures, but all bond length in the nicotinamide moiety are equivalent. An example for the latter would be *'7cat'*.

4.11. Performance of the Atom Type Perception

4.11.1. Quality of Assignments

Assessing the quality of the atom type perception and comparing it with existing state-of-the-art tools is not trivial, due to the facts discussed in the previous section. Each molecule in an appropriate test data set has to be inspected visually, comparing it with the reference literature and considering cases where also alternative assignments would be valid. Labute (2005) assembled a test set consisting of the ligands from 179 PDB complexes (see Table 3 in the original paper) that was also used by Zhao et al. (2007). Furthermore, Zhao also evaluated the results of OpenBabel (version 2.0.0) on this data set. Table 4.4 summarizes those previous results with the results obtained with *fconv*. In his study, Zhao reported nine cases where I-interpret failed to perceive the correct structure (Table 6 in the original paper). For curiosity, in three of the cases (*'1etr, 1rne, 2xim'*) the formula reported as correct is actually incorrect and

Table 4.4.: Success rates of correct assignments for the test set proposed by Labute (2005).

Program/Method	Success rate
fconv	97.2 %
I-interpret	96.6 %[a] (95.0 %[b])
Labute's method	93.9 %[b]
OpenBabel	79.3 %[b]

[a]Adjusted result (see text).
[b]Results cited from Zhao et al. (2007)

the one reported incorrect is the correct one. In case of *'1etr'* and *'2xim'*, it is not sure how Zhao et al. came to the "incorrect correct structure". For *'1rne'*, it is likely that they used the formula as depicted in the PDB as a reference. The latter one is wrong, as maybe other pictures in the PDB, because they are also generated automatically.

For Table 4.4, the results of Zhao were corrected in a way that these three cases are counted as correct, which leads to an increased success rate. Neither is it approved that there are no further errors in the evaluation by Zhao, nor is this the case for Labute's results or even the results obtained for *fconv*. Human inspection will never be free of errors, thus the numbers given should not be overestimated. An at least valid conclusion would be that *fconv*, I-interpret and Labute's method perform on a similar level, while results obtained with OpenBabel are worse.

From the five cases where *fconv* fails to assign correct bond or atom types, three are shared with the eleven incorrect results for Labute's method. In case of *'1aaq'*, there is an ester group with a non-planar carbonyl carbon and a dihedral angle of 69° for the ester bond. Consequently, the carbonyl is perceived as alcohol and the complete functional group becomes a hemiacetal. In case of *'1aqb'*, one of the double bonds is part of a dihedral angle of 23.3° and is thus perceived as single bond. In case of *'2r04'*, the 4,5-dihydro-2-oxazolyl moiety is

Table 4.5.: A comparison between run-times of *fconv* and OpenBabel for the atom type perception of 165 222 CSD structures.

Program	Time[a]
fconv	161 s
OpenBabel	297 s

[a]Measured on an Intel(R) Core(TM) i7 CPU Q 720, 1.60GHz

perceived as aromatic oxazole ring, because it is perfectly planar.

Another incorrect perception from *fconv* is shared with I-interpret. In *'8xia'*, the aldehyde group of D-xylose (open chain form) is perceived as alcohol. It has exactly the same bond length as the terminal hydroxyl oxygen on the other side of the chain. Thus, due to missing hydrogens an automatic method can never correctly perceive this structure. It would either assign a hydroxyl group at both ends of the chain (like *fconv* and I-interpret) or an aldehyde at both ends. Interestingly, Labute did not report a problem with this structure.

Finally, there is also one case where only *fconv* obtained an incorrect result. In *'3fx2'*, one of the three hydroxyl groups in flavine mononucleotide is perceived as carbonyl group. In contrast to the other two hydroxyles, it has a trigonal planar geometry and also a shorter bond length of only 1.4 Å (1.5 Å for the others).

4.11.2. Computational Speed

Apart from the achieved quality, also the computational speed of the atom type assignment is an important aspect. *fconv* is designed to handle input data with millions of compounds and its libraries were also developed to be an integral part of the scoring function *DSX* (see Part II). The latter is intended as a method of especially high speed and this speed should not suffer from an integrated atom type perception.

For a benchmark, the CSD sample introduced in subsection 4.4.1 was used

again. Table 4.5 informs about the time that was needed to read all 165 222 molecules from MOL2, reassign atom types and bond orders, and write out the result as MOL2. For comparison, also the time OpenBabel (version 2.2.3) needs for the same task is reported.

Chapter 5.
5 Additional Exemplary Features of fconv

Many features implemented in *fconv* rely on the internal atom types that were described in the previous chapter. Examples would be the function to add hydrogens to a molecule, or the function to determine the number of freely rotatable bonds.

Other features only make use of connectivity information or attributes that are related with element types (like vdW-radii). Two categories of such features shall be described briefly within this chapter.

For all other features, the *fconv* help function in combination with the available source code serves as a documentation.

5.1. RMSD Values, Alignments, and Substructure Search

The calculation of RMSD values, spatial superimpositions, and substructure searches are frequently occurring tasks in the daily work with molecule data. The *fconv* solutions to all these tasks have a common foundation and are therefore presented within the same section.

5.1.1. Definitions

Spatial superimposition of two molecules is often referred to as "alignment". However, the term "alignment" can be interpreted differently. To clarify subsequent explanations, the definitions of different alignment types, as well as relevant definitions correlated with this topic, will be given in the following.

- *Pairwise alignment*: A pairwise alignment is an alignment of exactly two objects, whereas an alignment of more objects is called *multiple alignment*. In the following, the term alignment always refers to pairwise alignments.

- *Functional alignment*: This kind of alignment does not comprise any geometric changes. Given two objects A and B, a functional alignment is a pairwise match between elements of A and elements of B. An example is shown in Figure 5.1a. Both objects in the figure consist of three elements (a_1, a_2, a_3 and b_1, b_2, b_3 respectively) and the function of these elements shall be determined by their shape. The functional alignment between the objects would be $F(A, B) = \{(a_1, b_3), (a_2, b_1), (a_3, b_2)\}$. Thus, performing a functional alignment is only the retrieval of a one-to-one correspondence between the elements of both objects.

- *RMSD*: The root mean square deviation between two objects A and B is calculated on the basis of a functional alignment $F(A, B) = \{(a_m, b_n), \dots\}$ $= \{f_1, \dots, f_i\}$. If $d(f)$ denotes the Euclidean distance between $a(f)$ and $b(f)$, the RSMD is defined as

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{k=1}^{k=i} d(f_k^2)}{i}}$$

- *Spatial alignment*: A spatial alignment is the optimal superimposition of two objects, given their functional alignment. It consists of rotation and translation of one of the objects, such that the RMSD between both objects is minimized. Figure 5.1b shows the spatial alignment from the left onto the right structure, based on the functional alignment from (a).

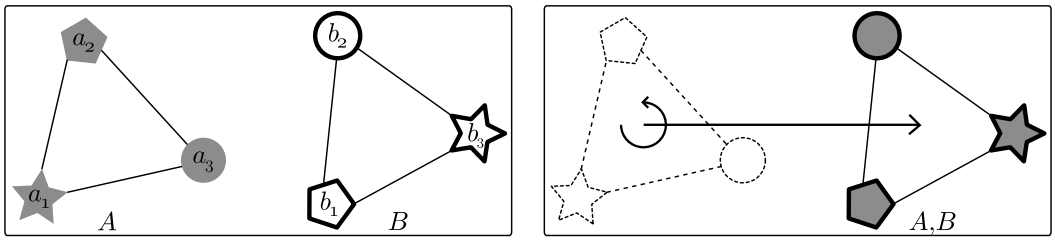


Figure 5.1.: (a) Two objects A and B, consisting of elements a_1, a_2, a_3 and b_1, b_2, b_3 respectively. If the function of an element is characterized by its shape, then the functional alignment between the two objects would be $\{(a_1, b_3), (a_2, b_1), (a_3, b_2)\}$. (b) Spatial alignment of the left object onto the right object, performed by rotation and translation.

- *Partial alignment*: A partial alignment is a functional alignment $F(A, B)$, where at least one element $a \in A$ or $b \in B$ is not part of the alignment, hence has no match in the other object $((A \cup B) \setminus F(A, B) \neq \{\})$.

- *Complete alignment*: In contrast to a partial alignment, all elements are matched in a complete alignment $((A \cup B) \setminus F(A, B) = \{\})$.

5.1.2. Method for Spatial Alignments

Given a pairwise functional alignment between two objects A and B, the spatial alignment from A onto B is the transformation of A that minimizes the RMSD. The challenging part of this transformation is to find the optimal rotation. An analytic solution for this problem was proposed by Kabsch (1976, 1978) and calculates an orthonormal rotation matrix for the optimal rotation. Computational calculations with real-valued numbers are performed with limited precision and can lead to numerical unstable results. In case of the Kabsch algorithm, this can lead to non-orthonormal matrices that must be re-normalized subsequently in order to obtain proper rotations. This re-normalization is a non-trivial problem that must be addressed when using the Kabsch algorithm and after each manipulation of rotation matrices.

Another analytic solution was presented by Horn (1987) and calculates a unit quaternion for the optimal rotation. Also, these quaternions must be re-normalized to guarantee numerical stability, but in contrast to orthonormal matrices, this re-normalization is straightforward. Therefore, the method of Horn was implemented to *fconv*.

5.1.3. Functional Alignment and Derived Features

A variety of different approaches to generate functional alignments exist. The choice of an appropriate algorithm depends on the definition of "function", and the size of the problem (hence the molecule size).

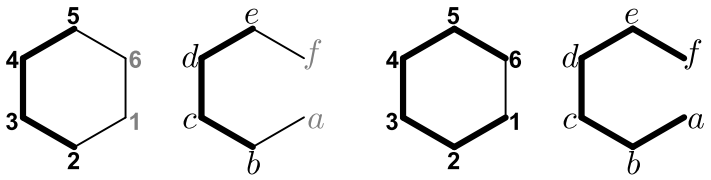
5.1.3.1. Small Molecules

In case of small (ligand-sized) molecules, a graph-based solution is usually the method of choice. This subsection is structured as follows:

1. The graph definition for small molecules will be given.
2. The graph theoretical approach used for functional alignments is described in general.
3. The functional alignment for small molecules will be defined in detail.
4. The general approach is specified with respect to implications from the alignment definition.
5. RMSD calculations, spatial alignments and substructure searches with *fconv*.

Small Molecule Graphs A small molecule is represented by an undirected graph $G(V, E)$, where the set of vertices V corresponds to the atoms except for all hydrogens, and the set of edges E to the bonds. Vertices and edges can be labeled corresponding to various attributes like element type, atom type, valence or bond type.

A line graph $L(G)$ is a graph where each vertex corresponds to an edge in G, and an edge between two vertices exists, if their corresponding edges in G share a common vertex.



(a) MCIS **(b)** MCES

Figure 5.2.: The common subgraphs are depicted as bold edges. (a) Neither '1' and 'a', nor '6' and 'f' are matched in a maximum common induced subgraph (MCIS), because an edge exists between '1' and '6' but not between 'a' and 'f'. (b) In a maximum common edge subgraph (MCES), all common edges are present.

General Approach The problem of finding functional alignments between two molecules A and B corresponds to the problem of finding maximum common subgraphs of $G(V_A, E_A)$ and $G(V_B, E_B)$. In case of partial alignments, it must be differentiated between maximum common induced subgraph (MCIS) and maximum common edge subgraph (MCES). Figure 5.2 illustrates the difference between MCIS and MCES. Atoms '1', '2', 'a', and 'f' would not be part of an MCIS, because the connectivity of '1-2' differs from that of 'a-f'. Finding MCESs for $G(V_A, E_A)$ and $G(V_B, E_B)$ can be easily traced back to the problem of finding MCISs for the line graphs $L(G_A)$ and $L(G_B)$ (Koch, 2001). Therefore, the method will be described only for MCISs.

A popular method to determine MCISs is to re-formulate the task as the problem to find maximum cliques in a product graph $H(V_P, E_P) = G(V_A, E_A) \circ_P G(V_B, E_B)$. In this context, the graph product is usually defined as follows:

- The vertices of the product graph are the Cartesian product $V_P = V_A \times V_B$ and an edge between $v_P = (v_A, v_B)$ and $u_P = (u_A, u_B)$ exists, if
- $v_A \neq u_A$ and $v_B \neq u_B$ and
- v_A, u_A are adjacent in A and v_B, u_B are adjacent in B, or both are not adjacent in the respective graphs.

It was shown by Levi (1972) that a maximum clique in such a product graph corresponds to an MCIS in the factor graphs. The terms "vertex product

graph", "associative graph" or "modular graph" are used in the more recent literature (Koch, 2001; Raymond et al., 2002), but they are not well defined (Note that there exist also well defined graph products for other purposes.). Levi used the term "compatibility table", which characterizes the relation between MCIS in the factor graphs and cliques in the product graph much better. Searching for a clique corresponds to the search for a set of vertex pairs (from A and B) that are all compatible to each other. The definition of compatibility can be varied with respect to the underlying problem. For example in case of vertex labeled graphs, vertex pairs $(v_A, v_B) \in H(G_A, G_B)$ are only allowed if v_A and v_B share a common label (Levi, 1972).

After definition of a product graph, the clique search can be applied. For *fconv*, a variation of the Bron and Kerbosch algorithm (Bron and Kerbosch, 1973) was implemented. Many other algorithms for the clique problem have been proposed, but the variations of the BK-algorithm are still considered the best ones for general graphs (Koch, 2001; Tomita et al., 2006).

Unfortunately, reporting all maximal cliques is an NP-complete problem. An optimal solution has a time bound of $O(3^{|V|/3})$, because there exist $3^{|V|/3}$ maximal cliques in a worst case graph (Tomita et al., 2006). In case of functional alignments, only maximum cliques are of interest, that is maximal cliques with maximum cardinality. Nevertheless, the time bound remains exponential even for maximum cliques. Therefore, a reduction of the number of edges and/or vertices in the product graph can be expected to speed up the computation significantly, if the reduction is in polynomial time. Possible reductions correspond to modifications in the definition of compatibility.

Definition of the Functional Alignment for Small Molecules Recall the difference between MCIS and MCES as shown in Figure 5.2. For some chemists, the MCES would fit better to their imagination of a maximum common substructure. Currently, *fconv* determines MCISs, but this could be extended in future versions, offering both options to the user.

Other chemists would even say there is no common substructure at all, because in one molecule all atoms are ring atoms and in the other there is

just an open carbon chain. These different points of view are correlated to the definition of compatibility in the product graph. The term "functional alignment" was defined in subsection 5.1.1 rather general. Prior to the discussion of compatibility rules, it shall be specified to a definition of a "*valid functional alignment for small molecules*". For *fconv*, this definition can be formulated as follows. A functional alignment $F(A, B)$ for two molecules A and B is valid, if and only if:

- There exists a sequence of rotations, translations, and torsions of free rotatable bonds that results in a superimposition such that a and b have identical coordinates for all atom-atom pairs $(a, b) \in F(A, B)$ (assuming equal bond lengths).
- a and b have the same element type for all $(a, b) \in F(A, B)$.
- a and b are either both part of a ring or both are no part of a ring for all $(a, b) \in F(A, B)$.

Specified Approach Due to the given definition of valid alignments, only atoms with equal element types and ring properties can be paired in the product graph. As already mentioned, it is desired to further reduce the number of vertices in the product graph. One possibility could be to use atom types instead of element types as labels. However, this cannot be considered a robust solution. If for example the RMSD value of docking solutions with respect to a reference shall be determined, then it could easily happen that some poses have a planar torsion angle that is non-planar in other poses. The result could be a different atom type assignment for the atoms involved in the torsion. In consequence, RMSD values would be calculated incorrectly, hence for only partial functional alignments.

As robustness is more important compared to speed, *fconv* does not use atom types to reduce the number of vertices. Nevertheless, it is also possible to use an atom's neighborhood information in an unambiguous manner. If only complete alignments are allowed (by user decision), *fconv* calculates two

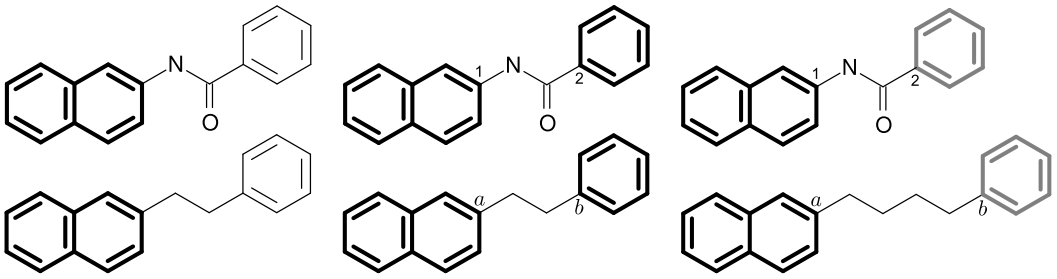
Algorithm 5.1: Calculation of atom compatibility hashes.

Input : A set of atoms A
Result: Two hash values hash1 and hash2 for each $a \in A$.

```
1  forall a ∈ A do
2      a.hash1 := 0
3      forall b ∈ a.bonded_atoms do
4          switch b.element do
5              case 'C' a.hash1 := a.hash1 + 1
6              case 'N' a.hash1 := a.hash1 + 5
7              case 'O' a.hash1 := a.hash1 + 21
8              case 'S' a.hash1 := a.hash1 + 85
9              case 'P' a.hash1 := a.hash1 + 341
10             case 'F' a.hash1 := a.hash1 + 1365
11             case 'Cl' a.hash1 := a.hash1 + 5461
12             case 'Br' a.hash1 := a.hash1 + 21845
13             case 'I' a.hash1 := a.hash1 + 87381
14 forall a ∈ A do
15     a.hash2 := 0
16     forall b ∈ a.bonded_atoms do a.hash2 := a.hash2 + b.hash1
```

different hash values for each atom. The calculation of these hash values is shown in Alg. 5.1. The rationale behind the numbers used for hash1 is the maximum atom valence of four. Four bonded carbons must result in a different value than one (or more) nitrogen, and four bonded nitrogens must result in a different value than one oxygen, and so on. Atom pairs (a, b) in the product graph are only possible, if a and b have equal hash values. Please note again that these hashes are not used in case of partial alignments.

To reduce the number of edges in the product graph, one could consider the bond types in the factor graphs in addition to the adjacency information. However, this would result in unstable assignments for the same reason as discussed for the usage of atom types to reduce vertices. Another possibility would be the usage of Euclidean distances d in the factor graphs, hence to connect (v_A, v_B) and (u_A, u_B), only if $d(v_A, u_A) \approx d(v_B, u_B)$. Thus, identical molecules with different conformations would not be matched, which is not in



(a) Connected **(b)** *fconv* **(c)** Disconnected

Figure 5.3.: The maximum common subgraphs are depicted as bold edges. (a) The naphthalene ring is the maximum connected subgraph. (b) If subgraphs are not required to be connected, the phenyl ring would also be part of the MCIS. In *fconv* the depicted subgraph would be detected, because the path length between '1' and '2' equals the path length between 'a' and 'b'. (c) The disconnected subgraph including the phenyl ring would not be detected by *fconv*, because the path length between '1' and '2' does not equal the path length between 'a' and 'b'. Therefore, *fconv* would determine only the naphthalene as MCIS.

agreement with the given definition of valid functional alignments. If one is interested in similar conformations, these can be identified by clustering them with respect to their RMSD values (e.g. "`fconv -clust multi.mol2`").

Fortunately, there is a two-dimensional (hence conformation independent) equivalent to the Euclidean distances, that is the shortest path between v_A, u_A in the molecule graph. An easy-to-implement method for calculating the shortest path information of all possible vertex pairs would be the Floyd Warshall algorithm (Floyd, 1962) that solves the problem in $O(|V|^3)$ time (note that *fconv* uses a BFS algorithm instead).

At this point a further differentiation of MCISs must be discussed. In the literature, algorithms for MCIS searches are classified whether they yield only connected or also disconnected subgraphs (Koch, 2001; Raymond et al., 2002). The difference is illustrated in Figure 5.3. Due to the compatibility criterion of equal path lengths between v_A, u_A and v_B, u_B, *fconv* neither belongs to the methods that determine only connected MCISs, nor to the methods that determine all disconnected MCISs. In Figure 5.3b, *fconv* would detect the

depicted disconnected MCIS, because the path lengths between atoms in the phenyl ring and atoms in the naphthalene ring are equal. In Figure 5.3c, the chain between the rings is elongated in one molecule. Thus, path lengths become different and the phenyl ring is no longer part of the MCIS.

This behavior is in agreement with the definition for valid functional alignments on page 77, because in case of Figure 5.3c, it would not be possible to transform one of the molecules such that both, the naphthalene and the phenyl atoms, superimpose at the same time. The definition was not chosen that way, to allow for the reduction of the product graph, but it fits with "chemical intuition" about equal substructures.

If one is interested to superimpose a number of compounds only with respect to their naphthalene ring and regardless of other matching substructures in the molecules, there is also a straightforward solution with *fconv*. The user only needs to supply an unsubstituted naphthalene and use *fconv* to spatially align it on a compound of choice. Subsequently, all other compounds can be aligned onto this template.

There remains one important point that must be considered with respect to the definition of valid functional alignments. Atom-atom pairs that are compatible with respect to their element types, ring properties, and connectivities can still be incompatible with respect to their stereochemistry. Two cases where stereochemistry makes a difference are shown in Figure 5.4. Neither E/Z-isomers, nor enantiomers should be aligned, because they represent different compounds.

Unfortunately, this cannot be easily transformed into an additional compatibility criterion for the product graph, because the stereochemistry can only be evaluated with respect to a particular functional alignment (MCIS). Therefore, *fconv* determines all MCISs initially and then evaluates the stereochemistry for each of them. If stereochemistry does not fit, the corresponding alignment is discarded.

This check is not only necessary for stereogenic centers and E/Z-stereobonds, but also in case of tetrahedral atoms or double bonds with two equal substituents. In Figure 5.5a, two equivalent maximum functional alignments

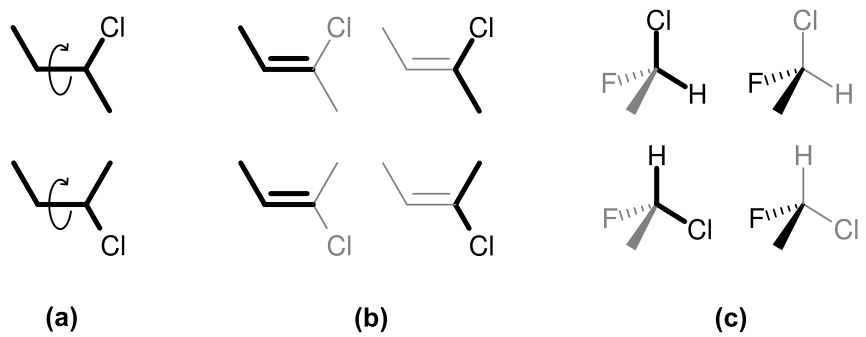


Figure 5.4.: Functional alignments shown in bold. (a) A complete alignment is possible, because of the freely rotatable bond. (b) The molecules in the first row are E/Z-stereoisomers of the molecules in the second row. It is not possible to change their conformation in a way that they become chemically equal. Therefore, a functional alignment of them is only partially possible. There are two such partial functional alignments shown on the left and the right, respectively. (c) The molecules in the first row are enantiomers of the molecules in the second row. It is not possible to superimpose all atoms by any conformational or rotational changes. Instead of a complete functional alignment, different partial alignments are possible.

are possible due to the symmetric molecule. In Figure 5.5b, the atoms d and e are not equivalent with respect to the other atoms a, b, c. Therefore, only one maximum functional alignment is possible.

The perception of stereoisomers is necessary to recognize whether an alignment corresponds to equal chemicals. Now the question may arise why it is necessary to recognize also double bonds and tetrahedral atoms with two equal substituents. If one is only interested in the identity of two molecules it does not matter whether only one or several complete alignments are possible. However, in case of RMSD calculations that are based on a functional alignment, the situation is different. For the molecules in Figure 5.5a, two RMSD values would be calculated, corresponding to the two maximum functional alignments. Then, the lower value will be reported by *fconv*. Depending on the spatial orientation of the molecules shown in Figure 5.5b, it would be possible to obtain a lower RMSD value for the invalid alignment, thus leading to an incorrect result.

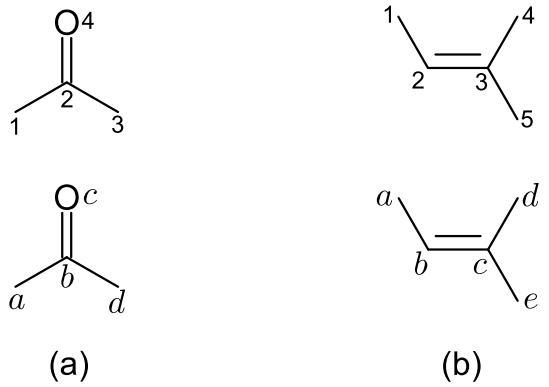


Figure 5.5.: (a) Valid alignments are $(1,a)(2,b)(3,d)(4,c)$ and $(1,d)(2,b)(3,a)(4,c)$. (b) A valid alignment is $(1,a)(2,b)(3,c)(4,d)(5,e)$. An invalid alignment would be $(1,a)(2,b)(3,c)(4,e)(5,d)$, because d and e are not equivalent with respect to a,b,c.

RMSD Calculations Usually, RMSD values are calculated only for complete functional alignments, hence for equal compounds. "`fconv -rmsd A.mol2 --s=B.mol2`" calculates three different RMSD values between molecule A and molecule B:

The first value reported, belongs to the RMSD that is based on the functional alignment only (no superimposition), without consideration of any hydrogen atoms. It can be used to measure how different certain docking poses are. Using "`fconv -clust multi.mol2`", it is also possible to cluster all poses within the multimol2 file with respect to this first RMSD value (for the various options of the clustering see "`fconv -h`").

The second value also corresponds to the RMSD without superimposition, but it includes hydrogen atoms for the calculation. Hydrogens are not used within the graph matching, but subsequently matched on the basis of a given functional alignment.

The third value is the RMSD after spatial alignment (superimposition) of A and B. It is useful to measure how different two conformations are (independent from translation and rotation).

If both molecules are enantiomers or diastereomers, *fconv* returns -2. If they are not equal for any other reason it returns -1.

If one is interested which atoms are aligned to each other in detail, the option "`---d`" can be used. Note, that the best (lowest RMSD) alignment with respect to the first and second value can differ from the best alignment after superimposition. Therefore, both functional alignments are reported.

If also an RMSD for non-equal compounds with a common substructure should be determined, the command "`fconv -rmsd2 A.mol2 --s=B.mol2`" can be used. Currently, the latter option does not consider stereochemistry, which is important when interpreting the results.

Spatial Alignments A spatial alignment is simply a call of the RMSD function, but in addition the aligned molecule is written out after applying the transformation that yields the third RMSD value (described in the previous paragraph). In consequence, there are two different options for spatial superimposition (optimal alignment): "`fconv -oa A.mol2 --s=B.mol2 --t=A_transformed`" for the superimposition of equal compounds and "`fconv -oa2 A.mol2 --s=B.mol2 --t=A_transformed`" for compounds with common substructure, respectively.

Instead of transforming the molecule in "`A.mol2`", it is also possible to supply an additional file "`C.mol2`" that shall be transformed, e.g. "`fconv -oa A.mol2 --s=B.mol2 --s2=C.mol2 --t=C_transformed.mol2`". The latter command calculates the spatial alignment from `A` onto `B`, but applies the corresponding transformation to `C`. A practical application of this feature would be the situation that two complexes show the same bound ligand in different proteins. In this situation it can be useful to superimpose the binding pockets based on the superimposition of the ligands.

Substructure Searches The command "`fconv -ss multi.mol2 --s=pattern.mol2`" can be used to search in the file `multi.mol2` for all occurrences of the complete structure contained in the pattern file. Several additional options are possible. For example, the user can mark particular atoms in the pattern file, to indicate that their valency (adjacency) must be matched exactly. Another example is the possibility to require exact hybridization states for

certain atoms.

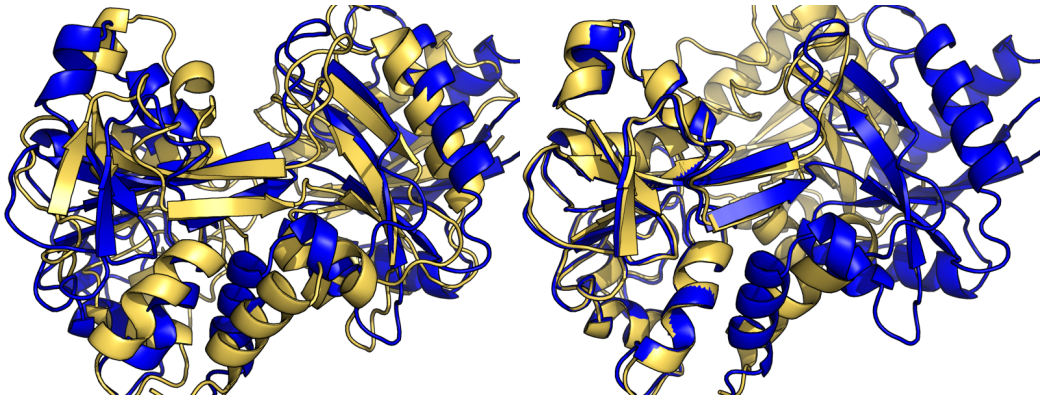
5.1.3.2. Large Molecules

The MCIS search introduced in the last section can also be applied to large molecules, but instead of milliseconds for ligand-sized molecules, a match of two complete proteins takes minutes up to weeks. Furthermore, the result would rarely fit with the expectations from a protein-protein alignment.

Sequence-based In case of small molecules, "function" was defined in terms of two-dimensional attributes (only two-dimensional connectivity information was used). In case of proteins, or other macromolecules such as DNA or RNA, a common approach is to use the one-dimensional sequence of amino acids, hence to use a sequence alignment as functional alignment. For this purpose, the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) was implemented into *fconv*. It makes use of dynamic programming and generates an optimal alignment with respect to a given scoring function. Scores are usually derived from a substitution matrix that gives a high score for exactly matching amino acids, and lower scores for mismatches depending on the probability for the two amino acids to be exchanged. Furthermore, a penalty for gaps in the sequence is applied.

Currently, no substitution matrix is implemented to *fconv*, and only a positive score for matches and a linearly scaling penalty for gaps is applied. Therefore, good alignments can only be expected for proteins with a high similarity. To perform a protein-protein superimposition that is based on a sequence alignment, the command "`fconv -oa A.pdb --s=B.pdb`" can be used.

Shape-based Another approach for protein alignment is to use 3-dimensional structural features. A number of literature-known tools for this purpose exist. DALI (Holm and Sander, 1993), CE (Shindyalov and Bourne, 1998) and MAMMOTH (Ortiz et al., 2002) are examples for programs that perform 3D-alignments, based on a partition of the proteins into short peptides that



(a) sequence **(b)** structure

Figure 5.6.: Superimposition of two crystal structures (*'1a8e,1bp5'*) of the N-lobe of human transferrin. Both structures have 100 % sequence identity. *'1a8e'* (yellow) is the closed form with a ferric ion and crystallized in spacegroup $P2_12_12_1$. *'1bp5'* (blue) is the apo form and crystallized in spacegroup $P2_1$. (a) The superimposition is based on a sequence alignment. (b) Superimposition with *fconv*, based on a backbone shape alignment. One domain is aligned very well.

are compared by means of distance matrices or various angles. The focus of these programs is on shape-based measures of protein similarity and protein classification with respect to folding patterns. Furthermore, there are also approaches to perform flexible 3D-alignments where also conformational changes are possible. An example for the latter would be the web service RAPIDO (Mosca and Schneider, 2008).

For *fconv* a 3D-structure-based alignment was developed with a different focus. It aims at the alignment of only those parts of two proteins that lead to very low RMSD values, if superimposed. The goal is best illustrated by an example. Figure 5.6 shows two PDB structures with 100 % sequence identity, but different conformations. After superimposition based on a sequence alignment (Figure 5.6a), the similarity between both structures is visible, but it is hardly possible to determine at which point the conformational changes occur. In contrast, Figure 5.6b shows the superimposition after shape-based alignment with *fconv* ("`fconv -oa2 1a8e.pdb --s=1bp5.pdb`"). There, it can

be seen that one of the two domains (left part) is superimposed very well. Now it becomes obvious that the major difference between both conformations is found in the two beta sheets that connect both domains. Only 55 from 328 $C\alpha$-carbons were used for the spatial alignment.

Another use-case would be the alignment of polyalanine chains that can occur in intermediate steps of crystallographic structure refinement. In this case, the lack of sequence information also demands for a shape-based approach.

Method First, the idea behind the shape-based protein alignment will be outlined in analogy to the graph matching of small molecules. Proteins can be represented as graphs $G(V, E)$, where vertices correspond to amino acids and edges to the adjacency of amino acids. The vertices can be labeled with respect to backbone properties describing the shape, hence dihedral angles. To align two proteins A and B, a product graph $H(V_P, E_P)$ can be generated, where each $v_p = (v_A, v_B)$ is a pair of equally labeled amino acids. An edge between two vertices (v_A, u_A) and (v_B, u_B) in the product graph could be generated, if they do not share a vertex from the factor graphs and the distances $d(v_A, u_A)$ and $d(v_B, u_B)$ are equal within a given tolerance. In addition to the distances, also the spatial orientations of the amino acids can be used.

In case of huge proteins, product graphs would have more than a million vertices. Apart from the general problem of exponential time for the clique search, there are three more problems that prohibit an effective computation. First, it would not be possible to use an adjacency matrix for the product graph, due to memory limitations. Second, also the distance matrices for the factor graphs would require an enormous amount of memory. And third, even if there would be enough memory for the distance matrices, the computation of all-against-all distances would take much time.

For these reasons, a heuristic method was developed that makes use of the described compatibility criteria. It also starts with a product graph, where amino acids are paired, if their dihedral angle ψ is equal within a tolerance of $\pm 9°$. In the literature, ψ is usually measured as the torsion (N, CA, C, N'), but this definition is unsuitable to characterize an individual amino acid, because

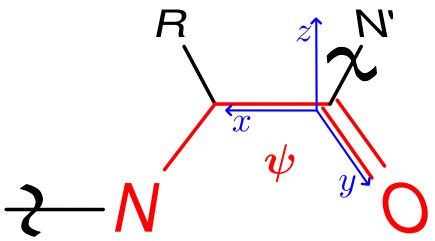


Figure 5.7.: Geometric representation of an amino acid: The dihedral angle $\psi('N','CA',' C',' O')$ (red) characterizes the backbone conformation. Three vectors $x('CA',' C')$, $y('O',' C')$, and $z = x \times y$ (blue) characterize the orientation in space.

the angle is calculated from atoms of two different amino acids (N' is from the next amino acid in the sequence). Therefore, *fconv* defines ψ as the angle formed by (N, CA, C, O), which is essentially just 180° different from the literature definition. The angle ψ and three vectors that will be explained later are shown in Figure 5.7. The ψ-tolerance of ±9° was chosen empirically. To get an idea about the discriminating power of the corresponding 18° interval, one can consult a Ramachandran plot (Ramachandran et al., 1963; Kleywegt and Jones, 1996; Hovmöller et al., 2002).

Two vertices (v_A, u_A) and (v_B, u_B) from the product graph are connected, only if v_A, u_A are adjacent in A and v_B, u_B are adjacent in B. Thus only amino acid pairs, where all amino acids are directly connected in the respective protein are connected in the product graph. Therefore, maximum subgraphs of A and B do not correspond to cliques in the product graph. Instead, the algorithm presented in Alg. 5.2 iteratively determines compatibility clusters in the product graph. It starts with a number of clusters equal to the number of vertices in the product graph, hence each cluster contains one vertex initially (line 5). In the main loop (line 6), each cluster is checked for possible extensions. Only vertices that are adjacent to those vertices $c.N$ that were added in the last cycle are checked for extensions. Recall that two vertices $v_p = (v_A, v_B)$ and $u_p = (u_A, u_B)$ are adjacent, only if v_A, u_A are adjacent and v_B, u_B are adjacent. In consequence, the iteratively growing clusters correspond to growing protein chain fragments without any gaps.

Algorithm 5.2: Shape-based alignment.

Input : A product graph $H(V, E)$.
Result: A match $F(V \subseteq V(H))$.

```
1  Let v.A be the set of vertices adjacent to v ∈ V(H)
2  Let c.V be a set of v ∈ V(H), belonging to a compatibility cluster c
3  Let c.N be the set of v ∈ V(H) that were last added to c
4  C := {}                                   // set of compatibility clusters
5  forall v ∈ V(H) do C := C ∪ {c(v)}        // put v into c.V and c.N
6  while ∃c ∈ C|c.N ≠ {} do
7      forall c ∈ C do
8          N := c.N; c.N := {}
9          forall n ∈ N do
10             forall a ∈ n.A do
11                 if COMPATIBLE(c, a) then
12                     c.V := c.V ∪ a; c.N := c.N ∪ {a}
13 max_fragment := max(|c ∈ C|)/3
14 C := C \ {c ∈ C|(|c| < max_fragment)}
15 SORT(C)                                  // with respect to opt.-align-RMSD
16 MERGE_COMPATIBLE(C)
17 max_fragment := max(|c ∈ C|)
18 C := C \ {c ∈ C|(|c| < max_fragment)}
19 SORT(C)
20 F ← C[1]
```

The function COMPATIBLE(c, a) in line 11 performs checks whether the vertex a is a compatible extension for cluster c. An obvious check is found in line 8 of Alg. 5.3. If the cluster already contains one of the amino acids from a, then a is not compatible.

The checks in lines 4 to 7 of Alg. 5.3 need some preliminary explanation. In Figure 5.7, three vectors x, y, z were defined to fully describe the spatial orientation of an amino acid. The first corresponds to the bond between $C\alpha$-carbon ($'CA'$) and carbonyl-carbon, and the second to the bond between oxygen and carbonyl-carbon. They are calculated as origin vectors $x =' CA'.coord -' C'.coord$ and $y =' O'.coord -' C'.coord$. The third is the vector product of x, y, hence it is perpendicular to x and y. The length of all three

Algorithm 5.3: The function COMPATIBLE from Alg. 5.2.

Input : A compatibility cluster c and a vertex $a = (u_A, u_B)$.
Return: $True$, if c and a are compatible, else $False$.

1 Let $c[1].X_A, c[1].Y_A, c[1].Z_A, c[1].Q_A$ be reference points from $v_A(c[1])$
2 Let $c[1].X_B, c[1].Y_B, c[1].Z_B, c[1].Q_B$ be reference points from $v_B(c[1])$
3 Let ϵ be a user defined tolerance
4 **if** $abs(d(c[1].X_A, a.X_A) - d(c[1].X_B, a.X_B)) > \epsilon$ **then** return $False$
5 **else if** $abs(d(c[1].Y_A, a.Y_A) - d(c[1].Y_B, a.Y_B)) > \epsilon$ **then** return $False$
6 **else if** $abs(d(c[1].Z_A, a.Z_A) - d(c[1].Z_B, a.Z_B)) > \epsilon$ **then** return $False$
7 **else if** $abs(d(c[1].Q_A, a.Q_A) - d(c[1].Q_B, a.Q_B)) > \epsilon$ **then** return $False$
8 **if** $(u_A(a) \in c) \vee (u_B(a) \in c)$ **then** return $False$
9 **else** return $True$

vectors is scaled to 10 Å subsequently, and four reference points X, Y, Z, Q are calculated as $X =' C'.coord + x$, $Y =' C'.coord + y$, $Z =' C'.coord + z$, and $Q =' C'.coord$ respectively.

As proposed in the beginning, distances between the amino acids could be used to determine compatibility in the product graph. However, this would mean to compare the distances between each $v \in c$ and the possible extension a (two distances per comparison), hence the time complexity would scale linearly with the cluster size. In contrast, using the defined reference points enables a check in constant time. The initial element $v = c[1] = (v_A, v_B)$ of a cluster c defines its reference vectors $x_{A/B}, y_{A/B}, z_{A/B}$ and reference points $X_{A/B}, Y_{A/B}, Z_{A/B}, Q_{A/B}$. A new vertex $a = (u_A, u_B)$ is compatible, only if u_A has the same coordinates and orientation in the vector space spanned by x_A, y_A, z_A as u_B in the vector space spanned by x_B, y_B, z_B. Hence, not only the relative distances, but also the relative orientations must be equal. The subtle point is, if u_A and u_B have identical relative positions in the reference systems, they do have also identical relative positions in the reference systems of all other $v \in c$ (not the case when only using distances).

In the implementation, a comparison of the distances to the reference points is used (lines 4 to 7 of Alg. 5.3). The tolerance ϵ correlates with the maximum displacement an amino acid can have after superimposition with the final

Algorithm 5.4: The function MERGE_COMPATIBLE from Alg. 5.2.

Input : Set of clusters C.
Result: Merged clusters, corresponding to approximated maximum common subgraphs in $G(V_A, E_A)$ and $G(V_B, E_B)$.

```
1 forall i := 1 to |C| do
2     forall j := i + 1 to |C| do
3         if C_COMPATIBLE(C[i], C[j]) then C[i] := C[i] ∪ C[j]
```

alignment. The default value is 1.4 Å, which can be changed by the user (option "`--r`"). Note, that the tolerance must be lower than 10 Å, because the reference points are calculated with reference vectors scaled to 10 Å. With a higher tolerance, the reference points would no longer discriminate the orientation with respect to the cluster (but only the distance).

Back to Alg. 5.2, the main loop (line 6) terminates, if no further extension of a fragment is possible. Subsequently, all clusters (matched fragments) that consist of less amino acids than one-third of the largest cluster are discarded (lines 13,14). The remaining clusters are sorted with respect to the RMSD value after optimal superimposition of the corresponding fragments. Now, compatible clusters are merged (line 16) as explained in Alg. 5.4. The C_COMPATIBLE function in line 3 shall not be given in detail, as it is analogous to the COMPATIBLE function. Here, the usage of reference coordinates is even more important, because of constant time checks with respect to the cluster sizes. Otherwise, the time would scale with $|C[i]| \cdot |C[j]|$, if using distances.

Finally, all but the largest clusters are discarded and the maximum clusters are sorted again with respect to the RMSD after optimal superimposition. The best cluster in this respect is returned as functional alignment.

Time Complexity The worst case corresponds to an alignment of two identical amino acid chains A and B, where all amino acids $a \in A$ and $b \in B$ have exactly the same conformation and are also connected with identical geometry. The product graph would have a size $|V_P| = |V_A| \cdot |V_B|$. There would be

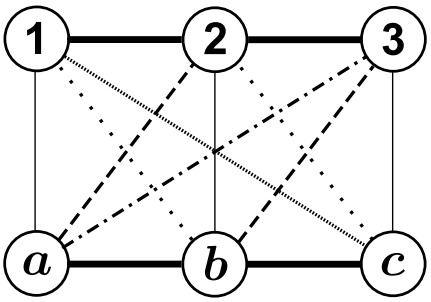


Figure 5.8.: Two amino acid chains symbolized as $A = 1, 2, 3$ and $B = a, b, c$. Different clusters are interconnected by different line types. If the chains are identical and all amino acids have identical conformations, Alg. 5.2 would find the cluster $(1a, 2b, 3c)$ three times (starting with $(1a)$, with $(2b)$ and with $(3c)$ respectively). The clusters $(2a, 3b)$ and $(1b, 2c)$ would be found twice and the clusters $(1c)$ and $(3a)$ once.

$|V_A| = |V_B|$ clusters of maximum size $|V_A|$, $2 \cdot (|V_A| - 1)$ clusters of size $|V_A| - 1$, $2 \cdot (|V_A| - 2)$ clusters of size $|V_A| - 2$, $2 \cdot (|V_A| - 3)$ clusters of size $|V_A| - 3$, and so on. Figure 5.8 gives an example for possible matches (clusters) for the case $|V_A| = |V_B| = 3$. The number of checks in the innermost loop of Alg. 5.2 is proportional to the number of extensions, and therefore proportional to the sum of cluster sizes $S = |V_A|^2 + 2 \sum_{i=1}^{i=|V_A|} (|V_A| - i)^2$, which leads to an $O(|V_A|^3)$ scaling. The orientation check in Alg. 5.3 is in constant time, but the check in line 8 could only be performed in constant time, if all cluster elements were put into a hash table. This is not the case in the *fconv* implementation, due to the memory requirements. Therefore, each element of the cluster must be checked which leads to another linear term. The worst case time bound is thus $O(|V_A|^4)$.

However, while the algorithm can be subjected to identical chains, it will never happen that all amino acids in these chains have identical conformations and are connected with identical geometry.

To give an impression of real-case runtimes, an all-against-all pairwise alignment of the 179 PDB entries introduced in section 4.11 was calculated $(179 + (179^2 - 179)/2 = 16110$ pairwise alignments). The runtime, measured on an Intel(R) Core(TM) i7 CPU Q 720 1.60GHz, was 1596 seconds, hence

a single alignment was performed in about 100 milliseconds on average. The by far most time consuming alignment in the data set was the alignment of bacterial glutamine synthetase (*'1LGR'*) on itself. This protein consists of twelve identical subunits and has a total of 5616 amino acids (41 496 atoms), and the alignment took 89 seconds.

Prospect As mentioned, the shape-based alignment was developed for superimposition of identical chain fragments or for cases where all or some amino acid types are unknown (more precisely not known from the input file). However, the method could also be extended in the direction of a protein classification tool. Currently, a lot of the retrieved information is discarded. A good example is the already shown example in Figure 5.6, where the shape-based alignment leads to a nearly perfect superimposition of one of the two domains. There also exists a very good match of about 50 amino acids from the other domain, but the relative orientations of these two matches are different and the algorithm returns only the best of them. If one is interested in the similarity of two proteins, the information of all fragments (with high size) would be relevant.

Nevertheless, even the current implementation can give a hint on how useful these matches are to estimate similarity. Given the size $|F|$ of the returned best match (alignment), a simple similarity measure is $(1 + RMSD(F))/|F|$. Based on this value, *fconv* can cluster PDB files using the command "`fconv -clust *.pdb`". The result are two SVG files (Scalable Vector Graphics), one with the dendrogram from the hierarchical complete linkage clustering, and the other with the corresponding similarity matrix.

Figure 5.9 shows the similarity matrix, calculated for the 179 PDB entries introduced in section 4.11. The obvious clusters are trivial cases, because only structures with very high sequence identity are found within them. Some additional conclusions like the similarity between the proteases in clusters 2 and 3, or the lower similarity between the proteases in cluster 3 and all other proteins are possible. At least, it seems to be possible to extend the functionality towards a protein classification tool, by using the information from all matches of Alg. 5.2 and an appropriately calculated similarity value. The computational

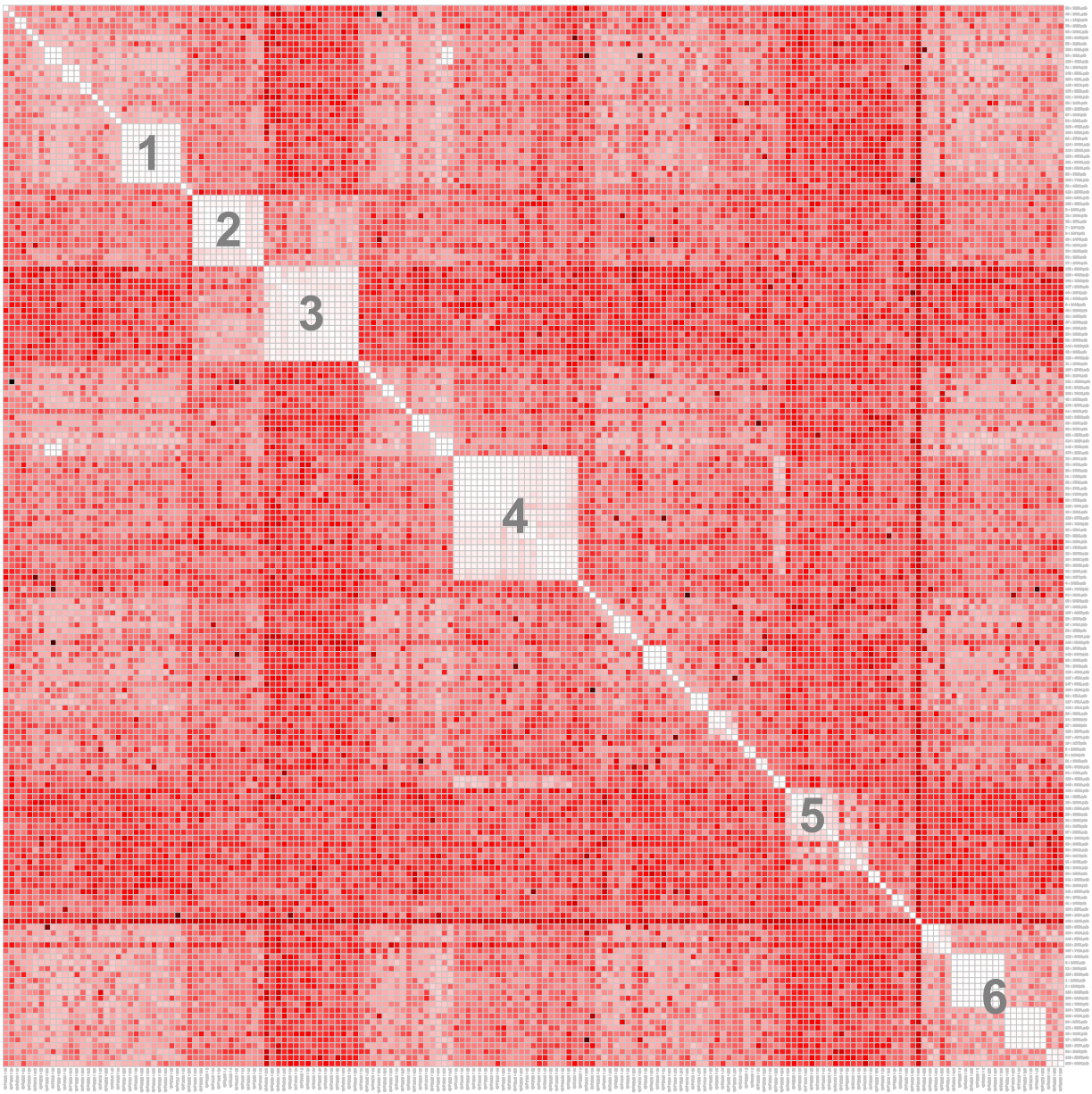


Figure 5.9.: Protein similarity matrix, generated by *fconv*. Colors range from white (highest similarity), over light red, red, dark red to black (lowest similarity). Cluster 1: thermolysin structures. Cluster 2: pepsin, renin, and cathepsin D structures. Cluster 3: aspartic proteases. Cluster 4: serine proteases. Cluster 5: immunoglobulines. Cluster 6: consists of 4 smaller clusters of triosephosphate-isomerase, L-arabinose binding protein, carboxypeptidase A, and ribonuclease structures.

speed of the described method is sufficient for the processing of large protein data bases.

5.2. Triangulation-Based Binding Site Detection

The unsupervised identification of putative ligand binding sites in a protein is an important step in large scale data processing like cross docking or pocket-based protein classification. Unfortunately, the term "binding site" is not well defined. Given a particular structure subjected to ten different scientists for binding-site assignment, it is likely to receive ten different classifications of which atoms belong to a binding site and which do not. Thus, even in case of a smaller number of targets, it can be beneficial to use an automatized approach that guarantees consistent results at least for identical input.

Almost all binding sites are also cavities, but by far not all cavities are also binding sites. Many cavities are either too small, or too flat and solvent-exposed for drug-like ligand binding. Hence, there is a correlation between a cavity's size and burial, and it's likelihood to be a binding site. Therefore, geometric approaches focus on the detection of large cavities with a certain degree of burial. Examples are the programs $POCKET$ (Levitt and Banaszak, 1992), $LIGSITE$ (Hendlich et al., 1997b), $LIGSITE^{CS}$ (Huang and Schroeder, 2006), $SURFNET$ (Laskowski, 1995), $CAST$ (Liang et al., 1998), and $FPOCKET$ (Le Guilloux et al., 2009). Additional information related to putative binding sites can improve the results. For example, the program $ConCavity$ (Capra et al., 2009) combines the geometric approach with a measure of evolutionary sequence conservation, and the program $SiteMap$ (Halgren, 2007) evaluates cavities with respect to their physicochemical properties.

The geometric approaches for cavity detection can be grouped into orientation-dependent and orientation-independent methods. $POCKET$, $LIGSITE$, and $LIGSITE^{CS}$ are examples for programs that yield different results depending on the initial orientation. They are based on algorithms that work on a rectangular grid, hence they can produce different results for identical, but

rotated proteins.

$SURFNET$, $CAST$, and $FPOCKET$ are examples for programs that are rotationally invariant. Similar to the two latter programs ($CAST$ and $FPOCKET$), *fconv* offers a cavity detection based on a Delaunay triangulation (more precisely a regular triangulation) of the protein atoms. Apart from being rotationally invariant, this triangulation offers some additional, useful features that will be explained later on.

5.2.1. Regular Triangulations

The Delaunay triangulation (DT) is an angle maximizing triangulation of a set of points P in $\mathbb{R}^d$. In three dimensions, the DT partitions the space into a set of tetrahedrons, where the vertices of tetrahedrons correspond to the triangulated points. For each tetrahedron with vertices p_1, p_2, p_3, p_4, the following regularity condition is fulfilled:

- The circumsphere $S(p_1, p_2, p_3, p_4)$ of T includes no other $p \in P$

This predicate already implies a first hint on how the DT of protein atoms can be used for cavity detection. The four points of the tetrahedron are located on the surface of the circumsphere, hence the volume of the sphere is void and therefore cavities correspond to regions with large circumspheres.

Of course, protein atoms are not points, but spheres with different radii. To consider these radii, a weighted DT can be used, which is then called regular triangulation RT (hence, the DT is just a special case of RTs). Usually, the atom's squared vdW-radii are used as weights. The reason for choosing the squared radii is best illustrated in terms of a Voronoi diagram, which is the dual graph to RTs. It decomposes the space into Voronoi cells and each vertex of a RT-tetrahedron is the center of a Voronoi cell. All coordinates $c \in \mathbb{R}^3$ within the borders of the cell are closer to the center than to any other $p \in P$. This is the dual formulation to the circumsphere condition for RTs. Thus, all coordinates on a border between two Voronoi cells must have the same distance to both centers.

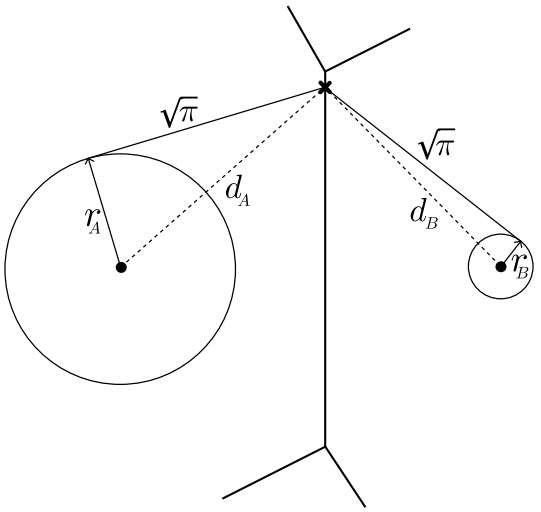


Figure 5.10.: The border between two Voronoi cells. The centers on both sides have different weights $w = r^2$. The power distance $\pi = d^2 - w$ between border and middle points is always equal. If Euclidean distances $dist = d - r$ would be used, the borders would be no lines (planes), but curves.

Figure 5.10 shows two Voronoi cells with centers of different weights. Using Euclidean distances $dist = d - r$, the border between both cells would be no line (plane in 3D). Instead the so-called power distance $\pi = d^2 - w$ is used which leads to planes that separate the Voronoi cells. Due to this definition, algorithms to calculate a DT can be easily adapted to calculate an RT. In the Figure, it can be seen that interpreting the weights as squares of circular (sphere) radii, the power distance is the squared tangent from a border point to the circles (spheres). It can also be seen that the DT is just an RT with zero weights for all points. It shall be noted that the circumsphere of four weighted points not necessarily has all points on its surface, and that its radius can be a complex number with a real part of zero (e.g. in a region of covalently overlapping atoms).

The algorithm that was implemented into *fconv* for regular triangulations was taken from Edelsbrunner and Shah (1996). It adds points incrementally to the triangulation and recovers regularity after each step by so-called flipping. The algorithm triangulates n points in $O(n \cdot log(n))$ time.

5.2.2. The Concept of Alpha Shapes

The usefulness of regular triangulations for cavity detection was already indicated and will now be specified. All programs that calculate cavities based on an *RT* or *DT* use different approaches to translate the triangulation information into binding sites. The implementation in *fconv* is no exception, but works very similar to the method described by Edelsbrunner et al. (1998) that is used in *CAST*. It is based on the concept of so-called alpha shapes that will be described in the following.

First, a definition for α-spheres is given:

- An *RT* consists of a set of tetrahedra T.
- Each $t \in T$ consists of four weighted points p_1, p_2, p_3, p_4 and four triangles (facets) $f_1(p_1, p_2, p_3), f_2(p_1, p_4, p_2), f_3(p_1, p_4, p_3), f_4(p_2, p_4, p_3)$.
- A sphere that can be attached to a triangle f, such that all three points are on the surface (in the unweighted case) and no other point is contained in the sphere, is called an α-sphere with respect to f and the radius of the sphere is called α-level.

One can check all triangles of a triangulation, with respect to a sphere with a particular α-level (radius). All triangles for which the sphere is an α-sphere correspond to the so-called α-complex. A 2D-example is given in Figure 5.11, where each line that fits into an α-circle is part of the α-complex, if no other point is located in the circle.

In three dimensions an α-complex consists of complete tetrahedra, individual triangles, lines, and points (all of them being simplices of the triangulation). The set of simplices that represent the border between α-complex and simplices that are not part of the α-complex is called α-shape. Note that the α-shape of an infinitely high α-level corresponds to the convex hull, and an α-level of zero corresponds to the set of points without any other simplices.

Figure 5.12 illustrates the usefulness of α-shapes for cavity detection in a 2D-example. If an appropriate lower and upper α-level is chosen, the difference between the corresponding α-shapes represents cavities. If the lower α-level is chosen too small, the complete space between all points could be detected as

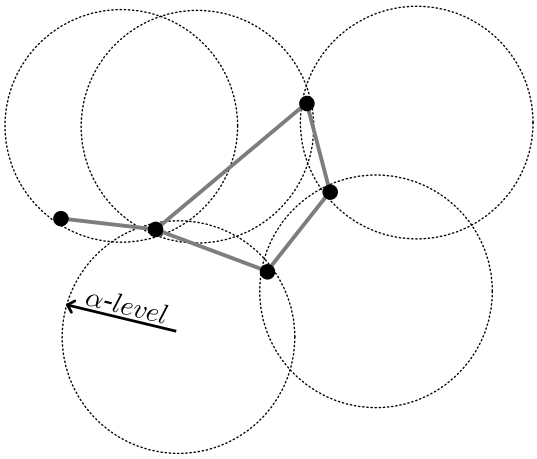
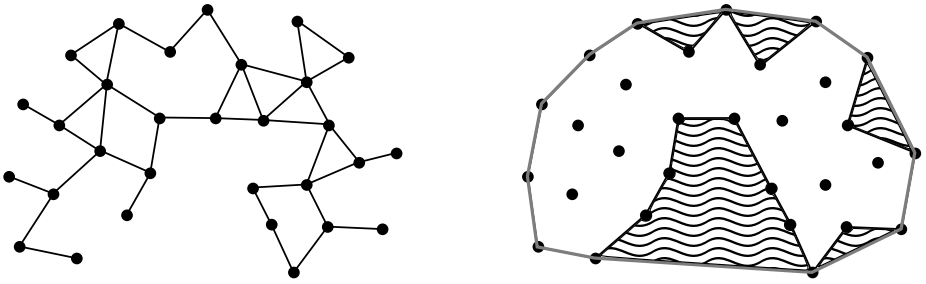


Figure 5.11.: An α-complex of five points depicted in gray. A line between two points is part of the α-complex, if a circle with the α-radius can be attached in such a way that no other point lies inside the circle.



(a) small α-level **(b)** structure

Figure 5.12.: (a) α-shape of a 2D point set at a rather small α-level. (b) α-shape at lower α-level (black line) and upper α-level (gray line). The difference between them corresponds to cavities.

a cavity. If the upper α-level is too high, also rather shallow cavities (on the protein surface) will be detected.

A 3D-example calculated with *fconv* for the PDB entry *'1hpv'* is given in Figure 5.13. The lower and upper α-level in (a) and (b) correspond to the default values used in *fconv*, which can be changed by the user.

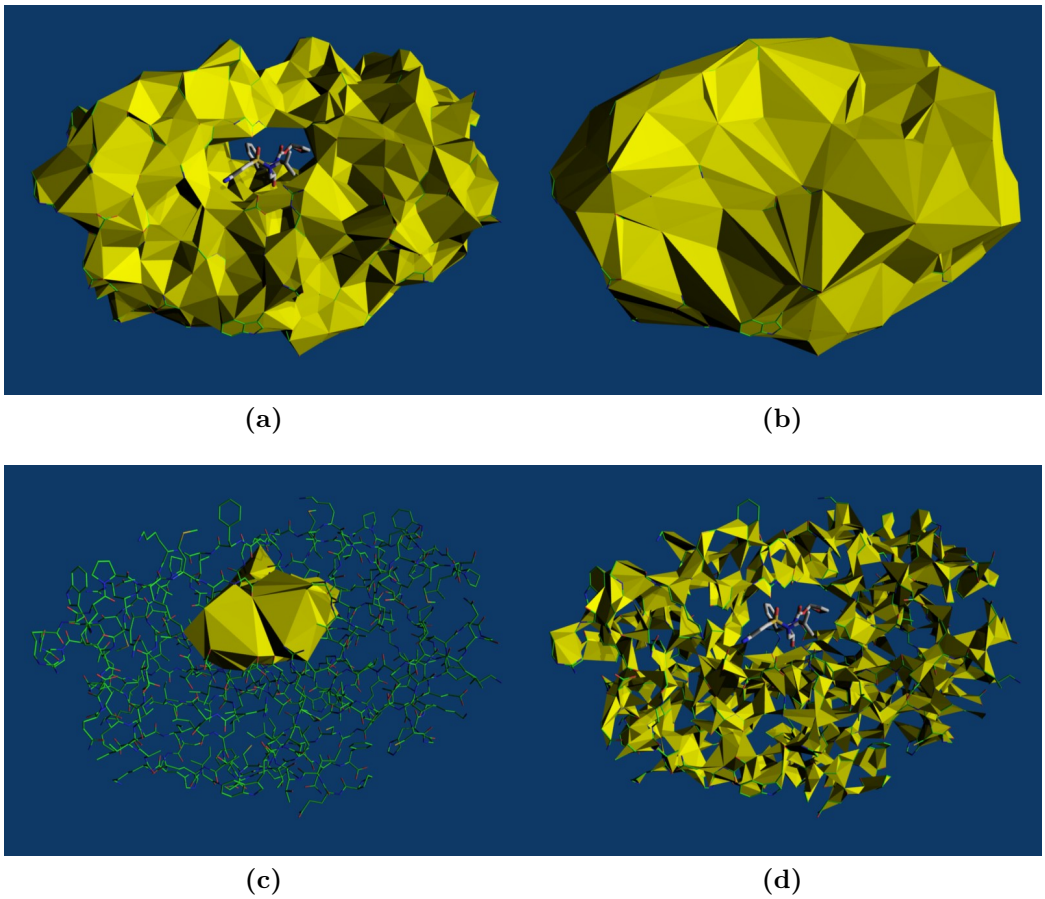


Figure 5.13.: The PDB entry *'1hpv'* (a) α-shape at lower α-level of $\alpha = 2.6\,\text{Å}$. (b) α-shape at upper α-level of $\alpha = 8.0\,\text{Å}$. (c) The difference complex between lower and upper α-level, corresponding to a cavity. (d) An example for a lower α-level that was chosen too small (1.2 Å).

For geometric objects that are represented by arbitrary point clouds, it is not trivial to determine appropriate α-levels. However, in case of molecules, the lower level is equally valid for all input. Changing its default to a lower value would result in a finer shape with more details, but it should never be chosen below 2.2 Å. If a higher value for the upper level is chosen, larger pockets and also rather shallow cavities will be detected.

5.2.3. Determining the Cavities

The difference between two α-shapes is a subset T_C of all tetrahedrons. Fortunately, it is very easy to determine this set for arbitrary α-levels. If the circumsphere $S(t)$ of a tetrahedron t is larger than the α-sphere $S\alpha$, the tetrahedron is part of the α-complex. Thus, the set of cavity tetrahedrons is defined as $T_C = \{t \in T | S(t) > S\alpha_{lower} \text{and} S(t) \leq S\alpha_{upper}\}$.

Certainly, the set T_C does not correspond to a single cavity in most of the cases. Tetrahedra of two pronounced pockets can be connect by some tetrahedra that are close to the upper α-level. Therefore, *fconv* first clusters the tetrahedra, subsequently discards all clusters below a given threshold volume, and finally reports all remaining clusters as cavities ordered by decreasing volume.

Volumes can be calculated analytically, which is another feature of the regular triangulation. In case of unweighted points, the volume of a subset of tetrahedra would just be the sum of tetrahedra volumes. In case of protein atoms, their vdW-volume must be subtracted from this sum. In detail, this is done following the inclusion-exclusion principle as proposed by Edelsbrunner et al. (1995). On a side note, the calculation of molecules' vdW-volumes with "`fconv -vol`" is also based on *RT* and the inclusion-exclusion principle.

5.2.4. Application

The command "`fconv -pa2 *.pdb`" reports all detected cavities ordered by their volume, and writes them out as individual PDB files. Using the debug flag "`---d`", *fconv* will generate VRML (Virtual Reality Modeling Language) files for visualization of lower and upper α-shapes, as well as the difference complex (see Figure 5.13a to c). Using the flag "`-pa3`", complete residues for all cavity atoms will be written.

5.2.5. Further Development

The described method is well suited to detect pronounced cavities, but needs further development to yield reliable results for arbitrarily shaped binding sites.

A logic and promising extension would be a combination with *DSX* hotspots (see section 11.1), to restrict putative binding sites to cavities that coincide with attractive *DSX* interaction fields.

Chapter 6.
6 Summary and Outlook

6.1. Summary

In this part of the thesis, the development of a programming framework for the processing of structural data, as well as the corresponding command line front-end *fconv* was explained. The framework is also an integral part of the scoring function *DSX* that is described in the second part of this thesis, and *fconv* has become an important tool within our work group.

Major achievements concerning the framework/*fconv* are:

- Parsers for the robust handling of the file formats PDB(QT), MOL2, SDF, CIF, and DLG were implemented. Special care was taken to use only essential data contained in input files, hence three-dimensional coordinates and information that allows for the determination of element types. Many errors, as they can especially occur in PDB files, are tolerated and corrected.

- A class hierarchy for a consistent and exhaustive representation of structural data was developed. For functions that rely on a logic evaluation of such data, the input is always mapped to this hierarchy first. This is a prerequisite for the segregation of such functions.

- The core competence of the framework is the automatic assignment of

atom and bond types. In a comparison with the best known approaches so far, it was shown that *fconv* is competitive, having the lowest failure rate on a literature known test set. With respect to the computational speed, *fconv* demonstrated superior performance compared to the widely used program OpenBabel. An internal set of atom types was developed based on a broad extension of the widespread used Sybyl atom types. For generated output files, these internal types are mapped onto the Sybyl types by default. Furthermore, an easy method was included that allows for arbitrary changes of this mapping (e.g. used in Part II). In addition, it was also enabled to easily change default protonation states, maximum ring sizes and other parameters.

- In context with the atom type assignment, the problem of ring perception was discussed briefly. In contrast to the majority of other cheminformatics tools, *fconv* is not determining an ambiguously defined SSSR, but the unique set of all relevant rings. A simple algorithm to obtain this set of rings was introduced. Though not being appropriate for the application of arbitrary data that can be represented as a graph, it has polynomial time complexity within the restrictions of biological molecules.

- Another simple algorithm was developed for the challenging problem of maximum weighted matching in non-bipartite graphs that is used for an optimal assignment of double bonds. In case of arbitrary graphs, the proposed method is not competitive compared to the best known algorithms for this problem. However, the latter are rather complicated to implement and it was shown that the newly developed algorithm is sufficient, if applied to real case molecules.

- From the many features that are available within *fconv*, the automatic cavity detection, RMSD calculation and structural superimposition, as well as substructure searches were introduced. These often occurring tasks are solved using robust implementations of well known algorithms, as well as self-developed algorithms. Other freely available programs for file

conversion (e.g. OpenBabel) have only limited additional features. There is of course also free software for each particular feature, but then these applications rely on atom types and connectivities as supplied in the input or they use own, and possibly different definitions. In contrast, computing different features based on an identical internal representation can be considered an advantage with respect to robustness and consistency.

- Since its publication, *fconv* is maintained as an open source program under GPL license and is available on `http://www.agklebe.de/drugscore/fconv_download.php`. *fconv* is not only used by scientists in the pharmaceutical industry and at other universities, but also the recently in our group developed program MiniMuDS depends on some of its features.

6.2. Outlook

With respect to the tasks the framework was aimed at in the beginning, there are currently no open demands, because all criteria of robustness, quality, computational speed, and possible differentiation of atom types are fulfilled. Nevertheless, many additional features are accumulated meanwhile and not all of them can be considered stable, as exhaustive test cases were not developed. Within the same context, the command line interface of *fconv* is crowded and not consistent in all aspects. Here a complete redesign is highly desired.

While *fconv* is highly useful to other people, the application of the framework in software development is currently limited. This is due to historical reasons. At the beginning of the PhD time, the author was new to programming and thus, design is not conform with what is considered as good programming practice. Time was however spent to scientific questions, instead of redesigning the libraries. Such a redesign should strictly fulfill the interface segregation and single responsibility principles. This would not only enhance the usefulness of the framework, but at the same time simplify the implementation of new features drastically.

With respect to the functionality of *fconv*, it might be promising to implement

a generic pattern recognition for arbitrary atom type definitions. Of course, this would demand an initial decision for a language to describe such patterns that must fulfill two criteria: First, being simple to adapt by unexperienced users and second, being distinct.

Other features that will be included in the future are supply for additional file formats. Especially the conversion between different formats that represent three-dimensional fields, like e.g. electron density maps, would be useful.

Part II.

The Program DSX and Derived Applications

Chapter 7.
7 Introduction and Motivation

The following chapters contain material that was already published by the author (Neudert and Klebe, 2011a).[1]

7.1. Virtual Screening

Supported by an increasing number of crystallographically derived structures that represent relevant drug targets, structure-based virtual screening has become an ever more important method in modern drug research. To find new binders for a given target, a typical virtual screen work flow comprises:

1. Molecular docking to the active site of a target to generate reasonable putative binding poses of small molecule structures from an appropriate *in silico* compound library.

2. Determine the best docking pose for each compound.

3. Rank the different compounds with respect to their expected affinity.

Today, a variety of different docking programs exists using different approaches to solve the ligand placement problem (Taylor et al., 2002). Popular and widely used examples are the programs AutoDock (Goodsell and Olson,

[1] DOI: 10.1021/ci200274q

1990; Morris et al., 1996, 1998), DOCK (Ewing et al., 2001), eHiTS (Zsoldos et al., 2007), FlexX (Rarey et al., 1996), Glide (Friesner et al., 2004, 2006), GOLD (Jones et al., 1995, 1997) and Surflex (Jain, 2003). It has been shown that these programs are able to generate near-native geometries (Dixon, 1997; Paul and Rognan, 2002; Kellenberger et al., 2004; Warren et al., 2006), that are docking poses with a considerably low Root-mean-square-deviation (RMSD) with respect to the experimentally determined binding pose. Thus, it has often been stated that the docking task is solved (Lipinski, 2003), but it is necessary to discuss this statement in more detail.

Roughly speaking, molecular docking consists of two different parts: (i) a global optimization strategy and (ii) a scoring function that is optimized.

As shown later on, no "perfect" scoring function exists up to now and therefore, also the docking problem cannot be solved completely. What is implied by the above-mentioned statement is only the global optimization (which can be split into the tasks: initial placement, global search strategy and local search strategy). However, except for a systematic sampling of the complete search space, all global optimization strategies are a trade-off between computation time and quality of the results.

Most docking programs consider only torsion angles, translation and rotation of the ligand as degrees of freedom. For a typical drug-like molecule with up to 20 freely rotatable bonds this results in six up to 26 degrees of freedom. When considering the geometries produced by docking programs it becomes obvious that the results are much more satisfying for small compounds with only a few rotatable bonds and get worse for large compounds with many such rotatable bonds (Plewczynski et al., 2011). If the complete protein, with a few hundred up to thousands rotatable bonds, would be considered flexible, the combinatorial explosion would make it impossible to find reasonable solutions in an acceptable range of time for all of the above-mentioned programs.

For AutoDock, eHiTS, FlexX, Glide, GOLD and Surflex, Plewczynski et al. (2011) demonstrated that the docking results strongly depend on the initial compound conformation given to the docking engine. Therefore, even in case of rigid proteins one can conclude that the global optimization problem is not

Table 7.1.: RMSD to native pose averaged over 1300 complexes. Results taken from Plewczynski et al. (2011)

	averaged RMSD in Å		
Docking program	*best pose*	*top pose*	difference in Å
GOLD	1.6	2.7	1.1
eHiTS	1.7	2.8	1.1
Surflex	1.9	3.3	1.4
Glide	2.2	3.7	1.5
LigandFit	2.1	4.1	2.0
AutoDock	3.2	4.4	1.2
FlexX	3.2	4.4	1.2

yet solved satisfyingly.

For the second step in the work flow mentioned on page 109 (determining the best docking pose), either the ranking with respect to the target function that was applied in the docking process can be used or a subsequent re-ranking with respect to an alternative scoring function. In the following, the docking pose that has the smallest RMSD with respect to the native pose will be called *best pose*. The docking pose that obtains the best score and is therefore ranked on top will be called *top pose*.

In the already mentioned study by Plewczynski et al. (2011) all docking programs under consideration revealed a significant gap between *best pose* and *top pose*. Table 7.1 shows results calculated from Figure 3 and Figure 4 in Plewczynski et al. (2011). These results are RMSD values for *best pose* and *top pose* averaged over 1300 complexes from the PDBbind database (Wang et al., 2004, 2005) and averaged over four different docking protocols (using one or ten input conformations of the ligand generated by either CORINA (Gasteiger et al., 1990) or OMEGA (Boström et al., 2003)). The significant differences between these averaged values represent the high potential for improvement in re-ranking of docking poses. When developing the program *DSX* it was aimed primarily for this task, hence as a function for the re-scoring of different protein-ligand geometries for a given type of ligand.

The last step in the work flow mentioned on page 109 (ranking of the different compounds) is the most challenging task. While the second step only demands the comparison of different geometries for the same compound, the third step requires a good estimation of binding energies for the different compounds.

7.2. The Use of Scoring Functions

Considering the level of simplification of the underlying biophysics, current scoring functions are far-off from being perfect in binding energy prediction. Protein flexibility, desolvation effects and especially configurational entropy are not sufficiently accounted for. Elaborate methods such as Linear interaction energy (LIE) (Aqvist et al., 1994), MM-PBSA/GBSA (Massova and Kollman, 2000; Hou et al., 2011) and thermodynamic integration (Jorgensen, 1989) can yield good results, but are not applicable in a high throughput virtual screening campaign with thousands of candidate compounds, because sufficient conformational sampling via Molecular Dynamics (MD) or Monte Carlo (MC) simulations would be computationally too demanding (Gilson and Zhou, 2007). Therefore, a pre-filtering must be applied using fast scoring functions.

Cheng et al. (2009) distinguish between three different tasks a scoring function should accomplish:

- If a distinct native binding mode for a compound exists, the function should identify the pose closest to the native conformation among a huge number of generated poses for this compound.

- If a set of different ligands binding to the same protein is given, a reliable scoring function must be able to rank the ligands according to their binding affinities.

- If a series of arbitrary protein-ligand complexes is given, the linear correlation between predicted scores and binding affinities should be as high as possible.

As proposed by Cheng et al. (2009), this work refers to the first criterion as *"docking power"*, to the second as *"ranking power"* and to the third as *"scoring power"*. The definition of *docking power* corresponds to the second step in the initially mentioned work flow on page 109 and is therefore somehow irritating. The name suggests the measurement of quality for a scoring function used as target function within the process of molecular docking. But due to the definition this is clearly not the case, because a function used for re-ranking of docking poses can be completely inadequate as target function in docking (e.g. it might have no term for steric clashes). In the following also the term *"pose recognition"* will be used synonymously with *docking power*.

The definition of *ranking power* and *scoring power* both correspond to the third step in the work flow mentioned on page 109. A perfect linear correlation with binding affinities also implies perfect ranking, but a perfect ranking does not necessarily imply high *scoring power*. Most likely, a near-native pose is a prerequisite to yield correct ranking.

In consequence, high *ranking power* is rather useless without high *docking power*. Therefore, a scoring function should be either adequate for both docking and ranking or better, the task should be split into a combination of two (or more) functions, each tailored for one goal. In the following the terms *"affinity correlation"* and *"affinity prediction"* will be used synonymously with *scoring power*.

Although the definition by Cheng et al. (2009) is not satisfying, it will be used throughout this part of the thesis, because *DSX* is validated on the same data set used in the mentioned paper and a comparison of the results is less confusing using the same terminology.

7.3. Classification of Scoring Functions

Depending on the methodological background, scoring functions are often classified into three categories:

- Force-field-based scoring functions (Ewing et al., 2001; Jones et al., 1995,

1997) use classical molecular mechanical force fields to evaluate binding energy.

- Empirical scoring functions (Tang and Marshall, 2011; Morris et al., 1998; Friesner et al., 2004; Wang et al., 2002; Eldridge et al., 1997; Krammer et al., 2005; Gehlhaar et al., 1995; Jain, 1996; Böhm, 1994, 1998; Sotriffer et al., 2008) decompose the total energy into several linear energy terms. The weighting of the individual terms is done by regression analysis using a training set with experimental binding affinities.

- Knowledge-based scoring functions (Huang and Zou, 2006a,b, 2010; Zhang et al., 2005; Gohlke et al., 2000; Velec et al., 2005; Pfeffer and Gohlke, 2007; Mooij and Verdonk, 2005; Xue et al., 2010; Shen et al., 2011; Muegge and Martin, 1999; Muegge, 2006) calculate the total score as sum of statistical potentials, which are derived from a database of known protein-ligand complexes.

This classification is rather crude and some scoring functions are difficult to assign to one of the three categories. For example, MotifScore (Xie and Hwang, 2010) does not apply the usual decomposition into individual atom-atom terms, but instead scores complete three-dimensional motifs.

Because they are trained at affinities, the key skills of empirical scoring functions should be *ranking power* and *scoring power*. As a shortcoming, their predictive power strongly depends on the similarity between important interactions in the complex under evaluation and important interactions in the training set complexes. Furthermore, they suffer from both, uncertainties in the structural data and experimental errors for the affinity data of the training set.

In contrast, knowledge-based functions do not rely on affinity data, but exploit comprehensive crystallographic information. Thus, they are more general and their key skill should be *docking power*, because the statistical

potentials reflect native binding geometries. When only distance-dependent atom-atom potentials are used, they are also faster to compute than empirical functions. This is important, as *docking power* needs many more function evaluations compared to *ranking power*.

In the following chapters the development, implementation and validation of the new knowledge-based scoring function *DSX* (DrugScore eXtended) is presented. *DSX* pair potentials are based on the DrugScore formalism (Gohlke et al., 2000; Velec et al., 2005) that was developed earlier in the same work group. This approach is extended with respect to a more detailed atom type assignment and a modification to overcome a problem with the reference state. Furthermore, statistically derived torsion-angle potentials are included, which allow for fast relaxation of docking poses and can improve *docking power* and *ranking power*. In addition, a new type of solvent-accessible-surface-dependent potentials is introduced. The validation of *DSX* is presented based on the carefully prepared and publicly available dataset of Cheng et al. (2009).

The next section supplies the theoretical background of knowledge-based scoring functions for subsequent discussions about reference states, volume corrections and the newly defined statistical potentials. It also clarifies inconsistencies in terminology and foundation of the formalisms, that are found in the literature. Furthermore, differences between the most popular knowledge-based functions, namely PMF (Muegge and Martin, 1999; Muegge, 2006), ASP (Mooij and Verdonk, 2005) and DrugScore (Gohlke et al., 2000; Velec et al., 2005) will be presented along with the modifications and extensions implemented in *DSX*.

Chapter 8.

8 Theory

8.1. Theoretical Background of Knowledge-Based Scoring Functions

A foundation of most knowledge-based scoring functions is based on statistical thermodynamics, although the used assumptions must be seen critically. The equation that is the base of these physical considerations is the Boltzmann distribution

$$\frac{n(i)}{N} = \rho(i) = \frac{e^{-\frac{E(i)}{kT}}}{Z(T)} \tag{8.1}$$

where $n(i)$ is the number of particles in a set of states i with the energy $E(i)$, N the total number of particles in the system, T the absolute temperature, k the Boltzmann constant and $Z(T)$ is the partition function (or Boltzmann sum over states).

$$Z(T) = \sum_i e^{-\frac{E(i)}{kT}} \tag{8.2}$$

The fraction $\rho(i)$ is a state-dependent probability density function. Equation 8.1 is the distribution function for the canonical ensemble, hence for a system in thermodynamic equilibrium with fixed temperature, volume and number of particles. Rearrangement leads to an equation which is often referred to as

inverse Boltzmann law.

$$E(i) = -kTln(\rho(i)) - kTln(Z(T)) \tag{8.3}$$

For systems where the partition function is known, all thermodynamic properties of the system can be calculated based on it. However, if it is unknown, one can still calculate energy differences compared to a reference state, because $Z(T)$ is constant at constant temperature (and constant number of particles which is generally assumed here).

$$\Delta E = E(i) - E_{ref} = -kTln(\rho(i)) + kTln(\rho_{ref}) = -kTln\left(\frac{\rho(i)}{\rho_{ref}}\right) \tag{8.4}$$

In the theory of liquids (Ben-Naim, 1992), free energies are calculated using radial distribution functions $g(r)$ corresponding to the fraction $\frac{\rho(i)}{\rho_{ref}}$. The Helmholtz free energy $W_{ab}(r)$ of two particles a and b in a homogeneous solvent is

$$\begin{aligned} W_{ab}(r) &= -kTln(g(r)) = U_{ab}(r) + \delta G_{ab} \\ g(r) &= \frac{\rho_{ab}(r)}{\rho_{ref}(r)} = \frac{P_{ab}(r)}{P_{ref}(r)} \end{aligned} \tag{8.5}$$

which is the reversible work spent or gained when transferring a and b from infinite separation to a distance r. In this case, the reference state is the ideal gas, thus $P_{ab}(r)$ corresponds to the probability to find two particles in solvent at a distance r, while $P_{ref}(r)$ corresponds to the probability to find them at the same distance in an ideal gas. Because $W_{ab}(r)$ corresponds to the mean force acting on the two particles due to their averaged interactions with the surrounding δG_{ab} and with each other $U_{ab}(r)$, it is called a *potential of mean force*.

In analogy to Equation 8.5, attempts were made to use potentials of mean force for protein folding prediction (Sippl, 1990, 1993, 1995; Jernigan and Bahar, 1996) and for scoring of protein-ligand complexes (Muegge and Martin,

1999). Here, the contact densities are calculated from the contact data found in protein data bases such as the PDB. However, it has been clearly pointed out that the statistical potentials derived from crystallographic data bases are no potentials of mean force (Ben-Naim, 1997; Koppensteiner and Sippl, 1998). In essence, the radial distribution functions for protein systems are derived for particles taken from different environments. That is, a and b have different interactions with their surrounding in different protein-ligand complexes. Thus, the δG_{ab} is different for each contact ab and averaging these data cannot yield a density function that corresponds to the $g(r)$ used in Equation 8.5. Furthermore, the $U_{ab}(r)$ in Equation 8.5 are additive, but the δG_{ab} are not (Ben-Naim, 1997). In consequence a partition of the total free energy into pairwise atom-atom contributions is not valid. When arguing on the basis of Equation 8.3 and Equation 8.4, the problem simply is that the distribution of atom-atom contact distances does not really follow the Boltzmann distribution and therefore these distances cannot be used to calculate energies based on this statistic. The reason is related to the problem of considering different environments. Two atoms a and b in a protein are not necessarily found at thermodynamic equilibrium distance even though the complete system might be at equilibrium, because the intramolecular structure of both, protein and ligand, prevents proper resemblance to a Boltzmann-like distribution.

Taking all this into account, one should strictly avoid terms like "potential of mean force" or "energy" when talking about statistical potentials. Koppensteiner and Sippl (1998) even proposed to avoid the term "potential", instead "preference" or "quantity" should be used. But as the term "potential" is not necessarily linked to an energy function, it will also be used in this work.

Given that the values computed by statistical potentials are not energies, one can drop the linear factor kT and replace the term "energy" by "score", which leads to the master equation for knowledge-based scoring functions.

$$Score(i) = -ln\left(\frac{\rho(i)}{\rho_{ref}}\right) \tag{8.6}$$

In the case of pairwise distance-dependent contributions, the total score for a given complex of protein atoms a_p and ligand atoms a_l is calculated as

$$TotalScore_{pair} = \sum_{a_p} \sum_{a_l} Score(p(a_p), l(a_l), r(a_p, a_l)) \tag{8.7}$$

$$Score(p, l, r) = -ln\left(\frac{\rho(p, l, r)}{\rho_{ref}}\right) \tag{8.8}$$

where $p(a_p)$ and $l(a_l)$ are the atom types and $r(a_p, a_l)$ is the distance of a_p and a_l. Equation 8.6 is not necessarily restricted to distance dependent atom-atom scores, but can also be applied to many other structural features like bond or dihedral angles. Using Bayesian probability theory, one can obtain similar equations (Hamelryck et al., 2010), but the problem of deriving meaningful probability functions remains and the prerequisite of pairwise independence is not fulfilled. Finally, the statistical potentials should be seen as a class of heuristics that have been proved to be useful.

8.2. Distance-Dependent Pair Potentials

Besides the choice of appropriate atom types and an appropriate data sample, the choice of a proper reference state is crucial for the quality of statistical potentials. In case of PMF (Muegge and Martin, 1999; Muegge, 2006) or ASP (Mooij and Verdonk, 2005), it has been selected as state of no interaction, referring to the (wrong) analogy to potentials of mean force. In contrast, the reference state used in DrugScore is chosen as state of mean interaction. In principle, ρ_{ref} can be seen as a kind of weighting function for $\rho(i)$ to successfully apply Equation 8.6. Another aspect often discussed is the volume correction for atom types i to account for the *de facto* available volume.

PMF, ASP and DrugScore are the most popular knowledge-based scoring functions and have been evaluated on the same validation test set used for *DSX*. With respect to the changes in the new scoring function *DSX*, a comparison to these functions concerning reference state and volume correction is presented. In

contrast to the original publications, the equations are presented in a rearranged form and the symbols were changed to those used in this thesis. This is necessary to make similarities and differences more obvious. All mentioned functions are based on Equation 8.8, but differ in the definition of the density functions.

In PMF, we have

$$\rho^{PMF}(p,l,r) = f(l,r)\frac{N(p,l,r)}{\Delta V(r)} \tag{8.9}$$

$$\rho_{ref}^{PMF} = \rho_{ref}^{PMF}(p,l) = \frac{\sum_{r}^{R} f(l,r)N(p,l,r)}{V(R)} \tag{8.10}$$

where $N(p,l,r)$ is computed from the database as the number of contacts between protein atom type p and ligand atom type l with a distance in the interval $[r, r + bin\_size[$. The contact numbers are normalized by the theoretically available volume $\Delta V(r)$ of the spherical shell corresponding to the interval $[r, r + bin\_size[$. The factor $f(l,r)$ is a correction of the theoretically available volume due the space that is occupied by other ligand atoms (averaged from the database). R is the cut-off radius of 12 Å and $V(R)$ the volume of the corresponding sphere. Strictly speaking, in this case the density functions are not probability functions. However, an applied normalization will not change the value of the fraction. Here, the reference density is clearly dominated by long range contacts and thus an approximation to a state of no specific interaction.

In ASP, we have

$$\rho^{ASP}(p,l,r) = \frac{N(p,l,r)}{\Delta V(r) f(l,r) f(p,r)} \tag{8.11}$$

$$\rho_{ref}^{ASP} = \rho_{ref}^{ASP}(p,l) = \left\langle \frac{N(p,l,r')}{\Delta V(r') f(l,r') f(p,r')} \right\rangle_{r'=6.0}^{r'=8.0} \tag{8.12}$$

where, in addition to a ligand volume correction $f(l,r)$, also a protein volume correction $f(p,r)$ is used. The angle brackets stand for the calculation of a mean value over all bins from 6 to 8 Å. As in PMF, the reference is chosen as a

state of no specific interaction and the density functions are not probability functions. The cut-off distance used for scoring is 6 Å.

In DrugScore, we have

$$\rho^{DS}(p,l,r) = \frac{N(p,l,r)}{\Delta V(r) \sum\limits_{r'} N(p,l,r')/\Delta V(r')} \tag{8.13}$$

$$\rho^{DS}_{ref} = \rho^{DS}_{ref}(r) = \frac{\sum\limits_{p'} \sum\limits_{l'} \rho(p',l',r)}{n_p \cdot n_l} \tag{8.14}$$

where n_p is the number of different protein atom types and n_l is the number of different ligand atom types. Here, the reference is selected as a state of mean interaction and the density functions are also probability functions. The latter fact is important, because averaging over all density functions without normalization would result in a reference dominated by contact types with high occurrence frequencies. As for ASP, the cut-off distance is 6 Å. This limit was chosen to assure that an interaction *pl* cannot be a water mediated interaction *pwl* (Gohlke et al., 2000).

It is not important whether the reference is chosen as a state of no interaction or a state of mean interaction. Its main responsibility is to weight $\rho(p,l,r)$ in best achievable agreement with experimental evidence. In Equation 8.10 and Equation 8.12, the reference depends on the contact type p_l, hence it is constant for a given contact type. As a consequence, the weighting between two different contact types $p_1\_l_1$ and $p_1\_l_2$ is constant for all distances and the extrema in the potentials will always correspond to the extrema in the $\rho(p,l,r)$. In Equation 8.14, the reference is solely a function of r. The weighting between different contact types is done by averaging over all possible contact types, but in contrast to PMF and ASP the weighting for short range interactions of two given contact types may differ from the weighting of long range interactions for the same types. As a result, the extrema of the DrugScore potentials can differ from the extrema of $\rho(p,l,r)$.

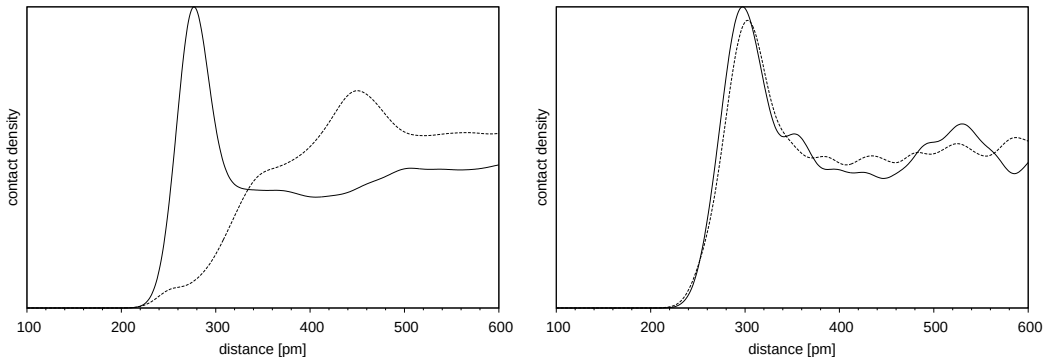
An advantage of the DrugScore reference state is the implicit inclusion of a volume correction. At short distances, generally fewer contacts are found

than theoretically expected. This is due to the inaccessibility of space actually occupied by other ligand or protein atoms, but it also implies that the reference state obtains lower values at short distances. Thus, the ratio $\frac{\rho(p,l,r)}{\rho_{ref}(r)}$ does not change in the mean. This implicit correction is an average correction for all atom types and it is sufficient as long as the available volume for particular atom types is not significantly different from the averaged value. However, Mooij and Verdonk (2005) demonstrated that there are considerable deviations from the mean value in case of protein atoms. Thus, also for DrugScore-like pair potentials, an explicit volume correction seems to be necessary when deriving contact data from protein complexes. Therefore, also an explicit volume correction will be investigated in the Results and Discussion section.

A putative disadvantage of the DrugScore reference state is the fact that any incorrect or erroneous density function will influence all resulting potentials or more generally speaking, there is only one reference function that will affect and therefore determine the quality of all potentials. The latter fact becomes even more important due to another problem of Equation 8.14 which will be discussed in the next section.

8.3. DSX Pair Potentials

DSX pair potentials are based on Equation 8.13 and Equation 8.14, but in contrast to DrugScore, *DSX* does not apply Sybyl atom types, but atom types defined by *fconv* (Neudert and Klebe, 2011b) (see Table A.2). The importance of the utilized atom-type set on the quality and reliability of statistical potentials has been shown in previous studies (Mooij and Verdonk, 2005; Ruvinsky and Kozintsev, 2005) and the choice of appropriate types is not trivial. In case of the Sybyl types (see Table A.1), one major concern regards the missing differentiation between oxygens with and without donor functionality (both O.3). In Figure 8.1a, two *fconv*-type-based density functions derived from the CSD (Allen, 2002) are shown. Using Sybyl types, O.3oh_O.carb and O.3et_O.carb would be merged into one single density function O.3_O.2. Depending on



(a) Dissimilar density functions **(b)** Similar density functions

Figure 8.1.: Density functions processed from the CSD. (a) O.3oh_O.carb (solid line): contact between hydroxyl oxygen and carbonyl oxygen, O.3et_O.carb (dotted line): contacts between oxygen in aliphatic ether and carbonyl oxygen. (b) N.3p_O.co2 (solid line): contact between a primary sp^3 nitrogen and oxygen in deprotonated carboxylates, N.3p_O.3oh (dotted line): contact to a hydroxyl oxygen.

the occurrence frequencies of hydroxyl and ether oxygens (both assigned as O.3), information about the hydrogen-bond interaction would be lost. In other cases, it is not really obvious whether differences in contact densities have to be expected. As already mentioned, one problem with the analogy to potentials of mean force is that particles present in different environments should also be treated as particles of different types. Thus, the more atom types are differentiated with respect to their environment, the more this problem will be reduced. The degree of differentiation is mainly limited by the available contact information in the knowledge base. For *DSX* from the 160 *fconv* atom types all hydrogen atom types as well as unusual metal types were excluded. Furthermore, some atom types with low occurrence frequencies were merged (see section 9.1). However, even if one assumes that all possible contact types of the remaining atom types will be sufficiently represented in the database, an increasing differentiation will raise another problem with respect to the reference state as defined in Equation 8.14. If an atom type p_1 is split into two new types p_{11} and p_{12}, the possible contact types $p_1\_l_x$ are considered twice

as $p_{11}\_l_x$ and $p_{12}\_l_x$. This is desired in case that all contact density functions $p_{11}\_l_x$ are different from the corresponding functions $p_{12}\_l_x$, but as shown in Figure 8.1b it is also possible that two contact types are essentially equal. In that case ($p_1\_l_x = p_{11}\_l_x = p_{12}\_l_x$), the only effect of splitting up p_1 is to double the weight of $p_1\_l_x$ in the reference state. Theoretically, one could split p_1 into a large number of subtypes p_{1y}, but if these subtypes do not differ from p_1, the result would be a reference state that is equal to the average of the $p_1\_l_x$ and the information of other contact types $p_y\_l_x$ would be significantly downscaled.

In this work, the applied strategy to reduce this problem is the clustering of the density functions by means of an appropriate similarity measure. In contrast to the merging of atom types, here it is possible to cluster two density functions $p_{11}\_l_1$ and $p_{12}\_l_1$, but to keep the differentiation between $p_{11}\_l_2$ and $p_{12}\_l_2$. The definition of the density functions for *DSX* results as

$$\rho^{DSX}(c,r) = \frac{\sum\limits_{p\_l \in c} N(p,l,r)}{\Delta V(r) \sum\limits_{p\_l \in c} \sum\limits_{r'} N(p,l,r')/\Delta V(r')} \tag{8.15}$$

$$\rho_{ref}^{DSX} = \rho_{ref}^{DSX}(r) = \frac{\sum\limits_{c'} \rho(c',r)}{n_c} \tag{8.16}$$

where c denotes an individual cluster of contact types and n_c is the number of clusters.

From a probabilistic point of view, Equation 8.15 is an estimator for the conditional probability to find a contact at distance r, given the contact type c. The reference is an estimator for the averaged probability to find an arbitrary contact at distance r and the resulting potential is a log-likelihood function.

$$\rho^{DSX}(c,r) = P(r|c) \tag{8.17}$$

$$\rho_{ref}^{DSX} = \frac{\sum\limits_{c'} P(r|c')}{n_c} = \bar{P}(r) \tag{8.18}$$

$$Score_{pair}^{DSX} = -ln\left(\frac{P(r|c)}{\bar{P}(r)}\right) \tag{8.19}$$

In principle, arbitrary likelihoods could serve as appropriate scoring measures, as long as the calculated density functions are good estimates for the corresponding probability functions. For example, equivalently to Equation 8.19, one could define,

$$Score_{pair} = -ln\left(\frac{P(c|r)}{\bar{P}(c)}\right) \tag{8.20}$$

$$\rho(c,r) = P(c|r) = \frac{N(c,r)}{F(c)\sum\limits_{c'} N(c',r)/F(c')} \tag{8.21}$$

$$\rho_{ref} = \rho_{ref}(c) = \bar{P}(c) = \frac{\sum\limits_{r'} \rho(c,r')}{n_r} \tag{8.22}$$

where $\rho(c,r)$ is the conditional probability to find a specific contact type c, given the contact distance r, and n_r is the number of used distance bins. With this definition, a normalization with respect to the (corrected) theoretically available volume would not be necessary. Instead, a normalization with respect to the occurrence frequencies of contact types would be mandatory, because otherwise the highly populated types would dominate $\rho(c,r)$. One possible normalization factor could be $F(c) = N(c) = \sum\limits_{r'} N(c,r')$, where $N(c)$ is the total number of contacts of type c found in the knowledge base.

8.4. DSX Torsion Potentials

To allow for a local relaxation of docking poses and to deal with unlikely torsion angles produced by docking programs, also knowledge-based torsion angle-dependent potentials were developed for *DSX*. Based on Equation 8.6, the state i of a torsion was defined as a function of the atom types a, b, c, d

being part of the torsion, a qualifier e and the actual torsion angle ϕ.

$$Score_{tors}^{DSX}(t,\phi) = -ln\left(\frac{\rho(t,\phi)}{\rho_{ref}}\right) = -ln\left(\frac{P(\phi|t)}{\bar{P}(\phi)}\right) \tag{8.23}$$

$$\rho(t,\phi) = \frac{N(t,\phi)}{\sum\limits_{\phi'} N(t,\phi')}$$

$$\rho_{ref} = \rho_{ref}(\phi) = \frac{\sum\limits_{t'} \rho(t',\phi)}{n_t}$$

$$t = t(a,b,c,d,e)$$

Four different values are used for e:

$e = 1$: neither b nor c is part of a ring system
$e = 2$: b or c (exclusive) is part of a ring system
$e = 3$: b and c are part of different (not fused) ring systems
$e = 4$: b and c are part of the same ring system

An example for all four types is given in Figure 8.2.

The primary intention for the torsion score is to penalize unlikely torsion angles rather than a good correlation with correct torsional energies. As a particular bond can be part of more than one torsion, the score for each bond is calculated as the mean of all torsions it participates in. A clustering of torsion types t is not necessary, as only a set of rather general atom types is considered (see section 9.2).

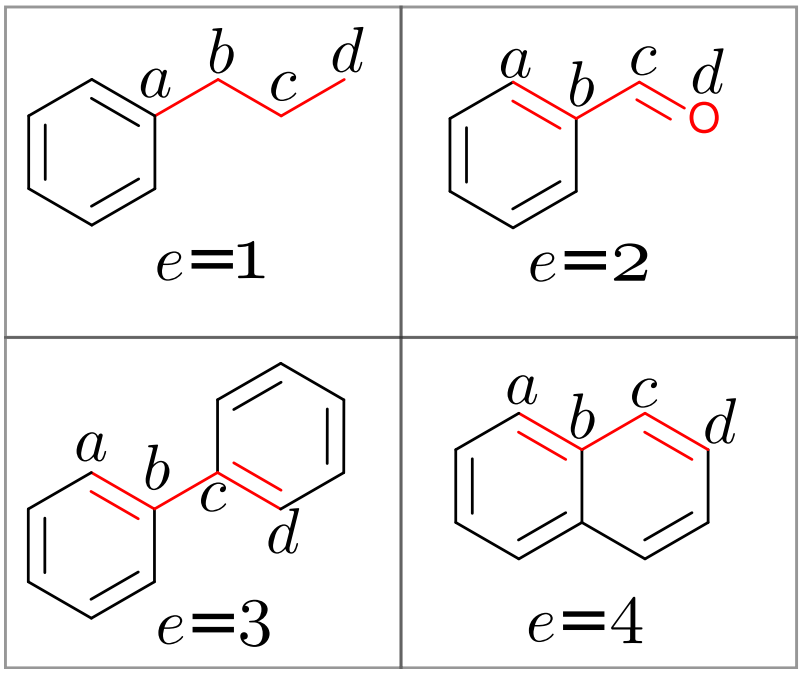


Figure 8.2.: Examples to illustrate different values for the qualifier e.

8.5. DrugScore SAS- and DSX SR-potentials

To account for desolvation effects, Gohlke et al. (2000) introduced a statistical potential in DrugScore that is based on the solvent-accessible surface (SAS). Individual SAS potentials for either protein atom types p and ligand atom types l can be derived from a database, but only l will be used in the following equations, as the formalism is identical.

$$Score^{DS}_{SAS}(l, SAS) = -ln\left(\frac{\rho(l_{complexed}, SAS)}{\rho_{ref}(l_{uncomplexed}, SAS)}\right) \tag{8.24}$$

$$\rho(l, SAS) = \frac{N(l, SAS)}{\sum\limits_{SAS'} N(l, SAS')}$$

For an isolated atom a_l, its SAS corresponds to the surface of a sphere with radius $r(l) = r_{vdW}(l) + 1.4\,\text{Å}$, because $1.4\,\text{Å}$ is the approximate radius of a sphere occupied by a water molecule. The SAS for an atom in the complexed

state is calculated as the part of its surface that is not in contact with any other protein or ligand atom. The SAS for a protein atom in uncomplexed state does not consider ligand atoms and the SAS for a ligand atom in uncomplexed state does not consider protein atoms, respectively. Gohlke et al. (2000) denote the SAS in uncomplexed state as SAS_0 and i will use both terms synonymously. Parts of SAS_0 that correspond to polar atoms are not excluded from SAS if the contacting atom in the complex is also polar, because hydrophilic groups transferred from the solvent to a polar protein environment should exhibit roughly balanced desolvation contributions. In equations 7 and 8 of the original paper, Gohlke et al. (2000) used $\Delta W_i(SAS, SAS_0)$, which requires a more detailed explanation. As illustrated in Figure 6 of the original paper, the potentials for a given atom type only depend on the SAS of atom i. In detail, the definition would be $\Delta W_i(SAS, SAS_0 = SAS)$, which becomes more obvious considering the probabilistic definition given in Equation 8.25:

$$Score_{SAS}^{DS}(l, SAS) = -ln\left(\frac{P(SAS|l_{complexed})}{P(SAS|l_{uncomplexed})}\right) \tag{8.25}$$

Given an atom type p or l, the score is the preference to find a complexed atom with a particular SAS compared to the same SAS in the uncomplexed state.

The use of an SAS-dependent term as defined in Equation 8.25 to score desolvation effects can be questioned, because only changes in the solvent accessible surface ΔSAS can contribute to binding energy, but the SAS as calculated here only contains averaged information about this difference. Therefore, it does not allow to deduce ΔSAS for a specific complex. In other words, the original DrugScore SAS potentials define a score based on the probability to find a specific atom type with a defined degree of burial in protein-ligand complexes, but they do not measure effects depending on the actual ΔSAS. In analogy to Equation 8.19 one could define a ΔSAS-dependent potential as

$$Score(l, \Delta SAS) = -ln\left(\frac{P(\Delta SAS|l)}{\bar{P}(\Delta SAS)}\right) \tag{8.26}$$

but instead, the decision was made to use a potential that holds information about both the preference for a distinct SAS (as in DrugScore) and the amount of ΔSAS. Therefore, for each atom a_l a ratio

$$SR(a_l) = \frac{SAS(l_{complexed})}{SAS(l_{uncomplexed})} = 1 - \frac{\Delta SAS}{SAS_0} \tag{8.27}$$

is calculated and the DSX-SR-potential is defined as shown in Equation 8.28,

$$Score_{SR}^{DSX}(c, SR) = -ln\left(\frac{P(SR|c)}{\bar{P}(SR)}\right) = -ln\left(\frac{\rho(c, SR)}{\rho_{ref}}\right) \tag{8.28}$$

$$\rho(c, SR) = \frac{\sum\limits_{l \in c} N(l, SR)}{\sum\limits_{l \in c} \sum\limits_{SR'} N(l, SR')}$$

$$\rho_{ref} = \rho_{ref}(SR) = \frac{\sum\limits_{c} \rho(c, SR)}{n_c}$$

where c denotes a cluster of ligand-atom types and n_c is the number of clusters. It is important to point out again that in case of Equation 8.24 the SAS in uncomplexed state is an averaged value across the entire database, whereas in Equation 8.27 it is a specific value for each individual atom in a specific protein-ligand complex. The SR potentials for protein atoms are calculated analogously.

Chapter 9.
9 Methods

For all purposes of atom-type perception, ring perception or generally parsing of input files, the *fconv* libraries (Neudert and Klebe, 2011b) (see Part I) were used. A major reason for the development of an own library for consistent atom type perception was the easy integration into other self developed programs together with the ability to drive the perception in the direction needed by these other programs. Most other non-commercial scoring functions rely on the atom types as they are supplied with the input files. In consequence they perform well, only if the input is prepared carefully. As explained in Part I there are cases where a perfect automatic atom type perception is simply not possible. For example a primary alcohol (without set hydrogens in the structure) will be typed as aldehyde, if the C-O bond length is shorter than a program specific threshold. In such cases it is especially important to apply identical atom type perception routines in both processes, the derivation of contact data from a database and the scoring. This should reduce the bias of systematic errors in the atom-type perception. Furthermore, reassigning atom types in re-scoring makes the program independent from differences in the docking solutions with respect to their atom types (which may differ among different docking programs). For higher consistency, the decision was made to generally ignore any predefined hydrogens and set standard protonation states (see section 9.1).

The definition of distance-dependent pair potentials $Score_{pair}^{DSX}$, torsion angle potentials $Score_{tors}^{DSX}$ and SR potentials $Score_{SR}^{DSX}$ was explained in chapter 8. Whereas torsion angle potentials are derived from the CSD only, the SR potentials originate from the PDB only and pair potentials are derived from both sources. The pair potentials are also used for evaluation of intramolecular interactions of the ligand, but only for those contacting atoms that are separated by at least four bonds (on the shortest path). The total DSX-score for a given protein-ligand complex is given by Equation 9.1,

$$\begin{aligned} Score_{total} &= w_p \cdot Score_{tot}^{pair} + w_t \cdot Score_{tot}^{tors} \\ &\quad + w_s \cdot Score_{tot}^{SR} + w_i \cdot Score_{tot}^{intra} \end{aligned} \tag{9.1}$$

$$Score_{tot}^{pair} = \sum_{a_i \in P} \sum_{a_j \in L} Score_{pair}^{DSX}(c(a_i, a_j), r(a_i, a_j)) \tag{9.2}$$

$$Score_{tot}^{tors} = \sum_{b} \sum_{T \in b} \frac{Score_{tors}^{DSX}(t(T), \phi(t))}{n_T}$$

$$Score_{tot}^{SR} = \sum_{a \in P} Score_{SR}^{DSX}(c(a), SR(a)) + \sum_{a \in L} Score_{SR}^{DSX}(c(a), SR(a))$$

$$Score_{tot}^{intra} = \sum_{a_i \in L} \sum_{a_j \in L} \begin{cases} 0 & \text{if } SP(a_i, a_j) < 4 \\ Score_{pair}^{DSX}(c(a_i, a_j), r(a_i, a_j)) & \text{if } SP(a_i, a_j) \geq 4 \end{cases}$$

where a is an atom from either the set of protein atoms P or the set of ligand atoms L, c is a cluster type, b is a central bond of a torsion T, t is a torsion type, n_T is the number of torsions for a given bond, SR is the SAS-ratio for a protein or ligand atom, $w_{p/t/s/i}$ are the weighting factors used and SP is the shortest path between two particular atoms of a ligand.

To enable an unbiased comparison (not trained for a particular test set), the weightings of the individual potentials were not adjusted in the validation. Instead the individual terms were only toggled on or off with a weighting of 1.0 or 0.0. For validation, ten different schemes are used as shown in Table 9.1, where DSX^{CSD} denotes derivation of pair potentials from the CSD and DSX^{PDB} denotes derivation from the PDB: In addition, results for $w_i = 1.0$ and $w_i = 0.0$ are presented in the evaluation of ranking power (see section 10.2).

Table 9.1.: Setups for *DSX* validation. A '•' indicates a weighting of 1.0

Abbreviation	w_p	w_t	w_s
DSX^{PDB}::Pair	•		
DSX^{PDB}::SR			•
DSX^{PDB}::PairSR	•		•
DSX^{CSD}::Pair	•		
DSX^{CSD}::PairSR	•		•
DSX^{CSD}::Tors		•	
DSX^{CSD}::PairTors	•	•	
DSX^{CSD}::All	•	•	•
DSX^{CSD}::Pharm	•		
DSX^{PDB}::Pharm	•		

9.1. Pair Potentials

Similar to DrugScore, also for *DSX* two different knowledge bases were used to derive the potentials. The first is the PDB (Berman et al., 2002) and the second is the CSD (Allen, 2002).

In the PDB-case an initial list of 37 067 Xray-structures with a resolution up to 2.4 Å and containing at least one ligand was used (a complete list of PDB codes is available as supporting information to Neudert and Klebe (2011a)). Only contacts between atoms with B-factors $\leq$40 Å^2 and occupancies $\geq$0.5 were considered. Also a set of potentials was derived after exclusion of all structures being part of the primary test set, but no difference in the validation as presented in chapter 10 was observed. This is not surprising, as the test set represents only 0.5 % of the knowledge base. All HETATM molecules (including cofactors) with more than five non-hydrogen atoms and also water molecules were considered as ligand. When processing one of these ligands, the remaining part of the HETATM molecules was considered as part of the protein.

In the CSD-case, ConQuest (Bruno et al., 2002) was used to query the database for all structures with an R-factor $\leq$ 0.075, at least one carbon, no error flag set and completeness of all coordinates. After removal of duplicates (some structures have two entries, with and without hydrogens respectively), for

the resulting 345 726 structures the crystal packings were generated using *fconv* (see Part I). To evaluate the contact data, the central molecule of each packing was treated as ligand and the surrounding molecules as protein. Therefore, the packings were generated with a size that guarantees to consider all atoms within a range of 6 Å around the central molecule. In case of different molecules in the unit cell, each of them was treated as the "ligand" once.

All contact data were derived symmetrically, hence contact type *A_B* is equal to *B_A*. Although this appears obvious only in the CSD-case, also in the PDB-case better results were achieved when not using asymmetric data.

To account for the inherent limitation of low occurrence frequencies for some of the desired atom types, not the full set of *fconv* atom types was used, but some types were merged initially. As a result, there are 17 types for carbon, 24 for nitrogen, 10 for oxygen, 4 for sulfur, 2 for phosphorus, one for F, Cl, Br, I and 7 different metal ion types. The mapping from internal *fconv* atom types onto these merged types is available in Table A.3 on page 202. The following protonation flags (see Part I) were set in the atom type perception: *protonate_amine*, *protonate_amidin* and *protonate_guanidin*.

Only contact types with more than 1000 contacts (within 6 Å) in the database were considered for further processing. Furthermore, all types were neglected, where not at least one of the two atoms was either a carbon, nitrogen, oxygen, sulfur or phosphorus. After applying these filters, 930 contact types were obtained in the PDB-case and 1561 in the CSD-case. The lower number of different contact types in the PDB case is not just due to the smaller database, but due to the fact that many atom types are never part of a protein molecule. If a contact type which remained unconsidered due to too low occurrence frequency has to be handled in re-scoring, it is mapped to the most similar contact type with sufficient occurrence frequency. The criterion for similarity is the same as the one used for clustering (see below). In case a completely unknown contact type appears, it will not be considered. However, this situation will be rather rare and not of significant influence as many additional known contact types for the involved atoms will be regarded.

Furthermore, a second set of pair potentials was derived that is tailored

for usage in hotspot analysis (see section 11.1). Here, a reduced set of pharmacophoric atom types was used, in detail: donor, acceptor, donor-acceptor, aromatic, hydrophobic and metal. If an *fconv* atom type could not uniquely be assigned to one of these 6 categories, the element type was used as a dummy. The corresponding mapping from the internal *fconv* types is also available in Table A.3.

In contrast to DrugScore and most of the other knowledge-based scoring functions a bin size of 0.01 Å instead of 0.1 Å is used for both deriving the contact data and the resulting potentials. It is important to note that, due to the smoothing function subsequently applied to the data, this has no impact on the statistical significance. Whether one would use a 0.0001 Å or 0.01 Å binning is irrelevant as long as the same smoothing function is applied. For *DSX* a Gaussian smoothing is used, hence it is the sigma (parameter determining the width of the Gaussian function) that is relevant for an appropriate signal to noise ratio. A value of $\sigma = 0.15$Å was chosen, as this is in good agreement with the triangular smoothing applied by Gohlke et al. (2000). In the original DrugScore paper, it was argued that the uncertainties in crystallographically determined coordinates are the rationale for smoothing. However, these uncertainties are already considered while averaging over a large number of complexes from the database. The actual rationale for smoothing is to increase the signal to noise ratio. This implies that there should be an optimal sigma for each individual contact type, depending on the number of such contacts in the database and the distribution of the contact distances. Generally, a higher value for sigma should be used for lower occurrence frequencies, but in case of Equation 8.16, averaging over density functions with different sigma levels (lower sigma for higher occurrence frequency) would increase the impact of highly populated contact types in the reference. Moreover, it would complicate the similarity measurement for the density functions. Thus, the decision was made to use a constant sigma for all contact types. A narrower binning of 0.01 Å was chosen to avoid a second smoothing in the scoring process. If a distance falls close to the border between two bins, an average value of both bins should be applied. If this smoothing is neglected, a high difference in the score is possible for

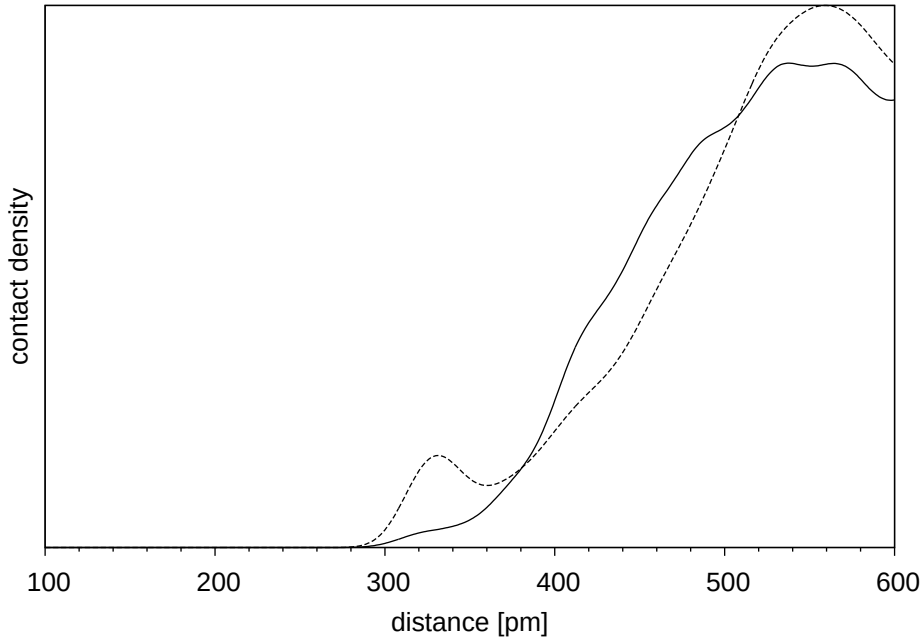


Figure 9.1.: Density functions for two contact types processed from the CSD. Cl.0_P.o (solid line): contact between an organic chlorine and phosphorus bound to at least one oxygen, Cl.0_P.3 (dotted line): contact to other phosphorus atoms.

rather small differences in the contact distances. Using a small bin size of only 0.01 Å, the differences between two bins are negligible and a smoothing with neighboring bins can be avoided in the scoring process, which speeds up computing.

To cluster the contact types $c = p\_l$, a hierarchical approach with complete linkage was implemented. Different distance metrics for the $\rho(p, l, r)$ were evaluated and the best results were obtained using squared Euclidean distances:

$$dist(\rho_a, \rho_b) = \sum_{r=1.00}^{r=5.50} (\rho_a(r) - \rho_b(r))^2 \tag{9.3}$$

Distances were multiplied by a factor 10.0, if a and b were contact types that not only differed with respect to the atom types but also with respect to the element types. There is no general rule how to choose an appropriate distance

threshold for clustering. After visual inspection of some density functions of different distance levels, the PDB potentials were merged down to a set of 300 contact types and the CSD potentials to 600 contact types, corresponding to a maximum distance of 0.0028 and 0.0026 respectively. Figure 9.1 shows the unmerged density functions with the lowest distance in the CSD case. Using the mentioned thresholds, no contacts with different element types were merged.

In case of the pharmacophoric types, 66 different contact types were merged into 62 clusters for the CSD-knowledge base. For the PDB case, 50 different contact types were obtained and used without any clustering, because all of them were dissimilar enough.

For low contact distances, the reference density in Equation 8.16 is not well defined, simply because no structural data are available in this distance range. Usually, for simple re-scoring no problem should occur, as the docking programs avoid clashes. However, to enable minimization on the potential functions, a repulsive term is attached in the range from 0 Å up to the first maximum that is followed by a negative potential value. The actual functional form of the repulsive term can be selected arbitrarily, but a smooth connection is desired. A function starting with a gradient of 0.025 and linearly decreasing this gradient to 0 with decreasing distance to 0 Å was attached to all potentials.

9.2. Torsion Angle Potentials

To derive torsion angle potentials, the same CSD dataset as for the *DSX*-pair potentials was used. A 2°-binning was used for a range from $\phi = 0°$ to $\phi = 180°$. A Gaussian smoothing with $\sigma = 5°$ was applied and only torsions with more than 50 occurrences in the database were considered. To cover most of the torsion types that occur in ligands, the atom types are reduced according to the following scheme: "$Element.Hybridization$" in case of carbon, nitrogen and oxygen, "Hal" in case of halogens and "$Element$" for all other elements. The result are 4464 different torsion types. With respect to the primary test set (see section 9.7), only seven different torsion types were not sufficiently represented

by the database. However, in only four ligands there is an unconsidered torsion type where not at least one other torsion potential for the corresponding bond is available.

9.3. SR Potentials

The structures used to derive PDB pair potentials were also used to derive SR potentials. To approximate the SAS spherical grids with precomputed coordinates for each element type are used. All grids consist of 162 points which were calculated by 2-fold subdivision of an icosahedron and subsequent scaling to a radius of $r_{vdW} + 1.42\,\text{Å}$ (to account for the space occupied by a water molecule). For nitrogen, oxygen and sulfur a vdW-radius decreased by 0.2 Å was used to account for putative hydrogen-bond formation. For a nitrogen for instance, this results in $r_{grid} = r_{vdW} + 1.42 = 2.77\,\text{Å}$ and a grid of a mean closest point-to-point distance of 0.81 Å with a standard deviation of 0.01 Å. The mean distance to the closest 6 points is 0.88 Å with a standard deviation of 0.08 Å. The SR for each ligand atom is then calculated with Equation 9.4

$$SR(a_l) = \frac{points_{complexed}(a_l)}{points_{uncomplexed}(a_l)} \tag{9.4}$$

with a_l as ligand atom, $points_{uncomplexed}$ as the number of grid points not contacted by other ligand atoms and $points_{complexed}$ is the number of grid points not occupied by other ligand- or protein atoms. If a_l is a nitrogen or oxygen atom, any contacting nitrogen and oxygen atoms of the protein are not considered for $points_{complexed}$. The SR for protein atoms is calculated correspondingly. It is calculated for those protein atoms that are in SAS-contact to at least one ligand atom.

A bin size of 0.01 was used for the SR and a Gaussian smoothing with $\sigma = 0.08$ was applied. The same atom type classification as used for the pair potentials has been considered and only types with more than 50 occurrences in the database were regarded for further processing.

For clustering, squared Euclidean distances were used again:

$$dist(\rho_a, \rho_b) = \sum_{SR=0.00}^{SR=1.00} (\rho_a(SR) - \rho_b(SR))^2 \tag{9.5}$$

In case of protein atoms, no types are merged as they are sufficiently different. Also for ligand atoms a very low distance threshold was chosen, resulting in 50 clusters (from 68 atom types).

9.4. Ligand Relaxation

To enable a local relaxation of docking poses, Powell's method, as described in the Numerical Recipes (Press et al., 2007a), was implemented into *DSX*. *SR* potentials are not used in the minimization and for torsion- and pair-potentials the same weightings as specified for re-scoring are used. Intramolecular interactions are also evaluated as defined in Equation 9.1.

9.5. Volume Correction

To calculate volume corrections for the PDB-derived contact data, a spherical grid-based approach similar to that described for the *SR* potentials was applied. Here, a 5-fold icosahedron subdivision resulting in 10 242 points was used to generate the grid's coordinates. These coordinates on a sphere were scaled according to the radius under investigation, whereat a 0.2 Å binning was used. For each radius bin, the corresponding points were evaluated with respect to their neighboring atoms. If no surrounding atom was closer than its vdW-radius plus 1.4 Å, the point under investigation was counted as unoccupied. The additional 1.4 Å were used to account for the volume of a putative contacting atom. Strictly speaking, one should derive individual data for each possible contacting element type using the vdW-radii of these elements. However, as one is interested in the relations between different atom types and not in absolute values, the fixed radius should be a sufficient approximation. The available

volume fraction for an atom type a was calculated as

$$f(a,r) = \frac{points_{unoccupied}(a,r)}{points_{total}(a,r)} \tag{9.6}$$

9.6. Implementation Details

DSX is implemented in ISO C++ and binaries for Linux and MacOS were made freely available on `www.agklebe.de/drugscore/dsx_download.php`.

Valid input formats for *DSX* are PDB or MOL2 for proteins and MOL2 or DLG for ligand files. Cofactors, metals and water molecules can be supplied separately or together with the protein (when in MOL2 format). The user can choose between different interaction modes that specify whether cofactors, metal ions and/or water molecules should be handled individually or as part of the protein. If for example the cofactor was kept rigid during the docking process, it should be considered as part of the protein in re-scoring. But if it was kept flexible upon docking, also the interactions between cofactor and protein should be re-scored. In consequence, when choosing an interaction mode with individual cofactor, this cofactor must be supplied as additional MOL2 file with a number of cofactor poses equal to the number of generated ligand poses.

As mentioned on page 131, *DSX* always redefines the atom types using the same routines that were used for atom-type perception when deriving the potentials. Additionally, the program generally ignores hydrogens in the input files, thus the results are independent from any predefined protonation states.

For docking solutions obtained by AutoDock, where amino acid side chains of the protein were kept flexible, the user can switch-on a flag to consider the correct side chain conformations for each solution. In that case, the original DLG file has to be supplied. For docking results using GOLD with flexible side chains, the necessary information is included in the MOL2 output files, but currently the protein must be supplied in PDB format if *DSX* is supposed to consider the correct side chain conformations.

If GOLD was used to dock with explicit water molecules, there is a flag to consider the corresponding information in the GOLD result files.

If the check for covalently bound ligands is turned on, *DSX* will ignore all atoms participating in a protein-ligand bond and also their neighboring atoms.

The weightings used for the different types of potentials can be freely assigned by the user.

Furthermore, a PyMOL-based[1] visualization similar to the work of Block et al. (2007) was implemented. Favorable and unfavorable per-atom scores are visualized by blue and red spheres respectively, where the sphere's radius scales with the absolute value of the score. Additionally, single contacts with very high or low scores are visualized as red or blue lines and also unfavorable torsion angles are displayed.

The most significant speedup of *DSX* is done by using a modified octree data structure (see section A.1).

9.7. Test Sets and Validation

To evaluate *docking-, ranking-* and *scoring power*, the test set prepared by Cheng et al. (2009) is used. It consists of a primary set of 195 protein-ligand complexes and four additional sets to assess *ranking-* and *scoring power*. The authors of this test set evaluated 16 different scoring functions, making it one of the most comprehensive comparisons up to now. The primary set was compiled from the PDBbind database (Wang et al., 2004, 2005), regarding quality and diversity of the structures and considering only complexes with experimentally determined binding constants. It covers 65 diverse targets and each target is represented by three complexes, one of them with high binding affinity, one with low affinity and one close to the mean. Up to 100 highly diverse decoy poses were generated for each complex using various docking programs followed by a subsequent cluster analysis. This primary set can be downloaded from the PDBbind[2], including all decoy structures and thus enabling a comparison

[1] The PyMOL Molecular Graphics System, Schrödinger, LLC.

[2] `www.pdbbind.org`

using identical input as in the original publication. The four additional test sets consist of 112 HIV protease-, 73 trypsin-, 44 carbonic anhydrase- and 38 thrombin complexes with known binding constants respectively.

To assess the *docking power* of *DSX*, five different success rates on the primary test set are calculated and compared to the results given in the supporting information (part VI) by Cheng et al. (2009). In two cases, a success is defined as finding the native pose on rank 1 or among the first five ranks, respectively. In the other three cases a success is defined as finding a docking pose approximating the crystal structure with an RMSD ≤ 2.0 Å on rank 1, among the first five ranks or on rank 1 excluding the native pose, respectively. In a personal communication, Cheng et al. (2009) reported that there are five complexes where all decoys have an rmsd ≤ 2.0 Å (1df8, 1fcx, 1fcz, 1fd0, 2f01) and seven complexes where all decoys have an rmsd > 2.0 Å (*'1a30, 1elb, 1nhu, 1tyr, 1u1b, 2fzc, 6rnt'*). Therefore, they computed the success rate using Equation 9.7, where S is the number of success cases.

$$Success\_rate = \frac{S-5}{195-5-7}100\% \tag{9.7}$$

Meanwhile it was found that only one complex exists where all decoys have an rmsd ≤ 2.0 Å (*'2f01'*), hence the correct success rate would be $(S-1)/(195-8)$. However, Equation 9.7 is used in this work, because it was used by Cheng et al. (2009) for all results tabulated in their paper and therefore it must also be used here for a fair comparison with the cited values.

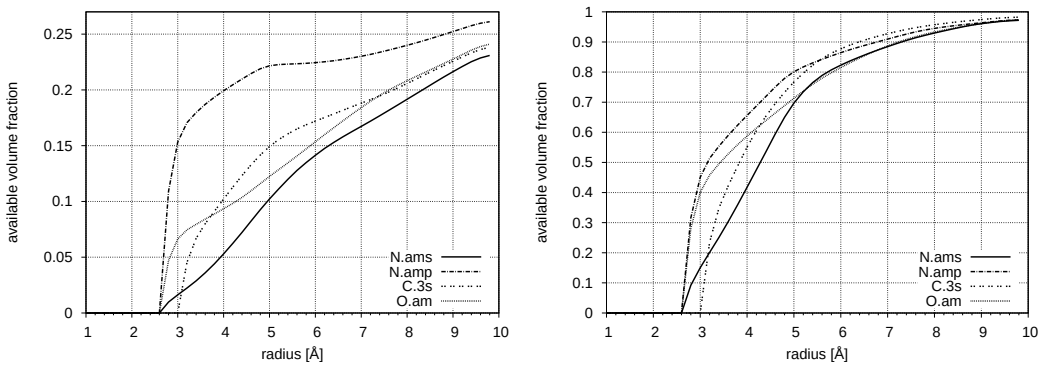
To assess *ranking-* and *scoring power*, Spearman and Pearson correlation coefficients are calculated for the four additional test sets and compared with the results given in the supporting information (part VII) by Cheng et al. (2009). Also a success rate for *ranking power* based on the primary set is calculated and compared with the results given in Table 4 by Cheng et al. (2009). Here, a success is achieved if the three complexes for one of the 65 targets are ranked in correct order with respect to their binding constant. Furthermore, the Pearson correlation between experimental affinities and scores is calculated for the complete primary test set.

For all results on this test set, *DSX* was used in version 0.88. CSD and PDB potentials were used in version 05/11.

10 Chapter 10. Results and Discussion

From the results of Cheng et al. (2009), only the best performing variants of each scoring function are listed except for DrugScore, where all results are given. The missing results are available in the supporting information by Cheng et al. (2009). Also the pharmacophoric pair potentials were applied to the test set, although they are mainly intended for hotspot analysis (see section 11.1). They are abbreviated by DSX^{CSD}::Pharm and DSX^{PDB}::Pharm.

In section 8.2 it was mentioned that a volume correction is of higher importance for protein atoms, because there are significant differences in the available volume, especially for backbone atoms compared to side chain atoms. To assess the influence of a volume correction for PDB-based potentials, the available volume fractions were calculated for protein and ligand atoms, respectively. Qualitatively, the results shown in Figure 10.1 are similar to what was found by Mooij and Verdonk (2005) (see Figure 4 and 5 in their paper). As the fractions were derived for a distinct classification into protein- and ligand atoms, they can only be applied to asymmetric PDB data. When using a DrugScore-like reference state, Mooij and Verdonk (2005) found the potential for a contact between protein backbone amides and aromatic nitrogens to never approach negative (favorable) values (Figure 6 in the original paper). In contrast, Figure 10.2 shows a minimum at negative values not only when potentials are derived symmetrically, but also for asymmetric potentials from the PDB. If



(a) Protein atoms **(b)** Ligand atoms

Figure 10.1.: Available volume fractions. N.ams: nitrogen in secondary amides; N.amp: nitrogen in primary amides; C.3s: secondary sp^3 carbon; O.am: amide oxygen.

the volume correction is applied in the asymmetric case, the expected effect of more pronounced minima is observed and also an improvement in *docking-* and *ranking power* in the validation. Surprisingly, the symmetric variant with less pronounced minima performs even better than both asymmetric variants. One can only speculate about the reasons. In case of asymmetric data, certain contact types A_B have very low occurrence frequencies, whereas B_A is well populated. If such a type A_B has negative impact on the reference, the performance of asymmetric potentials decreases. In the symmetric case A_B and B_A are merged to one type that is dominated by B_A contacts, hence a possible bias of statistically underrepresented A_B is alleviated. For now, it was decided to neglect the volume correction, but it could be an interesting aspect for further investigation.

Corresponding to the density functions shown in Figure 8.1a, Figure 10.3 shows the resulting *DSX* pair potentials.

Figure 10.4 gives an example for torsion angle potentials. The solid line corresponds to a carbon chain where all atoms have hybridization sp^3, whereas the central bond is formed by two sp^2 hybridized carbons in case of the dotted line.

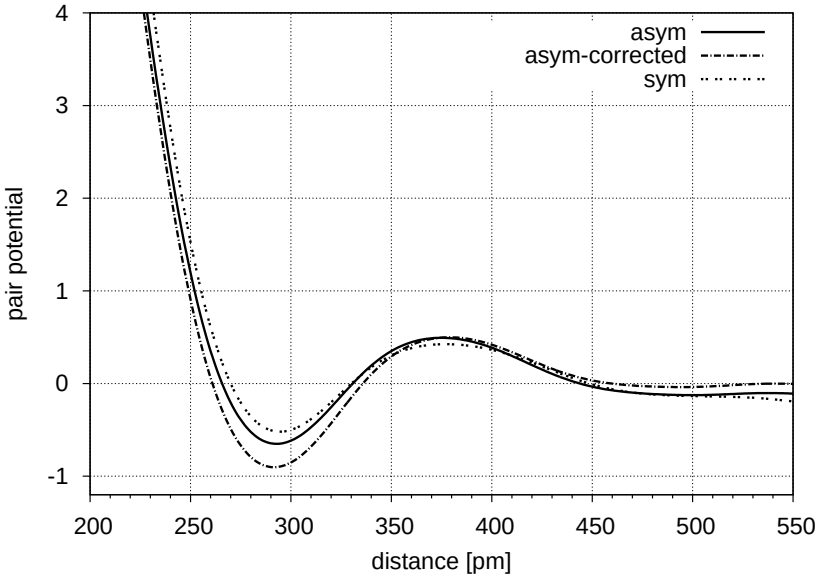


Figure 10.2.: PDB pair potentials for contacts between N.ams (secondary amide) on protein side and N.ar6 (aromatic nitrogen) on ligand side. asym: derivation with differentiation between N.ams_N.ar6 and N.ar6_N.ams, but without volume correction; asym-corrected: with volume correction; sym: symmetric contact types without volume correction

An example for SR potentials is shown in Figure 10.5. The amount of solvent accessible surface that becomes buried upon ligand binding increases with decreasing SAS ratio given on the x-axis. Higher magnitudes in case of ligand atoms indicate higher changes of the SAS upon complex formation compared to protein atoms.

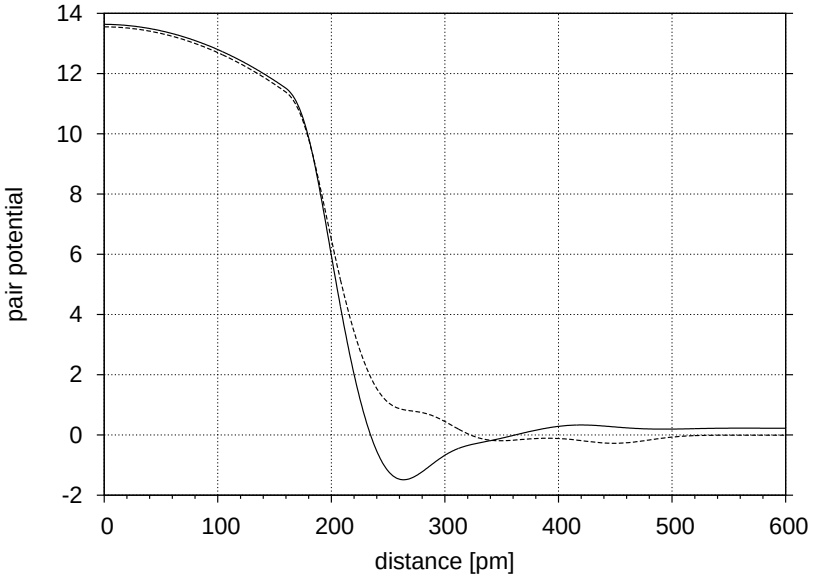


Figure 10.3.: Pair potentials for two contact types processed from the CSD. O.3oh_O.carb (solid line): contact between hydroxyl oxygen and carbonyl oxygen, O.3et_O.carb (dotted line): contacts between oxygen in aliphatic ether and carbonyl oxygen.

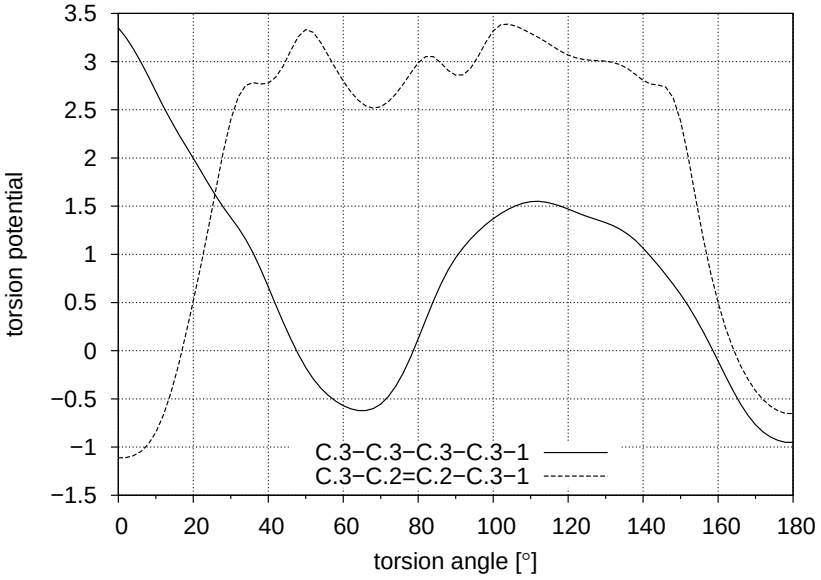


Figure 10.4.: Torsion angle potential for an sp^3 carbon-chain (solid line) and for a carbon chain with a double bond (dotted line)

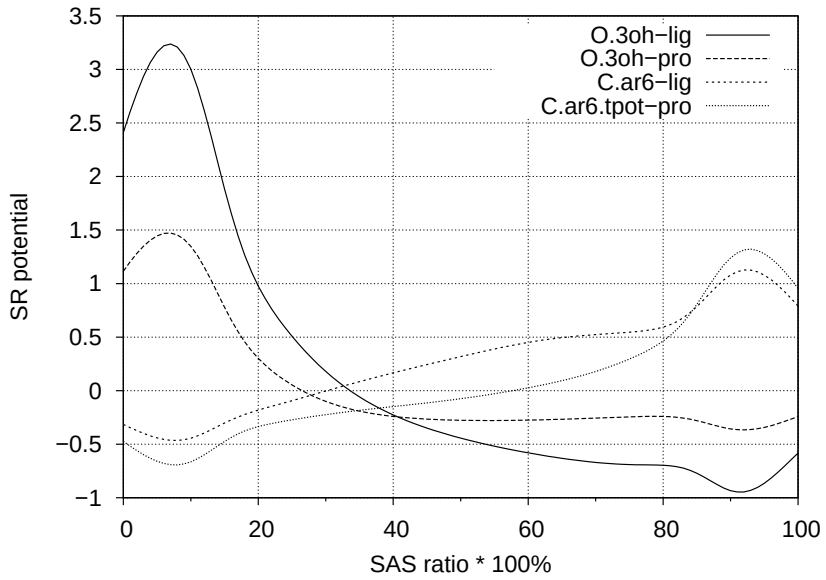


Figure 10.5.: Example for *SR* potentials. O.3oh-lig: hydroxyl oxygen in ligands, O.3oh-pro: hydroxyl oxygen in proteins, C.ar6-lig: aromatic carbon in ligands, C.ar6-pro: aromatic carbon in proteins.

10.1. Docking Power

Table 10.1 shows the validation similar to the results recorded in Table S6, S7 and S8 from the supporting information of Cheng et al. (2009). The most important number can be found in the last column and corresponds to the success rate of finding solutions with RMSD ≤ 2.0 Å on rank 1, when the native, crystallographically determined ligand geometry is excluded from the decoy set. In a virtual screening run, this docking solution would be the relevant pose that is compared to the top ranks of other compounds. Therefore, it must be as close as possible to a native geometry to allow for a reliable compound selection.

For both, the CSD- and the PDB-case, the combination of pair and SR potentials shows an improvement compared to the pair potentials alone. It has to be noted that the DSX^{CSD}::PairSR mode is a combination of information retrieved from the CSD and the PDB, whereas the DSX^{PDB}::PairSR mode

only relies on PDB data. Interestingly, the differences between CSD- and PDB-derived potentials and their combinations are only marginal with respect to *docking power*.

A combination with torsion potentials (DSX^{CSD}::PairTors) increases the recognition of native geometries, but decreases the success rate when the native pose is excluded. This indicates that they are very sensitive to deviations from ideal geometries as found in the CSD. Native geometries that are ranked on first place by DSX^{CSD}::Pair, but not by DSX^{CSD}::PairTors are *'2g94, 7cpa, 2bok'* and *'1bma'*. In contrast, DSX^{CSD}::PairTors ranks native geometries of *'1xgj, 2azr, 1sl3, 2bz6, 2std, 1a30'* and *'1rnt'* on first place, but DSX^{CSD}::Pair does not. From the mentioned eleven structures, eight have very large ligands with many rotatable bonds. For such structures, there is a higher chance for docking programs to fail with one of these numerous bonds, hence it is easier to differentiate between native pose and docking solutions with respect to torsion angles. One could also speculate that, in contrast to small molecule crystal packings, higher deviations from ideal geometries are possible in protein-ligand complexes. In that case, the CSD-derived torsion angle potentials could penalize native poses too strongly.

It must be pointed out that the discussed improvements when applying SR and torsion potentials are only marginal. It is not sure whether these terms generally improve the results on other data sets.

With respect to *docking power*, *DSX* outperforms all other functions tested, except for ASP which is on a similar level. For the used test set, the best results are obtained using all three types of potentials in combination.

Table 10.1.: Success rates (%) for the evaluation of *docking power*. Results (excluding *DSX*) cited from Cheng et al. (2009).

	Native pose on		≤ 2.0 Å pose on		
Scoring function	Top pose	Top 5 poses	Top pose	Top 5 poses	Top pose no cryst.[a]
DS::Jain	1.5	15.4	44.8	79.2	44.8
DS::LigScore2	17.9	49.7	71.6	92.9	69.4
DS::LUDI2	9.7	29.2	57.4	83.6	56.8
DS::PLP1	40.5	56.4	75.4	97.3	68.3
DS::PMF	19.5	44.1	43.7	67.2	39.3
DrugScore^{CSD}::Pair	50.3	79.5	58.5	94.0	25.7
DrugScore^{CSD}::PairSurf	44.6	80.0	54.1	95.6	25.1
DrugScore^{PDB}::Pair	40.0	73.8	74.3	93.4	68.9
DrugScore^{PDB}::PairSurf	39.5	74.9	74.3	95.1	69.4
DrugScore^{PDB}::Surf	3.6	20.0	32.8	80.3	32.2
DSX^{CSD}::Pair	50.8	77.4	83.6	95.6	77.6
DSX^{CSD}::PairSR	51.3	79.0	84.7	96.2	78.1
DSX^{CSD}::PairTors	52.3	77.4	84.2	95.1	77.0
DSX^{CSD}::All	52.8	77.9	85.2	96.2	79.2
DSX^{CSD}::Tors	8.7	20.0	38.3	76.5	36.1
DSX^{PDB}::Pair	50.3	78.5	84.2	95.6	75.4
DSX^{PDB}::PairSR	51.8	77.9	84.7	95.6	78.7
DSX^{PDB}::SR	3.6	16.9	39.3	82.5	38.3
DSX^{CSD}::Pharm	47.2	76.4	79.8	95.6	73.2
DSX^{PDB}::Pharm	41.5	72.3	77.6	94.0	69.4
GOLD::ASP	36.9	71.8	82.5	95.6	77.6
GOLD::ChemScore	17.9	50.8	70.5	86.9	69.4
GOLD::GoldScore	8.2	28.7	68.9	89.6	68.3
GlideScore::SP	18.5	50.3	73.2	93.4	72.7
SYBYL::F-Score	21.5	49.2	64.5	90.7	60.1
X-Score1.2	32.3	64.6	67.2	91.3	63.4
X-Score1.2::HMScore	30.3	57.9	68.3	90.7	62.3

[a]The native geometry was not part of the decoy set.

10.2. Ranking Power

Table 10.2 shows the validation similar to the results recorded in Table 4 by Cheng et al. (2009). In the original paper only the best performing version of each scoring function was evaluated. For comparison, also here only the results for the best CSD- and PDB-based *DSX* mode are presented.

In case of Discovery Studio, Glide, GOLD and Sybyl, ligand optimization was performed by the functions implemented into these programs (Cheng et al., 2009). For DrugScore and X-Score, Discovery Studio was used to minimize the ligands in the CHARMm force field (Cheng et al., 2009). For *DSX*, the program's own local minimization was used in the CSD case, while in the PDB case no minimization is possible due to the lack of torsion angle potentials.

Table 10.3 shows achieved ranking correlations for the additional test sets and it corresponds to the results listed in Tables S13, S14, S15 and S16 of the supporting information of Cheng et al. (2009). The results shown in brackets were obtained when additionally applying intramolecular interactions with a weighting of $w_i = 1.0$. Interestingly, this improves the correlations except for the case of thrombin.

Applying the torsion angle potentials in addition to the pair potentials improves ranking in case of HIV protease and trypsin, but makes the results for carbonic anhydrase worse. Remarkably, applying only torsion angle potentials without any assessment of protein-ligand interactions produces the best ranking correlation of all scoring functions in case of HIV protease. This emphasizes the disappointing performance of all scoring functions under assessment for this target. The ligands for HIV protease are rather large and have many rotatable bonds. A possible explanation for the (at least significant) correlation with the torsion score is that in case of ligands with lower affinities, there are often higher deviations from ideal torsion angles.

Intramolecular- and torsion angle potentials exhibit different (positive or negative) impact on *ranking power*, depending on the target. This implies that there is no unique best weighting scheme for the different potentials, but instead a tailored set of weighting parameters should be identified for each

Table 10.2.: Success rates (%) for the evaluation of *ranking power* on the primary test set. Results (excluding *DSX*) cited from Cheng et al. (2009).

Scoring function[a]	on original complex structures	on optimized complex structures
X-Score::HSScore	58.5	52.3
DSX^{CSD}::All	55.4	52.3
DS::PLP2	53.8	46.2
DSX^{PDB}::PairSR	52.3	/
$DrugScore^{CSD}$::PairSurf	52.3	49.2
SYBYL::Chemscore	47.7	52.3
SYBYL::D-Score	46.2	46.2
SYBYL::G-Score	46.2	36.9
GOLD::ASP	43.1	49.2
DS::LUDI3	43.1	43.1
DS::Jain	41.5	35.4
DS::PMF	41.5	35.4
SYBYL::PMF-Score	38.5	33.8
GOLD::ChemScore	36.9	41.5
DS::LigScore2	35.4	47.4
GildeScore::XP	33.8	35.4
NHA[b]	32.3	32.3
GOLD::GoldScore	23.1	38.5

[a]Scoring functions are ranked by their success rates. [b]Ranking by the number of heavy atoms of each ligand.

target of interest.

DSX performs best in case of thrombin, is second best after X-Score in case of trypsin and performs second best after PLP2 in case of carbonic anhydrase. Astonishingly, for the latter, the pharmacophoric potentials show significantly higher correlations compared to the highly specialized pair potentials.

As in the case of DrugScore, also for *DSX* the CSD-based potentials have a higher *ranking power* compared to the PDB-based analogs. In contrast to *docking power*, the application of *SR* potentials decreases *ranking power* in most cases.

Table 10.3.: Spearman correlations for the four additional test sets. Results (excluding *DSX*) cited from Cheng et al. (2009).

Scoring function	HIV protease	trypsin	carbonic anhydrase	thrombin
DS::Jain	0.023	0.698	0.133	0.491
DS::LigScore1	0.106	0.536	0.330	0.371
DS::LigScore2	0.167	0.418	0.143	0.424
DS::LUDI2	0.047	0.791	0.405	0.558
DS::PLP2	0.168	0.774	0.772	0.666
DS::PMF04	0.200	0.395	0.612	0.022
DS::PMF	0.200	0.693	0.389	0.275
$\text{DrugScore}^{\text{CSD}}$::Pair	0.129	0.737	0.542	0.622
$\text{DrugScore}^{\text{CSD}}$::PairSurf	0.147	0.768	0.535	0.617
$\text{DrugScore}^{\text{PDB}}$::Pair	0.163	0.744	0.488	0.515
$\text{DrugScore}^{\text{PDB}}$::PairSurf	0.170	0.743	0.468	0.535
$\text{DrugScore}^{\text{PDB}}$::Surf	0.170	0.743	0.468	0.535
DSX^{CSD}::Pair	0.199 (0.225)	0.762 (0.789)	0.559 (0.611)	0.709 (0.682)
DSX^{CSD}::PairSR	0.184 (0.198)	0.733 (0.756)	0.496 (0.547)	0.679 (0.642)
DSX^{CSD}::PairTors	0.300 (0.319)	0.782 (0.797)	0.413 (0.442)	0.703 (0.668)
DSX^{CSD}::All	0.267 (0.291)	0.752 (0.776)	0.429 (0.454)	0.660 (0.636)
DSX^{CSD}::Tors	0.423	0.744	0.089	0.226
DSX^{PDB}::Pair	0.179 (0.199)	0.753 (0.782)	0.575 (0.580)	0.671 (0.637)
DSX^{PDB}::PairSR	0.160 (0.174)	0.728 (0.758)	0.519 (0.537)	0.663 (0.657)
DSX^{PDB}::SR	0.006	0.239	0.139	0.419
DSX^{CSD}::Pharm	0.109 (0.123)	0.744 (0.762)	0.708 (0.703)	0.744 (0.647)
DSX^{PDB}::Pharm	0.140 (0.151)	0.710 (0.708)	0.753 (0.761)	0.708 (0.588)
GOLD::ASP	0.140	0.744	0.486	0.287
GOLD::ChemScore	0.138	0.280	0.572	0.489
GOLD::GoldScore	0.232	0.052	0.079	0.603
GlideScore::SP	0.183	0.177	0.280	0.525
SYBYL::ChemScore	0.228	0.773	0.631	0.587
X-Score1.2::HSScore	0.214	0.824	0.595	0.586
X-Score1.3::HPScore	0.373	0.815	0.494	0.558
X-Score1.3::HSScore	0.291	0.809	0.555	0.593

10.3. Scoring Power

Table 10.4 shows the obtained affinity correlations for the primary test set and corresponds to the results in Table S11 of the supporting information of Cheng et al. (2009).

After X-Score, *DSX* is second best in this category. The minimization was applied as described for 10.2.

At this point, it is worth discussing the importance of *scoring power* (affinity prediction) for re-scoring. First, its influence on *docking power* will be discussed: One has to keep in mind that each re-scoring of docking solutions is a consensus scoring, because it is a combination of the scoring function used in docking and the function applied subsequently. To generate reasonable poses, the target function used in docking should regard all contributions to binding energy and must weight these contributions correctly. Such an ideal scoring function will be called "complete" in the following, hence completeness is a measure for the amount of considered affinity contributions and the quality of the weighting of these terms. However, some aspects of binding energy can only be evaluated as rough approximations and other aspects can even be neglected. For example, covalent bond energies must not be evaluated, because docking programs usually do not modify bond lengths, hence scoring functions used for docking can be incomplete with respect to bond energies. The same holds true for functions used in re-scoring. They can be incomplete with respect to some binding energy contributions that were evaluated by the docking program. Moreover, they can also assign much higher weights to contributions that discriminate between near native and decoy poses. For example, in case of hydrogen bonds, a docking function has to consider distances and angles. If it relied on distances only, it would generate unrealistic geometries. In contrast, a function for re-scoring could solely rely on the H-bond angles produced by docking programs and give a much higher weight on H-bond distances (in case these distances are especially valuable to penalize decoy poses). Assigning higher weights to certain terms can decrease binding affinity correlations for native poses, but at the same time increase *docking power*. Thus, one cannot generally conclude from

Table 10.4.: Pearson correlations for the primary test set. Results (excluding *DSX*) cited from Cheng et al. (2009).

Scoring function	on original complex structures	on optimized complex structures
DS::Jain	0.316	0.339
DS::LigScore2	0.464	0.479
DS::LUDI3	0.487	0.477
DS::PLP1	0.545	0.529
DS::PMF	0.445	0.294
$\text{DrugScore}^{\text{CSD}}$::Pair	0.561	0.589
$\text{DrugScore}^{\text{CSD}}$::PairSurf	0.569	0.585
$\text{DrugScore}^{\text{PDB}}$::Pair	0.524	0.543
$\text{DrugScore}^{\text{PDB}}$::PairSurf	0.531	0.536
$\text{DrugScore}^{\text{PDB}}$::Surf	0.520	0.542
DSX^{CSD}::Pair	0.597	0.588
DSX^{CSD}::PairSR	0.598	0.591
DSX^{CSD}::PairTors	0.607	0.599
DSX^{CSD}::All	0.609	0.602
DSX^{CSD}::Tors	0.481	0.478
DSX^{PDB}::Pair	0.567	/
DSX^{PDB}::PairSR	0.571	/
DSX^{PDB}::SR	0.445	/
DSX^{CSD}::Pharm	0.560	/
DSX^{PDB}::Pharm	0.547	/
GOLD::ASP	0.534	0.518
GOLD::ChemScore	0.441	0.528
GOLD::GoldScore	0.295	0.329
GlideScore::XP	0.457	0.555
SYBYL::ChemScore	0.555	0.622
X-Score1.2::HMScore	0.644	0.649

high *scoring power* to high *docking power*. An example is GoldScore, which achieves the lowest affinity correlation for the primary test set (Table 10.4), but has higher *docking power* than X-Score1.2::HMScore on this test set (68.3 % vs. 62.3 % in the most relevant category), although the latter achieves the highest affinity correlation. As a matter of fact, a function that achieves Pearson correlations of 1.0 for arbitrary datasets could still fail with respect to *docking power*, because it could still calculate better scores for certain decoy poses compared to the native pose. Only a really complete function would be perfect in both aspects. Such an ideal function is unlikely to be developed in the near future and as mentioned in section 7.2 even functions that can produce good approximations are computationally too demanding to be used in an initial screening of large compound libraries. Therefore, functions intended to exhibit high *docking power* should focus on terms discriminating near-native from decoy geometries and they must not be trained with respect to affinities. In contrast, *scoring power* (which is a measure of completeness) should be the key value while developing a target function for a docking engine. *DSX* is particular suited for high *docking power*, as it is not designed to calculate binding energies, but relies on likelihoods for given geometries. It is likely that this complements the functions typically used in the docking process, which explains the generally high *docking power* of knowledge-based scoring functions as an effect of consensus scoring.

The influence of affinity correlation on *ranking power* is more straight forward: Of course, high *scoring power* implies high *ranking power*. However, Table 10.3 suggests that for different targets, different scoring functions are most suitable. This is again a consequence of the incompleteness of the scoring functions used nowadays. Functions intended to optimize *ranking power* should therefore be tailored towards a specific target or they should offer an option to train them for a target. The weightings for the different scoring terms in *DSX* allow at least for a moderate training with respect to a specific target.

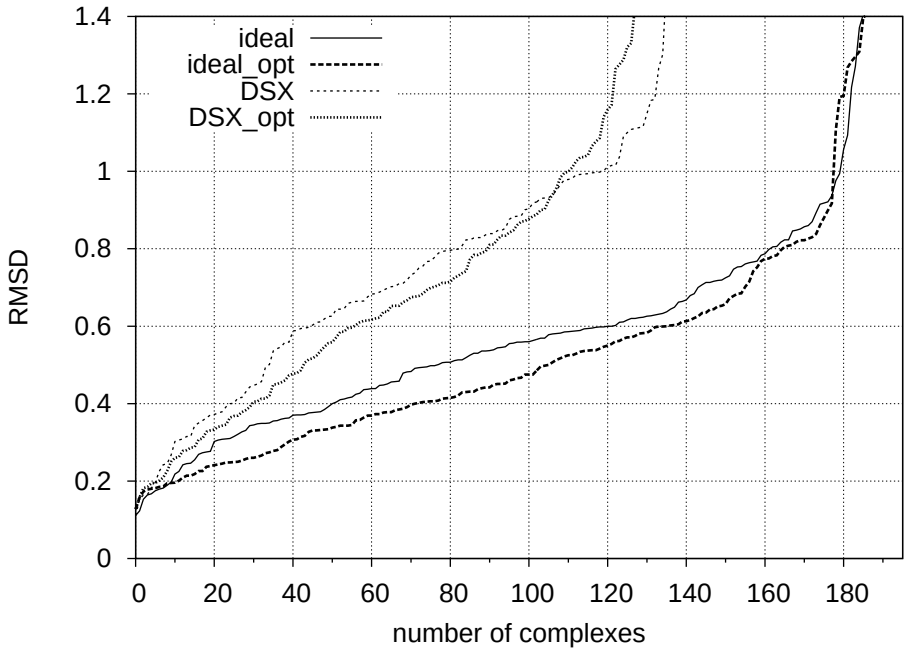


Figure 10.6.: Primary test set docking solutions ordered by increasing RMSD values. ideal: this curve corresponds to the geometry closest to the native pose; ideal_opt: also the geometry closest to the native pose, but after minimization of all docking solutions; DSX: the poses ranked on first place by DSX^{CSD}::All; DSX_opt: the poses ranked on first place by DSX^{CSD}::All after local minimization

10.4. Influence of Local Minimization

Figure 10.6 shows the influence of local minimization applied to the primary test set.

Both, the RMSD of the best poses and the RMSD of the poses ranked on first place by *DSX*, slightly improve as long as the starting geometry has an RMSD of less than 1 Å. Beyond this threshold, the results obtained after minimization get worse compared to the case without minimization. This observation reveals information about the typical size of a potential valley on the *DSX* score landscape, at least about the valley where the native pose resides. Poses with an RMSD larger than 1 Å usually give rise to minimization into different local

minima.

Unfortunately, the ranking with *DSX* becomes worse when applying local minimization to the native poses (Table 10.2) and even the affinity correlation decreases (Table 10.4). A possible explanation might be the incompleteness of *DSX*. For example, proper geometry of H-bonds is only implicitly considered to some degree in the sum of pair potentials. Thus, during a minimization the contact distances may be optimized at the price of unrealistic H-bond angles. Furthermore, the weightings of intra- and intermolecular distance-dependent potentials and torsion angle-dependent potentials are not trained on affinities.

The largest improvement upon minimization in the functions native "energy" landscapes is achieved by scoring functions of Gold, Glide and Sybyl that are also used as target functions during the docking process (Table 10.4 and Table 10.2). This is not surprising, because especially the improvement in affinity correlation upon minimization is a measure of the completeness of the used scoring function and as suggested above, completeness is a key feature of target functions used for docking.

10.5. Runtime Performance

Table 10.5 gives some information about the required runtime on the primary test set for DSX^{CSD} compared to $\mathrm{DrugScore}^{\mathrm{CSD}}$. To apply the DrugScore Surf potentials (SAS potentials), it is necessary to precalculate binding pockets. For this purpose, DrugScore is bundled with a program named CalcPocket. This program was used to precalculate the 7 Å pockets (with complete residues) around the ligands for each protein respectively. Both measurements for DrugScore, Pair and PairSurf, were performed with these binding pockets, whereas for *DSX* the complete and unmodified protein structures were used. For the calculations including a solvent accessible surface term, *DSX* is faster by a factor of 15.7 compared to DrugScore even without considering the time needed by CalcPocket. Also in case of scoring based on pure pair potential evaluations, *DSX* is significantly faster, although DrugScore uses smaller input

Table 10.5.: Comparison of runtime for DrugScore and *DSX* on the primary test set.

Scoring function	runtime in seconds
DrugScore^{CSD}::CalcPocket	249
DrugScore^{CSD}::Pair	67
DrugScore^{CSD}::PairSurf	3779
DSX^{CSD}::Pair	50
DSX^{CSD}::PairSR	241
DSX^{CSD}::Pair-Opt	3921

structures and *DSX* runtime includes full atom type perception for protein and ligand structures. The last row in the table corresponds to the *DSX* runtime with the built-in local minimization. All values were measured on an Intel Core2Duo E6600 (2.4 GHz).

11 Chapter 11. Exemplary Applications of DSX Potentials

Apart from the program *DSX* itself, the described pair potentials were successfully integrated into many other tools, with promising results, but also implications for further development. Some of these applications will be described in the following sections.

11.1. The Program HotspotsX

Up to this point, only molecular docking and rescoring was presented as a tool for structure-based drug design. Of course, there are many other computational approaches that can be valuable in this process. One of them is the generation of so-called hotspots of binding, that are regions within a binding pocket that favor particular functional groups. Such hotspots can be useful in several applications. For example, they can be a starting point for protein-based pharmacophore models or they give hints on how to expand a lead structure in direction of unoccupied parts of the binding pocket.

A simple method to generate such hotspots is to calculate a discrete scalar field of energy values that can be visualized as isocontour map. Popular examples for the latter are the programs GRID (Goodford, 1985) or SUPERSTAR (Verdonk

et al., 1999).

For the same purpose, the program *HotspotsX* was developed along with *DSX*. It puts the protein or just a binding site into an evenly spaced rectangular grid and calculates a *DSX* pair score for a probe atom on each grid point.

The program has the same interface as *DSX*, hence it also accepts PDB or MOL2 files and is able to take, e.g., cofactors as additional input. By supplying a so-called mapfile, the user can specify which probe atoms shall be used. Furthermore, one can assign the same name to different probe atoms and as a result contour maps will be generated as a combination of these different probes. The program uses a default grid spacing of 0.5 Å, which can also be adjusted by the user. Another highly recommended option is Gaussian smoothing. It reduces the noise without significant changes in the relevant parts of the contour maps.

The output of *HotspotsX* is an ascii contour (ACNT) file for each probe atom. The data within these files can be visualized for a defined isocontour level. A possible program for visualization is PyMOL[1].

Many promising results from the application of *HotspotsX* can be found in Ritschel et al. (2009); Behnen et al. (2012); Craan (2011).

At this point one representative example will be used to outline both, the usefulness in lead extension and an implication for further development. Figure 11.1 shows the human cAMP-dependent protein kinase (PKA) in complex with a fragment-like ligand. A *HotspotsX* calculation was applied to the empty binding pocket, using Gaussian smoothing, a grid spacing of 0.5 Å, and aromatic carbon as probe atom. The resulting map is displayed at an isocontour-level of 65 % of the global minimum within the pocket. Obviously, the ligand's phenyl ring and also its hydrophobic methyl group fit very well into the predicted hotspot field in the preselected part of the binding site. Furthermore, it can be seen that aromatic carbons in another, yet empty part of the pocket are predicted.

Figure 11.2 shows the same binding site with a larger ligand that can be interpreted as an extension of the fragment shown in Figure 11.1 (the methyl

[1]The PyMOL Molecular Graphics System, Schrödinger, LLC.

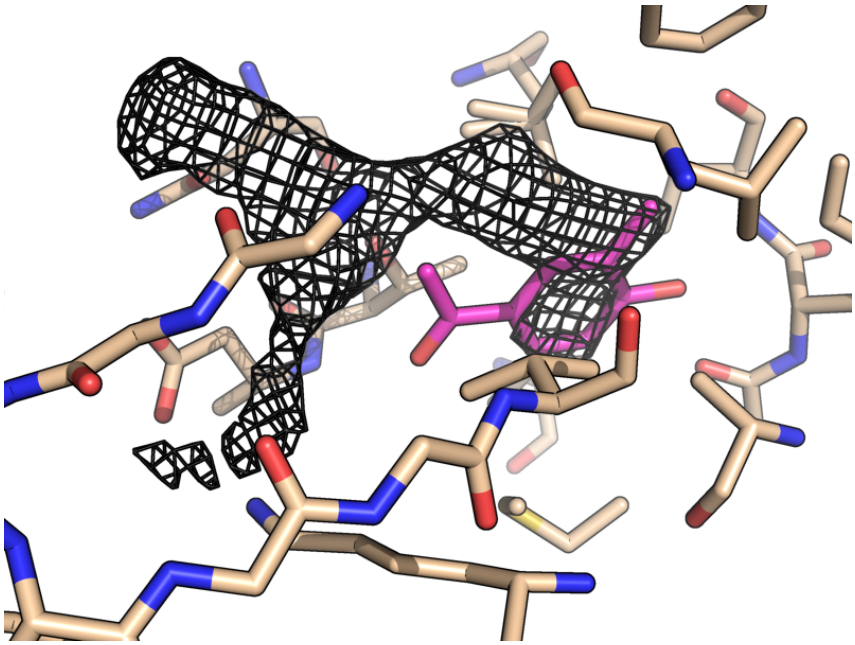


Figure 11.1.: The human cAMP-dependent protein kinase (PKA) in complex with a fragment-like ligand (magenta, PDB code *'3oog'*). The isomesh shown in black corresponds to an isocontour-level of 65 % with respect to the global minimum (best value) for *C.ar*6 as probe atom.

group was replaced by a similarly hydrophobic bromine). The second phenyl group of this ligand is located in a part of the pocket that already showed a favorable hotspot field for aromatic carbons, even without considering the fragment.

However, when looking at the picture, one might ask an apparent question: Why is the second phenyl ring not rotated to that part of the pocket that is shown in the upper left of Figure 11.2, as the hotspot field seems to be more pronounced in that area? A first idea might be that in case of a rotation, the primary amine would be located in a less favorable position. Actually, consulting the hotspot field for primary nitrogens in Figure 11.3, the opposite seems to be the case, because it indicates a favorable nitrogen position at one edge of the second phenyl.

Figure 11.4 shows the complex from the same direction that was used in

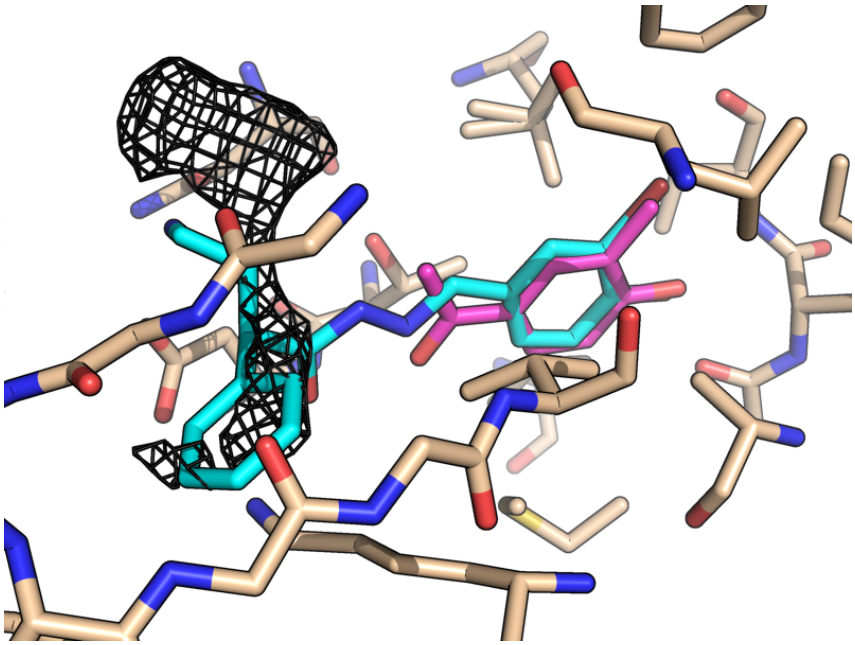


Figure 11.2.: PKA in complex with a ligand (cyan, PDB code *'3p0m'*) that can be seen as an extended version of the fragment shown in magenta. The isomesh shown in black corresponds to an iso-level of 65 % with respect to the global minimum (best value) for *C.ar*6 as probe atom. The fragment (magenta) was considered as part of the protein for hotspot calculation.

Figure 11.3, but this time together with the protein surface. On the left-hand side, the usual Connolly-surface is displayed, and on the right-hand side the protein is shown with solvent accessible surface (calculated as vdW-surface where all vdW-radii were increased by 1.3 Å). The described more pronounced hotspot field for aromatic carbons is completely solvent exposed, while the region of the second phenyl ring is not solvent accessible. That might explain the experimentally observed orientation of the phenyl and at the same time the preferred location for the hydrophilic amino group.

This observation suggests a necessary enhancement of *HotspotsX*. Recalling the findings in section 8.5, one could also consider the *SR* potentials in the hotspot calculation. This should correct for effects that are related to favorable or unfavorable solvent exposure of certain parts of a ligand in the binding pocket.

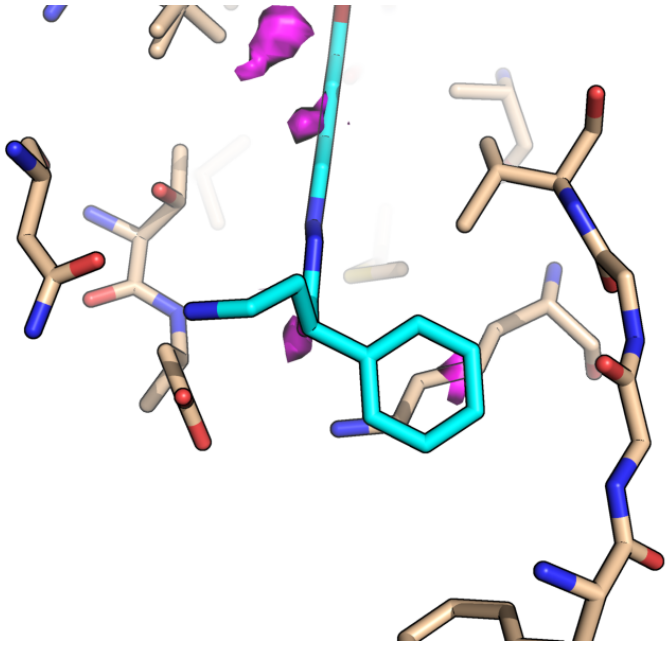
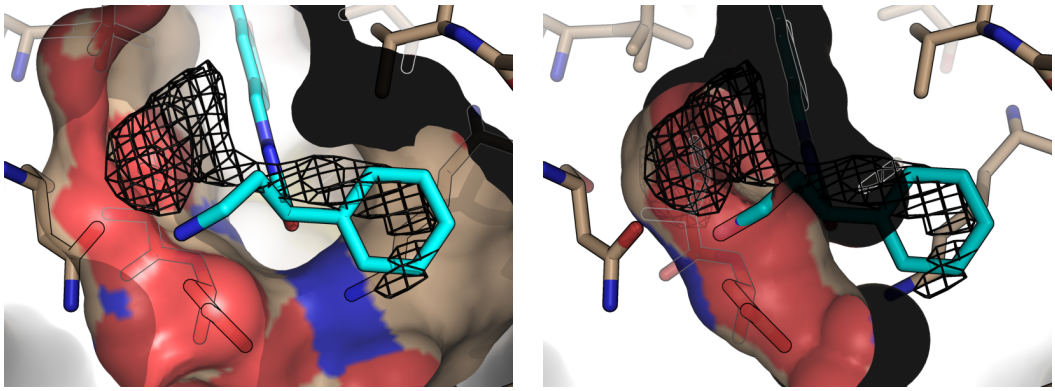


Figure 11.3.: PKA complex *'3p0m'*. The isosurface shown in magenta corresponds to an iso-level of 40 % with respect to the global minimum (best value) for $N.3p$ as probe atom.



(a) **(b)**

Figure 11.4.: PKA complex *'3p0m'*. The isomesh for aromatic carbons is shown in black. (a) Connolly-surface of the protein. (b) solvent accessible surface (vdW-radii plus 1.3 Å) of the protein.

Finally, another possible improvement of the hotspot calculation should be discussed as a perspective of future research. Currently, *HotspotsX* applies the pair potentials

$$Score(c,r) = -ln\left(\frac{\rho(c,r)}{\rho_{ref}}\right) \tag{11.1}$$

with the density functions as they are defined in Equation 8.15 and Equation 8.16. Given a grid position $\vec{q}$ and a probe atom type l, the hotspot score is calculated as

$$HS_{score}(l,\vec{q}) = \sum_{a\in P} Score(c(p(a),l), r(a,\vec{q})) \tag{11.2}$$

where P is the set of all protein atoms. The scores for atom-atom pairs that are summed up in Equation 11.2 are likelihoods that depend on the contact type c and the distance r.

What might be much more interesting in case of a hotspot analysis is a likelihood for a single atom (the probe) that depends on atom type l and position $\vec{q}$ in the grid. To clarify the difference between the latter and the definitions in Equation 11.1 and Equation 11.2, the new definition is given in the following:

$$HS_{score}(l,\vec{q}) = -ln\left(\frac{\sum_{a\in P}\rho(c(p(a),l), r(a,\vec{q})}{\sum_{l'\in T}\sum_{a\in P}\rho(c(p(a),l'), r(a,\vec{q}))}\right) \tag{11.3}$$

where T is the set of all possible ligand atom types.

In Equation 11.2, all likelihoods are considered to be independent and summed up. Furthermore, all possible contact types are considered in the reference, regardless whether they can occur on position $\vec{q}$ or not.

Equation 11.3, in contrast calculates a single likelihood where the sum of probabilities for atom type l is set in relation to the sum of probabilities from all contact types that are possible at the exact position $\vec{q}$. From a theoretical point of view, it would be more reasonable to multiply the probabilities instead

of summing them up. However, this could lead to highly unstable results, due to the higher impact of individual density functions that may be erroneous.

11.2. The Programs TransCent and DSX_rota

Enzyme design is an important technology to alter function and stability of known enzymes or even to generate completely new enzymes from first principles, with the aim to create highly specific and efficient biocatalysts. An improved activity or possible catalysis of a novel substrate can often be achieved by small changes of a well-known active site, e.g. by site-directed mutations (Toscano et al., 2007). During the last years, computational methods were developed to assist the design and first success stories of this novel approach have been reported (Pinto et al., 1997; Bolon and Mayo, 2001; Röthlisberger et al., 2008).

The program *TransCent*, developed by Fischer et al. (2009), is an enzyme design tool that was created for the transfer of active sites from known enzymes to alternative scaffolds. It utilizes RosettaDesign (Kuhlman and Baker, 2000) for modeling and optimization, but extends the energy function by features that are important for enzyme design.

Relevant aspects for the optimization of transferred active sites are protein stability, ligand (substrate) binding, pKa-values of the residues, and structural features. *TransCent* is implemented modular, hence there is one module for each of the aforementioned tasks.

As a part of the ligand binding module, the program *DSX_rota* was developed to score side chain conformations. Its input is a list of possible side chain rotamers. Given a ligand conformation, *DSX_rota* evaluates these rotamers on the basis of *DSX* pair potentials and thus assists the optimization of side chain conformations.

In a validation, *TransCent* was able to recapitulate a considerable fraction of active site residues for a given template, demonstrating its usefulness for enzyme design. These results were obtained, using an older version of *DSX* pair

potentials, but the program is under continual development and meanwhile utilizes the most recent *DSX* potentials that are described in this thesis.

11.3. The Program DSFP

It has already been mentioned that currently no scoring function is able to calculate accurate binding energies. The best one can expect is to obtain a relative ranking of different docking poses and/or different compounds.

If two docking poses appear in essentially the same binding mode, they address the same protein residues and thus, a similar number of comparable atom-atom contacts is evaluated. Even if the scores for individual contacts do not have a good correlation with the actual energies, there is a good chance that the relative differences allow for a proper ranking. If in contrast rather different binding modes are evaluated, the chance for a reliable ranking is much lower, because different contact types must be compared with one another and also a different number contacts.

In case of large ligands there are usually not many degrees of freedom to allow for several entirely different binding modes within a binding pocket. At least, if different modes are possible, it is likely that there are significant differences in binding energies between these binding modes. The higher the differences are, the better becomes the discrimination of the correct binding pose when using an imperfect scoring function. This could be an explanation why even simple pairwise additive scoring functions like *DSX* yield good results in case of average sized ligands.

However in case of small, e.g. fragment sized ligands, the situation is different. For such small molecules there may be two or more different positions within a binding pocket where a particular fragment can be placed with ideal interaction distances to the protein atoms. In other words, they can address very different residues without experiencing further restraints or creating unfavorable clashes. The *DSX* pair potentials are rather sensitive with respect to non-ideal distances. But comparing different contact types, all of them with ideal distances, is a

much more challenging problem.

Fortunately, when studying a particular target, it is often possible to use the knowledge from known binders. Often, a particular binding pattern is found among the different binders. The program presented in this section aims at a combination of *DSX* pair potentials and a method to assess such binding patterns. Thus, it is a scoring function that can be trained with respect to a given target, if a set of experimentally determined structures with known ligands is available.

Therefore, an alternative approach was developed that ranks ligand poses not according to the sum of pair potentials, but according to the difference of two vectors. One vector, the so-called consensus reference vector, is generated from a set of known X-ray structures for a target of interest. Then for each docking pose, query vectors are generated by the same formalism and compared to the reference by means of appropriate distance metrics. In validation studies the Manhattan distance turned out to be most valuable.

As the vectors can be interpreted as target specific fingerprints, the developed program was called *DSFP*, which stands for DrugScore-FingerPrints.

11.3.1. Fingerprint Generation

First, the generation of the reference fingerprint will be described. Consider a set of crystallographically known complexes K for a given target of interest. In an initial step a consensus set of protein atoms is determined that consists of those atoms that have a distance $< 6\,\text{Å}$ to at least one ligand atom in one of the $k \in K$. The result is the set of protein binding site atoms. As the same binding site atom may have different coordinates in the different complexes, $P_k = \{p_1, \ldots, p_i\}$ will denote the binding site atoms of one particular complex k ($|P_k| = \text{const} \forall k \in K$). Depending on the atom type of a protein atom p, there is a set of possible contact types $C_p = \{c_1, \ldots, c_j\}$ (more precisely a set of contact type clusters, see section 8.3). Given a particular binding site atom p, for each contact type c a *DSX* score is calculated by summing up the scores for all contacts of type c between p and all ligand atoms $l \in L_k(c)$ that fit to

the contact type. Note that the latter might also be an empty set, if no ligand atom is present that leads to the particular contact type. The concatenation of the scores leads to a vector $v(p) = \{v_1(c_1(p)), \ldots, v_j(c_j(p))\}$ for each p, where an element $v_x(c_x(p))$ is defined as

$$v_x(c_x(p)) = \frac{1}{|K|} \sum_{k \in K} \sum_{l \in L_k(c_x(p))} Score_{pair}^{DSX}(c_x(p), r(p,l)) \tag{11.4}$$

Finally, the concatenation of all $v(p)$ leads to the reference fingerprint $V = \{v(p_1), \ldots, v(p_i)\}$. If $|K| > 1$, it is called the consensus reference fingerprint.

The aim of the *DSFP* approach is to easily filter out ligand poses with different binding modes. However, using the pair potentials that are based on the highly specific *fconv* atom types would also result in high differences for identical binding modes but different atom types, due to different contact types that are related to these atom types. Therefore, *DSFP* uses the pair potentials that are based on pharmacophoric atom types (see section 9.1). Thus, two different compounds that address the same protein residue with different atoms that have the same pharmacophoric (interaction) type, will lead to identical fingerprints.

The query fingerprint is generated accordingly, hence the only difference is that it is not averaged over all k.

11.3.2. Results

In several validation scenarios, *DSFP* achieved high enrichment rates of known binders within a set of many non-binders. Here, it clearly outperformed *DSX*. In case of trypsin, 59 X-ray structures of known binders were available. The reference fingerprint was generated from only three of them (the most dissimilar) and then all 59 binders were docked together with 1800 assumed non-binders from the NCI diverse data set. In the subsequent *DSFP* ranking, there were only three "non-binders" among the first 62 ranks. The trypsin binding affinity of two of them could be determined experimentally as $K_i = 465\,\mu M$ and $K_i = 566\,\mu M$ respectively. In Figure 11.5 the corresponding ligands are shown.

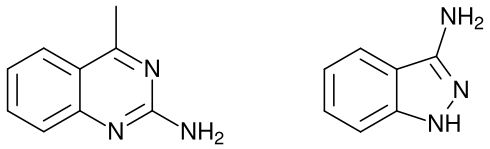


Figure 11.5.: Two fragment hits from a trypsin enrichment study using *DSFP*.

Considering their low fragment-type size, they can be seen as binders, though having relatively low binding constants.

Development and validation of *DSFP* was performed in an equally contributing cooperation with Dr. Patrick Pfeffer. More details about the method and validation results are documented in his PhD thesis (Pfeffer, 2009). It has to be noted that the results of this study have been obtained using an older set of *DSX* potentials. Studies with the most recent set of potentials are subject of ongoing research.

Another example of successful application of *DSFP* for the target TGT (tRNA-guanine transglycosylase) can be found in Ritschel (2009).

11.4. The Program DSX_wat

11.4.1. Introduction

The incorporation of solvent, especially water in the evaluation of given protein-ligand complexes is a very important, but yet not satisfactorily solved task. Water is a very special molecule; it can serve as a donor and acceptor at the same time, forming multiple hydrogen bonds. Therefore, water molecules mediating interactions between protein and ligand are frequently found in many crystallographically determined protein-ligand complexes. Furthermore, the ability to form hydrogen bonds in tetrahedral geometry allows for stable networks between multiple water molecules that can also have a partial order. Apart from translational degrees of freedom, water molecules can rotate and switch between various equivalent hydrogen bonds. This results in an exponential number of energetically similar (hence equally probable) microstates

and thus in a high entropy. It is the basis for the classical hydrophobic effect, hence the gain of entropy if the contact surface between water molecules and hydrophobic parts of a molecule is reduced upon complex formation (the water molecules that are part of the local solvation shell are ordered and can form less different hydrogen bonds). This effect is one of the driving forces in protein folding and it can also be a major contribution to Gibbs free energy upon ligand binding.

There are two generally different approaches for the incorporation of solvent effects in computational methods. The first is to use a continuum solvent model. In this case, not individual water molecules are considered, but the solvent is regarded as a homogeneous medium that is characterized only by its dielectric constant ϵ. The electrostatic part of solvation free energy can then be calculated by solving the Poisson-Boltzmann (PB) equation that corresponds to the solvent-solute model. As solving the PB equation is computationally quite demanding, more frequently a popular approximation, the Generalized Born (GB) model, is used. A very important part of solvation free energy is missing in the values calculated by the PB or GB methods, because they do not account for the entropic part that corresponds to the hydrophobic effect. This part is often approximated by an empirical term that depends on the size of the solvent accessible surface area (SA). Thus, many molecular mechanical calculations utilize the combination PBSA or GBSA to estimate the total solvation free energy.

The second approach for the incorporation of solvent effects is to consider solvent molecules explicitly. In molecular dynamics (MD) simulations for example, the system under consideration can be placed into a box of water molecules that are described by an appropriate force-field model (e.g. TIP3P or TIP4P (Jorgensen et al., 1983)). The electrostatic part of solvation free energy is calculated for all water molecules using Coulomb's law to describe the attractive interactions and Lennard-Jones potentials to incorporate repulsive interactions.

11.4.2. Motivation

A knowledge-based approach to estimate the hydrophobic effect was already presented in terms of the SR potentials introduced in section 8.5. What is still missing is an estimation of directed electrostatic interactions between water molecules, the protein, and the ligand.

The observation that water hotspots calculated with *HotspotsX* (see section 11.1) often coincide very well with crystallographically determined water positions lead to the idea of a program that takes a given protein-ligand conformation as input and calculates a water network around the ligand that is optimal with respect to the *DSX* pair potentials. The generated network should then be used to improve the rescoring especially of those complexes where interactions are mediated by water molecules. Furthermore, hydrophilic parts of the binding pocket that are not properly filled by the ligand should be filled with water molecules. As a result of these considerations, the program *DSX_wat* was developed.

11.4.3. Method

The strategy of the proposed method is to generate a consistent water network that (i) includes all structurally conserved waters and (ii) fills empty space within the binding site as effectively as possible.

The first part of the method only aims at reducing the number of putative water positions, to speed up the computationally more demanding part. Given a protein-ligand complex, the work flow is as follows:

1. An evenly spaced rectangular grid embedding the binding site is generated. Currently a grid spacing of $s = 0.5\,\text{Å}$ is applied.

2. All grid points that clash with protein or ligand, as well as all grid points that are not in contact (distance $> 6\,\text{Å}$) with any ligand atoms are removed.

3. The *DSX* score for each grid point is determined as described in section 11.1, using the atom type $O.h2o$ as probe.

4. All grid points are removed that have a *DSX* score worse than a predefined threshold.

5. To overcome the discretization from the applied grid, the position of each grid point is locally minimized with respect to the *DSX* pair potential. In the following, these optimized grid points will be called 'water points'. Powell's method (Press et al., 2007a) is applied for the minimization. An "elastic" penalty term is used to avoid that a water point moves further than $\frac{\sqrt{3s^2}}{2}$ Å in space, which corresponds to half of the diagonal between the initial grid points.

6. The water points are clustered in a way that no two points within a cluster exhibit a distance $> \frac{\sqrt{3s^2}}{2}$ Å.

7. For each cluster the best point with respect to the *DSX* score is determined and all other points of the cluster are removed.

8. To each remaining water point w, a list $w.C$ of clashing water points is associated, that is all water points that have a distance < 2.4 Å to w.

9. To each water point w, a second list $w.N$ of neighboring water points is associated, that is all water points that have a distance < 4 Å and that are not in $w.C$.

All steps described so far are rather fast to calculate. Even the minimization is no complex task, because water is treated as a point and thus has only three degrees of freedom.

A simplified set of putative water points is shown in Figure 11.6. Positions 2 and 3 are too close to each other and thus both positions cannot be occupied by a water at the same time. The same holds true for positions 4 and 5. The task is to find a mutually exclusive set of positions that maximizes the total *DSX* score.

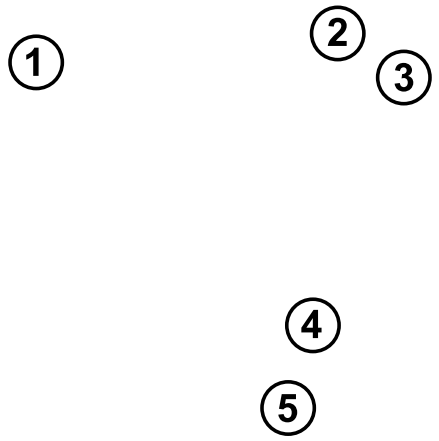


Figure 11.6.: Putative water positions. The positions 2,3 and 4,5 are in clash distance (short contacts $< 2.4\,\text{Å}$) to each other. Possible exclusive sets are '124', '134', '125', and '135'.

In the first implementation of *DSX_wat* this problem was formulated as the problem to find maximum cliques of a graph $G(V, E)$. Hence the water positions were represented as vertices and two vertices were connected by an edge if their distance was larger than the clash distance of 2.4 Å. For very small systems this strategy worked properly, but for larger binding pockets or even complete proteins, dense graphs with up to millions of edges were generated, resulting in unacceptable time scales for the computational evaluation of such systems. In an improved version the pocket was partitioned into several smaller parts. Then cliques for these smaller parts were calculated and the best among them were merged subsequently. The aggregation of these locally best cliques was again formulated as a clique problem. In comparison to the clique search for the complete system, the latter has to be considered as a heuristic approach, because only best cliques from each part are taken into account (loosing some points that could otherwise be a part of the total best clique). It was not possible to find a good balance in the parameters that would allow for both, acceptable runtimes for each input and results that are close to the result of a complete clique search.

Therefore, an alternative heuristic approach had to be developed. It should

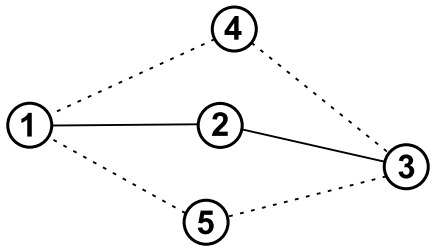


Figure 11.7.: Two possible water clusters. '1,2,3' is preferred, if '2' has a higher score compared to '4' and '5', even if '1,3,4,5' has a better total score.

be straight-forward to control by only a few parameters and at the same time it should yield good results in acceptable runtime complexity. The idea was to formulate the task in terms of a clustering problem. A first thought in that direction was to use a distance function that results in an infinite cluster distance for clashing water positions and in all other cases a cluster distance that is inversely proportional to the sum of *DSX* scores of the water positions. However, a usual hierarchical or K-means clustering would always prefer individually high scored water positions over proper space filling networks. In the example shown in Figure 11.7, there is a clash '2,4' and a clash '2,5'. The cluster '1,2,3' would be preferred by the clustering, if the score for '2' is higher than that for '4' and '5' and the scores of the interactions '1-2', '1-4', '1-5', ..., are in a similar range.

In consequence, a modified clustering approach was implemented. It works agglomerative and initially allows for a particular water point to be part of several different clusters. The corresponding algorithm is shown in Alg. 11.1. Initially, each water point corresponds to one cluster and at the same time it is a possible extension point of the respective cluster (line 6). For each cluster (line 8), the set of possible extension points is used (line 10) to iterate over all neighboring points. If such a point does not clash with one of the cluster's elements, the cluster is extended by the new point (lines 12,13). Furthermore, each new point becomes an extension point for the next cycle of the main loop in line 7.

The sets of neighboring water points are sorted with respect to their *DSX*

Algorithm 11.1: A heuristic algorithm to determine optimal water clusters.

Input : A set of putative water points W
Result: A list of water clusters R, ordered by total *DSX* score.

```
1  Let w.N be the set of neighboring, but not clashing water points for w ∈ W,
   sorted by their DSX score
2  Let w.C be the set of clashing water points for w ∈ W
3  Let r.W be the set of water points of a cluster r
4  Let r.E be the set of water points that were last added to r
5  R := {}
6  forall w ∈ W do R := R ∪ {r(w)}                    // put w into r.W and r.E
7  while ∃r ∈ R|r.E ≠ {} do
8      forall r ∈ R do
9          E := r.E; r.E := {}                          // ordered by DSX score
10         forall e ∈ E do
11             forall n ∈ e.N do
12                 if (n ∉ w.C∀w ∈ r.W) ∧ (n ∉ r.W) then
13                     r.W := r.W ∪ n; r.E := r.E ∪ {n}
14     REMOVE-DUPLICATES(R)
15     REDUCE-NON-ISOLATED(R)
16 MERGE-ISOLATED(R)
17 MINIMIZE(R)
18 SORT(R)
```

score before the algorithm starts. Thus, the best neighboring points are always added first to a cluster. In addition, they become also the first extension points that will be evaluated in a subsequent cycle of the main loop in line 7.

Referring back to the example shown in Figure 11.7, this would mean that starting from the initial cluster with point '1', the final cluster '1,2,3' would be found. The same final cluster is generated when starting either with '2' or '3', respectively. However, starting with '4' and '5' the final clusters will be twice '1,3,4,5'. The possibility of identical clusters even exists before the algorithm terminates and thus implies the removal of such duplicates (line 14).

In case of a higher number of putative water points, the step in line 15 of Alg. 11.1 is the crucial part to guarantee acceptable runtimes. Here, a

Algorithm 11.2: The MERGE-ISOLATED function that is utilized by Alg. 11.1.

Input : The set of final clusters R after the main loop of Alg. 11.1
Result: A list merged clusters R, ordered by total *DSX* score.

```
1 Let r.C be the set of clusters that are not isolated from r (clashing clusters)
2 SORT(R)                                        // with respect to DSX score
3 forall r ∈ R do
4     forall e ∈ R ∧ ¬e ∈ r.C do
5         r := r ∪ e
6         r.C := r.C ∪ e.C
```

percentage of clusters is removed from R, particularly those clusters which achieve the worst total *DSX* score. Note that the total score of two water points is not simply the sum of their initially determined scores, but in addition the score for the contact between these two points is regarded. The percentage starts with a value that depends on the number of initial water points $|W|$ and grows linearly with the number of cycles that are performed for the main loop in line 7. It is important to note that this removal is applied individually to all isolated sets of clusters. For example, there could be a cluster of three particular water points in one part of a binding pocket that is not in contact with other clusters (has no neighbors). It clearly would achieve a lower total score compared to clusters which comprise much more elements and that reside in other parts of the pocket. But as it is independent (isolated) it is not removed.

If no further extension of clusters is possible, all isolated clusters are merged (line 16). The merging algorithm is shown in Alg. 11.2.

Finally, a local minimization is performed for the complete clusters (line 17 of Alg. 11.1, Powell's method). In contrast to the minimization of individual water positions (as described for the preprocessing steps), the waters also "see" each other in this minimization.

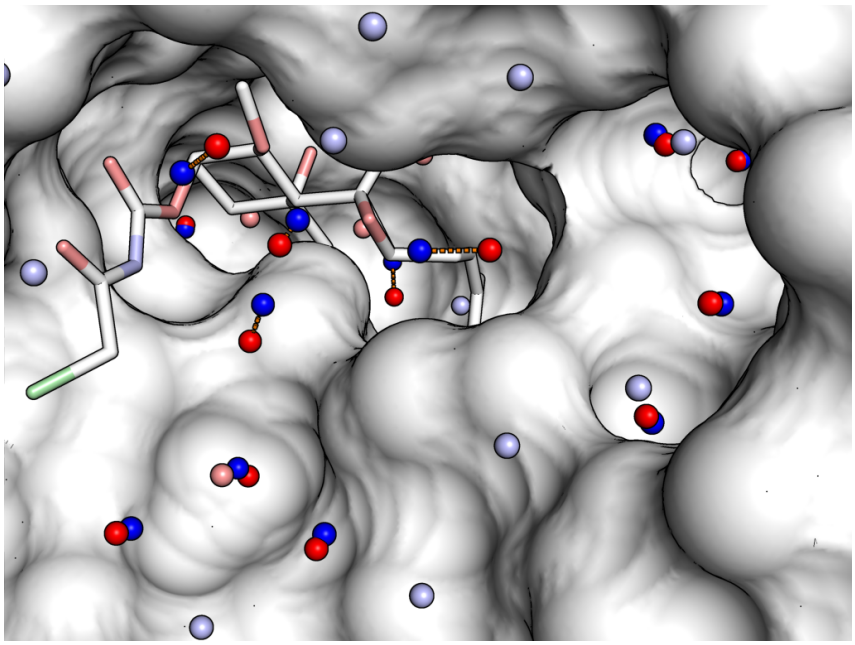


Figure 11.8.: The PDB complex *'1b6a'* with X-ray waters (red) and predicted waters (blue). If the closest distance between experimental and predicted position is smaller than 1.5Å, it is considered as a match and dark colors are used for the corresponding pair. Unmatched waters (predicted or experimental) are depicted in light red or blue.

11.4.4. Results and Implications

The development and validation of *DSX_wat* is subject of ongoing research. At this point only one example will be given that uncovers a shortcoming of the *DSX* scoring function and thus implies an option for further improvement.

Figure 11.8 shows crystallographic and predicted water molecules of the PDB entry *'1b6a'* (methionine aminopeptidase). The crystallographically determined water molecules are depicted in red and the water molecules determined by Alg. 11.1 are shown in blue. The shortest distance between a predicted and an experimentally determined water molecule defines a pair. If the distance is $< 1.5\,\text{Å}$, the pair is considered to be a match. Otherwise both water molecules are considered unmatched in the following.

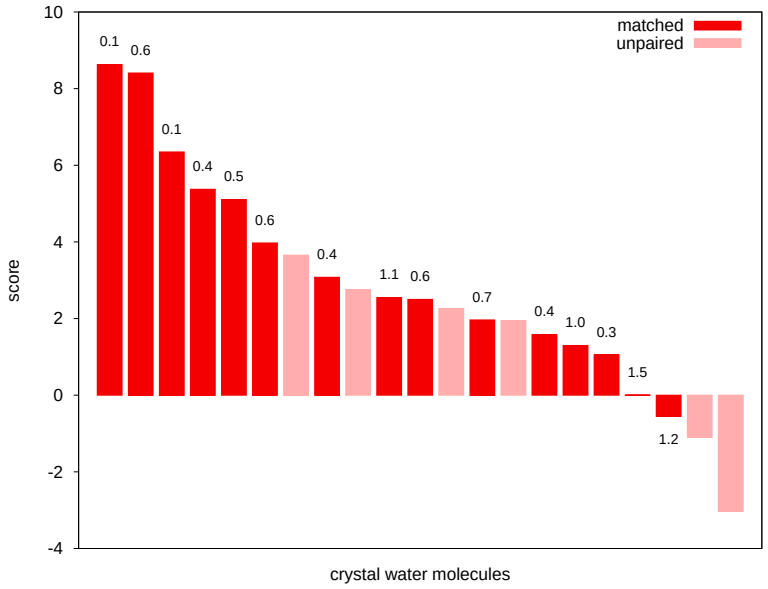


Figure 11.9.: X-ray waters and their *DSX* score (multiplied by −1). Dark red is used if the corresponding water is matched by a predicted water. The numbers above the boxes correspond to the distance between matched and predicted water (in Å). This figure was prepared by Michael Betz, who continues the *DSX_wat* project.

Figure 11.9 shows the *DSX* scores of the individual crystallographic water molecules. It can be seen that those experimental water molecules with a very favorable *DSX* score are also predicted by *DSX_wat* with a deviation of less than 1 Å. In case of the best scored water molecule, the predicted water position is found in only 0.1 Å distance.

Figure 11.10 shows the *DSX* scores of the individual predicted water molecules. The most remarkable observation in this diagram is that the best scored prediction has an extremely high value that is twice as high as the best scored experimental water. Considering the complete range of scores, this seems to be an unrealistic, rather artificial value. The predicted water position corresponding to this score is shown in Figure 11.11. It has seven perfect H-bond distances to aspartate oxygens and one to the ligand's carbonyl group. It appears unreasonable that a water molecule should form eight H-bonds at

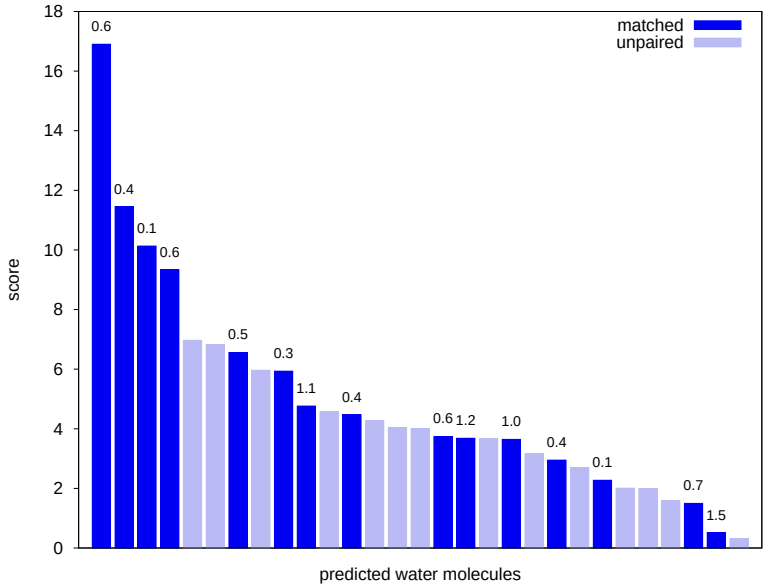


Figure 11.10.: Predicted waters and their *DSX* score (multiplied by −1). Dark blue is used if the corresponding water is matched by an X-ray water. The numbers above the boxes correspond to the distance between matched and predicted water (in Å). This figure was prepared by Michael Betz, who continues the *DSX_wat* project.

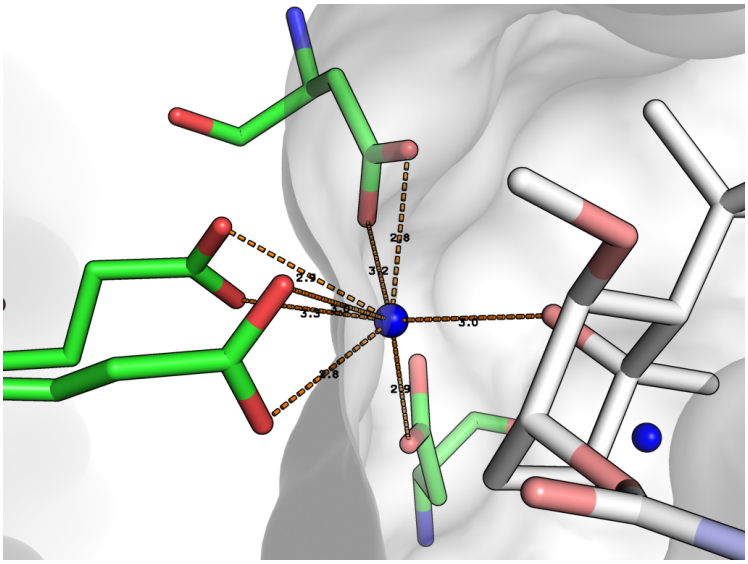


Figure 11.11.: A predicted, yet artificial water in a position that offers eight perfect H-bond distances.

the same time, in contrast, a sodium ion would fit perfectly to such a position. Of course, the water cannot form eight H-bonds at the same time (in contrast, a sodium would fit perfectly at this position).

This example implies that the *DSX* scoring function should be extended to restrict the number of possible directed interactions. For the rescoring of docking solutions, this should have not much impact, because the docking programs already account for this limitation. Otherwise, water molecules and sodium ions exhibit nearly the same diffraction power and therefore, a fair number of corresponding misassignments are present in the PDB. This deficiency of the reference data has to be taken into account. At least, for subsequent minimization in *DSX*, as well as for the water prediction, the restriction of directed interactions seems to be a mandatory extension.

12 Chapter 12. Summary and Outlook

12.1. Summary

A reliable scoring of putative protein-ligand complexes as generated by docking programs is still one of the most urgent tasks in structure-based drug design. In this part of the thesis, the development of the knowledge-based scoring function *DSX* together with an exhaustive validation was described. Furthermore, some examples for applications that originate from *DSX* were given. Major achievements concerning this work are:

- New knowledge-based pair potentials were derived, corresponding the original DrugScore formalism, but based on a novel classification scheme of atom types that was introduced in part one of this thesis. A major problem with respect to the reference state was faced in this concern. It was solved by clustering of atom-atom contact types with similar contact distance distributions. The performance of the newly derived potentials was validated on a literature known test set that also enabled a comparison with a majority of other popular and frequently used scoring functions. With respect to this test set, the *DSX* pair potentials perform not only significantly better than the old DrugScore potentials, but also better than all other scoring functions in terms of *docking power*. In the assessment of *ranking power*, *DSX* was among the best functions under

evaluation.

- DrugScore also included singlet potentials based on a solvent accessible surface term. It was shown that the original definition of these potentials has some deficiencies and an improved definition has been proposed. Using this new definition, also solvent accessible surface-dependent potentials were derived for *DSX*. In the validation, it was possible to further improve *docking power* using these potentials.

- In addition, newly defined torsion angle potentials were derived from the CSD. This was a necessary step to enable a local relaxation of given protein-ligand complexes. Furthermore, application of these potentials was shown to increase *docking-* and *ranking power* in some cases.

- Aimed at the use for generating hotspots in protein binding pockets, also pair potentials based on pharmacophoric (generic) atom types were derived. Interestingly, even this rather limited differentiation of atom types lead to improved results compared to the application of original Sybyl types. In case of *ranking power*, evidence was found that these potentials might deliver more robust results averaged over a number of different targets.

- Compared to DrugScore, *DSX* is much faster, more consistent due to the application of own atom type assignments, more flexible with respect to possible input formats, and also more robust with respect to errors in the input.

- The option to adjust the weights of *DSX* scoring terms enables the possibility to enhance the quality of the results with respect to a particular target.

- *DSX* was made freely available for the scientific community on `http://www.agklebe.de/drugscore/dsx_download.php` and has been used over the last years by many scientists world-wide.

Many additional features were included to be prepared for all demands that may arise from a given docking protocol.

- The concept of interaction modes was introduced. It enables the individual treatment of cofactors, metals and waters if necessary. If the cofactor was used as rigid part of the protein during docking, it should be handled as such in rescoring, hence only interactions between ligand and cofactor should be considered. If in contrast the cofactor was flexible in docking, then also interactions between cofactor and protein should be considered. *DSX* allows for all possible combinations of flexible/non-flexible waters, metals, and cofactors.

- It is possible to rescore covalently bound ligands.

- It is possible to consider flexible protein parts in case of docking results from Gold and AutoDock, where this flexibility was enabled.

- It is possible to rescore water that was specially treated by the docking program Gold.

- The PyMOL-based visualization that was introduced for DrugScore by Peter Block and Hans Velec was enhanced and fully integrated into *DSX*. It is especially useful to medicinal chemists to see at which positions molecules do not fit properly into a binding pocket.

Additional tools were developed, based on *DSX* potentials.

- The program *HotspotsX* was developed alongside with *DSX*. It was successfully applied in several studies of colleagues and the author.

- In a cooperation, the program *DSX_rota* was developed and became an integral part of the program TransCent, that was developed in the group of PD Dr. Rainer Merkl at the University of Regensburg.

- The program *DSFP* was developed in an equally contributing work with Dr. Patrick Pfeffer. It can be used as a fingerprint-based ranking

function that uses already known X-ray structures for a particular target to differentiate between likely and unlikely binding modes. The generated fingerprint vectors are based on pharmacophoric *DSX* pair potentials. The program was successfully applied in a virtual screening study and has been shown to be especially valuable in case of fragment-sized compounds.

- The program *DSX_wat* was developed to generate reliable water networks within a binding pocket, based on *DSX* pair potentials for water atoms. A heuristic algorithm was developed to retrieve optimal networks. First validation results showed that the predictions are in good agreement with experimental results, though further improvement of the method is necessary.

12.2. Outlook

12.2.1. DSX

The work with *DSX* and tools that are based on *DSX* potentials, implies several possible improvements for future research.

- Minimization of protein-ligand complexes with *DSX* pair potentials as target function revealed that the score often improves, while unrealistic geometries for directed interactions are generated. An explicit consideration to adjust and rank these geometries is therefore demanded.

- An observation in development of *DSX_wat* underscores related deficiencies similar to the aforementioned problem. It is not only possible that interaction distances are optimized at the cost of unrealistic interaction angles, but it is also possible that a larger amount of directional interactions is scored than an atom under consideration could actually experience. Therefore, it seems necessary to restrict the number of putative directional interactions.

- A still unresolved problem that goes along with the introduction of an increasing amount of atom types, is an appropriate handling of atom-atom contacts that are not well represented in the database. Even for well populated contact types, the potential curves show several small maxima and minima, although in theory most interaction types should be unimodal or bimodal. Especially in case of derivation from the CSD, there seems to be a periodicity in the long-range contacts that might artificially result from the periodicity in crystal packings. With respect to these problems, it might be beneficial to use a robust method to models the potentials in terms of sums of Gaussian functions that are parameterized based on the observed density functions.

12.2.2. HotspotsX

Two possible improvements of *HotspotsX* were proposed. First, the inclusion of solvent accessible surface-based potentials, and second, a new method to calculate the hotspots from *DSX* density functions instead of *DSX* pair potentials.

12.2.3. DSFP

It is an advantage of *DSFP* to include existent knowledge from X-ray structures in a screening for a particular target. But at the same time it is also a drawback, because the approach can only be used if crystallographic structures for the target under consideration are available. Furthermore, the quality of the results depends on the diversity of these known structures.

A method that determines consensus binding sites from similar targets would be able to use more input for the generation of a reference fingerprint. Finally, binding patterns for small, frequently occurring binding site motifs could be derived using the wealth of information from the complete PDB.

12.2.4. DSX_wat

One possible improvement, which addresses the limitation of possible directional interactions, was already mentioned in the outlook for *DSX*. But further improvement is required:

- The major goal of *DSX_wat* is to improve the ranking of docking solutions with *DSX*. First validations in this direction imply that it is necessary to find a proper weighting for the water interactions, because the total score for the high number of generated water molecules can easily exceed the total score of the remaining protein-ligand interactions.

- The proposed algorithm to generate optimal water networks produces promising results within a few seconds for arbitrary protein-ligand complexes. However, for the application to virtual screening even a few seconds are far to long, if thousands of docking poses should be evaluated. It is likely that the problem can be addressed by a much more efficient algorithm. Nevertheless, it will be necessary to balance out between accuracy and acceptable runtime.

- Up to now, water molecules are treated as spheres, hence the geometry of possible H-bonds is not considered explicitly. Introducing the tetrahedral coordination geometry of water molecules would complicate the optimization of the networks significantly. On the other hand it could also improve the quality and, at the same time, this would also solve the problem of limiting the number of interactions.

- In addition to consideration in the optimization process, geometric H-bond constraints could also be used to further reduce the initial water points. This would be done in linear time and save exponential time in the optimization.

Part III.

Appendix

Appendix A.

A Appendix

A.1. Modified Octree Structure for Neighborhood Partition

Processing a set of objects with three-dimensional coordinates gives rise to the problem of efficiently determining the neighborhood of an individual object. One type of such objects may be **Atom**s, but the data structure presented here is a generic implementation that supports all objects with associated coordinates. The meaning of a neighborhood $N_c(r)$ is the set of all atoms $a \in A$ that have a distance $dist(c, a) \leq r$, whereas c could be an atom itself or just Cartesian coordinates.

An important example application is the usage in *DSX*. The usual input to this scoring function is a protein file, and a file with one, up to millions of docking poses that shall be evaluated. As shown in Equation 9.2, the total pair score is calculated by the evaluation of all possible protein-ligand contacts. In a naive approach, this would mean to calculate all against all protein-ligand distances. Now consider first that all distances longer than 6 Å obtain a score of zero. Second, consider that the protein might have a diameter of more than 100 Å. Usually docking poses are cumulated in specific part of the protein, hence thousands, up to millions (depending on the number of docking poses) of protein-ligand distances would be measured with no purpose.

Of course, there maybe also docking solutions residing in different areas of the protein. It is therefore not possible to take just a subset of docking poses and discard not addressed areas of the protein. Instead, a much more general and efficient technique is applied that is also useful in other situations.

The basic idea is to use octrees (Press et al., 2007b). Compared to the literature, there are some important differences in how this well known data structure is implemented and used in context of this thesis. These modifications were applied in order to fit better to the requirements of the specific problems involved in this thesis.

First, the usual literature-known definition will be outlined briefly. To simplify illustrations, quadtrees will be used in the following. They are the two-dimensional equivalent to octrees and all explanations will be equally valid for the latter.

A quadtree structure that should store a given set of two-dimensional objects starts with a box that encloses all of these objects. In the internal representation, this box is the root of the quadtree. An example is shown in Figure A.1, where the initial box contains eight objects a,b,c,d,e,f,g,h. Now the box is subdivided into four child boxes in such a way that the child boxes are an exact bisection of the parent box in each dimension. All child boxes that contain more than one object are further subdivided, continuing until each object has its own box.

If one is interested e.g. in the nearest neighbor of a, only the neighboring boxes must be checked. If the internal data structure for the tree was chosen appropriately, traversing the tree can be implemented very efficient by accessing parents and children of a box via a hash.

However, in the context of this thesis, the nearest neighbor problem is of no relevance. Instead, neighborhoods as defined in the first paragraph of this section must be retrieved efficiently. This could also be done using the above described method and traversing neighboring boxes until the neighborhood distance is exceeded.

Actually, the octree implementation developed for *fconv* and *DSX* works differently, as described in the following.

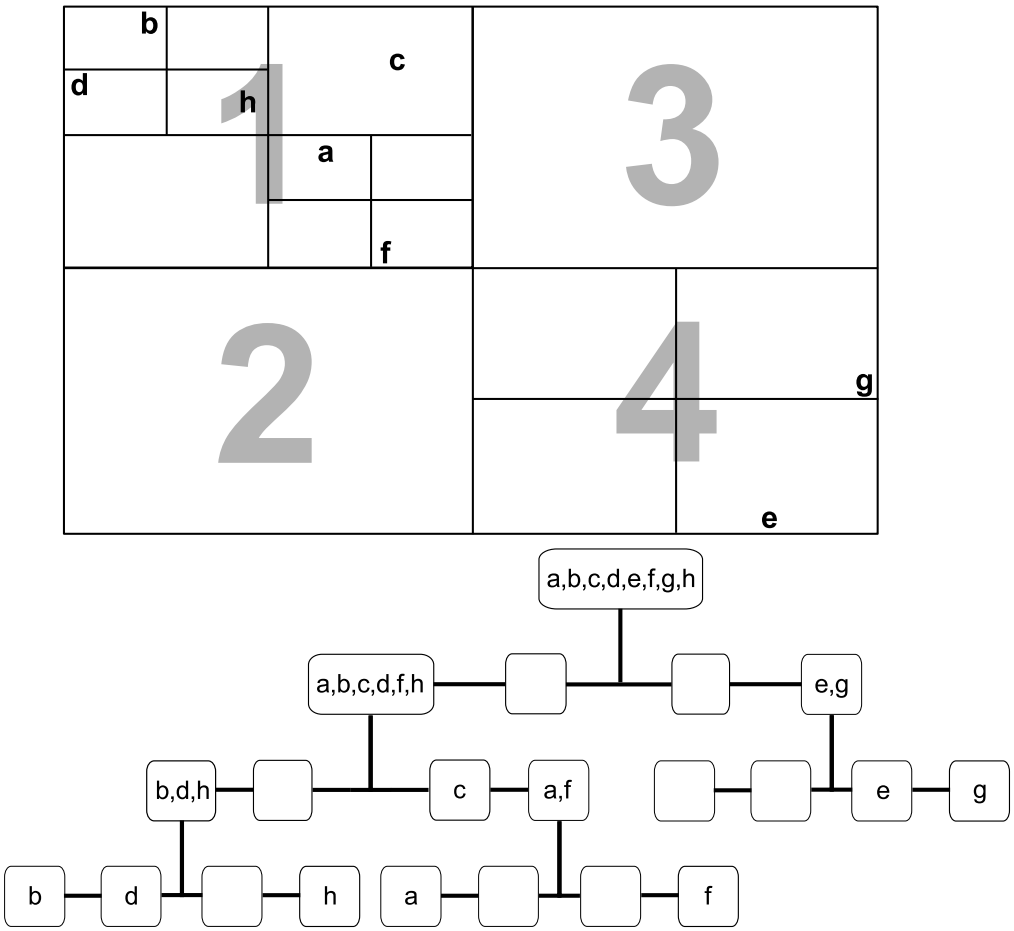


Figure A.1.: Generation of a quadtree for a set of eight objects a,b,c,d,e,f,g,h. The order of children in the tree corresponds to the numbering of the first subdivision.

A.1.1. Implementation

The octree library is a generic implemetation that is part of the *fconv* framework and hence available as open source code (see section 1.1).

An **Octree** object is constructed using a list of atoms A and the desired neighborhood radius r as input parameters. At that stage nothing else is processed, hence partition follows only at request. Considering the initially mentioned example with millions of ligands and just one protein, calculating the full tree once at the beginning or only relevant parts of it would make no

significant difference in total computation time. However, it is e.g. also possible to have a huge set of only a few ligands that were docked to many different targets. In that case it would be a waste of time to compute the full tree for each protein, although only a little part of it will be accessed.

Now the **Octree** object can return an iterator on neighboring atoms, given coordinates Q and a spacing s (or alternatively a minimum size of elements). If the demanded spacing is greater than the longest edge of the initial box, an iterator on A is returned. If it is smaller, the initial box will be subdivided (if it was not already divided in previous requests). So a first difference compared to usual octrees/quadtrees is that the boxes are not partitioned until each box contains only one element. Instead, subdivision (hence traversing) of the tree stops, if the longest edge of a box falls below a requested limit. Of course, a box is not further subdivided if it contains only one element. Alternatively, a different minimum number of elements can be specified at iterator request.

The second difference compared to standard octrees/quadtrees is that object references are not only assigned to those boxes which contain them geometrically, but also to all other boxes that are within the range r to the object. Thus, an object can be referenced by multiple boxes. An example for a neighborhood request of point Q is given in Figure A.2. In the first subdivision, a reference on f is also assigned to the second box (bottom left corner), because the distance from f to this box is $< r$. In the same way, c and f are also assigned to the third box.

Q is located in the first box and therefore only this box is further subdivided. After the third partition, the box length is shorter than the requested spacing s, and an iterator to a,c,f is returned. All subsequent requests for coordinates that are located within the same box as Q result in a,c,f without any additional subdivision. In the example, c is not really within the neighborhood radius of Q, but it might be for other points in the same box.

The computational costs for the subdivision is higher compared to an implementation as visualized in Figure A.1, because the decision to which box a particular object corresponds must be evaluated more frequently. In return, there can be thousands of following requests without the necessity to calculate

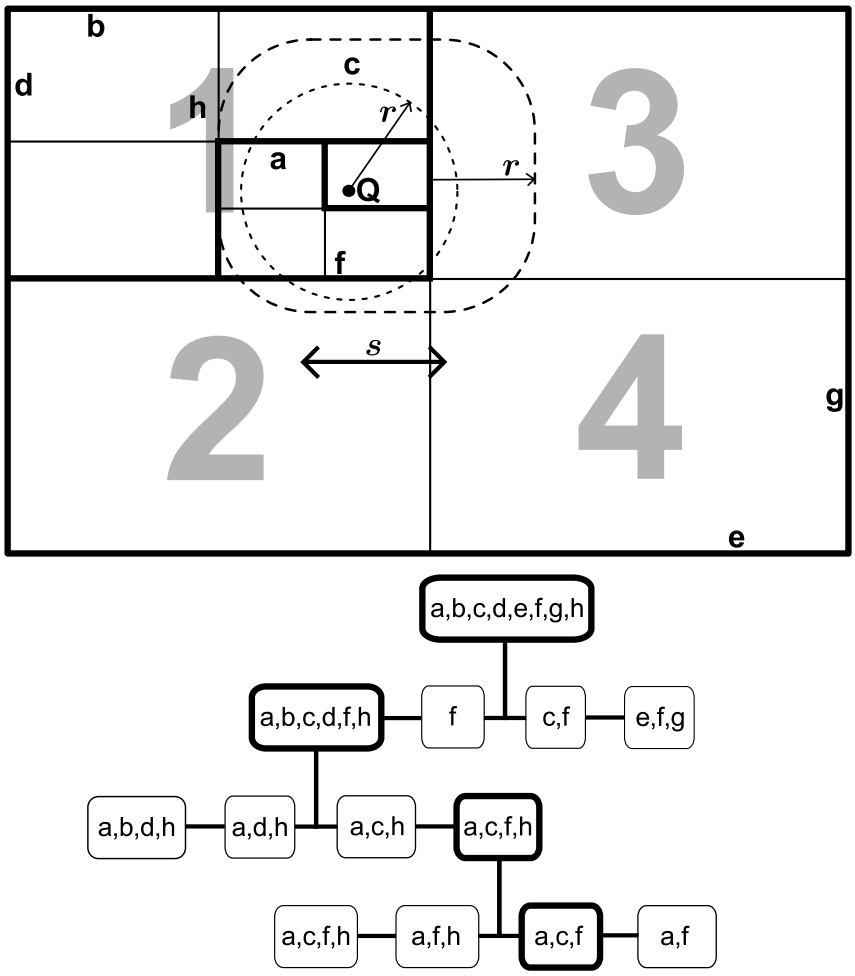


Figure A.2.: An r-neighborhood request for the coordinates Q and spacing s. The tree is traversed as marked with the thick boxes.

indices or traverse the tree for neighbors. Furthermore, the subdivision is usually performed in only a small part of the object cloud.

Last not least, also a heuristic is used. In Figure A.2, the dotted line around the final box is depicted with rounded edges, suggesting that the shortest distance between each point of the curve and the box equals r. In the real implementation it is strictly rectangular. In consequence, also points that have a distance $> r$ maybe referenced by the box. The advantage however is that no distances must be calculated to decide whether an object belongs to a

particular box or not. A simple sequence of conditional statements that compare x, y and z coordinates with the (by r extended) box boundaries is sufficient. Extending the octree library to optionally use distance calculations would be trivial. Having many neighborhood requests compared to the complexity of subdivision this might be favored, due to less neighbors with distances $> r$.

Finally, it shall be noted that usual octrees must properly handle numerical uncertainties, that are cases where an object is located exactly on a bisection line (or plane). In the modified implementation such cases are irrelevant, because the object is assigned to both boxes.

A.2. Atom Type Definitions

Table A.1.: Sybyl Atom Types used by the original DrugScore implementation

Atom type[a]	Definition
C.3	sp^3-hybridized carbon
C.2 (C.1)	sp^2- and sp-hybridized carbon
C.ar	carbon in aromatic ring systems
C.cat	carbon in amidino or guanidino groups
N.3 (N.4)	sp^3-hybridized nitrogen
N.ar (N.2)	nitrogen in aromatic ring systems and sp^2-hybridized nitrogen
N.am	nitrogen in amide bonds
N.pl3	nitrogen in amidino or guanidino groups
O.3	sp^3-hybridized oxygen
O.2	sp^2-hybridized oxygen
O.co2	oxygen in carboxylate groups
S.3 (S.2)	sp^3 and sp^2-hybridized sulfur
P.3	sp^3-hybridized phosphor
F	fluorine
Cl	chlorine
Br	bromine
I[b]	iodine
Met	calcium, iron, zinc, nickel[c]

a) Types in brackets are merged into the aforementioned type due to insufficient observations of the original type in the PDB structures. b) Iodine was only considered in $\text{DrugScore}^{\text{CSD}}$. c) Nickel was only considered in $\text{DrugScore}^{\text{PDB}}$.

Table A.2.: Internal fconv atom types and their definitions.

Atom type	Definition
H.ac	acidic H (bonded to O.3ac, N.im, N.sam or N.ohac)
H.onh	amide NH
H.n	bonded to other nitrogens
H.o	bonded to other oxygens
H.0	all other hydrogens
C.ar6p	sp^2-hybridized carbon with a positive charged resonance structure in a protonated 6-membered heteroaromatic ring
C.ar6x	sp^2-hybridized carbon in a 6-membered heteroaromatic ring
C.ar6	sp^2-hybridized carbon in a benzene ring
C.arp	sp^2-hybridized carbon with a positive charged resonance structure in other protonated heteroaromatic rings
C.arx	sp^2-hybridized carbon in other heteroaromatics
C.ar	sp^2-hybridized carbon in other aromatics
C.2r3o	carbonyl carbon in cyclopropanone or cyclopropenone
C.2r3x	sp^2-hybridized carbon in heterocyclic 3-membered rings
C.2r3	sp^2-hybridized carbon in 3-membered rings
C.3r3x	sp^3-hybridized carbon in heterocyclic 3-membered rings
C.3r3	sp^3-hybridized carbon in 3-membered rings
C.1n	sp-hybridized carbon in cyano groups
C.1p	sp-hybridized carbon with one heavy atom bonded
C.1s	sp-hybridized carbon with two heavy atoms bonded
C.co2h	sp^2-hybridized carbon in explicitly protonated COOH groups
C.co2	sp^2-hybridized carbon in COO^- groups (also set if protonation state is unknown)
C.es	carbonyl carbon in ester groups or anhydrides
C.hal	carbonyl carbon in acidhalogenides
C.am	carbonyl carbon in amides
C.o	other carbonyl carbon
C.s	thionyl carbon
C.gu	sp^2-hybridized carbon in unprotonated guanidino groups
C.guh	sp^2-hybridized carbon in protonated guanidino groups (also set if protonation state is unknown)
C.mi	sp^2-hybridized carbon in unprotonated amidino groups
C.mih	sp^2-hybridized carbon in protonated amidino groups (also set if protonation state is unknown)

Table A.2.: fconv atom types *(continued)*.

Atom type	Definition
C.n	sp^2-hybridized carbon in imines
C.2p	other sp^2-hybridized carbon with one heavy atom bonded
C.2s	other sp^2-hybridized carbon with two heavy atoms bonded
C.2t	other sp^2-hybridized carbon with three heavy atoms bonded
C.et	sp^3-hybridized carbon in ethers
C.ohp	sp^3-hybridized carbon in primary alcoholes
C.ohs	sp^3-hybridized carbon in secondary alcoholes
C.oht	sp^3-hybridized carbon in tertiary alcoholes
C.3n	other sp^3-hybridized carbon bonded to nitrogen
C.3p	other sp^3-hybridized carbon with one heavy atom bonded
C.3s	other sp^3-hybridized carbon with two heavy atoms bonded
C.3t	other sp^3-hybridized carbon with three heavy atoms bonded
C.3q	other sp^3-hybridized carbon with four heavy atoms bonded
N.ar6p	positive charged nitrogen in 6-membered aromatics (e.g. pyridinium or NAD+)
N.ar6	sp^2-hybridized nitrogen in 6-membered aromatics
N.arp	sp^2-hybridized nitrogen in protonated aromatics (e.g. both nitrogens in protonated imidazole)
N.ar2	sp^2-hybridized nitrogen in aromatics with two bonded atoms (corresponding to sybyl type N.2)
N.ar3	sp^2-hybridized nitrogen in aromatics with three heavy atoms (corresponding to sybyl type N.pl3)
N.ar3h	sp^2-hybridized nitrogen in aromatics with two heavy atoms and one hydrogen (corresponding to sybyl type N.pl3)
N.r3	sp^3-hybridized in aziridine or azirene rings
N.az	middle nitrogen in azides
N.1	other sp nitrogen
N.o2	nitrogen in nitro groups
N.ohac	nitrogen in hydroxamic acids
N.oh	nitrogen in hydroxylamines
N.ims	imide nitrogen with two heavy atoms bonded
N.imt	imide nitrogen with three heavy atoms bonded
N.amp	carbon- or thionamide with one heavy atom bonded
N.ams	carbon- or thionamide with two heavy atoms bonded
N.amt	carbon- or thionamide with three heavy atoms bonded

Table A.2.: fconv atom types *(continued)*.

Atom type	Definition
N.samp	sulfonamide with one heavy atom bonded
N.sams	sulfonamide with two heavy atoms bonded
N.samt	sulfonamide with three heavy atoms bonded
N.gu1	NH nitrogen in unprotonated guanidino group (only if explicitly protonated)
N.gu2	NH_2 nitrogen in unprotonated guanidino group (only if explicitly protonated)
N.guh	nitrogen in protonated guanidino group (also set if protonation state is unknown)
N.mi1	NH in unprotonated amidino group (only if explicitly protonated)
N.mi2	NH_2 in unprotonated amidino group (only if explicitly protonated)
N.mih	nitrogen in protonated amidino group (also set if protonation state is unknown)
N.aap	primary aromatic amine (hybridization can't be determined exactly)
N.aas2	sp^2-hybridized secondary aromatic amine
N.aas3	sp^3-hybridized secondary aromatic amine
N.aat2	sp^2-hybridized tertiary aromatic amine
N.aat3	sp^3-hybridized tertiary aromatic amine
N.2n	sp^2-hybridized nitrogen bonded to another nitrogen
N.2p	other sp^2-hybridized nitrogen with one heavy atom
N.2s	other sp^2-hybridized nitrogen with two heavy atoms
N.2t	other sp^2-hybridized nitrogen with three heavy atoms
N.3n	sp^3-hybridized nitrogen bonded to another nitrogen
N.3p	sp^3-hybridized nitrogen with one heavy atom bonded
N.3s	sp^3-hybridized nitrogen with two heavy atoms bonded
N.3t	sp^3-hybridized nitrogen with three heavy atoms bonded
N.4q	sp^3-hybridized nitrogen with 4 bonded heavy atoms
N.4h	sp^3-hybridized nitrogen with 4 bonded atoms (at least 1 hydrogen)
O.ar	aromatic oxygen
O.r3	oxygen in oxiran ring
O.h2o	water oxygen
O.n	oxygen in nitro groups
O.noh	sp^3-hybridized oxygen in hydroxylamine or hydroxamic acid
O.2co2	sp^2-hybridized oxygen in COOH (sp^2-hybridized bonded to C.co2h)
O.2es	sp^2-hybridized oxygen in esters or anhydrids
O.2hal	sp^2-hybridized oxygen in acidhalogenides
O.am	oxygen in carbonamides
O.carb	oxygen in other carbonyl groups

Table A.2.: fconv atom types *(continued)*.

Atom type	Definition
O.co2	oxygen in COO^- or CSO^-
O.2po	sp^2-hybridized oxygen in P=O (non deprotonated groups)
O.2so	sp^2-hybridized oxygen in S=O (non deprotonated groups)
O.2p	sp^2-hybridized oxygen in OPO_3H^- or PO_3H^- or POO^-
O.2s	sp^2-hybridized oxygen in OSO_3^- or SO_3^- or POO^- or deprotonated sulfonamides
O.3po	sp^3-hybridized oxygen with two heavy atoms bonded to at least one phosphor
O.3so	sp^3-hybridized oxygen with two heavy atoms bonded to at least one sulfur
O.o	oxygen in peroxo groups
O.3ac	OH oxygen in COOH, CSOH, $PO(OH)_2$, POOH or SO_2OH
O.ph	oxygen in phenolic hydroxyl group
O.3oh	oxygen in hydroxyl group
O.3es	sp^3-hybridized oxygen in esters or anhydrids
O.3eta	oxygen in aromatic ether
O.3et	oxygen in aliphatic ether
S.ar	aromatic sulfur
S.r3	sulfur in thiiran ring
S.thi	sulfur in thionyl group
S.o	sulfur in SO
S.o2h	sulfur in protonated sulfonamide or other SO_2
S.o3h	sulfur in SO_3
S.o4h	sulfur in OSO_3
S.o2	sulfur in SO_2 or deprotonated sulfonamides (or unknown protonation state)
S.o3	sulfur in SO_3^- (or unknown protonation state)
S.o4	sulfur in OSO_3^- (or unknown protonation state)
S.2	sulfur in CSO^-, COS^- or other sp^2-hybridized sulfur
S.sh	sulfur in SH groups
S.s	suflur in S-S bonds
S.3	other sp^3-hybridized sulfur
P.r3	phosphor in phosphiran rings
P.o	phosphor in PO groups
P.o2h	phosphor in not deprotonated PO_2 groups
P.o3h	phosphor in not deprotonated PO_3 groups
P.o4h	phosphor in not deprotonated PO_4 groups
P.o2	phosphor in deprotonated PO_2 groups (or unknown protonation state)

Table A.2.: fconv atom types *(continued)*.

Atom type	Definition
P.o3	phosphor in deprotonated PO_3 groups (or unknown protonation state)
P.o4	phosphor in deprotonated PO_4 groups (or unknown protonation state)
P.3	other sp^3-hybridized phosphor
F.0	bonded fluorine
F.i	fluoride ion
Cl.0	bonded chlorine
Cl.i	chloride ion
Br.0	bonded bromine
Br.i	bromide ion
I.0	bonded iodine
I.i	iodide ion
Li	lithium
Na	sodium
Mg	magnesium
Al	aluminium
Si	silicon
K	potassium
Ca	calcium
Cr.th	chromium (tetrahedral)
Cr.oh	chromium (octahedral)
Mn	manganese
Fe	iron
Co	cobalt
Cu	copper
Zn	zinc
Se	selenium
Mo	molybdenum
Sn	tin
Ni	nickel
Hg	mercury
B	boron
As	arsenic

Table A.3.: Mapping of fconv atom types onto Sybyl types, specialized *DSX* types and pharmacophoric *DSX* types. The type 'X' is generally ignored by *DSX*.

Internal type	Sybyl type	*DSX* type	Pharmacophoric type
H.ac	H	X	X
H.onh	H	X	X
H.n	H	X	X
H.o	H	X	X
H.0	H	X	X
C.ar6p	C.ar	C.arp	ARO
C.ar6x	C.ar	C.ar6x	ARO
C.ar6	C.ar	C.ar6	ARO
C.arp	C.2	C.arp	ARO
C.arx	C.2	C.ar	ARO
C.ar	C.ar	C.ar	ARO
C.2r3o	C.2	C.o	C
C.2r3x	C.2	C.2	C
C.2r3	C.2	C.2	HYD
C.3r3x	C.3	C.3	C
C.3r3	C.3	C.3	HYD
C.1n	C.1	C.1	C
C.1p	C.1	C.1	HYD
C.1s	C.1	C.1	HYD
C.co2h	C.2	C.co2h	C
C.co2	C.2	C.co2	C
C.es	C.2	C.o	C
C.hal	C.2	C.o	C
C.am	C.2	C.am	C
C.o	C.2	C.o	C
C.s	C.2	C.o	C
C.gu	C.2	C.guh	C
C.guh	C.cat	C.guh	C
C.mi	C.2	C.guh	C
C.mih	C.2	C.guh	C
C.n	C.2	C.2	C
C.2p	C.2	C.2	HYD

Table A.3.: fconv atom type mappings *(continued)*.

Internal type	Sybyl type	*DSX* type	Pharmacophoric type
C.2s	C.2	C.2	HYD
C.2t	C.2	C.2	HYD
C.et	C.3	C.3	HYD
C.ohp	C.3	C.3s	HYD
C.ohs	C.3	C.3t	HYD
C.oht	C.3	C.3q	HYD
C.3n	C.3	C.3	C
C.3p	C.3	C.3p	HYD
C.3s	C.3	C.3s	HYD
C.3t	C.3	C.3t	HYD
C.3q	C.3	C.3q	HYD
N.ar6p	N.pl3	N.arp	DON
N.ar6	N.ar	N.ar6	ARO
N.arp	N.pl3	N.arp	DON
N.ar2	N.2	N.ar2	ACC
N.ar3	N.pl3	N.ar3	ARO
N.ar3h	N.pl3	N.ar3h	DON
N.r3	N.3	N.3s	AnD
N.az	N.1	N.1	N
N.1	N.1	N.1	ACC
N.o2	N.pl3	N.o2	ARO
N.ohac	N.am	N.ams	DON
N.oh	N.3	N.oh	ACC
N.ims	N.am	N.ams	DON
N.imt	N.am	N.amt	N
N.amp	N.am	N.amp	DON
N.ams	N.am	N.ams	DON
N.amt	N.am	N.amt	N
N.samp	N.am	N.sams	DON
N.sams	N.am	N.sams	DON
N.samt	N.am	N.amt	N
N.gu1	N.2	N.guh	AnD
N.gu2	N.pl3	N.guh	AnD
N.guh	N.pl3	N.guh	DON

Table A.3.: fconv atom type mappings *(continued)*.

Internal type	Sybyl type	*DSX* type	Pharmacophoric type
N.mi1	N.2	N.guh	AnD
N.mi2	N.pl3	N.guh	AnD
N.mih	N.pl3	N.guh	DON
N.aap	N.pl3	N.aa	AnD
N.aas2	N.pl3	N.aa	N
N.aas3	N.3	N.aa	ACC
N.aat2	N.pl3	N.aat	N
N.aat3	N.3	N.aat	ACC
N.2n	N.2	N.2n	ACC
N.2p	N.2	N.2p	AnD
N.2s	N.2	N.2s	ACC
N.2t	N.pl3	N.3t	N
N.3n	N.3	N.3n	AnD
N.3p	N.3	N.3p	AnD
N.3s	N.3	N.3s	AnD
N.3t	N.3	N.3t	ACC
N.4q	N.4	N.4q	N
N.4h	N.4	N.4h	DON
O.ar	O.3	O.3et	ARO
O.r3	O.3	O.3et	ACC
O.h2o	O.3	O.h2o	AnD
O.n	O.2	O.n	ACC
O.noh	O.3	O.3ac	AnD
O.2co2	O.2	O.2co2	ACC
O.2es	O.2	O.carb	ACC
O.2hal	O.2	O.carb	ACC
O.am	O.2	O.carb	ACC
O.co2	O.co2	O.co2	ACC
O.2po	O.2	O.carb	ACC
O.2so	O.2	O.carb	ACC
O.2p	O.co2	O.co2	ACC
O.2s	O.co2	O.co2	ACC
O.3po	O.3	O.3et	ACC
O.3so	O.3	O.3et	ACC

Table A.3.: fconv atom type mappings *(continued)*.

Internal type	Sybyl type	*DSX* type	Pharmacophoric type
O.carb	O.2	O.carb	ACC
O.o	O.3	O.o	ACC
O.3ac	O.3	O.3ac	AnD
O.ph	O.3	O.3oh	AnD
O.3oh	O.3	O.3oh	AnD
O.3es	O.3	O.3es	ACC
O.3eta	O.3	O.3et	ACC
O.3et	O.3	O.3et	ACC
S.ar	S.3	S.3	ARO
S.r3	S.3	S.3	ACC
S.thi	S.2	S.2	ACC
S.o	S.o	S.o	S
S.o2h	S.o2	S.o	S
S.o3h	S.o2	S.o	S
S.o4h	S.o2	S.o	S
S.o2	S.o2	S.o	S
S.o3	S.o2	S.o	S
S.o4	S.o2	S.o	S
S.2	S.2	S.2	ACC
S.sh	S.3	S.sh	AnD
S.s	S.3	S.3	ACC
S.3	S.3	S.3	ACC
P.r3	P.3	P.3	P
P.o	P.3	P.o	P
P.o2h	P.3	P.o	P
P.o3h	P.3	P.o	P
P.o4h	P.3	P.o	P
P.o2	P.3	P.o	P
P.o3	P.3	P.o	P
P.o4	P.3	P.o	P
P.3	P.3	P.3	ACC
F.0	F	F.0	HAL
F.i	F	X	X

Table A.3.: fconv atom type mappings *(continued)*.

Internal type	Sybyl type	*DSX* type	Pharmacophoric type
Cl.0	Cl	Cl.0	HAL
Cl.i	Cl	X	X
Br.0	Br	Br.0	HAL
Br.i	Br	X	X
I.0	I	I.0	HAL
I.i	I	X	X
Li	Li	X	X
Na	Na	X	X
Mg	Mg	Mg	MET
Al	Al	X	MET
Si	Si	X	X
K	K	X	X
Ca	Ca	Ca	MET
Cr.th	Cr.th	X	X
Cr.oh	Cr.oh	X	X
Mn	Mn	X	MET
Fe	Fe	Fe	MET
Co	Co	Co	MET
Cu	Cu	Cu	MET
Zn	Zn	Zn	MET
Se	Se	X	MET
Mo	Mo	X	X
Sn	Sn	X	X
Ni	Ni	Ni	MET
Hg	Hg	X	X
As	As	X	X
B	B	B	X

Glossary

Active site In case of enzymes, the active site is that region where substrates bind. In case of receptors, the active site is that region where signaling molecules bind in a step of signal transduction. 109, 208

B-factor Also known as *temperature factor* or *Debye-Waller factor*. In X-ray crystallography it is a measure for the thermal oscillation of each individual atom around its position specified in the model. The final B-value is a combination of coordinates uncertainty due to true atom mobility and due to the overall uncertainty in structure refinement. The mean square displacement U^2 of an atom in $Å^2$ can be calculated as

$$U^2 = \frac{B}{8\pi^2}$$

. 133

Bin In context of this work a bin corresponds to an interval $[a, b[$ of a function's independent variable. 22, 24, 135, 208

Binder see ligand. 109, 169, 212

Binding affinity The higher the affinity between a ligand and a protein is, the more stable the complex they form will be. Thus the binding affinity is usually measured by the dissociation constant. 112

Binding constant see dissociation constant. 142

Binding energy Difference in Gibbs free energy between a protein-ligand complex and the uncomplexed state. It is related to the dissociation constant

by the following equation:

$$\Delta G = -RTln(K_d)$$

where R is the molar gas constant and T a constant temperature. 112

Binding pocket see binding site. 83

Binding pose The conformation and configuration of a ligand when complexed with a particular target. 109, 210

Binding site The region of a biomolecule where a ligand binds. Often this is the active site, but also other regions of binding are possible. 94, 208

Binning The merging of neighboring elements into one bin. 137

Clique A clique is a subgraph $M(V', E') \subseteq G(V, E)$, where all possible pairs of vertices are adjacent ($(u, v) \in E' \quad \forall u, v \in V'$). 75, 175

Combinatorial explosion The number of possible values of a function grows rapidly with the number of variables (n), e.g. 2^n in case of only boolean variables. A combinatorial explosion exists, if n exceeds the value that allows for a computation of function values for all possible combinations in a reasonable amount of time. 110

Degree Within the thesis, the term "degree" is defined as an attribute of a vertex v in a graph. The degree of v corresponds to the number of vertices that are adjacent to v. 36

Dissociation constant The equilibrium constant K_d for the dissociation of a protein-ligand complex PL into the protein P and the ligand L.

$$K_d = \frac{[P][L]}{[PL]}$$

where the brackets stand for concentrations. If the dissociation constant is determined indirectly via competitive inhibition in a kinetic assay with

respect to another ligand binding to the same protein, it is named K_i. 207

Docking pose A conformation of a ligand that was generated by molecular docking. 82, 109

Drug target A drug target is a biomolecule whose inhibition or activation either results in a therapeutic effect with respect to a disease or in an (subjective) improvement of life quality. Usually these biomolecules are macromolecules like proteins or nucleic acids. 109, 212

Enzyme Enzymes are biomolecules that catalyze a chemical reaction. 167, 207

Gibbs free energy The work obtainable from a thermodynamic system at constant temperature T and constant pressure p.

$$G(T, p) = H - TS$$

where H is the enthalpy and S the entropy. 207

Graph A graph $G(V, E)$ is a structure that consist of a set of vertices V and a set of edges E. Each edge connects to vertices, hence an edge is also a set of two vertices. Such connected vertices are called adjacent, while edges that share one vertex are called incident. 18, 28, 40, 74, 95

Helmholtz free energy The work obtainable from a thermodynamic system at constant temperature T and constant volume V.

$$A(T, V) = U - TS$$

where U is the internal energy. 118

Heuristic A heuristic is a simplification of a complicated problem that leads to a significant speedup in solving the problem and yields sufficient results

in the majority of possible cases. However, using heuristics an optimal solution cannot be guaranteed. 29, 57, 86, 120, 175

In silico This is a coinage that was created as an extension to the existing Latin phrases *in vivo* (processes within organisms) and *in vitro* (processes in chemical equipment). It stands for processes simulated with a computer. 109

Ligand A molecule that binds to a particular target. 6, 11, 15, 18, 109, 207, 208

Manhattan distance Also known as 'Taxicab distance' or 'city block distance'. For two vectors x and y in an n-dimensional vector space, the Manhattan distance is $d = \sum_{i=1}^{n} |x_i - y_i|$. 169

Molecular docking A computational method that performs a global optimization of a protein-ligand system with respect to a given scoring function. Mostly, only translational, torsional and rotational degrees of freedom of the ligand are considered while the protein and all bond lengths and angles are kept rigid. 209

Native pose see binding pose. 111, 142

Occupancy In X-ray crystallography, the occupancy of the nth atom a in a model corresponds to the fraction of molecules in the measured crystal where the nth atom had the same coordinates as a. Hence, if an atom has an occupancy of 0.5 it is found at the specified position in only half of the unit cells of the entire crystal used in the diffraction experiment. 133

Pearson correlation The Pearson product-moment correlation coefficient $\rho_{X,Y}$ measures the linear relationship between two quantities X and Y. For a sample of n pairwise measures $(x_1, y_1), \ldots, (x_n, y_n)$ it can be calculated as $\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}}$. 142, 211

Quaternions Also known as Hamilton numbers $\mathbb{H}$. An extension of the real numbers by three imaginary numbers. $\mathbb{H}$ forms a 4-dimensional normed division algebra over $\mathbb{R}$. 74

R-factor In X-ray crystallography, the R-factor is a measure for the difference between experimentally observed diffraction pattern and the diffraction pattern that is calculated from the model. A lower R-factor stands for a better agreement between the refined structure and the experimental data and therefore the R-factor is used to describe the quality of a model. 133

Receptor Receptors are biomolecules that receive chemical signals in a signal transduction chain. 207

Resolution In optics, two objects (observed under a given angle) can be separated, if the central peaks of their diffraction patterns (depending on the resolution) do not overlap. A higher resolution (lower values) in X-ray crystallography corresponds to shorter distances between atoms that can be separated with respect to their electron densities. 59, 133

Scoring function An arbitrary function that is correlated with binding affinity and/or that can discriminate between correct and incorrect ligand geometries. 5, 110, 111, 210

Signal transduction A process where extracellular signals (in terms of chemical structures) are received and lead to an intracellular response. 207, 211

Spanning tree A spanning tree of a graph $G(V, E)$ is a connected, acyclic subgraph $T(V', E') \subseteq G(V, E)$ with $V' = V$. 30

Spearman correlation In contrast to the Pearson correlation that measures only linear dependency, Spearman's rank correlation coefficient r_S can be used to evaluate the correlation between two variables that are related by an arbitrary monotonic function. Therefore, the measured values are

transformed into ranks, and then the Pearson correlation for these ranks is calculated. In case of tied values, the rank is calculated as the mean of the values' positions in the ranking. 142

Target see drug target. 94, 109, 141, 152, 169

Virtual screening A computational method that screens huge libraries of small molecules for putative binders with respect to a particular target. 109

Bibliography

Allen, F. H. (2002). The Cambridge Structural Database: a quarter of a million crystal structures and rising. *Acta Crystallogr., Sect. B: Struct. Sci.*, 58:380–388.

Allen, F. H., Bellard, S., Brice, M. D., Cartwright, B. A., Doubleday, A., Higgs, H., Hummelink, T., Hummelink-Peters, B. G., Kennard, O., Motherwell, W. D., and et al. (1979). The Cambridge Crystallographic Data Centre: Computer-Based Search, Retrieval, Analysis and Display of Information. *Acta Crystallogr., Sect. B: Struct. Sci.*, B35:2331–2339.

Aqvist, J., Medina, C., and Samuelsson, J. E. (1994). A new method for predicting binding affinity in computer-aided drug design. *Protein Engineering*, 7(3):385–391.

Baber, J. C. and Hodgkin, E. E. (1992). Automatic Assignment of Chemical Connectivity to Organic Molecules in the Cambridge Structural Database. *J. Chem. Inf. Model.*, 32(5):401–406.

Behnen, J., Köster, H., Neudert, G., Craan, T., Heine, A., and Klebe, G. (2012). Experimental and Computational Active Site Mapping as a Starting Point to Fragment-Based Lead Discovery. *ChemMedChem*, 7(2):248–261.

Ben-Naim, A. (1992). *Statistical Thermodynamics for Chemists and Biochemists*. Plenum Press, New York. chap. 5,6.

Ben-Naim, A. (1997). Statistical potentials extracted from protein structures: Are these meaningful potentials? *J. Chem. Phys.*, 107(9):3698–3706.

Berger, F., Flamm, C., Gleiss, P. M., Leydold, J., and Stadler, P. F. (2004). Counterexamples in chemical ring perception. *Journal of Chemical Information and Computer Sciences*, 44(2):323–331.

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D., and Zardecki, C. (2002). The Protein Data Bank. *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, 58:899–907.

Block, P., Weskamp, N., Wolf, A., and Klebe, G. (2007). Strategies to search and design stabilizers of protein-protein interactions: A feasibility study. *Proteins: Struct., Funct., Bioinf.*, 68(1):170–186.

Bolon, D. N. and Mayo, S. L. (2001). Enzyme-like proteins by computational design. *Proceedings of the National Academy of Sciences of the United States of America*, 98(25):14274–14279.

Boström, J., Greenwood, J. R., and Gottfries, J. (2003). Assessing the performance of OMEGA with respect to retrieving bioactive conformations. *J. Mol. Graphics Modell.*, 21(5):449–462.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

Bruno, I. J., Cole, J. C., Edgington, P. R., Kessler, M., Macrae, C. F., McCabe, P., Pearson, J., and Taylor, R. (2002). New software for searching the Cambridge Structural Database and visualizing crystal structures. *Acta Crystallogr., Sect. B: Struct. Sci.*, 58:389–397.

Bush, L. B. and Sheridan, R. P. (1993). PATTY: A Programmable Atom Typer and Language for Automatic Classification of Atoms in Molecular Databases. *J. Chem. Inf. Comp. Sci.*, 33:756–762.

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.

Böhm, H. J. (1998). Prediction of binding constants of protein ligands: a fast method for the prioritization of hits obtained from de novo design or 3D database search programs. *J. Comput.-Aided Mol. Des.*, 12:309–323.

Capra, J. A., Laskowski, R. A., Thornton, J. M., Singh, M., and Funkhouser, T. A. (2009). Predicting Protein Ligand Binding Sites by Combining Evolutionary Sequence Conservation and 3D Structure. *PLoS Computational Biology*, 5(12):18.

Chakravarty, S. and Kannan, K. K. (1994). Drug-protein interactions. Refined structures of three sulfonamide drug complexes of human carbonic anhydrase I enzyme. *J. Mol. Biol.*, 243(2):298–309.

Cheng, T., Li, X., Li, Y., Liu, Z., and Wang, R. (2009). Comparative assessment of scoring functions on a diverse test set. *J. Chem. Inf. Model.*, 49:1079–1093.

Craan, T. (2011). Fragment based Drug Discovery; Design and Validation of a Fragment Library; Computer-based Fragment Screening and Fragment-to-Lead Expansion. *Dissertation*, pages 53–66, 80–90.

Deo, N., Prabhu, G., and Krishnamoorthy, M. S. (1982). Algorithms for Generating Fundamental Cycles in a Graph. *ACM Transactions on Mathematical Software*, 8(1):26–42.

Dixon, J. S. (1997). Evaluation of the CASP2 docking section. *Proteins: Struct., Funct., Bioinf.*, Suppl 1:198–204.

Drake, D. E. and Hougardy, S. (2003). A Simple Approximation Algorithm for the Weighted Matching Problem. *Information Processing Letters*, 85:211–213.

Edelsbrunner, H., Face, M., Fu, P., and Liang, J. (1995). Measuring proteins and voids in proteins. In *In Proc. 28th Ann. Hawaii Int'l Conf. System Sciences*, pages 256–264.

Edelsbrunner, H., Facello, M., and Liang, J. (1998). On the definition and the construction of pockets in macromolecules. *Discrete Applied Mathematics*, 88(1-3):83–102.

Edelsbrunner, H. and Shah, N. R. (1996). Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Stand.*, 69:125–130.

Eldridge, M. D., Murray, C. W., Auton, T. R., Paolini, G. V., and Mee, R. P. (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.

Ewing, T. J., Makino, S., Skillman, A. G., and Kuntz, I. D. (2001). DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J. Comput.-Aided Mol. Des.*, 15:411–428.

Fischer, A., Enkler, N., Neudert, G., Bocola, M., Sterner, R., and Merkl, R. (2009). TransCent: Computational enzyme design by transferring active sites and considering constraints relevant for catalysis. *BMC Bioinformatics*, 10(1):54.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Fredman, M. L. and Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615.

Friesner, R. A., Banks, J. L., Murphy, R. B., Halgren, T. A., Klicic, J. J., Mainz, D. T., Repasky, M. P., Knoll, E. H., Shelley, M., Perry, J. K., Shaw, D. E., Francis, P., and Shenkin, P. S. (2004). Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. *J. Med. Chem.*, 47:1739–1749.

Friesner, R. A., Murphy, R. B., Repasky, M. P., Frye, L. L., Greenwood, J. R., Halgren, T. A., Sanschagrin, P. C., and Mainz, D. T. (2006). Extra precision glide: docking and scoring incorporating a model of hydrophobic enclosure for protein-ligand complexes. *J. Med. Chem.*, 49:6177–6196.

Froeyen, M. and Herdewijn, P. (2005). Correct bond order assignment in a molecular framework using integer linear programming with application to molecules where only non-hydrogen atom coordinates are available. *J. Chem. Inf. Model.*, 45(5):1267–1274.

Gabow, H. N. (1976). An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs. *Journal of the ACM*, 23(2):221–234.

Gabow, H. N., Galil, Z., and Spencer, T. H. (1989). Efficient implementation of graph algorithms using contraction. *Journal of the ACM*, 36(3):540–572.

Galil, Z., Micali, S., and Gabow, H. N. (1986). An O(EV log V) Algorithm for Finding a Maximal Weighted Matching in General Graphs. *SIAM Journal on Computing*, 15(1):120–130.

Gasteiger, J., Rudolph, C., and Sadowski, J. (1990). Automatic generation of 3D-atomic coordinates for organic molecules. *Tetrahedron Comput. Methodol.*, 3(6):537–547.

Gehlhaar, D. K., Verkhivker, G. M., Rejto, P. A., Sherman, C. J., Fogel, D. B., Fogel, L. J., and Freer, S. T. (1995). Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programming. *Chem. Biol. (Cambridge, MA, U. S.)*, 2:317–324.

Gilson, M. K. and Zhou, H.-X. (2007). Calculation of Protein-Ligand Binding Affinities. *Annu. Rev. Biophys. Biomol. Struct.*, 36(1):21–42.

Gohlke, H., Hendlich, M., and Klebe, G. (2000). Knowledge-based scoring function to predict protein-ligand interactions. *J. Mol. Biol.*, 295:337–356.

Goodford, P. J. (1985). A Computational Procedure for Determining Energetically Favorable Binding Sites on Biologically Important Macromolecules. *J. Med. Chem.*, 28(1):849–857.

Goodsell, D. S. and Olson, A. J. (1990). Automated docking of substrates to proteins by simulated annealing. *Proteins: Struct., Funct., Bioinf.*, 8:195–202.

Halgren, T. (2007). New method for fast and accurate binding-site identification and analysis. *Chemical biology drug design*, 69(2):146–148.

Hamelryck, T., Borg, M., Paluszewski, M., Paulsen, J., Frellsen, J., Andreetta, C., Boomsma, W., Bottaro, S., and Ferkinghoff-Borg, J. (2010). Potentials of mean force for protein structure prediction vindicated, formalized and generalized. *PLoS One*, 5:e13714.

Hendlich, M., Bergner, A., Günther, J., and Klebe, G. (2003). Relibase: design and development of a database for comprehensive analysis of protein-ligand interactions. *J. Mol. Biol.*, 326(2):607–620.

Hendlich, M., Rippmann, F., and Barnickel, G. (1997a). BALI: Automatic Assignment of Bond and Atom Types for Protein Ligands in the Brookhaven Protein Databank. *J. Chem. Inf. Model.*, 37(4):774–778.

Hendlich, M., Rippmann, F., and Barnickel, G. (1997b). LIGSITE: automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of molecular graphics modelling*, 15(6):359–363.

Holm, L. and Sander, C. (1993). Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1):123–138.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629.

Horton, J. D. (1987). A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366.

Hou, T., Wang, J., Li, Y., and Wang, W. (2011). Assessing the Performance of the MM/PBSA and MM/GBSA Methods. 1. The Accuracy of Binding Free Energy Calculations Based on Molecular Dynamics Simulations. *J. Chem. Inf. Model.*, 51(1):69–82.

Hovmöller, S., Zhou, T., and Ohlson, T. (2002). Conformations of amino acids in proteins. *Acta Crystallographica Section D Biological Crystallography*, 58(5):768–776.

Huang, B. and Schroeder, M. (2006). LIGSITEcsc: predicting ligand binding sites using the Connolly surface and degree of conservation. *BMC Structural Biology*, 6(1):19.

Huang, S. Y. and Zou, X. (2006a). An iterative knowledge-based scoring function to predict protein-ligand interactions: I. Derivation of interaction potentials. *J. Comput. Chem.*, 27:1866–1875.

Huang, S. Y. and Zou, X. (2006b). An iterative knowledge-based scoring function to predict protein-ligand interactions: II. Validation of the scoring function. *J. Comput. Chem.*, 27:1876–1882.

Huang, S. Y. and Zou, X. (2010). Inclusion of solvation and entropy in the knowledge-based scoring function for protein-ligand interactions. *J. Chem. Inf. Model.*, 50:262–273.

Hückel, E. (1931a). Quantentheoretische Beiträge zum Benzolproblem. I. Die Elektronenkonfiguration des Benzols und verwandter Verbindungen. *Zeitschrift Für Physik*, 70(3-4):204–286.

Hückel, E. (1931b). Quantentheoretische Beiträge zum Benzolproblem. II. Quantentheorie der induzierten Polaritäten. *Zeitschrift Für Physik*, 72:310–337.

Hückel, E. (1932). Quantentheoretische Beiträge zum Problem der aromatischen und ungesättigten Verbindungen. III. *Zeitschrift Für Physik*, 76(9-10):628–648.

Jain, A. N. (1996). Scoring noncovalent protein-ligand interactions: a continuous differentiable function tuned to compute binding affinities. *J. Comput.-Aided Mol. Des.*, 10:427–440.

Jain, A. N. (2003). Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *J. Med. Chem.*, 46:499–511.

Jernigan, R. L. and Bahar, I. (1996). Structure-derived potentials and protein simulations. *Curr. Opin. Struct. Biol.*, 6:195–209.

Jones, G., Willett, P., and Glen, R. C. (1995). Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J. Mol. Biol.*, 245:43–53.

Jones, G., Willett, P., Glen, R. C., Leach, A. R., and Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking. *J. Mol. Biol.*, 267:727–748.

Jorgensen, W. L. (1989). Free energy calculations: A breakthrough for modeling organic chemistry in solution. *Acc. Chem. Res.*, 22:184–189.

Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., and Klein, M. L. (1983). Comparison of simple potential functions for simulating liquid water. *The Journal of Chemical Physics*, 79(2):926.

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

Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828.

Kellenberger, E., Rodrigo, J., Muller, P., and Rognan, D. (2004). Comparative evaluation of eight docking tools for docking and virtual screening accuracy. *Proteins: Struct., Funct., Bioinf.*, 57:225–242.

Kleywegt, G. J. and Jones, T. A. (1996). Phi/psi-chology: Ramachandran revisited. *Structure*, 4(12):1395–1400.

Koch, I. (2001). Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1-2):1–30.

Koppensteiner, W. A. and Sippl, M. J. (1998). Knowledge-based potentials – back to the roots. *Biochemistry (Moscow)*, 63:247–252.

Krammer, A., Kirchhoff, P. D., Jiang, X., Venkatachalam, C. M., and Waldman, M. (2005). LigScore: a novel scoring function for predicting binding affinities. *J. Mol. Graphics Modell.*, 23:395–407.

Kuhlman, B. and Baker, D. (2000). Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences of the United States of America*, 97(19):10383–10388.

Labute, P. (2005). On the Perception of Molecules from 3D Atomic Coordinates. *J. Chem. Inf. Model.*, 45(2):215–221.

Langmuir, I. (1919). THE ARRANGEMENT OF ELECTRONS IN ATOMS AND MOLECULES. *J. Am. Chem. Soc.*, 41(6):868–934.

Laskowski, R. A. (1995). SURFNET: a program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of Molecular Graphics*, 13(5):323–30.

Lawler, E. L. (1976). *Combinatorial Optimization: Networks and Matroids*, volume 40. Holt, Rinehart and Winston.

Le Guilloux, V., Schmidtke, P., and Tuffery, P. (2009). Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics*, 10(1):168.

Levi, G. (1972). A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4):341–352.

Levitt, D. G. and Banaszak, L. J. (1992). POCKET: a computer graphics method for identifying and displaying protein cavities and their surrounding amino acids. *Journal of Molecular Graphics*, 10(4):229–234.

Liang, J., Edelsbrunner, H., and Woodward, C. (1998). Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Science*, 7(9):1884–1897.

Lide, D. R. e. (2009). *CRC Handbook of Chemistry and Physics, 89th Edition*. CRC Press/Taylor and Francis, Boca Raton, FL, 89th (Internet Version) edition.

Lipinski, C. A. (2003). Chris Lipinski discusses life and chemistry after the Rule of Five. *Drug Discovery Today*, 8(1):12–16.

Massova, I. and Kollman, P. A. (2000). Combined molecular mechanical and continuum solvent approach (MM-PBSA/GBSA) to predict ligand binding. *Perspect. Drug Discovery Des.*, 18:113–135.

Meng, E. C. and Lewis, R. A. (1991). Determination of Molecular Topology and Atomic Hybridization States from Heavy Atom Coordinates. *J. Comput. Chem.*, 12(7):891–898.

Mooij, W. T. and Verdonk, M. L. (2005). General and targeted statistical potentials for protein-ligand interactions. *Proteins: Struct., Funct., Bioinf.*, 61:272–287.

Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., and Olson, A. J. (1998). Automated docking using a Lamarckian

genetic algorithm and an empirical binding free energy function. *J. Comput. Chem.*, 19:1639–1662.

Morris, G. M., Goodsell, D. S., Huey, R., and Olson, A. J. (1996). Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4. *J. Comput.-Aided Mol. Des.*, 10:293–304.

Mosca, R. and Schneider, T. R. (2008). RAPIDO: a web server for the alignment of protein structures in the presence of conformational changes. *Nucleic Acids Research*, 36(Web Server issue):W42–W46.

Muegge, I. (2006). PMF scoring revisited. *J. Med. Chem.*, 49:5895–5902.

Muegge, I. and Martin, Y. C. (1999). A general and fast scoring function for protein-ligand interactions: a simplified potential approach. *J. Med. Chem.*, 42:791–804.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

Neudert, G. and Klebe, G. (2011a). DSX: A Knowledge-Based Scoring Function for the Assessment of Protein–Ligand Complexes. *J. Chem. Inf. Model.*, 51(10):2731–2745.

Neudert, G. and Klebe, G. (2011b). fconv: format conversion, manipulation and feature computation of molecular data. *Bioinformatics*, 27:1021–1022.

Ortiz, A. R., Strauss, C. E. M., and Olmea, O. (2002). MAMMOTH (Matching molecular models obtained from theory): An automated method for model comparison. *Protein Science*, 11(11):2606–2621.

Paul, N. and Rognan, D. (2002). ConsDock: A new program for the consensus analysis of protein-ligand interactions. *Proteins: Struct., Funct., Bioinf.*, 47:521–533.

Pfeffer, P. (2009). Optimiertes Design kombinatorischer Verbindungsbibliotheken durch Genetische Algorithmen und deren Bewertung anhand wissensbasierter Protein-Ligand Bindungsprofile. *Dissertation*, pages 17–49.

Pfeffer, P. and Gohlke, H. (2007). DrugScoreRNA–knowledge-based scoring function to predict RNA-ligand interactions. *J. Chem. Inf. Model.*, 47:1868–1876.

Pinto, A. L., Hellinga, H. W., and Caradonna, J. P. (1997). Construction of a catalytically active iron superoxide dismutase by rational protein design. *Proceedings of the National Academy of Sciences of the United States of America*, 94(11):5562–5567.

Plewczynski, D., Łaźniewski, M., Augustyniak, R., and Ginalski, K. (2011). Can we trust docking results? Evaluation of seven commonly used programs on PDBbind database. *J. Comput. Chem.*, 32:742–755.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007a). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3rd ed. edition. pp 507-514.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007b). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3rd ed. edition. pp 1149-1152.

Ramachandran, G. N., Ramakrishnan, C., and Sasisekharan, V. (1963). Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7:95–99.

Rarey, M., Kramer, B., Lengauer, T., and Klebe, G. (1996). A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.*, 261:470–489.

Raymond, J. W., Gardiner, E. J., and Willett, P. (2002). Heuristics for similarity searching of chemical graphs using a maximum common edge

subgraph algorithm. *Journal of Chemical Information and Computer Sciences*, 42(2):305–316.

Ritschel, T. (2009). TGT a Drug Target to Study pka Shifts, Residual Solvation and Protein-Protein Interface Formation. *Dissertation*, pages 80–83.

Ritschel, T., Kohler, P. C., Neudert, G., Heine, A., Diederich, F., and Klebe, G. (2009). How to replace the residual solvation shell of polar active site residues to achieve nanomolar inhibition of tRNA-guanine transglycosylase. *ChemMedChem*, 4(12):2012–2023.

Ruvinsky, A. M. and Kozintsev, A. V. (2005). The key role of atom types, reference states, and interaction cutoff radii in the knowledge-based method: new variational approach. *Proteins: Struct., Funct., Bioinf.*, 58:845–851.

Röthlisberger, D., Khersonsky, O., Wollacott, A. M., Jiang, L., DeChancie, J., Betker, J., Gallaher, J. L., Althoff, E. A., Zanghellini, A., Dym, O., and et al. (2008). Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195.

Shen, Q., Xiong, B., Zheng, M., Luo, X., Luo, C., Liu, X., Du, Y., Li, J., Zhu, W., Shen, J., and Jiang, H. (2011). Knowledge-based scoring functions in drug design: 2. Can the knowledge base be enriched? *J. Chem. Inf. Model.*, 51:386–397.

Shindyalov, I. N. and Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(9):739–747.

Sippl, M. J. (1990). Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. *J. Mol. Biol.*, 213:859–883.

Sippl, M. J. (1993). Boltzmann's principle, knowledge-based mean fields and protein folding. An approach to the computational determination of protein structures. *J. Comput.-Aided Mol. Des.*, 7:473–501.

Sippl, M. J. (1995). Knowledge-based potentials for proteins. *Curr. Opin. Struct. Biol.*, 5:229–235.

Sotriffer, C. A., Sanschagrin, P., Matter, H., and Klebe, G. (2008). SFCscore: scoring functions for affinity prediction of protein-ligand complexes. *Proteins: Struct., Funct., Bioinf.*, 73:395–419.

Spitzmüller, A., Velec, H. F. G., and Klebe, G. (2011). MiniMuDS: a new optimizer using knowledge-based potentials improves scoring of docking solutions. *J. Chem. Inf. Model.*, 51(6):1423–1430.

Stewart, R. F. and Jensen, L. H. (1967). Redetermination of the crystal structure of uracil. *Acta Crystallographica*, 124(2):305–308.

Tang, Y. T. and Marshall, G. R. (2011). PHOENIX: A Scoring Function for Affinity Prediction Derived Using High-Resolution Crystal Structures and Calorimetry Measurements. *J. Chem. Inf. Model.*, 51:214–228.

Taylor, R. D., Jewsbury, P. J., and Essex, J. W. (2002). A review of protein-small molecule docking methods. *J. Comput.-Aided Mol. Des.*, 16:151–166.

Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42.

Toscano, M. D., Woycechowsky, K. J., and Hilvert, D. (2007). Minimalist active-site redesign: teaching old enzymes new tricks. *Angewandte Chemie International Edition*, 46(18):3212–3236.

Velec, H. F., Gohlke, H., and Klebe, G. (2005). DrugScore(CSD)-knowledge-based scoring function derived from small molecule crystal data with superior recognition rate of near-native ligand poses and better affinity prediction. *J. Med. Chem.*, 48:6296–6303.

Verdonk, M. L., Cole, J. C., and Taylor, R. (1999). SuperStar: a knowledge-based approach for identifying interaction sites in proteins. *Journal of Molecular Biology*, 289(4):1093–1108.

Vismara, P. (1997). Union Of All Minimum Cycle Bases Of A Graph. *The electronic Journal of Combinatorics*, 4:73–87.

Wang, R., Fang, X., Lu, Y., and Wang, S. (2004). The PDBbind database: collection of binding affinities for protein-ligand complexes with known three-dimensional structures. *J. Med. Chem.*, 47:2977–2980.

Wang, R., Fang, X., Lu, Y., Yang, C. Y., and Wang, S. (2005). The PDBbind database: methodologies and updates. *J. Med. Chem.*, 48:4111–4119.

Wang, R., Lai, L., and Wang, S. (2002). Further development and validation of empirical scoring functions for structure-based binding affinity prediction. *J. Comput.-Aided Mol. Des.*, 16:11–26.

Warren, G. L., Andrews, C. W., Capelli, A. M., Clarke, B., LaLonde, J., Lambert, M. H., Lindvall, M., Nevins, N., Semus, S. F., Senger, S., Tedesco, G., Wall, I. D., Woolven, J. M., Peishoff, C. E., and Head, M. S. (2006). A critical assessment of docking programs and scoring functions. *J. Med. Chem.*, 49:5912–5931.

Xie, Z. R. and Hwang, M. J. (2010). An interaction-motif-based scoring function for protein-ligand docking. *BMC Bioinf.*, 11:298.

Xue, M., Zheng, M., Xiong, B., Li, Y., Jiang, H., and Shen, J. (2010). Knowledge-based scoring functions in drug design. 1. Developing a target-specific method for kinase-ligand interactions. *J. Chem. Inf. Model.*, 50:1378–1386.

Zhai, L., Shukla, R., and Rathore, R. (2009). Oxidative C-C bond formation (Scholl reaction) with DDQ as an efficient and easily recyclable oxidant. *Organic Letters*, 11(15):3474–3477.

Zhang, C., Liu, S., Zhu, Q., and Zhou, Y. (2005). A knowledge-based energy function for protein-ligand, protein-protein, and protein-DNA complexes. *J. Med. Chem.*, 48:2325–2335.

Zhao, Y., Cheng, T., and Wang, R. (2007). Automatic Perception of Organic Molecules Based on Essential Structural Information. *J. Chem. Inf. Model.*, 47(4):1379–1385.

Zsoldos, Z., Reid, D., Simon, A., Sadjad, S. B., and Johnson, A. P. (2007). eHiTS: a new fast, exhaustive flexible ligand docking system. *J. Mol. Graphics Modell.*, 26:198–212.

Danksagung

Ich danke *Prof. Dr. Gerhard Klebe* für die Aufnahme in seiner Arbeitsgruppe und das Ermöglichen der vorliegenden Arbeit. Er gewährte mir stets großen Freiraum bei der Verfolgung neuer Ideen und war immer offen für alle Vorschläge. Er gab Impulse ohne Zwänge und motivierte durch sein entgegengebrachtes Vertrauen.

Prof. Dr. Eyke Hüllermeier danke ich für die Bereitschaft die Arbeit als Zweitgutachter zu beurteilen.

Ich danke allen *fconv*- und *DSX*-Usern für ihr wertvolles Feedback. Insbesondere *Tobias Craan*, *Andreas Spitzmüller* und *Patrick Pfeffer* haben mit ihren Wünschen und Kritiken dafür gesorgt, dass Funktionsumfang und Zuverlässigkeit ständig zugenommen haben.

Meinem Bürokollegen *Sven Siebler* danke ich für viele interessante Diskussionen, für die gemeinsam geleistete Arbeit bei der Installation und Administration unserer IT-Ausrüstung, sowie für das immer angenehme Arbeitsklima.

Patrick Pfeffer und *Oliver Koch* danke ich für die Zusammenarbeit beim *DSFP*-Projekt. *Michael Betz* danke ich für Diskussionen über das Programm *DSX_wat* und seine Bereitschaft dieses Projekt weiterzuführen. *Timo Krotzky* danke ich für die Weiterführung des *DSX*-Projektes.

Besonderer Dank gilt *Antje Großmann*, die mich über einen großen Zeitraum der Promotion unterstützt hat und die maßgeblichen Anteil daran hatte das ich in der hervorragenden Arbeitsgruppe von Prof. Klebe gelandet bin.

Ich danke *Cornelia Koch*, die mich ebenfalls großartig unterstützt hat und die dafür gesorgt hat, dass ich beim Schreiben der Arbeit ein Ende finde.

Danken möchte ich auch allen Freunden die mir die Zeit in Marburg verschönt haben, insbesondere *Patrick Pfeffer, Tina Ritschel, Alexander Hillebrecht, Lisa Englert, Sven Siebler, Tobias Craan und Michael Betz.*

Lydia Hartleben danke ich für ihre ruhige Art mit der sie die Arbeitsgruppe organisatorisch zusammenhält.

Meinen Eltern danke ich dafür das sie mich immer in all meinen Vorhaben unterstützt haben und dafür das sie meine naturwissenschaftliche Weltanschauung geprägt haben.

Erklärung

Ich versichere, dass ich meine Dissertation

„Development and Improvement of Tools and Algorithms for the Problem of Atom Type Perception and for the Assessment of Protein-Ligand-Complex Geometries“

selbstständig ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen bedient habe.

Die Dissertation wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Marburg, den

.....

(Gerd Neudert)