Stratagems for Effective Function Evaluation in Computational Chemistry

Gwyn S. Skone

Keble College

Doctor of Philosophy

University of Oxford
Mathematical, Physical, and Life Sciences Division
Computing Laboratory

May 2010

Version 1.0 (beta) submitted November 2009 and examined February 2010.
Version 1.1 (revised) submitted April 2010 and approved May 2010.
Final printing July 2010.

Abstract

In recent years, the potential benefits of high-throughput virtual screening to the drug discovery community have been recognized, bringing an increase in the number of tools developed for this purpose. These programs have to process large quantities of data, searching for an optimal solution in a vast combinatorial range. This is particularly the case for protein-ligand docking, since proteins are sophisticated structures with complicated interactions for which either molecule might reshape itself. Even the very limited flexibility model to be considered here, using ligand conformation ensembles, requires six dimensions of exploration — three translations and three rotations — per rigid conformation. The functions for evaluating pose suitability can also be complex to calculate. Consequently, the programs being written for these biochemical simulations are extremely resource-intensive.

This work introduces a pure computer science approach to the field, developing techniques to improve the effectiveness of such tools. Their architecture is generalized to an abstract pattern of nested layers for discussion, covering scoring functions, search methods, and screening overall. Based on this, new stratagems for molecular docking software design are described, including lazy or partial evaluation, geometric analysis, and parallel processing implementation. In addition, a range of novel algorithms are presented for applications such as active site detection with linear complexity (*PIES*) and small molecule shape description (*PASTRY*) for pre-alignment of ligands. The various stratagems are assessed individually and in combination, using several modified versions of an existing docking program, to demonstrate their benefit to virtual screening in practical contexts. In particular, the importance of appropriate precision in calculations is highlighted.

Contents

List of Symbols and Abbreviations

Editions of the *DOX* software, testing various stratagems from this thesis, are denoted by boxed mnemonic codes in this document. A table describing these is given in Table E.3 (*p*.202).

3D	Three dimensions/dimensional	19
Å	Angstrom (length unit = 10^{-10}m = 0.1nm)	19
FFT	Fast Fourier Transform (algorithm)	36
GA	Genetic Algorithm (search method)	36
HAC	Heavy Atom Count (molecular size property = number of non-hydrogen atoms)	98
LUT	Look-Up Table (pre-calculated data structure)	65
MD	Molecular Dynamics (modelling technique)	39
MKB	Molecular Knowledge Base (data storage)	143
MOL2	Tripos Mol2 file (molecular file format)	213
NMR	Nuclear Magnetic Resonance (imaging method)	19
$\mathcal{O}$	Order of magnitude (worst-case complexity notation)	89
PASS	Putative Active Sites with Spheres (pocket detection method)	96
PASTRY	Pocket Alignment with Spheres and Thin Radial Yokes (shape descriptor/alignment method)	107
PDB	Protein Data Bank (structure repository and molecular file format)	19
PIECE	Pocket Identification by Encroaching Cubes for Efficiency (pocket detection method, derived from PIES)	95
PIES	Pocket Identification by Encroaching Spheres (pocket detection method)	87
PLP	Piecewise Linear Potential (scoring function)	61
PRM	Probabilistic Road Map (search method)	46

RCD	Rigid Cluster Decomposition (molecule analysis)	41
RMSD	Root-Mean-Square Deviation (displacement measure)	19
SA	Simulated Annealing (search method)	38
SAS	Solvent-Accessible Surface (molecular boundary)	24
SDF	Structure-Data File (molecular file format)	144
TCMD	Traditional Chinese Medicines Database (structure repository)	19
USR	Ultra-fast Shape Recognition (shape descriptor)	29
VdW	Van der Waals (atomic radius/interaction)	24
XML	Extensible Markup Language (structured data format)	178
ZINC	ZINC Is Not Commercial [Is Not Commercial...] (structure repository)	19

List of Figures

List of Tables

List of Listings

Acknowledgements

Any project of this nature is dependent on many people, both directly and indirectly. I am grateful to all those who have helped me in the last four years, even if simply with their friendship. Since attempting to list everyone would inevitably omit someone, I refrain from an absurd fawning roll-call here*, and mention only a significant few:

My great supervisory double-act —
Dr. Stephen Cameron (who took me on despite three years as my undergraduate tutor) and
Dr. Irina Voiculescu (who was largely responsible for my original arrival in Oxford);
InhibOx Ltd. for funding, facilities, and conversations over coffee;
Ben Fellows and Colin Gray for accommodation and encouragement;
Martin Esslin, Anthony O'Reilly, and TAFF for healthy distraction;
My family for unflinching support;
Rex and Pépe for entertainment; and
Carolina Johnson, my wife, for everything.

* I could offer an example from the references, but this margin is too narrow to contain it.

Chapter 1

Introduction

> "Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."
>
> *'Alice's Adventures in Wonderland', Lewis Carroll*

1.1 Biochemistry

The study of proteins is a significant field for chemistry, biology, and other sciences because of their central importance to the human body. These large (thousands of atoms) molecules have a clear structure: a backbone chain of amino acids folded into a stable globular arrangement. They can be found in many of the physiological systems: keratin is the principal component of skin and hair, myosin is required for muscle contraction, myoglobin transfers oxygen to muscles for respiration, and haemoglobin carries that oxygen in red blood cells. About 99% of the protein ingested as food is retained for use in the body, of which they comprise around 20–30%. An explanation of some basic protein terminology is given in Appendix A (*p*.171); see also [Branden & Tooze, 1999; Berg *et al.*, 2002].

Unsurprisingly, these molecules are often a major factor in the cause and effect of diseases. They can become damaged, their balance with other molecules can be upset, or foreign proteins (such as viruses) can interfere with normal activity. For example, the human recombinant beta-secretase protein is a cleaving enzyme, an overabundance of which is a cause of Alzheimer's disease. Many other diseases have had proteins identified as pharmacological targets, including Huntington's, Parkinson's, cystic fibrosis, and some cancers [Soto, 2001]. Consequently, investigating their behaviour and interactions — and how these may be influenced — is a major topic of medical research.

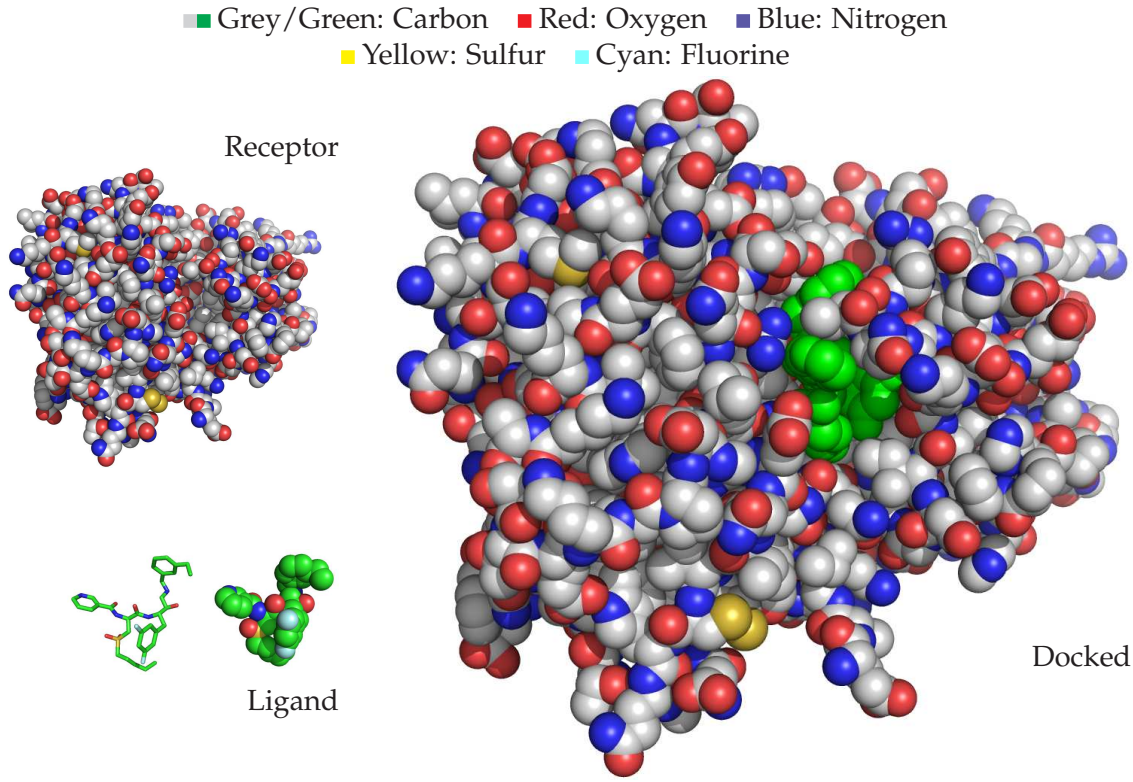


Figure 1.1: The human recombinant beta-secretase protein with a ligand proposed as a treatment for Alzheimer's disease (PDB code 2HM1)

1.1.1 Drug Discovery

Many drugs are now understood to work by delivering ligands (small molecules) into the body that chemically bind with the problematic protein, interrupting the undesirable behaviour [Lybrand, 1995]. For example: a potential treatment for Alzheimer's by inhibiting the beta-secretase protein is under development [Freskos *et al.*, 2007]. Figure 1.1 shows the molecular structures as space-filling atomic spheres, with the proposed ligand (isophthalate hydroxyethyl amine) bound in place. Identifying suitable ligands and developing them into pharmaceutical compounds like this is called rational drug discovery. There are three significant questions to be considered when assessing potential ligands: can it bind, how does it bind, and what effect does it have once bound? The first two are particularly suited to computer modelling *in silico*, since pursuing them by laboratory experiments *in vitro* (in glass (test tubes)) is extremely costly, and experimenting *in vivo* (in the body) without prior research is obviously unacceptable.

The rapid improvement of information technology through recent decades has facilitated the development of computational chemistry. Software models of molecules

have been devised, and these may be shared, studied, and used to simulate chemical properties. Techniques such as x-ray diffraction and nuclear magnetic resonance (NMR) imaging can reveal the physical arrangement of atoms. The ability to visualize simply this three-dimensional (3D) shape is of great use for understanding a molecule. However, it is the scope for algorithmic analysis which provides most interest, and computerized structural calculations have been used for more than 30 years [Bourne & Weissig, 2003; Richards, 2007]. Today, this is supported by the prevalence of publicly-accessible repositories such as the Protein Data Bank (PDB) [Bernstein *et al.*, 1977, `www.pdb.org`] with over 50000 entries, ZINC Is Not Commercial [Irwin & Shoichet, 2005, `zinc.docking.org`], and the Traditional Chinese Medicines Database (TCMD) [Kan *et al.*, 1996].

The search for ligands that could bind to an identified target is often now performed by computer, using high-throughput virtual screening. This refers to the processing of a database of ligand-like molecules — perhaps millions — and attempting to place each in a stable arrangement bound to the protein. Exploring these positions is called docking to a receptor, and the relative geometry of a docked ligand is called a pose.

Poses can be compared quantitatively: for two positions of the same ligand, the root-mean-square deviation (RMSD) is widely used as the measure of how close they are:

$$\mathbf{RMSD}(A, B) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{a}_i - \boldsymbol{b}_i|^2}$$

where the pose $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\}$, the $\boldsymbol{a}_i$ are the vectors of the atomic centres, and pose B is similarly defined. When performing a docking for which a naturally-occurring (or native) bound pose is known, an RMSD of 2Å (2×10^{-10} m) from that is generally considered to be an acceptable result, with 1Å being very good.

This application of computers to the problem of rational drug discovery — filtering possible inhibitor ligands before synthesis in the laboratory — is a proven science. Approved medicinal compounds, such as the human immunodeficiency virus (HIV) treatments Viracept and Sustiva and the influenza drug Zanamivir [von Itzstein *et al.*, 1993], have already been developed using these structural tools. Before computer modelling was introduced to the field, the preliminary research to identify and develop a new drug candidate would typically take 15 years [Lunney, 2001]. Virtual screening

is now a standard tool for pharmaceutical research [McGaughey *et al.*, 2007], and continues to improve as computational power increases. In order to leverage massive processing power, several public projects have used screensavers to distribute molecular modelling tasks across the world: *FightAIDS@home*, *Folding@home*, *Rosetta@home*, and the Cancer Screensaver Project (which also studied anthrax [Richards, 2002]) being notable examples.

Protein-ligand docking for high-throughput virtual screening remains an active area of interest, and has the potential to reduce further the time and financial cost of bringing new medicinal drugs to widespread availability [Lunney, 2001; Leach *et al.*, 2006].

1.2 Computer Science

Computer science can offer a different perspective on the existing techniques and tools, perhaps contributing new approaches to the problem of molecular modelling and docking in particular. Whilst the task may be concerned with microscopic objects, there are parallels with more readily visualized situations, and this appeals to the overarching field of spatial reasoning. Indeed, parallels can be drawn with robotics for motion simulation, as reviewed in [Parsons & Canny, 1994]. Besides geometry, though, general principles from pure computer science can be relevant to bioinformatics, offering refinements to the way molecular data are processed while still making use of established, proven methods.

Ligand docking is an ideal application for computational modelling because the combinatorial search required to find optimal alignments of even rigid bodies is massive. Although the issue of representation — how to describe chemical phenomena mathematically so that poses can be evaluated — is central, it must be surrounded by the question of what to do with those models once designed. That is a broad algorithmic problem with many general features worth studying, and is the subject of this work.

Certainly, there are some things that I do not propose to do here. Devising new models of molecular interaction would require a more detailed understanding of biochemistry. Comparisons of existing methods are useful (some are referenced in Chapter 2), but others would be better-placed to contribute them. Developing and/or

applying new search algorithms to the problem of molecular docking could improve efficiency, as could introducing more sophisticated behaviour for adaptively varying the search parameters or scope according to the contextual requirements and data. These possibilities are discussed in §8.2.3 (*p*.162), but investigating a new search method properly was beyond the available timeframe for this work. A related problem is that of how to represent molecular flexibility in (for example) a hierarchical pattern and use this for interaction simulation. Although I briefly study this in §3.2 (*p*.54), it has not been pursued with a detailed exploration, making way for the wider scope of investigation discussed instead.

1.2.1 This Thesis

The plan for this thesis was not to build an entirely new system for protein-ligand docking. It was to take a contemporary tool as an example, dismantle it to identify the significant aspects of its operation, and then try to reassemble the parts with improvements into a more efficient form. As a result, this work aims to demonstrate how a range of techniques and principles can be applied to virtual screening software in general. Chapter 3 (*p*.53) makes some abstractions about docking for virtual screening, providing a framework in which important features can be identified. This can also help to understand the components of such systems, clarifying the parts to be refined by strategic processing designs. Chapter 4 (*p*.69) then introduces stratagems for this purpose, discusses their relationships, and outlines the sequence in which they were originally devised and developed for application.

This work considers several aspects of docking program design, ranging over all levels of the problem from scoring function evaluation up to search method management, with the topics collected into three parts:

Geometric Guidance (Chapter 5) studies the representation of molecular shapes and their applications for guiding docking searches.

Efficient Exploration (Chapter 6) considers how best to perform these searches, and in particular how to avoid unnecessary work.

Properties, Priority, and Parallelization (Chapter 7) discusses the separation of docking procedures into independent units, their distribution for efficient processing, and how information might be gathered about them to learn from previous results.

All the suggested techniques presented in those chapters have been implemented in a commercial docking tool, *DOX* (introduced in §3.4 (*p*.64)). The principles encapsulated, however, are not specific to that system. Any existing or future molecular modelling tool for virtual screening could reasonably take advantage of the methods espoused here, supporting biochemistry research with efficient, effective computation.

Chapter 2

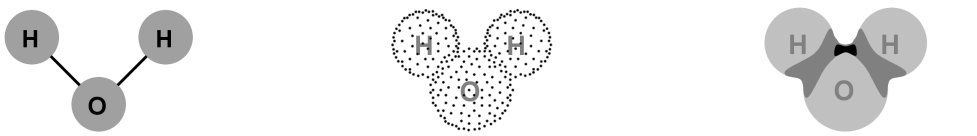
Literature Review

> Any man who reads too much and uses his own brain too little falls into lazy habits of thinking, just as the man who spends too much time in the theatres is apt to be content with living vicariously instead of living his own life.
>
> *Albert Einstein*

This chapter presents a survey of the field of molecular computation, and protein modelling in particular. It is a broad overview of recent and established work, with a focus on docking tools in preparation for my own foray into that area.

2.1 Molecular Representation

All computational biochemistry relies, to some degree at least, on a representation of the molecule(s) being studied. Depending on the particular focus of investigation, the precise data required may vary, and so there have been several different methods proposed. The simplest is the ball and stick model: this is based on the chemistry teacher's visual aid of small hard spheres representing atoms connected by thin rigid rods or springs. A cross-sectional ball and stick diagram of a water molecule is shown in Figure 2.1a. Since this is a direct analogy of currently-understood molecular physics, it is often taken as a basis from which other representations are formed. One common variation, known as space-filling, uses the atomic (or other) radii to draw solid spheres, overlapping where they bond. Several models are developments from this, including those mentioned below.



(a) Ball and stick (b) Dot surface (c) Surface patches

Figure 2.1: Three molecular structure representations

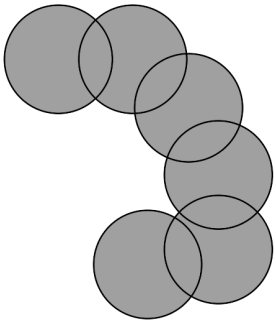
An excellent review of many common modelling designs may be found online [Connolly, 1996]; this includes details of 3D graphical output programs, links to organizations producing the work referenced, and summaries of known applications of the techniques described. More recent reviews discuss general shape representations [Putta & Beroza, 2007] and the use of pharmacophores [Leach *et al.*, 2010].

Connolly is a significant figure in the field, having contributed the dot surface representation [Connolly, 1983a] for calculating the external boundary of a molecule. This is a solvent-accessible surface, i.e. it is a shell of points around the molecule whereby a sphere (representing the solvent) may make contact with any such point without being penetrated by any other. Figure 2.1b shows a sketched projection of the dot surface for water. A simple description of the construction algorithm and methods of displaying the surface graphically are given in [Connolly, 1983b].

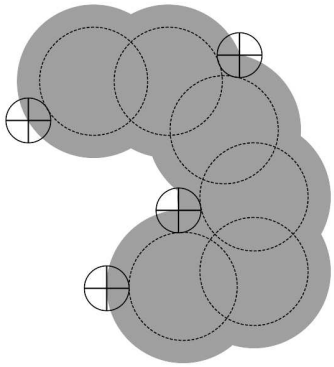
An extension of Connolly's method was developed by [Lin *et al.*, 1994]. This used the dots generated by the existing algorithm to identify critical points on the surface and the patches around them, categorized according to their approximate shape (convex, saddle, etc.), thus simplifying the model. In Figure 2.1c, the light grey patches are caps, the dark grey areas are belts (saddles), and the black regions are pits.

Since atoms are not solid, but a cloud of electrons around a central nucleus, defining their surface (and hence a molecule's) is a problem with several solutions. Figure 2.2 shows three common surface definitions, all based on the Van der Waals (VdW) radius. This is the contact distance between atoms of the same element, and so provides a good basic notion of the size of an atom. The solvent-accessible surface (SAS) is often used for interaction because it allows for the fact that some crevices are too small to be useful in docking (§2.3.2 (*p*.32)). It is defined as all points corresponding to the centre of a probe sphere in contact with the VdW surface. The smooth molecular surface is similar to the SAS, but using the probe's boundary instead, leaving a more rounded contour where atomic interactions are likely to occur.

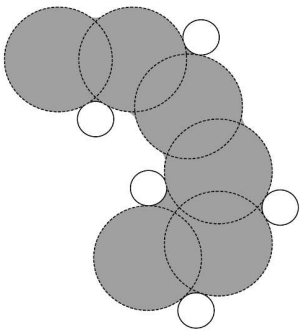
An alternative geometric representation, based on Delaunay triangulation, is given by the alpha-shape algorithm [Edelsbrunner & Mücke, 1994]. This allows a surface to be built from triangles around the molecule, with a parameterized approximation, and has been used for identifying significant features in the shape of a protein's surface.



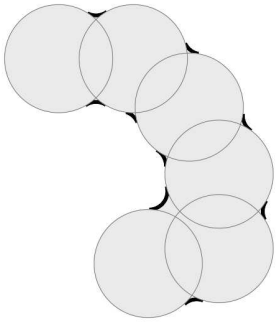
Van der Waals surface: equal to union of atoms' VdW spheres



Solvent-accessible surface: determined by probe spheres rolling on VdW surface



Smooth molecular surface (with probe spheres)



Smooth molecular surface (re-entrant sections emphasized)

■ Dark grey shading indicates the interior of the molecule

Figure 2.2: Four molecular surface definitions

It is sometimes useful to identify the secondary structure (regular patterns discussed in §A.1.2 (*p*.173)) of a protein, as well as its surface, perhaps for noting the less flexible sections of its backbone. One program for such analysis is *JOY* [Mizuguchi *et al.*, 1998], which takes data from the PDB and annotates the residues that form alpha helices, beta sheets, hydrogen bonds, or are accessible to solvents. It can also align common sequences of residues between two proteins.

Depending on the application for which a model is used, the geometric arrangement of atoms and the outer boundary of a molecule may well be insufficient for any useful results to be obtained. If a representation is to be physically and dynamically accurate, it must incorporate some element of the interactions between particles, as well as the attributes of the atoms themselves. Bonds between atoms (the stick in ball and stick) can take several forms, each of which has properties such as strength, torsional constraints, and elastic limits. These bonds introduce significant constraints to the molecule, which are often critical to the understanding of a dynamic system.

A review of early molecular modelling software is given by [Cohen *et al.*, 1990]. Two of the earliest programs developed for analysing these forces in detail were *AMBER* in the 1970s [Case *et al.*, 2005] and *CHARMM* [Brooks *et al.*, 1983; Brooks *et al.*, 2009], but these were preceded by a method reported for refining structural data by force field calculations [Levitt & Lifson, 1969]. They have been reused as calculation bases for other frameworks, such as *NAMD* [Phillips *et al.*, 2005].

Adaptations of Newton's methods have also been tried, with some success. An example, with discussion of progress and potential work, is [Ponder & Richards, 1987]; many of the other refinement articles mentioned later in this document include or cite similar minimization approaches. More recently, methods such as the Mining Minima Optimizer [Head *et al.*, 1997] and Convex Global Underestimator [Foreman *et al.*, 1999] have emerged to improve the reliability of calculations founded on energy considerations. A further recent development of these has been the Iterative Convex Quadratic Approximation minimization algorithm [Marcia *et al.*, 2005], which is designed to cope with many local minima successfully.

Hierarchical Representations

When considering the flexibility of a molecule, in particular when rigid regions have been identified and locked, it is helpful to represent the structure in a hierarchical or multi-scale form. This prioritizes the modes of flexibility according to their likelihood and range, keeping the rigid regions together as leaves of the tree. The generation of such a tree may be done using a decomposition method such as in *FIRST* [Jacobs *et al.*, 1999], or using a simpler atom clustering technique such as the k-means reduced point method [Glick *et al.*, 2002a; Glick *et al.*, 2002b].

When exploring molecular conformation space, Monte Carlo methods can be initialized using information about protein domains to improve their efficiency. This allows some hierarchical information to be used for simplifying simulated annealing calculations [Maiorov & Abagyan, 1997].

Folding prediction can also be helped by such multi-scale methods by incrementally building stable units and composing those in turn. The applicability of multi-scale methods to molecular simulation is wide; a general framework in which to build such

systems is the *MMTSB* Tool Set, which allows existing force fields, models, and other simulation scripting programs to be combined with a hierarchical architecture [Feig *et al.*, 2004]. Even grid-based calculation methods can have multi-scale principles applied, where both long and short range factors are combined [Skeel *et al.*, 2002].

Another use for hierarchical representations is the comparison of molecular shapes. The removal of fine detail at intermediate levels allows more generalized shape descriptions to be matched [Sharf & Shamir, 2004].

Quantum Physics

In addition to geometry and dynamics, the minute scale of the systems under consideration means that quantum effects may not be negligible. This is an aspect of molecular modelling that seems to have remained a speciality, and not commonly incorporated into applications. Most software algorithms developed for protein modelling tend to ignore quantum behaviour, since it is hard to solve and possibly of uncertain relevance. However, it has been argued [Miller, 2005] that it is unsafe to dismiss these issues without consideration, and there are some tools in existence which do provide for quantum calculations in protein modelling [Peters *et al.*, 2006; Raha *et al.*, 2007; Taft *et al.*, 2008; Rosales-Hernandez *et al.*, 2009].

An example of an application of quantum physics to the molecular modelling problem is the representation of hydrogen bonds. [Morozov *et al.*, 2004] describes the significance of these concerns to the proper description of hydrogen bonds in the context of protein structure.

It is suggested that combining quantum and classical mechanical approaches to the protein modelling problem is a useful direction of enquiry, since it is likely that the larger-scale interactions may show insignificant quantum effects to be worth the higher complexity calculations. Hence, hybrid methods have been investigated, and a good survey of the implementations and the principles at work is [Sherwood, 2000].

An extremely detailed description of *Q-Chem 2*, an implementation of many important quantum calculations, is provided in [Kong *et al.*, 2000]. This is a general package for electronic structure analysis, and not a biomolecule-specific system. Another

program for performing quantum chemistry simulations is *GAMESS* [Schmidt *et al.*, 1993], with its associated graphical viewer program *QMView*. Quantum algorithms such as these are readily and usefully parallelized, and the advent of the internet and distributed computing has proved beneficial to scientists using such packages. [Baldridge *et al.*, 2002] discusses the practicalities of running a system such as *GAMESS* in a grid architecture.

2.2 Molecular Comparison

An area of study related to molecular representation is structural comparison. If two molecules have been stored in some computer-based description, it may be useful to have some measure of their structural similarity. This could be measured at the primary level using common subsequences of amino acids, or at higher levels with (for example) duplicated groups of helices. This may be useful for identifying families of proteins with common functional behaviour or ligand binding affinities. In theory, two proteins sharing a significant proportion of their primary or higher-level structure will have a relatively recent evolutionary ancestor, and so should have more in common than two completely dissimilar molecules. For a selection of practical examples demonstrating the relevance of structural likeness, see [Orengo *et al.*, 1999; Eckert & Bajorath, 2007].

Comparative techniques can assist with the folding problem (§2.3.1 (*p*.30)). If one structure is known in detail, and another protein is known to have a significantly similar primary structure, then there is a likely and useful starting point for the physical conformation search. This is explored in [Sánchez & Šali, 1997].

Restricting the structure comparison to a small section of a molecule can provide a useful tool for the docking problem (§2.3.2 (*p*.32)). By using a comparison algorithm to test the similarity of one molecule with the *inverse* of another (i.e. the complement of its spatial occupancy), the suitability of the two geometries for binding may be assessed. This is based on the assumption that ligands can only dock where there is sufficient geometric complementarity, like a jigsaw puzzle. An implementation of this, using techniques from computer vision and geometric hashing, is given in [Fischer *et al.*, 1993b]. A more recent geometric hashing method, based on neighbouring triples of atom types, is that of [Kinnings & Jackson, 2009].

A genetic algorithm for the comparison of Connolly dot surfaces is presented in [Poirrette *et al.*, 1997], using mostly geometric alignment with some hydrogen bond comparisons to rate the placement of a trial surface. Small molecule shape descriptors which can be used to compare ligand shapes include spherical harmonics [Ritchie, 1998; Ritchie & Kemp, 1999; Morris *et al.*, 2005] which have also been used for docking [Mavridis *et al.*, 2007], the Ultra-fast Shape Recognition (*USR*) method [Ballester & Richards, 2007], and the Gaussian functions and histograms presented in [Kazhdan, 2004].

A popular structure comparison engine is the *Dali* system, developed at the European Molecular Biology Laboratory. This provides both an interactive web server and an email server for receiving a protein structure (as a set of atomic coordinates) and comparing it with all the molecules in the PDB to return a list of close relatives. The algorithm is based on pairwise distances between alpha carbons, using statistics from empirical findings to weight the combinations. For a full explanation of the methods employed and some protein family trees, see [Holm & Sander, 1993]; a more succinct summary of *Dali*'s uses is given in [Holm & Sander, 1995].

The algorithms used by the *Dali* server have been incorporated into a stand-alone program of use for simple comparisons between two molecules, returning a result file showing the common sections of the backbones and a score indicating the closeness of the match. This program (which is also available as an online web interface) is called *DaliLite* [Holm & Park, 2000; Holm *et al.*, 2008]. The *Dali* team also produced a summary of three distinct methods (*Suppos*, *Comp3D*, and *Dali*) for structure comparison, and described some useful published datasets of protein groupings [Holm *et al.*, 1992]. These were generated from the PDB by a combination of these three programs.

2.3 Protein Structure Prediction

There are two major fields of computational research concerning proteins: structure prediction and automated docking. Both have been active areas of work for several years, and there are several organizations and facilities in place to support their development. Most notably, the Protein Data Bank (PDB) [Bernstein *et al.*, 1977; Berman *et al.*, 2000]

has made a large repository of existing protein structure data publicly available, with facilities for indicating the motifs of secondary and tertiary structure, other atoms involved in mediating hydrogen (or other) bonds, and notes concerning possible flaws and other details regarding the information. All these data are provided in a number of formats which may be parsed easily, and have been adopted by many protein structure based projects as a standard method for data transfer.

To aid the assessment and development of new methods in computational biochemistry, international meetings of researchers have been organized to demonstrate and compare results in a friendly competition environment. The most relevant of these are the Meeting on Critical Assessment of Techniques for Protein Structure Prediction (*CASP*) and its sibling Critical Assessment of Predicted Interactions (*CAPRI*) [Wodak & Méndez, 2004]. Many of the methods that have been introduced have reported on their results from these events, which helps to give an impression of their success and a reference point for comparison.

An indication of the importance of proteins as a sphere of study may be found in [Burley, 2000]. This cites developments in crystallographic imaging techniques, along with achievements such as the Human Genome Project [Collins & McKusick, 2001], as examples of important work that demands continued effort to develop their benefits. An excellent and extensive review of computation projects from all aspects of protein modelling is given in [Xu *et al.*, 2000]. A recent review, reporting from a pharmaceutical conference, is [Snow, 2008]. Assessment of protein structure models for folding prediction is discussed in [Kihara *et al.*, 2009].

2.3.1 Native Structure Prediction — The Folding Problem

The fold of a protein — the geometric conformation of the chain of backbone bonds in the molecule — is of interest in bioinformatics because it can be used to identify classes of proteins with similar function and/or behaviour (especially with regard to docking). In addition, an understanding of how proteins fold into their native conformation can help to analyse how and why they may fold incorrectly on occasion. Malformed proteins are known to be a cause of several debilitating diseases [Soto, 2001], and the role of folding prediction for drug discovery is discussed in [Grant, 2009].

The study of how proteins fold has been ongoing for around half a century. The development of tools such as nuclear magnetic resonance (NMR) and facilities such as the PDB have undoubtedly helped to sustain and propel this research, but the advent of computer technology for automating the prediction methods being proposed has been a significant step forward. It has been noted [Honig, 1999] that the prediction of protein structure is not the same as understanding how it folds, but that these are related. The former may be helped by the latter, but this is not the only means of determining a protein's atomic layout.

In 1999, the *Current Opinion in Structural Biology* journal included a section describing the latest understanding of native protein structures. The editorial reviews the work done, and summarizes the basic principles of folding [Dobson & Ptitsyn, 1999]. The following article in the same journal gives a much more detailed description of the topics and summarizes the Levinthal Paradox, the motivation for developing algorithms for folding other than naive combination evaluation [Dobson & Karplus, 1999].

Two significant theories in the biochemical folding problem are directed pathways (funnels in the intramolecular energy landscape [Bryngelson & Wolynes, 1987]) and chaperones (molecules present in the environment that guide the protein as it folds [Ellis & Hartl, 1999; Hardesty *et al.*, 1999]).

A review of computer algorithms for predicting protein folds, and in particular the sources of error in such methods, may be found in [Finkelstein, 1997]. A diverse range of techniques have been tried, including Markov models [Karplus *et al.*, 1998; Karplus *et al.*, 1999], Monte Carlo potential constraint simulations [Kolinski & Skolnick, 1994; Skolnick *et al.*, 1997], and the prediction of likely geometry using empirical knowledge based on recognized peptides [Baker, 2000]. Recently, methodology from machine learning has been applied to a scoring-based system of string kernels [Rangwala *et al.*, 2006]: this report also includes an excellent introduction to the essential aspects of folding and structure prediction. These reviews are updated by [Dill *et al.*, 2007; Chen *et al.*, 2008; Dill *et al.*, 2008].

The folding problem may also be reversed: given a desired structure, can an amino acid sequence be selected that would fold to that shape? This too has been investigated: a recent thesis describing four algorithms for the purpose is [Hom, 2005]. One potential

benefit of such work is the synthesis of good proteins that might be beneficial to the body in fighting disease, either by displacement of mis-folded proteins or supplementation of desirable ones.

2.3.2 Complexed Structure Prediction — The Docking Problem

The prediction of how two molecules might bind together is of great importance in bioinformatics because many proteins bind ligands as a central part of their function. For example: haemoglobin is the protein that provides the oxygen carrying capacity of blood. It shows four domains in the tertiary structure, each of which has a haem group at the centre, containing an iron atom. The usual ligand, an oxygen molecule, is able to bond with one of these irons until the blood carries it to a part of the body with low O_2 concentration and it breaks off again [Shikama & Matsuoka, 2003]. In fact, other molecules can bind to haemoglobin, notably carbon monoxide, which is the cause of carbon monoxide's toxicity.

The problem of automated docking prediction is complicated by the possibility that either molecule may flex as it interacts. To use the haemoglobin example again, the binding of an oxygen molecule causes a small shift in and around the domain containing it. As a consequence, the forces present become increasingly favourable for binding as the haem groups are filled, up to the point where all four are occupied. As oxygen is removed, the protein returns to its native conformation. This flexibility in proteins (and ligands) can be critical for binding.

There have been a number of reviews of computational tools for protein docking prediction. [Jones & Willett, 1995] summarizes the methods, programs, and directions of research from the time; this is updated by [Lengauer & Rarey, 1996; Taylor *et al.*, 2002; Smith & Sternberg, 2002], and more recently by [Perola *et al.*, 2004; Kontoyianni *et al.*, 2004; Leach *et al.*, 2006]. These articles differentiate between protein-ligand and protein-protein interactions: the principles are similar, but in the latter case the second molecule is much larger and may have an active site of its own. A review of free tools for all aspects of computational drug discovery, including website references, is given in [Villoutreix *et al.*, 2007].

One of the central assumptions about docking is that the two molecules must have a sufficient geometric complementarity to bind: that is, they must fit together physically. If the surfaces are of completely different topographies, there is unlikely to be sufficient surface contact to provide a good bond. Early docking prediction was achieved by manual methods, with the computers providing some form of scoring to assist the human operator in deciding how to improve a visually aligned pair of molecules. A graphics terminal would display a simple rendering of the two molecules (typically by their VdW surfaces), and the user would manipulate one of them until they appeared well juxtaposed. A very early example without energy calculation is [Busetta *et al.*, 1983]. A better demonstration of the technique is [Pattabiraman *et al.*, 1985], where the interaction energy (Coulombic plus VdW) of the complexed molecule is calculated and shown in real-time to facilitate a more interactive style of alignment.

A relatively recent and novel variation on the manual docking method is introduced in [Nagata *et al.*, 2002]. This prototype uses force feedback technology to drive a simulator for ligand docking. The user is again presented with a graphic rendering of the molecules' structures, and a joystick (albeit a specially designed one) is used to move the ligand into position. However, the system calculates the repulsion from the electrostatic forces between the molecules, and uses this to drive a tactile output in the joystick. The intention is to allow the operator to discover the optimal binding site for a ligand by feeling their way around the protein. The prototype system was limited by available computing power — the graphics and joystick feedback were handled by separate machines linked by a network — and several areas for improvement are identified at the end of the paper.

For a thorough exploration of the mechanisms, principles, and methods of protein docking, set in the context of the PDB see [Jones & Thornton, 1996]. Another review is [Halperin *et al.*, 2002]; one with particular emphasis on screening drug candidates is [Kitchen *et al.*, 2004]. Several significant figures from the field contributed an introductory chapter to a book describing the general process of drug discovery and how molecular docking is applicable [Lang *et al.*, 2007]. A more general summary is given by [Morris & Lim-Wilby, 2008], the concluding chapter to a molecular modelling book. For some recent advances and success stories, see [Song *et al.*, 2009; Villoutreix *et al.*, 2009].

Active Site Prediction

As briefly mentioned above, proteins often exhibit an active site: a particular area of the molecule where ligands (or some usual ligand) normally bind. This is sometimes a crevice or other prominent geometric feature, but may also be an area with smoother shape but favourable electrostatic forces (for example). Identifying the active site (also known as the binding site) of a protein is likely to help in the assessment of whether and how a ligand might be docked. For those proteins whose native and complexed structures are documented, it is sometimes possible to determine the active site quite simply; using knowledge derived from known structures, predictive methods may be developed.

Active site proposals can be generated as a side-effect of docking algorithms by assessing a scoring function with a probe ligand. [Kuntz *et al.*, 1982] gives an example of this, in the context of rigid geometric docking.

Reviews of active site prediction techniques are given in [Sotriffer & Klebe, 2002; Campbell *et al.*, 2003; Laurie & Jackson, 2006] with reference to its applications in the more general docking problem. It can be seen from this that the shape of the molecule alone is not always sufficient data from which to reliably identify the active site. Evolutionary tracing — based on functional and structural similarity (§2.2 (*p*.28)) — has been used with experimentally determined sites to provide a basis for predicting the sites on related proteins [Skolnick & Brylinski, 2009]. In addition, empirical data about the tendencies of certain peptides to be present at binding sites can be used to estimate probabilities that some section of a backbone is part of the active region [Lichtarge & Sowa, 2002; Yao *et al.*, 2003]. Alternatively, the understanding of a receptor's function in terms of known patterns of peptides folded near the surface can help identify sites [Wass & Sternberg, 2009].

An earlier algorithm for identifying active sites primarily on the basis of geometry (using the alpha-shape representation) is given in [Peters *et al.*, 1996]. For a method that does not consider the biochemistry of the molecules, some very good results were obtained: the author claims that 95% of the test cases were solved correctly, although rarely with all atoms involved in binding identified. However, restricting a docking

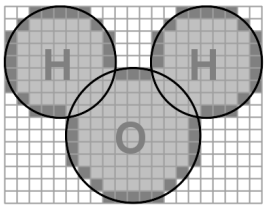
search to a particular region of a protein is a great improvement over a global search, and so this is still a useful achievement.

Analysis of a molecule's SAS (see *p*.24) can identify closed internal cavities, although this work may not detect open cavern active sites [Alard & Wodak, 1991]. Other geometric algorithms include the grid-based methods *POCKET* [Levitt & Banaszak, 1992] and *LIGSITE* [Hendlich *et al.*, 1997]. Sphere-based methods exist to provide arguably more precise results, such as *SURFNET* [Laskowski, 1995] and *PASS* [Brady & Stouten, 2000], which stack probes on the molecule's atoms and identify dense regions of these. It is often the case that geometry alone can be useful, since the largest cleft in a receptor's surface is often the correct one [Laskowski *et al.*, 1996].

A method that starts from geometric data but then refines its search using hydrophobicity and an atomic interaction scoring function is presented in [Ruppert *et al.*, 1997]. This places many probe spheres around the surface of the protein, and then ranks them according to their individual local binding affinity. It then clusters the most promising scores, and the highest ranked cluster is assumed to be the binding site. A more recent algorithm for active site prediction integrated with docking is implemented in the *LigandFit* program [Venkatachalam *et al.*, 2003].

Geometric Alignment

One of the longest-established rigid docking tools with an automated search is the *DOCK* program [Kuntz *et al.*, 1982; Lorber & Shoichet, 1998]. This remains a popular tool for generating conformations, sometimes used as a starting point for other refinement studies [Wei *et al.*, 2002]. This was initially a purely geometric method (although in recent years it has been extended to use many other techniques as well); one of the first improvements was the ellipsoid algorithm [Billeter *et al.*, 1987]. A near-contemporary geometric method was developed at Texas A&M University, applying combinatorial principles to both protein docking and graph clique detection problems [Kuhl *et al.*, 1986]. A review of five geometric matching algorithms, including Kuhl's, suitable for incorporation into the *DOCK* program (then in its fourth incarnation) was produced by *DOCK*'s authors [Ewing & Kuntz, 1997]. This concluded that Kuhl's work and the single graph matching method developed from it would be the best route forward for their rigid approach to the docking problem.



■ Black circles: true VdW radii of atoms.
□ White cells: external, zero score for FFT docking;
■ Dark grey cells: surface coating, positive score;
■ Light grey cells: buried interior, larger score of either sign.

Figure 2.3: One slice from a possible spatial occupancy grid for water (H_2O)

A simple but generally useful geometric method assigns values to cubes of the molecule's space in a lattice: positive for the surfaces/boundaries, negative for one interior, positive for the other, and zero externally. Figure 2.3 illustrates such a matrix for water. The score for a pose is the summation of the products of overlapping grid values. The precision of the method is determined by the spatial occupancy matrix: using more elements for a smaller volume each improves the match between representation and reality.

The calculation of this function is efficiently implemented with the convolution operation [Katchalski-Katzir *et al.*, 1992], observing that the inverse Fourier transform of the product of the Fourier transforms of two input functions is proportional to the convolution of those two inputs [Papoulis, 1962]. Hence, the well-established Fast Fourier Transform (FFT) algorithms can be used to implement a rapid version of the exhaustive search. Some extensions of the method are discussed in [Vakser, 1996; Vakser *et al.*, 1999]. A more recent validation of the basic procedure is given in [Tovchigrechko *et al.*, 2002].

Where a complete scoring function is not used over the entire molecular space, topographical features of the surface are used to reduce the amount of data being considered. These sections are then aligned, with little regard for other parts of the structure. One method, applying geometric hashing techniques for improved performance, is given in [Fischer *et al.*, 1993a]; a more recent variation is the Quadratic Shape Descriptor [Goldman & Wipke, 2000a; Goldman & Wipke, 2000b]. This method introduces a scale for assigning continuous real values to the contour shapes of surface patches, then uses this to compare patches from a protein and a ligand.

Genetic algorithms (GAs) have been investigated as another means of searching the large combinational space efficiently. The usual approach involves generating a population of 6-tuples (three rotation and three translation values), scoring the

population according to geometric overlap and/or some other function, then breeding a new generation from the fittest candidates. This process is repeated for a predefined number of generations, whereupon the highest-ranked surviving candidate is the result.

The *AutoDock* program [Morris *et al.*, 1998] was originally based on simulated annealing (see *p*.38), but has been adapted to use both traditional and Lamarckian GA searches. Lamarckian GAs use small local searches to refine the scores at each generation.

The *KENOBI* program uses the motifs of secondary structure to guide such an alignment, scoring according to the alignment of alpha helices and beta strands [Szustakowski & Weng, 2000]. Connolly's dot surface representation was used as the basis for another program [Gardiner *et al.*, 2001], developed from a GA for surface comparison [Poirrette *et al.*, 1997].

Geometry is not the sole factor determining a good docking solution, and as such is not the only direction from which the problem may be approached (although it may well be the easiest). A more chemistry-oriented method is used in the *LIGIN* program, described by [Sobolev *et al.*, 1996]. This algorithm categorizes the atoms in the protein and ligand using eight labels — hydrophilic, hydrogen-bonding, etc. — and uses a flexible polyhedron search method to refine many randomly-generated ligand positions according to both shape and the complementarity of adjacent atom types.

Electrostatics and Dynamics

The original FFT method of [Katchalski-Katzir *et al.*, 1992] was open to improvement, and a few attempts have been made to extend the procedure to incorporate other factors than surface contact alone. Electrostatic interaction — the forces of attraction and repulsion between the atomic charges — is an important consideration in molecular work. A straightforward solution is shown in [Gabb *et al.*, 1997]: this gives an algorithm whereby the Coulombic electric field of the molecules is represented in the same manner as their spatial occupancy for FFT-based convolution. This allows both geometric and electrostatic alignment to be carried out simultaneously.

A more comprehensive energy function computed over a grid and used with the FFT method is given in [Mandell *et al.*, 2001]. This combines both the VdW interaction

with Poisson-Boltzmann electrostatics, while accounting for any hydrophobic effects by weighting VdW terms. The total function is the summation of these. Despite the more precise energy function, however, the reported results are not significantly better than those of [Gabb *et al.*, 1997].

ZDOCK is a recent FFT-based program that uses a combined scoring function of electrostatics, desolvation, and shape complementarity [Chen *et al.*, 2003]. It produced some reasonable docking results, and is the basis for the *RDOCK* system (see below).

An attempt to improve the speed of the basic FFT method was proposed, and implemented as the program *Hex*, in [Ritchie & Kemp, 2000]. This suggests the use of spherical polar representations of the molecules, converting the traditional translation and rotation 6-tuples into a single intramolecular distance and five Euler angles. Most recently, a system called *PIPER* has been developed from the original FFT method using potential-based scoring and eigenvectors for calculation [Kozakov *et al.*, 2006].

An alternative to the FFT for fast grid searching is the use of Boolean operations over matrices [Palma *et al.*, 2000]. This method only allows geometric considerations in the grid search, but can exploit low-level bitwise manipulation to find initial alignments to be ranked using a function incorporating other considerations.

Simulated annealing (SA) is a time-step refinement procedure. A recent docking program based mostly on SA is *HADDOCK* [Dominguez *et al.*, 2003]. The two molecules are placed about 150Å apart, at random orientations. They are then subjected to energy minimization by rotation alone, then by translation and rotation. This, unusually, was implemented in the Python scripting language.

Another algorithm for energy minimization is the Mining Minima process [Head *et al.*, 1997]. This has been used to guide a ligand docking procedure [David *et al.*, 2001], and further refined to include flexibility into the molecules [Kairys & Gilson, 2002]. This has produced some very good results.

Explicitly geometric algorithms (such as Fischer's computer vision method [Fischer *et al.*, 1993b]) cannot have electrostatics so directly incorporated, owing to the lack of a grid-based function or universally defined scoring method. In these cases, it is sometimes useful to use the shape docking technique, and then attempt to refine the result using

energy considerations. The *RDOCK* program [Li *et al.*, 2003] is an example of this: it uses the authors' previous rigid docking project *ZDOCK* to produce initial guesses for docked conformations, then applies three energy minimization algorithms to each complex, searching for a better arrangement. Although the original *ZDOCK* performed well at the first *CAPRI* challenge, *RDOCK* has improved the docking results.

The *LigandFit* program also uses a grid-based method to identify binding sites and generate alignments, then refines the possibilities using energy calculations [Venkatachalam *et al.*, 2003]. The active sites are identified using a flood-filling algorithm, then the ligand is randomly placed in the site and tested for an approximate shape match. Successful placements are ranked in a list of the conformations seen. Finally, the best N arrangements are refined by a Broyden-Fletcher-Goldfarb-Shanno rigid-body method, a variation on gradient descent. This has resulted in some very good docking results.

A summary of the application of Molecular Dynamics (MD) and free energy calculations to the problem of ligand binding is given in [Åqvist *et al.*, 2002]; for comprehensive methods reviews see [Adcock & McCammon, 2006; Alonso *et al.*, 2006; Skjevik *et al.*, 2009].

Quantum Effects

For a discussion of the general techniques and implementations used for performing calculations of quantum effects in molecular representations, see *p*.27.

One algorithm that explicitly considers quantum mechanical issues is the Quantum Stochastic Tunnelling (*QSTUN*) method [Todorov *et al.*, 2003]. This is an optimization method based on SA, using Monte Carlo techniques to explore the potential energies of the conformations under consideration while avoiding traps in local minima by transforming the energy landscape non-linearly (the *STUN* algorithm). The *QSTUN* method has been implemented in the program *EasyDock* [Mancera *et al.*, 2004], and although this is not significantly more accurate than other tools available, it does run faster for moderately flexible ligands.

Another implementation that makes use of quantum effects in docking has been developed for the *CHARMM* [Brooks *et al.*, 2009] molecular calculations program. These

scripts combine quantum mechanics and classical dynamics with a Poisson-Boltzmann model to form a representation of the ligand for docking. An evaluation of the suitability of this method is given in [Gräter *et al.*, 2005], which concludes that the energy calculations are quite valid for the purpose, producing some good docking results.

Flexible Docking

In reality, the (widely-used) assumption that molecules are rigid ball and stick arrangements is invalid. Very few molecules are completely inflexible; even with the stability introduced by secondary structure motifs, proteins can bend or twist and often do when a ligand is introduced (such as in the description of haemoglobin on *p*.32). The necessity of flexibility in the molecular representation is demonstrated well in [Erickson *et al.*, 2004], where the amount of movement in a protein clearly affects the accuracy of docking results in well-established software tools; the observation is supported by [Fradera *et al.*, 2002; Teague, 2003].

The rigid docking problem is relatively simple to formulate, even if not so easy to perfect, since there are only six parameters to a solution: three dimensions each of translation and rotation for the ligand relative to the protein. Once either molecule is permitted to deviate from its initial shape, the complexity of the task is multiplied many times over [Cavasotto & Orry, 2007].

To avoid this complication, some rigid docking procedures allow a soft docking factor, whereby the surface of the protein is thickened to allow a non-perfect alignment to be accepted if it is close to an active site — this simulates a very limited amount of overall bending in either molecule. This can be seen in most of the FFT-based methods, and also in the feature-matching method of [Schneidman-Duhovny *et al.*, 2003].

A compromise between full receptor flexibility and none at all is the use of side chain rotation in the representation: while the backbone is held rigid in some chosen arrangement, the short amino acid side chains are twisted according to some force field. This technique has been growing in popularity for docking, such as in the *RosettaLigand* program [Meiler & Baker, 2006; Davis & Baker, 2009].

A hybrid solution to the problem is the refinement of approximate docking results. Generally, this would involve a rigid docking run to produce the best inflexible

conformation, followed by some adjustment mechanism to settle the arrangement into a more favourable interaction. The dynamics of the approximate complexed system have been used quite successfully in this way [May *et al.*, 2003].

The flexibility problem, when approached more thoroughly, may be divided into two separate issues: the determination of how a molecule could move, and the application of this information to the docking representation. A notable approach to the former task was implemented in a real-time program called *FIRST* [Jacobs *et al.*, 1999; Jacobs *et al.*, 2001]. This work exploits the fact that, although proteins are pliable in general, they do tend to form quite rigid substructures (helices, sheets, domains, etc.). The purpose of *FIRST* is to perform a rigid cluster decomposition (RCD), i.e. to identify which atoms within a protein are immobile relative to which others. This is achieved using an algorithm called the Pebble Game, a graph-based distance constraint analysis method. The relevance of such work is emphasized by the prevalence of rotatable bonds in protein side chains [Rader *et al.*, 2002].

This work has been used in various projects (*ROCK*, *FRODA*, and *RCNMA* to name three) to build conformation databases for later docking use [Lei *et al.*, 2004; Wells *et al.*, 2005; Gohlke & Thorpe, 2006]. The importance of such information to the docking problem has been recognized [Carlson & McCammon, 2000], and working applications have also been published [Zavodszky *et al.*, 2004].

Other methods of identifying protein flexibility have been explored, including neural networks trained with primary structure sequence data and predicted secondary structure motifs [Schlessinger *et al.*, 2006]. A review of some more direct approaches is given in [Teodoro *et al.*, 2001], along with an algorithm based on singular value decomposition of matrices derived from the dynamics of the molecular system. Another brief summary of flexible docking schemes may be found in the introduction to [Diller & Merz, 2001]. A description and comparison of four techniques (random walk, simulated annealing, stochastic approximation with smoothing, and terminal repeller unconstrained subenergy tunnelling (*TRUST*)) is performed in [Diller & Verlinde, 1999], where both speed and accuracy are assessed.

Conformation Ensembles The information gathered about molecular flexibility may be applied to most models of structure, even if only in a rudimentary manner. For a

number of years, flexibility has been incorporated mostly as a refinement step, typically by random perturbations of otherwise rigid models: [McMartin & Bohacek, 1997] for example. A similar, although arguably more systematic, approach is to use conformation databases or ensembles [Lorber & Shoichet, 1998]. This pre-generates a list of possible final states for the flexible molecule, then passes each in turn to a rigid model for assessment.

Plainly, this is a naive solution, but in some cases it may help to avoid significantly mis-evaluating good results. The main disadvantage of this is that it does not allow for induced changes in the course of molecular interaction. These conformation ensembles are the basis of the *FLOG* docking program [Miller *et al.*, 1994], which was based on the same methodology as the *DOCK* system, and also the work of [Diller & Merz, 2001].

The same principle has been applied to the receptor molecule. Clearly, since this involves a much larger number of atoms, there will be many more conformations to generate and test, but otherwise the process is the same. A recent example of this is [Cavasotto & Abagyan, 2004], where MD calculations are used to find backbone arrangements for the protein. Some good results appear to have been produced, which underlines the theory that receptor flexibility is important to the docking problem [Totrov & Abagyan, 2008]. Several significant researchers in the field collaborated on an extensive perspective article on the matter [Cozzini *et al.*, 2008]; other reviews of methods are [McCammon, 2005; B-Rao *et al.*, 2009].

Similarity Two modified versions of the *DOCK* program were developed to use similarity between structures as an additional factor for inclusion in a docking method [Fradera *et al.*, 2000]. This is designed to exploit whatever prior knowledge might exist about either the protein or ligand involved in a docking procedure by weighting conformations according to whether they do or do not resemble a hypothesis arrangement. For example, if a protein P is known to bind a ligand L, the structurally guided docking method for P with new ligand L' will favour conformations where L' is in some way similar to the bound L. This algorithm has produced good docking results, although it does depend on the pre-determined crystallographic structure data available.

A variation of *DOCK* uses similarity to organize ligand conformations from the ensemble into a hierarchy, and can then discard branches of this tree if their shapes do not fit the target binding site [Lorber & Shoichet, 2005].

The same techniques were developed into a program called *MacDOCK*, which was again built on the *DOCK* foundation [Fradera *et al.*, 2004]. This allowed the weighting of scores to be guided by either the structural similarity of the ligand or the arrangement of a few anchor points involved in covalent bonding. In some test cases, a hybrid of both ligand- and anchor-based weighting proved more successful at predicting docked conformations. One of the main contributions of *MacDOCK* was the demonstration that the consideration of covalent bonding can improve the accuracy of docking methods, as shown in its improvement on *DOCK*'s basic method.

GAs Genetic algorithms lend themselves to the flexible docking problem by allowing conformations to change in as many variables as required. An example of the successful application of this technique is the docking of a potential HIV inhibitor by evolutionary programming [Gehlhaar *et al.*, 1995]. Evolutionary programming is a variation on a typical GA, in which the scoring function is used to compete the members of each generation between themselves: random pairs of candidate solutions are compared using the scoring (energy) function, and those that win the most competitions are used to build the next generation by the usual genetic operations. Gehlhaar's method allowed all nine rotating bonds in the ligand to move, and was able to reproduce the correct docked arrangement several times.

The *GOLD* program is another GA implementation for fully flexible docking. It is described in great detail in [Jones *et al.*, 1997; Verdonk *et al.*, 2003], where much consideration is not only given to whether it works, but also why it fails when it does. One interesting case given is a fairly good docking result (judged by comparing graphical renderings of the molecules) with a relatively poor root-mean-square deviation (RMSD), used to make the point that mid-range RMSD values should be investigated carefully using graphical tools to ascertain whether they are in fact better than might be assumed. Indeed, using aggregated RMSD values for comparing programs is unreliable because factors other than the algorithms can influence their results, such as the quality and selection of input data [Hawkins *et al.*, 2008].

DARWIN is a GA-based program for docking [Taylor & Burnett, 2000]. It is based on the *CHARMM* system, and is designed to be run in a parallel environment. It has been developed not only for docking single protein-ligand pairs, but also for screening large databases of molecules as potential ligands for a protein. Moderately successful results are given, but *DARWIN* suffers from the local minima trap that plagues many docking programs.

AutoDock is another program whose GA has been used for flexible docking [Morris *et al.*, 1998]. A new scoring algorithm, *DrugScore*, was developed to replace the original regression-based energy function, and tested to provide a comparison [Sotriffer *et al.*, 2002]. *DrugScore* improved the ranking of results in moderately flexible cases, although not when molecules were rigid or showed complex structural changes in the docked state.

Energy/Dynamics An energy-mapping method of flexible docking in *AutoDock* is given in [Österberg *et al.*, 2002]. This uses multiple conformations of a protein to build a single, combined interaction energy map. In effect, this allows multiple dockings to be performed simultaneously. Consequently there is a problem with the risk of multiple high-scoring conformations presenting themselves where sufficient moderately acceptable mappings coincide. In tests, this did not manifest itself significantly, but it is noted as an area for work.

A docking algorithm that does represent both the receptor and ligand with full flexibility is given in [Tatsumi *et al.*, 2004]; this uses harmonics and dynamics to simulate the protein's movements, but then uses only the active site atoms to align the smaller molecule. An older procedure that explicitly considered the surrounding water molecules in SA calculations was that of [Mangoni *et al.*, 1999]; the execution is extremely slow, but for complexes where the solvent water is critical to the docking this is perhaps necessary. An extensive review of approaches to receptor flexibility and attempts to implement them concluded that the problem is a major focus for future work and probably requires multiple approaches to solve [Teodoro & Kavraki, 2003].

Incremental Construction Incremental construction algorithms are a very different approach to the flexible docking problem. Rather than contorting a ligand until it fits

well, the ligand is broken up into small pieces of a few atoms each, and then gradually reconstituted in possible active sites.

The *FLEXX* program [Kramer *et al.*, 1999] implements such an algorithm, breaking the ligand at all its single bonds — i.e. the points at which most flexibility is present. A small number of base fragments are used to create several well-fitting arrangements within the predicted active site. Another fragment is then added back in to each starting state, and a scoring function is used to settle the best arrangements again. This is repeated until the complete ligand has been formed, and the scoring function used to rank all the conformations found. Although this does permit full ligand flexibility to be considered, often successfully, a few large cases were not docked at all.

An improvement to that program was developed, primarily to include flexibility in the protein as well as the ligand [Claußen *et al.*, 2001]. *FLEXE* uses much the same method as *FLEXX*, but with a set (or ensemble) of possible protein conformations; these are compared and sections of the molecule that are very similar across structures are identified. As the alignment is built up, the various structural shapes present in the ensemble are used to allow for alternative protein conformations. In particular, the algorithm allows the segments of the ensemble members to be combined, thus creating new protein arrangements as required to accommodate the ligand. Although *FLEXE* is not significantly more accurate or successful than *FLEXX*, it does generally produce good results in less time.

Another tool, based on a similar method, is described by [Zsoldos *et al.*, 2003]. Alternatively, rather than breaking up a supplied ligand, a library of potential 'fragments' can be used to build a speculative ligand [Chen & Shoichet, 2009].

Path Planning Path planning is a field of robotics and spatial reasoning theory that has found applications in the molecular modelling area. Essentially, it involves treating the ligand as a robot, and attempting to navigate this automaton into the receptor's active site. In the simple, rigid case, this is a 6-dimensional problem; when flexibility is considered (i.e. the robot has joints), the situation is far less tractable. Algorithms for searching such conformation spaces are an active area of research; the kind attracting the most interest from the docking field is the roadmap model.

Roadmaps are graphs of possible transitions and states in a conformation space; to find a path from a start state to a goal requires a graph exploration algorithm — a well-researched field. Docking can be simulated either forwards or in reverse (and escape times predicted), using such methods [Apaydin *et al.*, 2002]. One of the most attractive features of these methods is that they do not perform an exhaustive conformational search; rather, they explore possible motion according to its potential usefulness. Consequently, they can be quite efficient, especially where effective tuning and look-up is applied to the exploration parameters [Jaillet *et al.*, 2005; Yershova & LaValle, 2006].

A ligand docking method of this kind is described in [Singh *et al.*, 1999]. This treats the ligand as a structure analogous to a robotic arm, with rotating bonds between rigid sections. One terminal atom is chosen to be the base of the robot, with five degrees of freedom (three for position and two angles for bond direction). The electrostatic potentials of the protein are calculated on a fine grid, and define the space in which the ligand robot moves. A probabilistic roadmap (PRM) algorithm is then used to navigate the ligand to a randomly docked low-energy position near the protein's surface. Each docking route and goal is scored by Dijkstra's algorithm, and these scores are used to predict the true active site, as well as the dynamics of the ligand's entry. This method does not appear suitable for reliable docking prediction, but may have some use for investigating the means by which docking occurs, as the authors note.

Other roadmap-based algorithms include combining PRM with A* search [Isto, 2003], weighted path variations [Apaydin *et al.*, 2001; Apaydin *et al.*, 2004; Apaydin, 2004], and enhanced-boundary PRM using interaction energies [Bayazit *et al.*, 2000; Bayazit, 2003]. Another exploration method is the rapidly-exploring random tree; this is used in [LaValle & Kuffner, 1999; Kuffner & LaValle, 2000]. The use of Voronoi volumes to guide the search through narrow passages in the conformation space is a recent development [Cortés *et al.*, 2005; Yershova *et al.*, 2005].

A related, but not map-based, algorithm is described by [Carpin & Pillonetto, 2005]. This uses a random-walk exploration method, rather than a mapping style, to explore the conformation space. The authors claim that it has shorter execution times than PRM methods and similar speeds to those using random trees. Another general conformation

space search method is given in [LaValle *et al.*, 1999], where kinematics (realistic smooth motion constraints) are applied to the exploration routes.

Hinge-bent docking, the term commonly used to identify the approach of dividing a molecule into several rigid sections, appears to be the usual technique for incorporating flexibility directly into a representation. It is reasonably simple, at least in comparison with full dynamic simulation, and is a fair model of the overall motion of many structures. Selecting the points at which hinges should be included is a problem, especially in larger molecules. A method of identifying the critical hinges is described in [Teodoro *et al.*, 2002], with particular reference to receptor flexibility. An example of the application of hinges to both molecules for docking is given in [Sandak *et al.*, 1998].

2.4 Current Areas of Research

None of the algorithms seen for either folding or docking prediction claims to be totally reliable or precise. This is hardly surprising, given the immense complexity of the problems. To properly model the chemistry of even a small molecule is a phenomenal challenge, and identifying any simplifications that can be made without sacrificing quality of representation is a major factor in achieving reasonable simulations in a useful time. A typical ligand might contain tens of atoms, each bond may have a torsional angle, and each atom might have a variable bond angle. Many bonds can vary in length slightly according to the surrounding forces and atoms involved. Hence, there are many degrees of freedom to be considered in any molecular simulation [Teodoro *et al.*, 2001].

The timing aspect is also a problem: most of the algorithms described take several minutes to perform a single run, in some cases taking hours to properly analyse a model. Although parallel processing improves this, only a massive array of machines could conceivably process a large library of molecules, as is often the ultimate goal for these programs.

The folding problem (§2.3.1 (*p*.30)) requires both biochemical and computational research, and this is ongoing [Chikenji *et al.*, 2006; Scott *et al.*, 2006]. Increased understanding of how proteins circumvent the Levinthal Paradox *in vivo* will probably aid their simulation *in silico*.

The docking problem (§2.3.2 (*p*.32)) certainly presents plenty of scope for exploration. The existing methods group into several general categories: FFT-based, GAs, SA/MD, and shape complementarity (including incremental construction).

FFT methods can provide a good starting point for a search, especially where multiple scoring functions are used, such as in [Gabb *et al.*, 1997]. They are, however, restricted in the scale of a search they can perform, owing to the size of the matrices required to accommodate a useful precision and/or range. An exhaustive search such as that is unlikely to be practicable.

GAs have proven useful [Morris *et al.*, 1998], but their non-deterministic nature makes proving their capabilities hard. In addition, the scoring functions used to compare candidates may have room for improvement to optimize their effect; recent achievements include [Pei *et al.*, 2006]. Simulated annealing and molecular dynamics approaches are based around attempts to directly model the physical interactions of proteins and ligands. Although these may prove complicated to design and implement, they have shown promise [Dominguez *et al.*, 2003]. The reviews of [Adcock & McCammon, 2006; Skjevik *et al.*, 2009] explore all the currently known methods of applying MD to simulation; the former identifies major areas for improvement as efficiency, stability, solvents, and electrostatics.

The purely shape-based approaches would seem to be naive, despite good results from the quadratic shape descriptor (*QSD*) algorithm [Goldman & Wipke, 2000b], primarily because they neglect all information about the model except geometry. They do provide a useful guide to the initial alignment of a ligand, but for an accurate judgement of binding affinity other considerations must be included, including electrostatic potential barriers and hydrophobic interactions.

As computational power increases over time, the models used to represent protein behaviour can be made more sophisticated, including more of the factors influencing molecular behaviour. In particular, quantum effects should be investigated, if only to establish their precise importance [Miller, 2005]. With the increases in biochemical understanding provided by (amongst other research) new methods of structure determination, the constituent algorithms for MD and GA simulations can be reviewed and improved. Hybrid methods, using one algorithm followed by or in parallel with

another, might be worth considering; the refinement and checking of approximate results is a common technique in programs already described, and should be considered as an overall pattern for simulation.

2.4.1 Lead Compound Identification

One of the primary applications for biochemical research, in particular the study of protein behaviour and interactions, is rational drug discovery/design (also known as lead compound identification). This is the exploration of molecules suitable for use in medicinal compounds. As already mentioned (*p*.17), many diseases are understood to be caused by unwelcome proteins in the body; binding a suitable ligand to these proteins can alter their interactions and thus inhibit the disease. Suitable ligands are those that may be delivered in some convenient form, may be synthesized easily, have low toxicity, will be metabolized by the body safely in a useful timeframe, and will only influence the intended target protein. Finding such ligands is the goal of lead compound identification.

Currently, a new drug can take fifteen years of research and development before it is ready to be offered to patients [Lunney, 2001]. Much of this is spent testing compounds in laboratories: clearly, the advancement of computational lead compound identification can reduce the cost of research and deliver new medicines sooner (and cheaper) than has been possible in the past.

There are two main approaches to lead compound identification in use: screening (drug discovery), in which a library of potential ligands is docked to a target protein in order to assess which may be worth further investigation, and *de novo* design (drug design), in which a hypothetical ligand is constructed and tested using biochemical details of the target.

Screening

In simplistic terms, screening requires only a docking program, a database of molecules, and a lot of time. For a given target protein, the docking procedure scores each of the potential ligands from the database with respect to its binding affinity, and eventually produces a list of the candidates ranked according to their predicted therapeutic potential. This is a slow process: ten years ago a typical database of commercially

available compounds — the Available Chemicals Directory [Symyx, 1982-2009] — contained around 50 000 molecules, today it has grown to more than a million. In 1996, the *DOCK* program could test 800 orientations of a ligand per second, and would have taken around two weeks to screen the (smaller) ACD. This is described in [Gschwend *et al.*, 1996], along with a summary of modifications to *DOCK* for improved screening.

A survey of the progress made with flexible docking methods in screening is given in [Carlson, 2002]. It includes the interesting suggestion of reverse screening, i.e. the selection of one potential ligand molecule, and docking it to a library of proteins in a search for an application. This is of use in testing for side-effects, as might occur if a ligand bound to the wrong protein. It does, however, rely on full flexibility in the representation to be reliable.

One of the most comprehensive screening programs to consider flexibility is *Glide*, which claims to perform a complete search of the conformational space of a ligand by a three-stage process [Friesner *et al.*, 2004; Halgren *et al.*, 2004; Friesner *et al.*, 2006].

As has been noted, screening requires a substantial quantity of computing power to complete in a useful time. Although supercomputers have been used, this is an expensive and still limited option. One cheap way of harnessing an immense quantity of processors is to employ grid computing, or massively parallel processing, and share the docking jobs amongst many mid-range desktop computers. A good example of this in practice is the recent Cancer Screensaver Project, which recruited 1.5 million computers in a year to screen billions of ligands for their applicability to cancer, and later, anthrax [Richards, 2002]. The vast supply of results from this project will require further refinement and investigation, providing a strong motivation for continued work on screening techniques.

De novo Design

The field of *de novo* ligand design, where the inhibitor for a protein is invented according to its desired function rather than being selected from a list, would require a review of its own. Designing molecules requires a significant amount of biochemical understanding, more than can be introduced here, but a justification for research and summary of techniques are given in [Zeng, 2000]. It can be noted here that a substantial intersection

between standard docking procedures and the ligand design problem occurs where incremental construction algorithms are used (*p*.44). Such a program, if allowed to combine fragments from many molecules or even single atoms, is one form of basic *de novo* design.

2.4.2 Reviews

For further information about the topics introduced by this chapter, see the following published reviews:

The drug discovery context: [Lybrand, 1995; Finn & Kavraki, 1999; Cavasotto & Orry, 2007; Lang *et al.*, 2007; Snow, 2008; Skjevik *et al.*, 2009; Song *et al.*, 2009; Villoutreix *et al.*, 2009].

Protein modelling: [Cavasotto & Orry, 2007; Putta & Beroza, 2007; Leach *et al.*, 2010].

Quantum effects: [Miller, 2005; Raha *et al.*, 2007; Taft *et al.*, 2008].

Molecular similarity: [Campbell *et al.*, 2003; Kazhdan, 2004; Eckert & Bajorath, 2007].

Folding prediction: [Dill *et al.*, 2007; Dill *et al.*, 2008; Chen *et al.*, 2008; Grant, 2009; Kihara *et al.*, 2009].

Docking in general: [Taylor *et al.*, 2002; Kitchen *et al.*, 2004; Leach *et al.*, 2006; Villoutreix *et al.*, 2007; Morris & Lim-Wilby, 2008; Kolb *et al.*, 2009].

Active site prediction: [Sotriffer & Klebe, 2002; Campbell *et al.*, 2003; Tompa *et al.*, 2005; Laurie & Jackson, 2006].

Scoring functions for docking: [Leach *et al.*, 2006; Warren *et al.*, 2006; de Azevedo & Dias, 2008].

Flexible docking: [Teague, 2003; Teodoro & Kavraki, 2003; Cozzini *et al.*, 2008; B-Rao *et al.*, 2009].

Chapter 3

Early Work

> You have talked so often of going to the *DOX* — and well, here are the *DOX*, and you have reached them, and you can stand it.
> It takes off a lot of anxiety.
>
> *'Down and Out in Paris and London'* [adapted], *George Orwell*

Before investigating stratagems for virtual screening, I worked on the smaller projects described in this chapter. These provided an opportunity to explore computational chemistry techniques and tools, such as docking algorithms and molecular representations. The last of these projects, implementing *XScore* in the *DOX* program, then inspired the major theme of this thesis.

3.1 FFT Alignment

The FFT grid search algorithm introduced by [Katchalski-Katzir *et al.*, 1992] has provided a basis for several docking methods (see §2.3.2 (*p*.36)) and is a good example of a simple molecular modelling algorithm. Since it is the most efficient means of performing an exhaustive search of translation space, it is a useful starting point for placing a ligand close to possible binding sites. Partly for this reason, I constructed a program to implement the algorithm as a way to start working with molecular models in software.

The original implementation described in that paper was done in Fortran for use on a Convex C-220. In summary, it uses the FFT to perform convolutions of scoring functions, recording peaks in the resulting functions that represent optimal alignments of receptor and ligand. I coded a version of this procedure using a combination of Java and C++. This would allow the algorithm to run on a wide range of modern workstations with minimal porting work. The development was initially done with Java alone, but eventually some incorrect FFT calculations could not be explained by any analysis of the

code and I decided that it was safer not to rely on the Java Virtual Machine for heavy computation. Consequently, I ported the FFT code to a separate C++ program, which was then used as a calculation engine by the Java-based user interface.

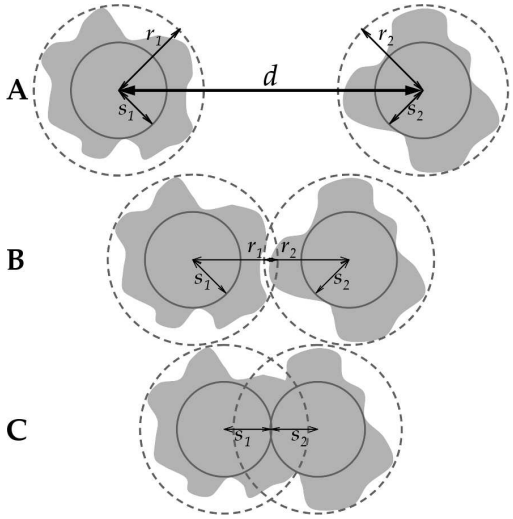
The FFT implementation was wrapped in an interface for pipelined docking. I designed this to provide a standard Java pattern for algorithms that align two molecules, such that they might be chained together to refine a pose according to different criteria. As an example, I added a rudimentary gradient descent method in this pattern, and the program allowed this to be used after the FFT to settle a pose more precisely. The methods appeared to be reasonably successful in their goal — that of docking a ligand — but were of only limited, academic interest. In comparison with current docking tools, they were overly simple, but could still provide a quick scan of possible translations if this were relevant as a preliminary stage for another algorithm.

Full details of this pilot project, which helped to inspire some of the stratagems of Chapter 4, are given in Appendix B (*p*.177).

3.2 Sphere Trees

The starting point for all physical modelling systems must be the form in which they describe the world of interest. By way of unifying the hierarchical flexibility and rigid cluster decomposition concepts mentioned in §2.1 (*p*.26), I began to consider the application of sphere tree models [Hubbard, 1996] to the representation of proteins and other molecules. Sphere Trees (or Sphere Hierarchies: both terms are used) are a means of representing shapes using successive refinements to an approximate boundary. The root of the model is a single sphere which bounds the entire object. For each node M in the model, its child nodes $\{N_1, N_2, \ldots, N_n\}$ are a set of smaller spheres whose union contains the same subset of the object's volume as M (and generally less volume than M itself). Hence, each complete level of the tree bounds the shape represented more precisely than the parent level.

The two primary advantages of using sphere trees are that point membership classification (and hence collision detection) may be performed quickly and efficiently, and that the model may be easily updated if the underlying object is rotated or translated.



Case A $(d > r_1 + r_2)$
is trivially not a collision, since the root nodes' outer boundaries do not intersect.

Case B $(r_1 + r_2 \geq d \geq s_1 + s_2)$
cannot yet be determined without exploring a deeper level of the hierarchy for more detail.

Case C $(s_1 + s_2 > d)$
is clearly a collision, since the inner boundaries meet and so the volumes cannot fail to touch.

Figure 3.1: Efficient collision detection using the roots of dual-concentric sphere trees

Collision detection is optimized by the ability to identify quickly when some subdivision of the shape cannot contain a test point. Updating the model when an orthogonal transformation (translation or rotation) is applied requires only that the spheres' centres be transformed similarly. All point tests are performed recursively through the tree structure from the root to the leaves, terminating early if sufficient information has been obtained from the nodes visited to avoid exploring every child.

An extended version of the sphere tree model is a dual-concentric sphere tree, in which each node sphere has a pair of radii associated with it — one guaranteed to completely enclose the shape, the other completely enclosed by the shape [Pitt-Francis & Featherstone, 1998]. This allows not only a trivial case to be caught when two shapes are sufficiently far apart not to touch, but also the converse, when they are too close together for there to be a gap. Figure 3.1 shows the three possible situations, with dotted lines indicating the outer radius and solid lines marking the inner radius for two top-level spheres enclosing irregular objects. The outer sphere is required to be the smallest possible enclosure, and the inner sphere is then concentric with it. These are both uniquely defined for any given shape.

3.2.1 Bonded Sphere Trees for Molecular Representation

The sphere tree paradigm lends itself perfectly to the representation of proteins and other molecules, especially where some hierarchical flexibility information is available. The leaves of the tree can directly correspond with the atoms: each sphere has its atom's

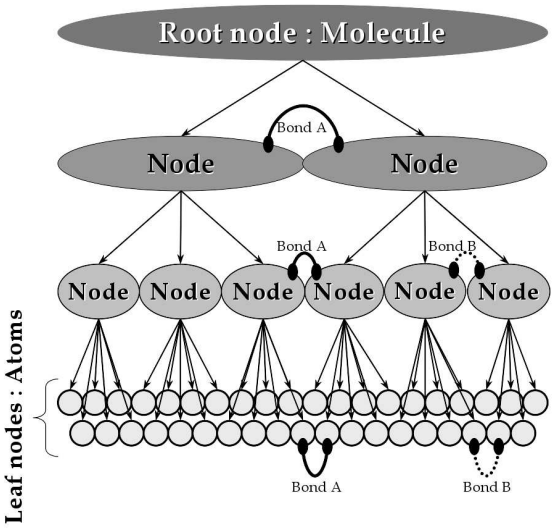


Figure 3.2: Abstract data structure for bonded sphere trees

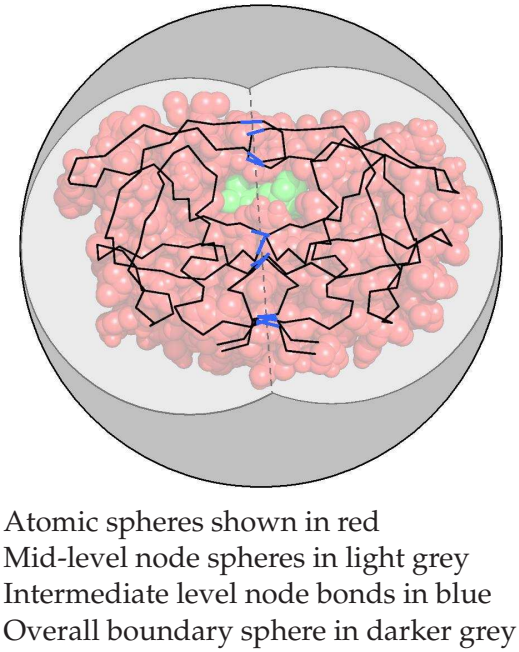


Figure 3.3: Illustration of a simple molecular bonded sphere tree with three levels

centre and VdW radius. This makes the sphere tree a perfect representation: unlike most geometric shapes which might be modelled to an approximation with very small spheres, the model fits the object exactly. The root of the model is, of course, the entire molecule with an enclosing boundary. Figure 3.2 shows the abstract data structure, as applied to a molecule. The number of levels in the hierarchy is arbitrary: in practice, it would probably be determined algorithmically.

The intermediate levels are, starting from the root, increasingly detailed rigid cluster decompositions (RCDs). Each level of the tree has an energy level associated with it — the energy required to induce movement between the constituent nodes at that depth. The child nodes of the root are thus formed from the output of a very strict decomposition that only separates the most loosely-coupled domains. Building such a model could be automated using a tool such as *FIRST* (see *p.*41), obtaining several decompositions at various energy levels, and using those with significant variation from their predecessor to extend the branches of the sphere tree.

To maintain the constraints on flexibility which exist between rigid clusters, it is necessary to include some model of inter-atomic bonds. Just as each level of the sphere tree contains a set of nodes, it should also hold a set of bonds. A bond B at level L may be modelled as $((a_1, n_1), (a_2, n_2)) : a_1 \neq a_2 \wedge n_1 \neq n_2$ where the a_x are the atoms (i.e. leaf

nodes) connected by the interaction, and n_x is the node in L which has a_x as a descendant. The effect of this arrangement is that the bonds at any given level are only those which cross the boundaries of that level's nodes. In particular, the root level has no bonds, since all are enclosed within the root node, and the leaf level includes all the molecule's bonds. Thus, it is intended that the intermediate levels collect only the information relevant to the flexibility permitted by their decomposition.

Figure 3.2 gives an illustration of this arrangement. Bond A is one of the least stiff interactions, and thus appears at all levels (except the root) linking some pair of nodes in each. Bond B, however, is less flexible, and is treated as rigid (and thus not present) in the first (coarsest) decomposition. Figure 3.3 artificially shows this on a picture of the HIV-1 protease. The black lines denote the bonds along the backbone, with the blue links indicating those retained at the level of the two clusters (pale grey circles). Note the successive improvements to the representation's precision by each level of the hierarchy.

3.2.2 Test Program (cSpheres)

To investigate the behaviour and usefulness of the bonded dual-concentric sphere tree concept, I built a simple test program called *cSpheres*, whose user interface is shown in Figure 3.4. This began with the classes for physical objects (atoms, bonds, and molecules) and their modelled counterparts (nodes (cluster and leaf), cluster bonds, and sphere trees). The design is such that a molecule may be loaded in (from either a PDB file or a custom sphere tree format) and modelled as a two-level bonded sphere tree: the leaves and the root. Any level may be replicated and then partitioned (to maintain completeness), with the bonds present in each level updated automatically.

If any node of the tree is moved, the bonds are used to ensure that the new position is acceptable. Bonds may also have their directions changed, in which case all connected nodes are repositioned together in accordance with the motion. If a cluster node's position changes, all its child nodes are relocated appropriately. These updates are applied lazily, however, to reduce the processing time required for large-scale movements: the cluster whose position is changed marks its children as dirty with the required transformation, but their position data are only recalculated when required. Collision detection and point membership classification (determining whether a point

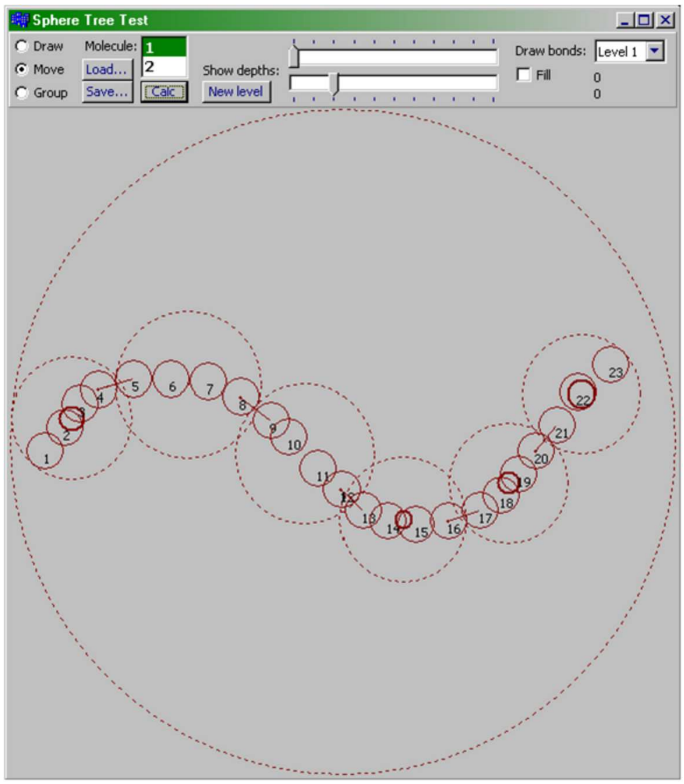


Figure 3.4: *cSpheres* interface for building and testing sphere tree representation

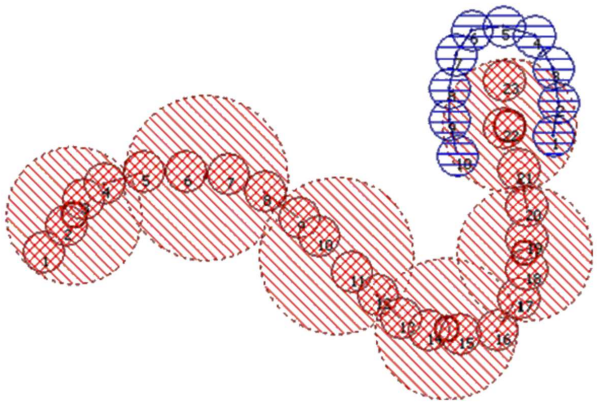


Figure 3.5: Two molecules juxtaposed manually in *cSpheres*

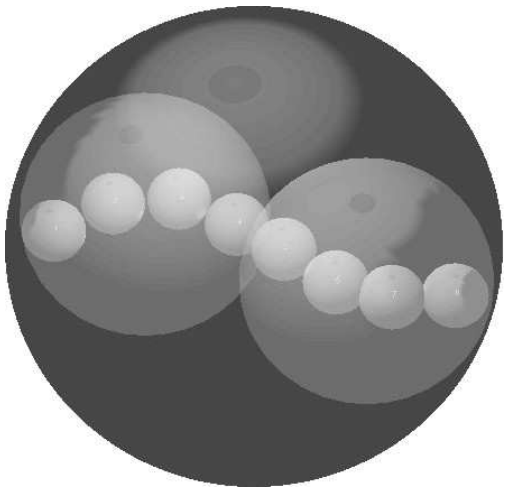


Figure 3.6: 3D view of a sphere tree in *cSpheres*

is inside, outside, or on the boundary) of molecules is implemented using the dual-concentric shortcuts discussed earlier.

This program includes a graphical user interface for loading or drawing molecules, manually grouping atoms into clusters to build a sphere tree, and interactively moving the bonds to adjust the conformation. The sphere tree is illustrated using dotted lines for

the outer spheres and thick solid lines for the inner spheres. Figure 3.4 shows a contrived test case with a three-level model. The bonds between the mid-level clusters are drawn as straight lines between atom centres, but spheres from all levels are shown. Dragging any circle moves the corresponding node of the model by adjusting bond angles and pulling other nodes to accommodate the new position, but preventing any motion beyond the bonds' range limits. Dragging the largest circle — the root node — moves the entire molecule. If two molecules are present, they are prevented from overlapping: changes that would result in atom clashes are rejected and undone. This means that a manual docking can be performed, as illustrated in Figure 3.5.

Although the underlying implementation classes use 3D coordinates, the interface only allows movement in the X-Y plane for simplicity in development. The Z coordinates are retained if molecular data are loaded. A view of the spatial arrangement is offered (Figure 3.6), but this is non-interactive except for camera movement. This display was reused for surface dot distribution tests; see Appendix C (*p*.187).

There were two main intentions for this work. One was to build an automated clustering algorithm (based initially on a *k*-means reduced point method [Glick *et al.*, 2002a], and later on *FIRST* decompositions), thus constructing a useful sphere tree for a molecule without human intervention. The other was to use this hierarchical model as part of an automatic docking procedure, such that the least rigid receptor bonds would be moved first if necessary to accommodate a ligand.

These projects were superseded, though, in favour of the scoring function stratagems work. It is still suggested that sphere tree representations are a valuable topic for research in this field, however, and this is discussed in §8.2.4 (*p*.163).

3.3 Scoring Functions and Search Methods

The screening of drug-like compounds for applicability to a particular target molecule (usually a protein) is an important aspect of the pharmaceutical industry's research and development. When docking ligands as part of a screening procedure, a lynchpin of the method is often the means by which a proposed arrangement may be rated: a scoring function.

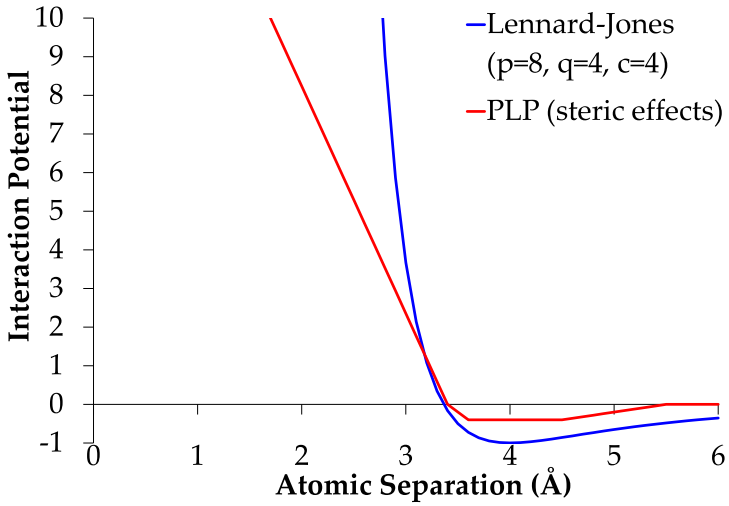
Many have been proposed and implemented, each focused on a different aspect of molecular interaction [Leach *et al.*, 2006]. Some are purely geometric, such as the FFT method, while others use electrostatic potential fields to calculate forces: [Kozakov *et al.*, 2006] for example. The hydrophobicity of molecular surface regions can be used in a scoring function by returning better values for poses that result in hydrophobic atoms being buried [Mandell *et al.*, 2001]. Certain kinds of bonding between molecules, hydrogen bonds in particular, add stability to a protein-ligand complex and so have been used to enhance scoring functions by rewarding well-aligned bonding atom pairs [Poirrette *et al.*, 1997; Schnecke & Kuhn, 2000]. Ionic interactions may be used similarly. Empirical functions, which combine several such terms in experimentally-determined proportions, are often found to be most generally applicable.

Equation 3.1 shows the general form of such a function: it accumulates a total score over all pairs of atoms and the interactions considered. The independent interaction functions are denoted by $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{N_{\text{int}}}$. The special case where the number of interactions $N_{\text{int}} = 1$ represents single physical functions, such as basic potentials.

$$\mathbf{Score}(R, L) = \sum_{l \in L_{\text{atoms}}} \sum_{r \in R_{\text{atoms}}} \sum_{i=1}^{N_{\text{int}}} \mathbf{F}_i(r, l) \tag{3.1}$$

An established and well-documented example of an early empirical scoring function is Böhm's [Böhm, 1994]. This accounts for several different molecular interactions (hydrogen bonds, ionic pairs, bond rotations, and lipophilic (similar to hydrophobic) interfaces) in the total score. The detail of how those terms are calculated is derived from observations of experimental cases and refined approximations based on training runs. This function provides a moderately sophisticated model for many other functions, including *ChemScore* [Eldridge *et al.*, 1997], *XScore* [Wang *et al.*, 2002], and those of *AutoDock* [Morris *et al.*, 1998].

Reviews of several recent and established scoring functions are numerous, often giving an overview of their methods and idiosyncrasies [Halperin *et al.*, 2002; Kitchen *et al.*, 2004]. In addition, comparisons have been performed to quantify, approximately at least, which functions are most suitable in which circumstances, both academically and industrially [Rocchia *et al.*, 2002; Wang *et al.*, 2003; Warren *et al.*, 2006].



$$\mathbf{LennardJones}_{p,q,c}(d) = \left(\frac{c}{d}\right)^p - 2\left(\frac{c}{d}\right)^q \quad (3.2)$$

Figure 3.7: Potential functions used by *PLP* and *XScore*

3.3.1 Piecewise Linear Potential

Any expectation that the most sophisticated functions are the most accurate or reliable is disproved by the usefulness of such simple methods as the Piecewise Linear Potential (*PLP*) function, which can screen comparatively large numbers of ligands very quickly and with fair success in many simple cases [Gehlhaar *et al.*, 1995; Böhm & Stahl, 2000; Wang *et al.*, 2003]. *PLP* simply models the potential between atoms using a five-part approximation of a Lennard-Jones function [Jones, 1924]. Figure 3.7 illustrates the linear form of this function. Atoms are classified according to whether they are likely to form a hydrogen bond, and the coefficients of the interaction function adjusted accordingly. Since this evaluation requires only a few arithmetic operations and treats all atoms of the same type as having the same radius (regardless of their element), it allows very fast, high-resolution scoring to be performed in reasonable time.

3.3.2 XScore

Many ligands in screening databases involve more complex interactions than *PLP* can consider. For this reason, it is generally necessary to have a selection of scoring functions, and screen each candidate with the appropriate choice. One possible choice for a more intricate assessment of docking suitability is the *XScore* function [Wang *et al.*, 2002; Wang, 2003]. This includes terms for hydrogen bonding, VdW potential, and three different hydrophobic interactions. To achieve the subtle variety of interactions, it requires a wider

range of atom types than *PLP*: polar, hydrophobic, and hydrogen bond donor, acceptor, or both. Rather than selecting one interaction between pairs of atoms, it calculates a score for all interactions (although these will be zero for some atom combinations). These various terms are then combined in weighted sums to produce three subtotals, whose mean is the overall docking score.

An overview of *XScore*'s definition is given in Equation 3.3: $\mathbf{X}(L,R)$ is the score returned for the pre-positioned ligand L and receptor R. This summarizes the terms used and how they are combined — note that it is an average of three functions (indexed by i). See [Wang *et al.*, 2002] for the full details of each . The k_i, t_i, h_i, v_i, and b_i are constants used as weighting factors, while $\mathbf{T}$, $\mathbf{H}_i$, $\mathbf{V}$, and $\mathbf{B}$ are functions of atoms l and r.

I observed that the averaging of three sub-functions is superfluous: distributing multiplication over commuted summation allows the division by 3 to be subsumed into the coefficients. These weightings may then be gathered and each term included only once, as shown in Equation 3.4. This reduction from 15 to 7 coefficients will make tuning their values much more straightforward.

$$\mathbf{X}(L,R) = \frac{1}{3}\sum_{i=1}^{3}\left(k_i + \sum_{l\in L}\left(t_i\mathbf{T}(l) + h_i\mathbf{H}_i(l,R) + \sum_{r\in R}\left(v_i\mathbf{V}(l,r) + b_i\mathbf{B}(l,r)\right)\right)\right) \tag{3.3}$$

$$= \sum_{i=1}^{3}\frac{k_i}{3} + \sum_{l\in L}\left(\sum_{i=1}^{3}\frac{t_i}{3}\mathbf{T}(l) + \sum_{i=1}^{3}\frac{h_i}{3}\mathbf{H}_i(l,R) + \sum_{r\in R}\left(\sum_{i=1}^{3}\frac{v_i}{3}\mathbf{V}(l,r) + \sum_{i=1}^{3}\frac{b_i}{3}\mathbf{B}(l,r)\right)\right)$$

$$= k' + \sum_{l\in L}\left(t'\mathbf{T}(l) + \sum_{i=1}^{3}h'_i\mathbf{H}_i(l,R) + \sum_{r\in R}\left(v'\mathbf{V}(l,r) + b'\mathbf{B}(l,r)\right)\right) \tag{3.4}$$

$$= k' + t'\sum_{l\in L}\mathbf{T}(l) + \sum_{i=1}^{3}h'_i\sum_{l\in L}\mathbf{H}_i(l,R) + v'\sum_{l\in L}\sum_{r\in R}\mathbf{V}(l,r) + b'\sum_{l\in L}\sum_{r\in R}\mathbf{B}(l,r) \tag{3.5}$$

$\mathbf{T}(l)$ is a rotor-count score for ligand atoms, returning either 0, 0.5, or 1. $\mathbf{V}(l,r)$ is an 8-4 Lennard-Jones function between l and r (Equation 3.2 in Figure 3.7, with p=8, q=4, and c determined by the radii of l and r). $\mathbf{B}(l,r)$ is a hydrogen bonding function which depends on the distance between atoms and the bond angles that would result, or zero if l and r do not form a hydrogen bond. $\mathbf{H}_1(l,R)$ is a function proportional to the SAS area associated with l and buried through the SAS of R, or zero if l is not hydrophobic. $\mathbf{H}_2(l,R) \equiv \sum_{r\in R}\left(\mathbf{H}_p(l,r)\right)$, where $\mathbf{H}_p(l,r)$ is a capped linear function of the atomic separation distance, or zero if l or r is not hydrophobic. $\mathbf{H}_3(l,R)$ is the

Coefficient		Originals		Updated
k	Constant	(2.69, 2.78, 3.10)	2.86	2.60
t	Rotor count	$(-0.159, -0.100, -0.169)$	-1.43×10^{-1}	4.86×10^{-2}
h_1	Hydrophobic burial		7.10×10^{-3}	1.53×10^{-3}
h_2	... atomic contact		3.73×10^{-2}	1.14×10^{-2}
h_3	... environment match		6.02×10^{-1}	3.60×10^{-1}
v	VdW potential	$(-2.01, -0.96, -2.14)\times10^{-3}$	-1.70×10^{-3}	-2.12×10^{-3}
b	Hydrogen bonding	(0.307, 0.412, 0.311)	3.43×10^{-1}	1.51×10^{-1}

Values in brackets are the separate values ($i \in \{1,2,3\}$) from the *XScore* authors' definitions.

Table 3.1: Weighting coefficients for *XScore* function

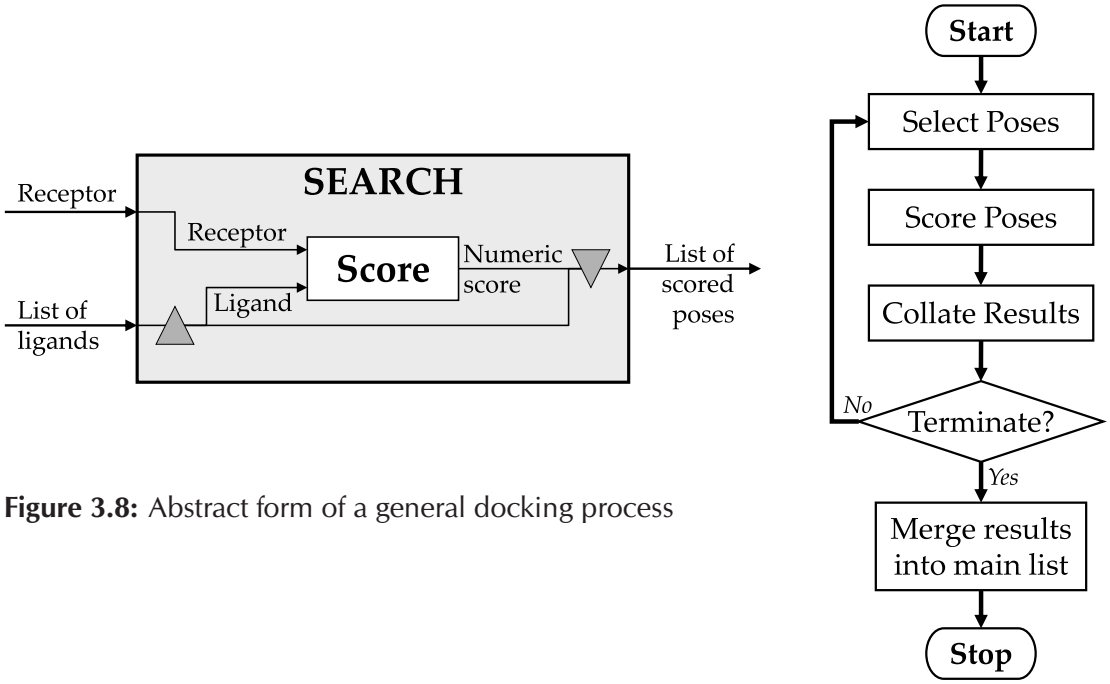


Figure 3.8: Abstract form of a general docking process

hydrophobic scale ('log P' value) of l if the sum of the log P values of all receptor atoms within 6Å is less than -0.5, or zero otherwise [Wang *et al.*, 2000].

XScore's authors chose coefficients using regression analysis with a training set. Their weightings are shown in Table 3.1. For a receptor R and two poses L_1 and L_2 of the same ligand, $\mathbf{X}(L_1, R) > \mathbf{X}(L_2, R)$ implies that L_1 has a better placement than L_2.

3.3.3 Search Methods

A scoring function does not select conformations and poses to consider: it merely evaluates the suitability of those presented to it. Before an implementation can be useful, some other algorithm is required to provide a framework in which it will be used, such as in Figure 3.8. Several techniques for this task have been published, including genetic

	For each...	*Order*
SEARCH	1. ligand	
	2. conformation	10^2
	3. pose	$(10^2)^6 = 10^{12}$
Score	4. ligand atom	10^1
	5. receptor atom	10^3
	6. interaction	10^1
	...calculate contribution	

Table 3.2: Nested loop composition of a general (naive) docking process

algorithms, probabilistic roadmaps, and molecular dynamics simulations. These are discussed, with references to reviews, in §2.3.2 (*p*.32). Whichever framework is chosen, it must read molecular data sources, rotate and position the protein and ligand into poses to be considered, call the scoring function with those poses as input, and aggregate the resulting scores to produce a ranked list of the best cases as the final output.

In summary, a docking system can be thought of as the bipartite Search/Score architecture of Figure 3.8, with a more detailed stratification of six layers as in Table 3.2. The orders of magnitude given therein are approximate scales, illustrating the problem's enormity. Note, however, that although myriad poses are possible even in the rigid, six-dimensional case, few search methods would consider them all exhaustively. Consequently, the third layer could consider anywhere from thousands to billions of arrangements, depending on how well it chooses the cases to score. The fourth, fifth, and sixth layers correspond directly to the summations of Equation 3.1 (*p*.60).

3.4 The DOX Family

3.4.1 DOX

I reimplemented *XScore* as an extension to an existing system called *DOX* [DOX, 2009], which performs a simple exhaustive search of rotational and translational space around a centre point given as input. Previously, *DOX* offered only *PLP*; however, the object-oriented code had been designed to allow interchangeable scoring function classes. The open source *OpenBabel* project [OpenBabel, 2008] provides the molecular file interfaces and chemical classes (`OBMol`, `OBAtom`, `OBBond`, etc.). The *Boost* libraries [Boost, 2009] are

used for some large-scale array manipulations, command line parsing, and file system interaction.

The existing *DOX* system included facilities for pre-calculated look-up tables (LUTs) for the interactions defined by a scoring function. These are 3D functions generated for each receptor R, interaction i, and ligand atom type t such that $\mathbf{F}^*_{R,i,t}(x,y,z)$ is the contribution from an atom of type t placed at position (x,y,z) to its ligand's interaction i with R. The general form of an empirical scoring function (Equation 3.1 (*p*.60)) can then be simplified to Equation 3.6.

$$\mathbf{Score}(R,L) = \sum_{l \in L_{\text{atoms}}} \sum_{i=1}^{N_{\text{int}}} \mathbf{F}^*_{R,i,l_{type}}(l_x, l_y, l_z) \tag{3.6}$$

PLP requires four LUTs per receptor since it models one interaction for four atom types: donor, acceptor, both, and non-polar. *XScore* required thirty-seven, as described later.

3.4.2 DOXGA

Since the exhaustive search performed by *DOX* is very slow to complete, its creators developed an alternative program called *DOXGA*, based on *GAlib* [Wall, 1996]. This (as the name implies) uses a genetic algorithm to explore poses, and produces approximately equivalent results. Since much of the code for *DOXGA* was shared with *DOX*, the same *XScore* implementation could be used in both search frameworks. Once the scoring function was seen to work exhaustively, the GA code has been used for all work since.

The appropriate parameters for the GA are very much dependent on the function used. In the original implementation, a population of 20 was evolved for 30 generations by default. This is barely sufficient for *PLP* and completely inadequate for the complexity of *XScore*. I introduced a scoring function complexity property, a constant static real-valued property of all scoring function classes, which may be used to set these parameters proportionally to the function in use. This number approximately corresponds to the value of N_{int} in Equation 3.1: it is defined as 2 by *PLP* and 6 by *XScore*.

3.4.3 OrthoDOX

Some of the later work discussed in this document, especially parallel execution and molecular property learning, led me to redesign the *DOX* software substantially.

The architecture of the code was too monolithic, and although some separation of components into classes was evident they were still too interdependent and inflexible. In addition, to make the screening of a database distribute between multiple processes, an overhaul of the design was unavoidable. The reworked software, whose client/server design for parallel execution took an extensive time to hone, is named *OrthoDox*. Technical details of this system are given in Appendix F (*p*.203).

3.5 XScore Implementation

My implementation of *XScore* calls for 37 LUTs. The three potential terms (**V**, **B**, and $\mathbf{H}_2$) each require 12 atom types (common ligand elements — carbon, oxygen, etc.) since atomic radii feature in their definitions. The summation part of $\mathbf{H}_3$ can be calculated as a single LUT only, since it is independent of the ligand. **T** is a constant property of the ligand and thus does not require a LUT. $\mathbf{H}_1$ can be partially pre-calculated, but not as an inter-atomic function, since it is based on whole molecules.

3.5.1 Surface Calculations

To estimate ligand surface area, I created a general `SpatialOccupancy` class to produce a 3D grid of cubes over any molecule's space. Each cube is classified as internal, external, or surface, and also notes the type of the atom occupying that point of space. This is somewhat akin to the matrix used in the FFT docking method described in §2.3.2 (*p*.36), but with more general element values. Its construction is summarized in Listing 3.1.

The volume of overlap between molecules can be estimated by obtaining the pairs of coinciding cubes from two such representations and comparing their classifications. To calculate *XScore*'s $\mathbf{H}_1$ term, the index list of surface cubes can be iterated and those points quickly checked against the receptor's volume. Counting those cubes that fall within the protein's space provides a passable approximation of the buried surface area in a method introduced by [Böhm, 1994]. A much more accurate solution would be to generate mathematically even distributions of random points on the smooth molecular surface of the molecule, and test these to obtain a proportion of the full, analytical surface area. I took a diversion to investigate this possibility, the details of which are discussed in Appendix C (*p*.187), but concluded that the fast indexed grid would suffice.

```
Create a lattice L larger than the spatial extents of the molecule
Set each element of L to 'external'
For each atom A in the molecule do
  For each element E in L within A's radius of A's centre do
    Begin
    Set E to 'internal'
    For each neighbour E' of E do
      If E' is not 'internal' then
        Set E' to 'surface'
    End
Create lists of lattice element references IE, IS, and II
For each element E in L do
  Begin
  If E is 'external' then Append E to IE
  If E is 'surface' then Append E to IS
  If E is 'internal' then Append E to II
  End
//SpatialOccupancy class then contains L, IE, IS, and II
```

Listing 3.1: Construction of `SpatialOccupancy` data

3.5.2 Recalibration

Inevitably, subtle discrepancies in the *XScore* algorithm's interpretation may have resulted in variations between my version and the original. The function as given in Equation 3.5 (*p*.62) is equivalent to the definition Equation 3.3, but the coefficients appear outside all the atom iterations. I thus demonstrate that the weightings can be applied after all other calculations, and hence that pre-calculating unweighted LUTs is a valid optimization to make. This point is crucial to the usefulness of LUT calculation: if the weightings are revised at a later stage or are varied for certain dockings, then the LUTs remain relevant without needing recalculation. The fact that coefficients only need to be applied at the very end of the function's calculation also allows swifter execution by eliminating their multiplications from the atom iteration loops.

Once the implementation was completed, I recalibrated the weighting coefficients using the training set provided by the function's authors. This involved scoring 100 protein-ligand complexes in their native complexes, and performing a linear multivariate regression with the individual function terms ($\mathbf{V}$, $\mathbf{H}_1$, etc.) against experimentally determined expected values supplied with the training set. The data used for the regression are shown in Appendix D (*p*.191), and my resulting coefficients appear as the last column in Table 3.1 (*p*.63). These mostly have the same order of magnitude, but some do show discrepancies with the function's authors' values. I cannot be certain that my

atom type assignments, the rules for which were encoded for me (see Listing E.2 (*p*.196)), always agree with theirs. Similarly, the log P function I used and my buried surface area estimates may vary, resulting in the different weightings. The rotor count received a very small coefficient of the opposite sign, but as a constant function of a ligand it is of no relevance to individual docking searches and should not affect results.

Chapter 4

Stratagems From Computer Science

stratagem
2 Any artifice or trick; a device or scheme for obtaining an advantage.

The Oxford English Dictionary, 2nd ed.

4.1 Context and Existing Technology

As already introduced by §3.4 (*p*.64), I had access to and was familiar with a working receptor-ligand docking tool, *DOX*. The exhaustive search version was discarded once the genetic algorithm variant *DOXGA* was demonstrated to work, and so that became known as *DOX* for brevity. These console-based programs were developed in C++ using a mostly object-oriented design, and was compiled for use on typical x86 Windows-based desktop computers. For large-scale screening of a ligand database, a small Linux cluster was available based on the *Rocks* platform. Consequently, a port to Linux was in progress, since the libraries employed in the project (*Boost* for general data processing, *OpenBabel* for molecular file parsing, and the *GAlib* classes) were all cross-platform compatible. These test platforms' specifications are given in §E.2 (*p*.200).

The basic *DOX* system using the *PLP* scoring function, according to its authors' own unpublished evaluation, was a reasonable ligand docking tool. It successfully implemented well-established methods for a rigid-body search, and produced acceptable poses for the examples tested — a particular target of interest was cyclin-dependent kinase (CDK 2) with the staurosporine inhibitor. Using my incorporation of the *XScore* function, when it was completed, the CDK 2 example was again docked correctly.

The use of stochastic search methods, such as GAs, particularly highlighted the balance to be made between speed and reliability. A comprehensive docking search is practically unattainable, because the scale of the problem is too expansive, and

so sampling techniques are an established way to cover a large combinatorial space efficiently. However, the fundamental problem with sampling is that there is a risk of false negatives: the omission of a good result from any output because it was never considered. Molecular interactions are a suitable candidate for this kind of search because the search space includes a large undesirable region and few small goal areas. Sampling allows the glut of poor ligand placements to be avoided and the most attractive examples pursued instead. Whilst it is an imperfect method, stochastic searching is one of the best practices available in the field, and *DOX* was a fair but unsophisticated demonstration of this.

4.2 Stratagems for Consideration

Over a long period of time working with the existing *DOX* implementation, I noted several points to consider regarding the best way to complete certain tasks (the surface area calculation of §3.5.1 (*p*.66), for example). These generally concerned either the accuracy of the results, the time and resources required to complete the procedure, or both. Even small improvements to either of these are potentially quite significant, since the practical application of such docking tools for virtual screening (§2.4.1 (*p*.49)) requires that millions of ligands be processed for any given receptor. Regardless of the search method in use, it is generally possible to divide the task into several layers, as shown in Table 3.2 (*p*.64). These can be summarized as scoring (atoms and interactions), screening (ligands and conformations), and searching (poses).

The problem can be abstracted thus: for a set $P = \{p_1, p_2, \ldots, p_n\}$ of n items, where n is very large, we define a comparison operator $\mathbf{C}(p_i, p_j)$ based on an evaluation function $\mathbf{S}(p)$ such that $\mathbf{C}(p_i, p_j) \equiv \big(\mathbf{S}(p_i) \prec \mathbf{S}(p_j)\big) \iff p_i$ is better than p_j. We seek the best p, such that $\forall q \in P, q \neq p : \mathbf{C}(p, q)$ is true.

4.2.1 Scoring

The evaluation $\mathbf{S}(p)$ generally employs a scoring function $\mathbf{Score}(R, L)$, as introduced in §3.3 (*p*.59). $\mathbf{S}(p)$ updates a ligand L to the pose specified by p, and then applies the scoring function to the new arrangement. The calculation of $\mathbf{S}(p)$ must be done quickly,

since it will be required a vast number of times, and the precision should be appropriate to the situation.

Look-Up Tables

As discussed in §3.5 (*p*.66), the *DOX* framework already permitted the pre-calculation of scoring function terms in look-up tables (LUTs) to speed the evaluation of docked poses. This produces a very large amount of data and takes a substantial amount of time for each receptor, although it needs to be done only once.

Interpolation Interpolation is important for the accuracy of LUTs. It may be impractical to generate LUTs at a very fine resolution — the memory demanded by *XScore*'s 37 tables becomes colossal for larger receptors — but subtle changes in ligand alignment need to be distinguishable. Thus, it is desirable to interpolate values when retrieving data.

Caching Although the pre-calculation of the LUTs is reasonable for a receptor that will be a repeated target, it may still be unnecessary. Rather than generating entire LUTs for generous bounding regions (to ensure that the ligand will not protrude out of bounds), I suggest that a caching mechanism could be employed. This then requires no lengthy pre-calculation to initialize the docking, but instead fills in the entries as and when required.

Early Rejection

The use of scoring functions to assess a ligand and receptor pose quantitatively is inexact. Values produced by any such calculation must be interpreted and used with care, treating them with appropriate precision. Consequently, there is little point in calculating unreasonable detail, especially if a judgement can be made about the result before it is completely evaluated.

4.2.2 Searching

It is unfeasible to evaluate $\mathbf{S}(p)$ for every $p \in P$ because P is far too large. Typically, a search must focus on a small population $P' \subset P$, and the selection of P' needs to be reasonable. We should avoid choosing elements that can be predicted to be unfavourable,

and those that are chosen should be handled in a way that allows their suitability to be established as quickly as possible.

Local Optimization

Local searches can be used to overcome the limits of a relatively coarse search of pose space, refining a moderately well-scoring result to find the best nearby pose [Hart, 1994; Morris *et al.*, 1998]. Similarly, they can be used to improve the likelihood of a GA finding the best result by ensuring that any strong genomes are optimized. However, their excessive use will most likely be counter-productive, expending time for only minor improvements to the poses found.

Prioritization

Where it is possible to choose the arrangement of data for processing, it should be done in such a way that the most significant information is processed first. If calculations are making use of early rejection, then this will ensure that better judgements can be made about what their appropriate level of detail should be.

Early Rejection

There is little to be gained — probably nothing whatsoever — from repeatedly evaluating a function for near-identical data. Instead, one could assume that $\mathbf{F}(x_1) \approx \mathbf{F}(x_2)$ if some similarity measure of x_1 and x_2 is above a threshold level; a simple example would be a scoring function and two poses of the same ligand. The computation time saved by using this shortcut should be used instead to cover a greater diversity of values.

Structural Decomposition

The geometry of molecules is crucial to their ability to dock, and thus their volumes can be compartmentalized into regions according to the likelihood and/or mode of binding. If substructural motifs can be identified, then they provide another source of information about the interactions being simulated. These clues about the possible arrangements and outcomes should be used to avoid searching for poses that are unlikely to be of interest.

Heuristics

To respond to the disproportionate range of unsuitable poses that a search method might examine, some bias is worth considering. Search methods often produce implicit information about their progress. Assuming that the algorithm iteratively refines a set of candidates, the current items for assessment provide an approximate indication of the general standard of results being discovered. If several distinct searches are necessary, but with an expectation that not all of them will produce favourable results, then processing may be biased to prefer those making encouraging progress.

4.2.3 Screening

The evaluation $\mathbf{S}(p)$ involves a particular ligand conformation L_k. Finding the best p for L_1 is only one part of the docking task: this search must be repeated for L_2, L_3, and so on. Over time, the set of poses P' considered by each search could be reduced using accumulated experience and general knowledge. The thoroughness of the searches could also vary, depending on the properties of L_k.

Spatial Indexing

Analysing the space around two solid bodies — a ligand and receptor, for example — will typically require frequent point membership classifications: categorizing space as inside, outside, or on the surface of a shape. To make such decisions quickly, volumetric LUTs can be constructed as a grid over the space of interest and labelling each cell in that lattice according to its occupancy. My work implementing *XScore* involved the development (described in §3.5.1 (*p*.66)) of a `SpatialOccupancy` class to perform this kind of grid-based decomposition. When generating the data for such a representation, it is perhaps also worthwhile to index the classifications assigned, so that they can be iterated separately and systematically.

Search Methods

Genetic algorithms are a well-established option for efficient searching of ligand placements, but they are not necessarily the best in all situations. Biochemistry literature

features numerous methods for applying GA work to docking, as reviewed in §2.3.2 (*p*.36) and discussed further in §8.2.3 (*p*.162). Since I already had a working GA system available to me, I chose to exploit this foundation as a test-bed for other stratagems, rather than starting with nothing. The intention was to explore abstract techniques for improving tools in general, not the construction of a particular complete system.

Early Rejection

The parameters for the output from a docking system provide some guidance for the search. Not only do they define the final processing of results, but they can be used to eliminate wasted effort at earlier stages. If it appears unlikely that a particular search job will produce any results that will be retained in the final output, then it could be rejected to make way for another that might prove more successful. The output from a docking search is often constrained by a count, and this quota defines a worst acceptable score. Once the required number of result poses has been collected, this is the least preferable score in the list. Any pose with a score worse than this can be guaranteed not to appear in the final output from the docking routine, and so should be avoided.

Deferred Evaluation

Permitting the suspension and resumption of searches according to their progress could make efficient use of processing resources. This would require some interface for the dynamic priority rating of jobs to select which should be started, continued, or stopped. Although a time and space cost is inevitably incurred by storing search states, the potential benefit is that ligand conformations that are proving unlikely to bind could be ignored until all alternatives have proven worse. This is a more cautious approach than early rejection.

Machine Learning

The application of prior knowledge learnt from completed dockings to the guidance of new cases is a field that has attracted much interest already. Ligands can be described and compared using many properties. If information can be collected about a particular target's previous results, this should help subsequent docking searches

to predict whether and how another ligand might bind. There could be a speed improvement if reasonable initial predictions of docking poses can be made and thus searches abbreviated. Even if the docking search is not directly improved, useful information may be collected about a receptor's behaviour of interest for other analysis.

Parallel Execution

High-throughput processing such as this cannot be discussed without considering the options for execution in parallel. Some search methods are embarrassingly parallel, including most population-based algorithms. These have clearly segmented implementations, and so it is relatively straightforward to consider distributing the members of the search population between several processors. At a higher level, where multiple complete searches are required with different inputs then these too can be performed on different processors and the results collated afterwards. This separation of tasks is the obvious extension of the job control design for deferred evaluation.

Shape Classification

As already noted with structural decomposition, the geometry of molecules determines how they might bind. The shape of a ligand may provide useful information to reduce the amount of repetitive work the docking search has to do. In combination with a machine learning mechanism, descriptors of ligands can be collected to identify a general pattern of what shapes dock well with a particular receptor, and where. Not only can the receptor provide guidance about where its surface pockets may support docking, each ligand processed can be used to illuminate better the likely outcomes of later searches.

4.3 Applications and Examples

Having collected these stratagems, I had to consider how to make sense of them in a practical situation. Starting with the simple list, I associated each with one or more aspects of the *DOX* system to which they were relevant and ideas for how they could be used. Several items appeared to be interrelated, with common areas for application or dependencies for implementation. After much thought and discussion, I began to

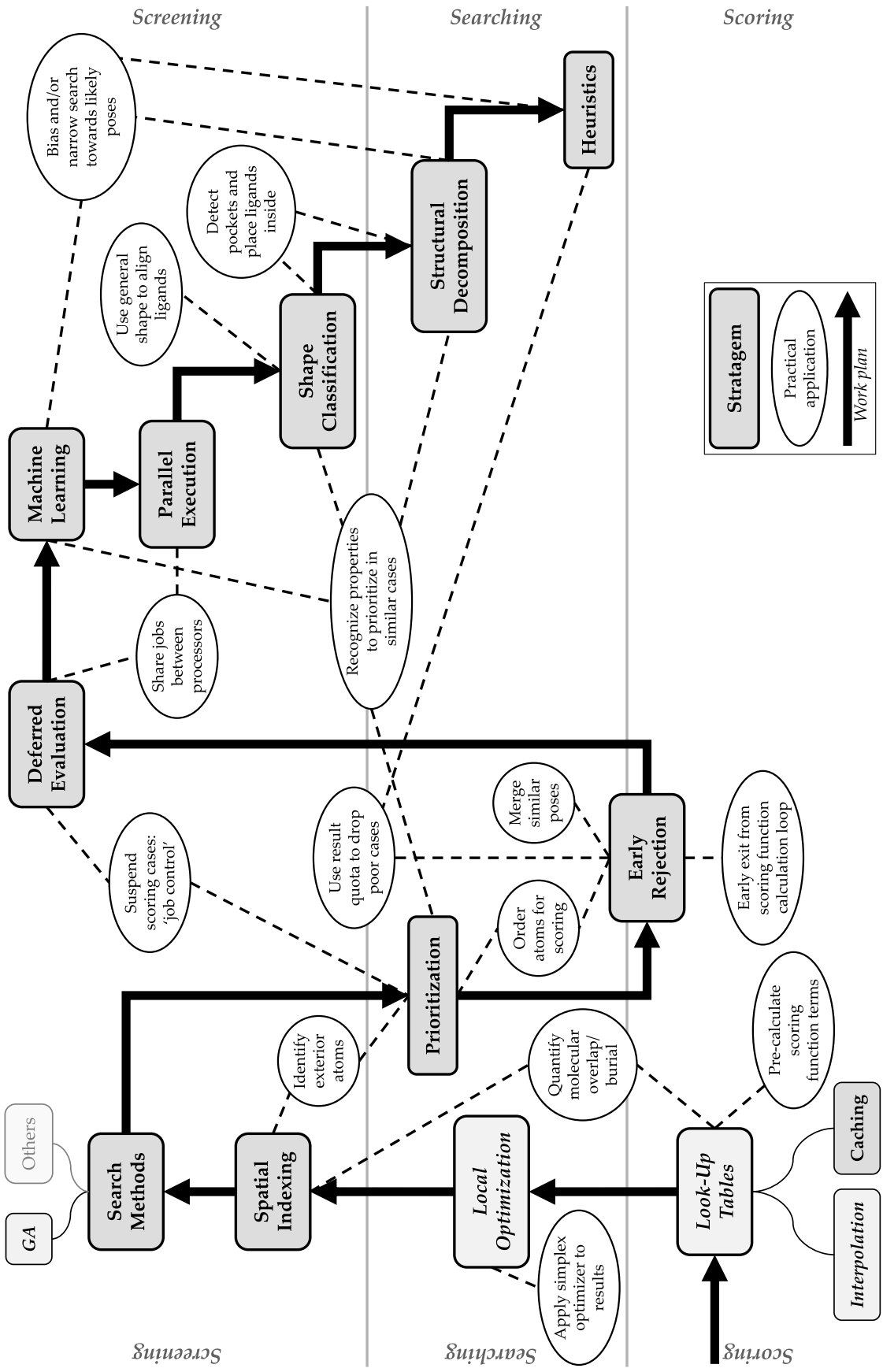


Figure 4.1: The research roadmap, linking the stratagems and practical applications

formulate a logical thread from one to another, and eventually, I constructed the roadmap shown in Figure 4.1. The diagram illustrates the stratagems (as shaded boxes) in their three layers, and their practical applications (in ovals), with dashed lines marking the connections. A combination of necessity and practicality then determined the order in which these should be investigated. That sequence is marked with heavy arrows, tracing the work plan from look-up table behaviour, through job-based parallel execution, to geometrically-guided heuristic searches.

The applications to be discussed have been divided, like Gaul, unevenly into three areas. These are not chronological with my experiments, but are rather a thematic presentation. The chapters cover the issues of geometry, searching and scoring algorithms, and higher-level task organization. Here, I introduce the experiments to be described and compared.

4.3.1 Geometric Guidance

To make much of the later geometric work possible, it is important that rotations in a docking system are represented using quaternions. These are an alternative mathematical construction to the traditional Euler angles method. An explanation of quaternions and the relative merits of the two systems are provided in §5.1 (*p*.83).

Structural decomposition for predicting the active site on a receptor can guide the initial placement of ligands. If an automatic pocket detection algorithm is employed, some likely poses could be selected to prime the search population. Depending on the information produced about the pockets, the ligand conformations could even be placed analytically in these volumes, avoiding unnecessary atomic clashes. A number of algorithms have been developed for this task in recent years, as discussed in §2.3.2 (*p*.34). I present my own novel method in §5.2 (*p*.87), which employs the **spatial indexing** already being used for *XScore*.

Although in some circumstances it is reasonable to expect manually-chosen limits to be imposed on a docking, investigating new molecule complexes may well require multiple binding sites — or even the receptor's entire surface. Automatic pocket detection may offer binding sites on which to concentrate a search, but several areas must

still be tried if any leads are to be found reliably. **Heuristics** for gradually narrowing those search extents are demonstrated in §5.2.5 (*p*.101).

Several general molecular shape representations exist (see §2.2 (*p*.28)), and these provide a good use for the learnable properties in §4.3.3: collecting geometric information about the best previous docking arrangements. Simple shape descriptors can be used to characterize a ligand in a manner that supports this kind of analysis better than a complete atomic model. The information thus gathered can then identify ligands that are likely to fit well again, and can also provide good example poses to which new ligands can be aligned rapidly, generating pertinent initial states for a search. **Shape classification** is used to pre-align ligands in §5.3 (*p*.106).

4.3.2 Efficient Exploration

Local optimization was already present in the *DOX* system, with an implementation of the simplex optimizer [Nelder & Mead, 1965] applied after every generation to each genome in the GA population. It would be faster to include this stage after every *n*th generation, or perhaps only the final population, and so avoid diverting excessive time to detailed local searches. The consequences for result quality are discussed in §6.1 (*p*.117).

Look-up tables are discussed and the benefit of **interpolation** is assessed in §6.2 (*p*.121). A lazy implementation, **caching** data rather than pre-calculating it all, is tested in §6.3 (*p*.125), having proved useful with the earlier pocket detection work in §5.2.5 (*p*.101).

Early rejection is relevant to all levels of docking. It may well be possible to deduce that a partially-calculated pose score can be expected to fall beyond any acceptable range if completed. If the number of atoms as yet unconsidered is insufficient to redeem it, then that incomplete score should be returned as the result so that the docking algorithm can continue. Since the atoms exposed on a molecule's surface tend to make the greatest contribution to any interaction, these can be prioritized for first consideration. Selectively approximating a scoring function with an early exit in this way is tested in §6.4.1 (*p*.128).

Any search method, including a GA, is constrained in its ability to find results by the population size used and the extent of the search bounds. If several of the poses under consideration become very similar, the exploration's performance will be diminished by

focusing too much one possibility, especially if large regions are still untested. Such duplications should be identified and merged, replacing one with an alternative case to try instead. This is applied to the GA search method in §6.4.2 (*p*.134).

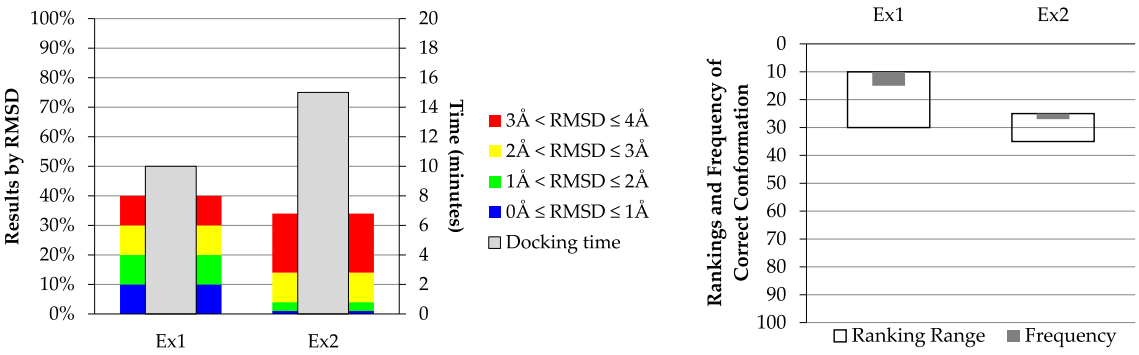
It may be reasonable to predict mid-way through a search whether it seems likely that any poses will be found that will be preferred to the worst-case score in the result quota. If not, the search should be terminated early and another conformation considered. Since the worst-case score for a receptor will only improve with each docking completed, the prioritization of conformations may be significant. This high-level rejection is discussed in §6.4.3 (*p*.137).

4.3.3 Properties, Priority, and Parallelization

A mechanism for associating information about molecules with their docking results must be devised to support **machine learning**. Somewhat akin to current pre-calculations of LUT data, a database should be created for each receptor storing properties of ligands and poses along with a measure of their success, updated automatically each time a conformation is docked. Making this usefully flexible and easily amended with new data is a primary requirement, allowing for the possibility of parallel screening (see below) needing shared access. The details of molecular knowledge bases (MKBs) are given in §7.1 (*p*.143), and the design and application of learnable properties are described in §7.2 (*p*.148).

Task **prioritization** should be done with a scheme that uses the shape classification work discussed above, such that jobs consistent with known good results are boosted and those with sufficiently low ratings are frozen or discarded altogether. In particular, the input conformation ensembles should be ordered before any searches begin, to collect results from the most promising cases first. This idea is discussed in §7.2.1 (*p*.150). Consequently, I replaced *DOX*'s existing database conformation evaluation loop with a job management procedure. To support this, I included a simplistic job priority function, based on intermediate GA population scores, to control **deferred evaluation** of conformations. This design is compared with sequential evaluation in §7.3 (*p*.151).

Parallel execution can be achieved by splitting the input database of ligands and running separate docking instances for each batch on separate computers. This can



Of these two fictitious editions, 'Ex1' is faster and much more precise in its docking results, and selects the correct ligand conformation more successfully.

Figure 4.2: Sample result graphs illustrating assessment criteria

be partially automated using well-designed shell scripts and job queueing tools. It is preferable to have a completely integrated implementation, dynamically balancing the workload between processors and coordinating data updates to and from the knowledge bases, so that a single command can be issued to start an autonomous distributed screening system. A working cross-platform example of this is presented in §7.4 (*p*.153), quantifying the potential acceleration of screening.

4.4 Assessment Criteria

The following three chapters discuss the preceding applications of the stratagems. For this purpose, I built many different editions of the *DOX* system, each with a code identifying the particular combination of modifications present. The edition codes are printed monospaced and boxed in this thesis. **A list of these program editions and the stratagems in each is given in Table E.3 (*p*.202).** When comparing the behaviour of these versions, assessing the relative merits of a particular technique or parameters, I have collated the following data from the program's standard output transcripts and result data files:

Loading time: The time taken to read and parse data for the docking run, and prepare all internal states up to the point of beginning the first search algorithm.

Docking time: The time taken to perform the docking search algorithms for all ligand conformations supplied. In the example of Figure 4.2, 'Ex1' takes 10 minutes while 'Ex2' takes 15 minutes, as shown by the grey bars. The inputs processed in this time would be specified by a 'Test used' caption in the figure, as explained in Appendix E (*p*.195).

Results by RMSD: The percentages of results within four ranges of RMSD (up to 1, 2, 3, and 4Å) from the crystallographically identified correct binding pose of the ligand(s) being docked. These data are shown as stacked bar charts to make the distribution of results clear. A standard colour code has been used: the proportion of poses within 1Å is shown by a blue bar; the proportion within 2Å is marked by the top of the green bar, those within 3Å with a yellow bar, and those within 4Å with red. An RMSD within 2Å is widely considered a good result: taller green bars are good and taller blue bars are better.

Figure 4.2 provides an illustrative example: of the result poses obtained from the 'Ex1' edition, 40% were within 4Å RMSD, 30% within 3Å, 20% within 2Å, and 10% within 1Å. This compares favourably with the 'Ex2' data, where only 4% of results were within 2Å, even though an almost similar proportion was within 4Å.

Rank and frequency of crystal structure: Where an input database of multiple conformations is screened, the output list indicates which conformation is represented in each pose result. In particular, the native bound conformation can be identified. Its number and ranks of appearances are shown using bar charts, with two bars per test case, as a measure of the screening method's tendency to select the correct ligand shape. On these charts, the black outer rectangles extend from the highest to lowest rankings of the true conformation, with an inner grey bar indicating the frequency of that conformation in the results. Higher, taller rectangles are therefore preferable, but full-height bars would indicate bias that would be undesirable in a screening situation.

Figure 4.2 shows that the native conformation appeared 5 times in the results from 'Ex1', from 10th position to 30th. 'Ex2' only returned it twice, at ranks 25 and 35.

These measures provide simple comparisons between editions and configurations of the docking system. In many published assessments, RMSD is used as a primary metric of success with the 2Å threshold used to identify correctly-placed results [Cavasotto & Abagyan, 2004; Erickson *et al.*, 2004; Meiler & Baker, 2006]. The selectivity measure, however, is of limited relevance to the question of whether a system is suitable for blind screening. Where an ensemble of ligand conformations is generated for docking to a hitherto untried target, it is unlikely that the exact ideal conformation will in fact be produced. Hence, the tendency of a search to choose ideal shapes is only of interest if the most similar example will be most preferred — not at all certain once conformations vary significantly.

Chapter 5

Geometric Guidance

"Round objects!"
"Who is Round, and to what does he object?"

'Yes Minister', Jonathan Lynn & Antony Jay

Protein-ligand docking is governed substantially by the shapes of the molecules involved. The interactions between them are often concentrated at their boundaries, and the extreme repulsion of atoms when colliding ensures that there cannot be any significant overlap. Consequently, the geometry of the molecules is an important source of guidance when searching for a good binding pose. This chapter discusses how best to represent, manipulate, and exploit raw shape data to improve the efficiency of docking searches. Where experimental results are presented, codes are used to identify the tests used: these are explained in Appendix E (*p*.195).

5.1 Quaternion Rotations

In order to represent the ligand poses being scored, it is necessary to use (at least) a translation and rotation. The position is trivially modelled as a 3D vector, but rotations lend themselves to multiple designs. The analogue of the translation's vector for precise, explicit use is a 3×3 matrix, which can be multiplied by a vector to perform the rotation. A more natural implementation is to employ three Euler angles, which correspond to three component rotations about prescribed axes. These are convenient to understand and describe in code, and conversions to matrix form are well-known. However, they have certain disadvantages:

- The convention of which axes the angles rotate about must be chosen and used consistently; this varies between implementations. A common choice is analogous to describing a person's longitude, latitude, and compass bearing from an equatorial origin.

- The problem of gimbal lock can occur: this is when a degree of freedom is lost because two of the axes are aligned. In the above convention, a person at the north pole has $+\frac{1}{2}\pi$ latitude, but longitude and bearing degenerate into the same axis.
- Discontinuities occur in the representation: for example, a person walking up the Greenwich meridian and passing the north pole instantaneously switches from 0 longitude to $\pm\pi$, and also from a bearing of 0 to $\pm\pi$. This makes calculating relative rotations and interpolation difficult.
- Composing rotations is difficult to do without incurring rounding errors, because the second rotation must be applied to the coordinate frame resulting from the first. In practice, this often has to be done by multiplying the corresponding matrices, an inefficient method and one less convenient for display to humans.
- When selecting rotations at random, as often must be done for a docking search, a uniform selection of angles will result in a significant clustering of rotations around polar arrangements. This can be avoided by selecting cosine values for the degenerating angles instead of the angle values themselves.
- The use of Euler angles relies heavily on trigonometric functions, and the computational cost these involve.

A third representation applies unit quaternions to the problem. Quaternions are hypercomplex numbers with one real and three imaginary parts (i, j, k) where $i^2 = j^2 = k^2 = ijk = -1$ [Hamilton, 1967]. Their use for rotation is an established technique in engineering and robotics since it allows a continuous representation of all possible rotations. Whereas the space of Euler angle rotations may be thought of as the volume of a cuboid of dimensions $(2\pi,\ \pi,\ 2\pi)$, the space of all quaternion rotations is the surface of a unit 4-sphere. A quaternion rotation $\boldsymbol{Q}$ may be written as a normalized 4-vector:

$$\boldsymbol{Q} = [a, x, y, z] : |\boldsymbol{Q}| = 1 \iff a^2 + x^2 + y^2 + z^2 = 1 \tag{5.1}$$

in which case the spatial interpretation of $\boldsymbol{Q}$ is a rotation of $2\arccos(a)$ about the axis in the direction $[x, y, z]$ from the origin. The special case of a zero axis, when $x = y = z = 0$, is consistent since the requirement that $\boldsymbol{Q}$ be normalized implies that $a = 1$, and thus $2\arccos(a) = 0$ defines the null rotation. Since this representation always explicitly defines the axis of rotation, there is no confusion about frames of reference. The angle-axis interpretation is reasonably easily understood by users.

Applying a rotation to a vector in space does not require any trigonometry but may be done directly by multiplication. Specifically, the transformation of a point v by Q is given by $[v_x, v_y, v_z] \mapsto [v'_x, v'_y, v'_z]$ where

$$\begin{aligned}
v'_x &= v_x + 2\,((-Q_yQ_y - Q_zQ_z)v_x + \quad (Q_xQ_y - Q_aQ_z)v_y + \quad (Q_aQ_y + Q_xQ_z)v_z) \\
v'_y &= v_y + 2\,(\quad (Q_aQ_z + Q_xQ_y)v_x + (-Q_xQ_x - Q_zQ_z)v_y + \quad (Q_yQ_z - Q_aQ_x)v_z) \\
v'_z &= v_z + 2\,(\quad (Q_xQ_z - Q_aQ_y)v_x + \quad (Q_aQ_x + Q_yQ_z)v_y + (-Q_xQ_x - Q_yQ_y)v_z)
\end{aligned} \tag{5.2}$$

Rotations may be composed by conventional quaternion multiplication: performing Q_1 then Q_2 is equivalent to Q' where

$$Q' = Q_2Q_1 = \begin{bmatrix} Q_{2a} \\ Q_{2x} \\ Q_{2y} \\ Q_{2z} \end{bmatrix} \begin{bmatrix} Q_{1a} \\ Q_{1x} \\ Q_{1y} \\ Q_{1z} \end{bmatrix} = \begin{bmatrix} (Q_{2a}Q_{1a} - Q_{2x}Q_{1x} - Q_{2y}Q_{1y} - Q_{2z}Q_{1z}) \\ (Q_{2a}Q_{1x} + Q_{2x}Q_{1a} + Q_{2y}Q_{1z} - Q_{2z}Q_{1y}) \\ (Q_{2a}Q_{1y} - Q_{2x}Q_{1z} + Q_{2y}Q_{1a} + Q_{2z}Q_{1x}) \\ (Q_{2a}Q_{1z} + Q_{2x}Q_{1y} - Q_{2y}Q_{1x} + Q_{2z}Q_{1a}) \end{bmatrix} \tag{5.3}$$

The inverse of a rotation Q is simply the same rotation angle about the reversed axis, corresponding to the hypercomplex conjugate:

$$Q^{-1} = [Q_a, -Q_x, -Q_y, -Q_z] \tag{5.4}$$

Thus, a relative rotation from orientation Q_1 to Q_2 is easily calculated as $Q_2Q_1^{-1}$.

Smoothly distributed rotations may be generated at random by picking values of a, x, y, z uniformly in the range $[-1, +1]$, discarding those for which $a^2 + x^2 + y^2 + z^2 > 1$ (i.e. that fall outside the unit 4-sphere), and normalizing the resulting quaternions. An alternative method, which avoids the wastage of rejected selections, is to pick the values of a, x, y, z each from a standard Gaussian distribution (expectation zero and arbitrary variance), and normalize these. This may be achieved by applying an inverse cumulative distribution function to uniform random variables in $(0, 1)$ [Shoemake, 1992].

5.1.1 Comparison with Euler Angles

A significant improvement over an explicit matrix representation is to store rotations as three angles, disallowing their direct composition, and calculating the matrix only when it is required. When matrices are used, their values must be kept normalized to ensure that a true rotation is represented — rounding errors could introduce skewing or distorting effects when applying the transformation to a molecule's pose.

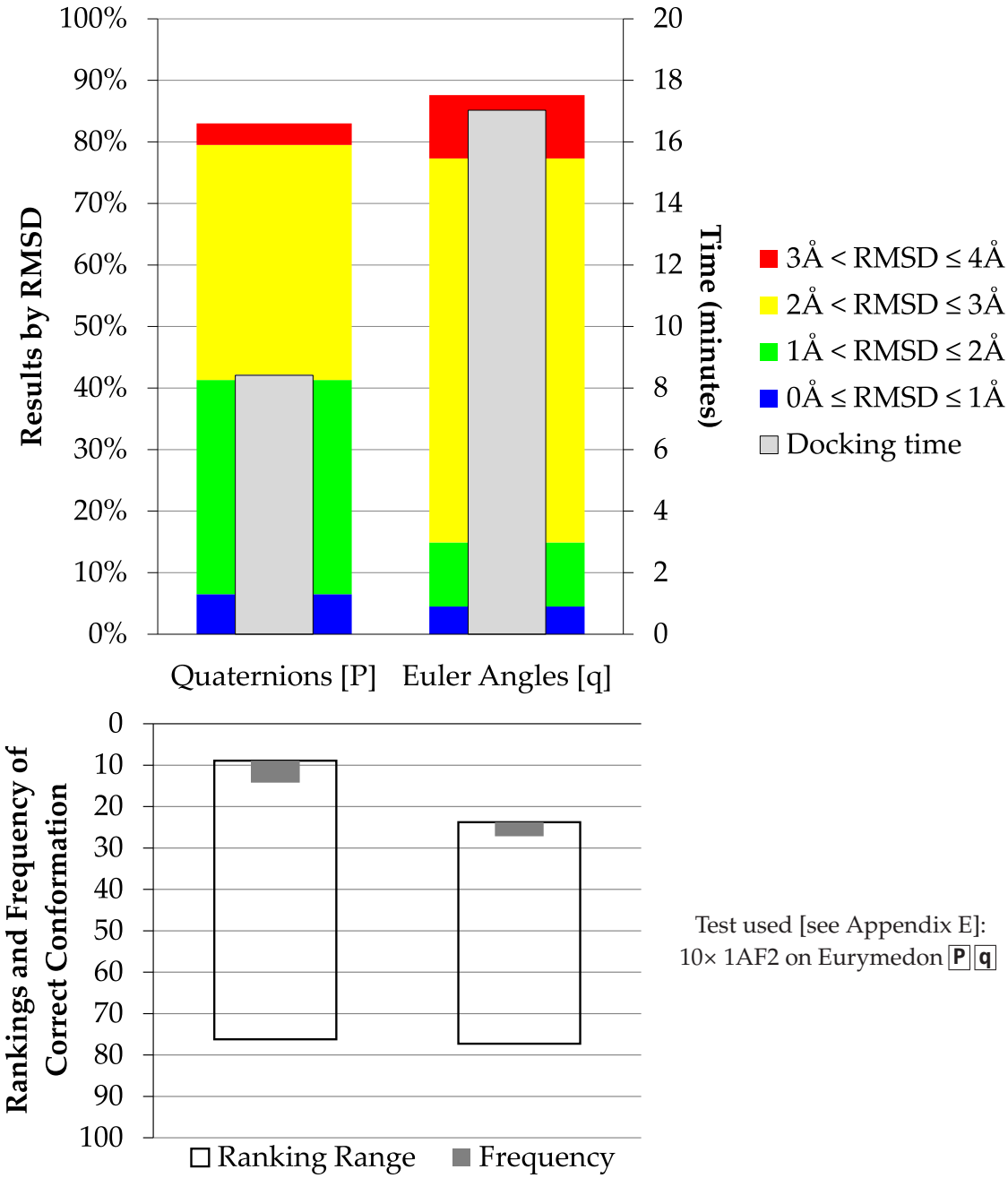


Figure 5.1: Comparison of quaternions with Euler angles for rotation representation, showing the improved speed and results when using quaternions

To demonstrate this, I created a general `Rotation` class to represent and apply rotations defined by three values or an angle-axis specification, making it possible to use either Euler angles or quaternions interchangeably according to compiler directives (see §E.3 (*p*.201)). A comparison of the P and q editions — identical except for the rotation implementation — demonstrates clearly the great improvement in efficiency offered by quaternions. Figure 5.1 shows a 50% reduction in run time with more results placed very close — within 1Å RMSD — to the true pose when docking the 1AF2 test

case. Faster execution is a consequence of the reduced data size and fewer arithmetic operations demanded by quaternions, and the improved results can be explained by the more uniform behaviour in the genetic algorithm.

The existing implementation using Euler angles used three angles as the search variables, which results in a bias towards selecting polar poses. To repair this, the inclination (latitude) parameter could be searched by cosine value instead, making random selections more evenly distributed. However, this will then make small mutations in the search have widely different effects depending on the value: a change in the cosine value from 0.00 to 0.01 represents a movement by 0.573° but 0.99 to 1.00 represents 8.11°.

5.2 Predicted Pocket Positioning

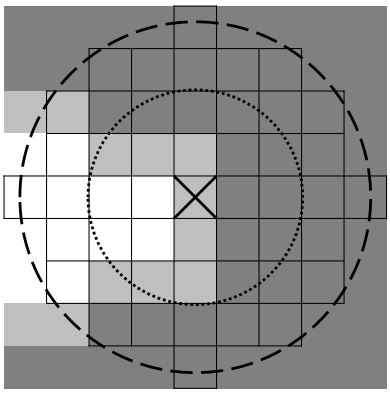
The task of searching all possible poses for a ligand conformation is one of huge combinatorial scope. Any means of focusing this into a smaller range of possibilities is helpful. This is why the prediction of active sites, discussed in §2.3.2 (*p*.34), is a topic of interest. If a receptor can be analysed before any ligands are presented to estimate where a ligand is likely to bind, then this information can be used to initialize any search poses and improve the likelihood of finding a good placement.

Since active sites are generally large crevices on the receptor's surface [Laskowski *et al.*, 1996], the identification of significant concave regions around a molecule is a desirable facility to have. If the representation of the pocket shapes can be used to calculate preferable orientations of the ligand as well as translations, then this is a bonus.

5.2.1 PIES

Although several approaches to pocket detection have been published and reviewed [Sotriffer & Klebe, 2002], the *XScore* implementation work led me to invent a geometric method of my own based on a predicted concavity measure calculated from the already-defined `SpatialOccupancy` data.

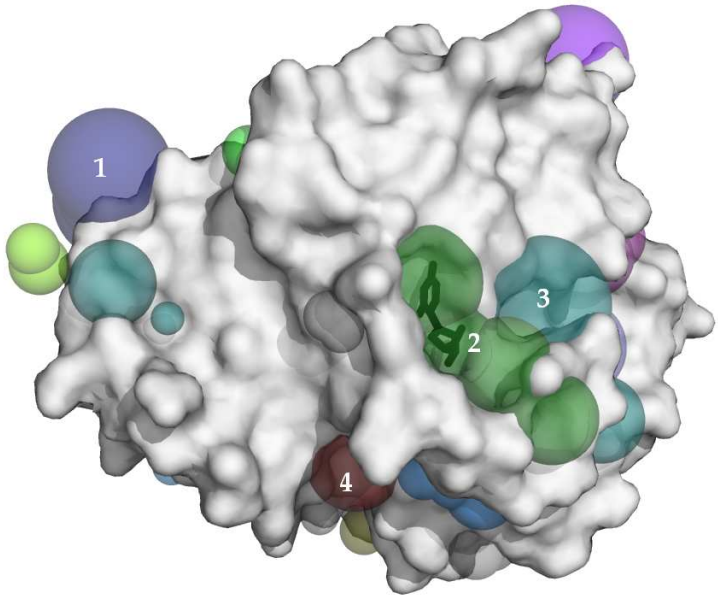
My Pocket Identification by Encroaching Spheres (*PIES*) algorithm provides a simple method for identifying cavernous sections of a molecular surface in a space-filling form,



■ Dark grey voxels: 'Internal'
■ Light grey voxels: 'Surface'
□ White voxels: 'External'

The dashed circle marks the *Cavern* radius and the dotted circle the *Fill* radius from the crossed surface point being considered.
To determine whether this is a concave point, the inverse proportion of external points within the dashed circle is counted: 49/9.
If this is greater than *Threshold*, the external voxels within the dotted circle are added to the pocket.

Figure 5.2: Grid-based concavity measure used by *PIES*



Ligand shown as dark sticks to left of label '2'.

Top four ranked pockets numbered. 28 pockets identified in total.

Ligand in second pocket, filling 48% of pocket volume.

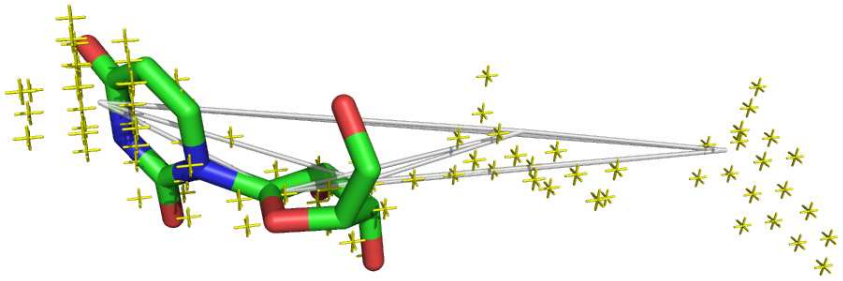
Correct pocket is 11% of total volume of all identified pockets.

Calculation took 4.3 seconds (using Eurymedon).

Figure 5.3: *PIES*-5-3-7 pocket detection analysis of 1AF2 protein

suitable for initializing a docking procedure. It uses a rectilinear lattice to represent the volume of the molecule, and the ratio of occupied to unoccupied cubes in spherical regions centred on the boundary to estimate the concavity at each point, as illustrated in Figure 5.2. The atomic radii are inflated by 0.5Å to eliminate any small crevices (thinner than 1Å), smoothing the molecular surface (akin to a solvent-accessible boundary), simplifying its processing, and avoiding an excess of undesirably small pockets.

The unoccupied volume near the concave boundary points is collected into spheres, and these are clustered together if they are sufficiently close (that is, the radius of their union is within 20% of the existing size, a limit chosen experimentally to balance output detail with simplicity). Finally, those clusters containing volumes above a given size



■ Yellow jacks mark lattice points filling the pocket.
■ Grey rods are the alignment axes identified.

Figure 5.4: Detail of 1AF2 active site as identified by *PIES*-5-3-7 showing native ligand pose

are recorded as the predicted pockets, with lines between the member spheres' centres and enclosed within the cluster recorded for use as alignment axes. The overall method, described in full in Listings 5.1 and 5.2, requires six inputs:

Mol: the molecule to be analysed,
Res: the resolution of the lattice to be used (typically 1Å),
Threshold, Cavern, Fill: the ratio and radii for concavity estimation,
MinSize: the minimum acceptable size for the clusters collected ($\frac{1}{4}\left(\frac{Fill}{Res}\right)^3$ by default).

The resolution and minimum cluster size are not generally specified and the defaults above used. *MinSize* is based on a minimum equivalence to the volume of a spherical cone of the *Fill* sphere with diameter also equal to *Fill* (approximately $\frac{1}{16}$ of the sphere), a value that seems to work well experimentally. The particular parameters used are denoted *PIES-cavern-fill-threshold*. Increasing the *Cavern* produces fewer clusters and with fewer spheres each. Using a larger *Fill* increases the number of clusters, making them larger. Raising the *Threshold* reduces the number and size of clusters.

The construction of the `SpatialOccupancy` data is linear in the number of atoms N, while the number of boundary points for iteration should be at most proportional to N for all reasonable receptors. The number of clusters that may be generated is bounded by the number of surface points, and these are only iterated in non-recursive patterns, so the overall complexity should be approximately $\mathcal{O}(N)$.

My `SphereCluster` implementation (used for all collections of volumes) keeps its list of voxels (volume elements) ordered by non-increasing distance from their centroid, and maintains the outer bounding sphere. This allows overlaps to be identified more efficiently, in the same manner as with sphere trees (shown in Figure 3.1 (*p*.55)). More

```
Strip any hydrogen atoms from Mol
Increase the radius of each atom in Mol by 0.5Å
Create Mesh, the spatial occupancy mesh for Mol with resolution Res
For each voxel V labelled 'Edge' in Mesh do
  Begin
  N := the number of voxels in Mesh within Cavern radius of V ★
  NO := the number of these labelled 'Outside'
  If NO > 0 and N/NO > Threshold then
    Begin
    Create cluster C1
    Add to C1 the 'Outside' voxels in Mesh within Fill radius of V ★
    If size of C1 ≥ MinSize then Append C1 to S
    End
  End

//S contains many small spherical clusters in the pockets

I := 1
For each cluster C1 in S do
  Begin
  Label C1 with I
  For each cluster C2 in S where C2 precedes C1 in S do
    If C1 and C2's labels differ and their separation ≤ Res then
      Label with I all clusters in S with the same label as C2
  Increment I
  End
Sort S by the clusters' labels

//S now has the same clusters labelled by pocket numbers

I := 0
N := 0
For each cluster C1 in S do
  Begin
  If I ≠ C1's label then
    Begin
    If I ≠ 0 and N > Cavern then
      Begin
      Label Group with N
      Append Group to Final
      End
    Clear Group list
    I := C1's label
    N := 0
    End
  Append C1 to Group
  N := N + size of C1
  End
If I ≠ 0 and N > Cavern then
  Begin
  Label Group with N
  Append Group to Final
  End

//Final is a list of cluster lists, one per sufficiently large pocket
```

★ marks lines modified in *PIECE* (§5.2.1 (*p*.94)). *continued in Listing 5.2...*

Listing 5.1: *PIES* pocket detection algorithm outline [part 1 of 2]

...continued from Listing 5.1

```
For each list of cluster Group in Final do
  Begin
  I := 1
  While I ≠ 0 do
    Begin
    I := 0
    For each cluster C1 in Group do
      Begin
      Find C2 in Group which merged with C1 has least radial increase
      If the radius increase < 20% then
        Begin
        Merge C2 into C1
        Remove C2 from Group
        Increment I
        End
      End
    End
  Append SphereAxis (Group, {}) to Output
  End

//Output contains a SphereAxis for each pocket
//SphereAxes are clusters with a list of SphereAxis references

For each array of SphereAxes X in Output do
  For each SphereAxis A in X do
    For each SphereAxis B in X where B ≠ A do
      Begin
      V := the midpoint between A and B
      If V is inside a member of X padded by Fill/8 then
        Append B's index in X to A's array of integer
      End
Return Output
//Output contains the list of pockets as lists of spheres
//with alignment axes inside them
```

Listing 5.2: *PIES* pocket detection algorithm outline [part 2 of 2]

importantly in this application, finding the minimum separation of points in two clusters subject to a maximum acceptable distance — an operation required for merging the spheres into coalescing clusters — can be accomplished quickly because it is easily possible to select and consider only those points that *may* be within that range.

My implementation of this algorithm, as the `PropPlacePIES` class, provides several outputs besides that used by the docking software and recorded with the receptor. The spheres used to enclose the pockets are recorded, colour-coded by cluster rank, to make it possible to see which pockets were identified with what preference. Figure 5.3 shows this representation for the 1AF2 receptor, where the second-ranked pocket contains the true binding site, and is approximately twice the size of the ligand.

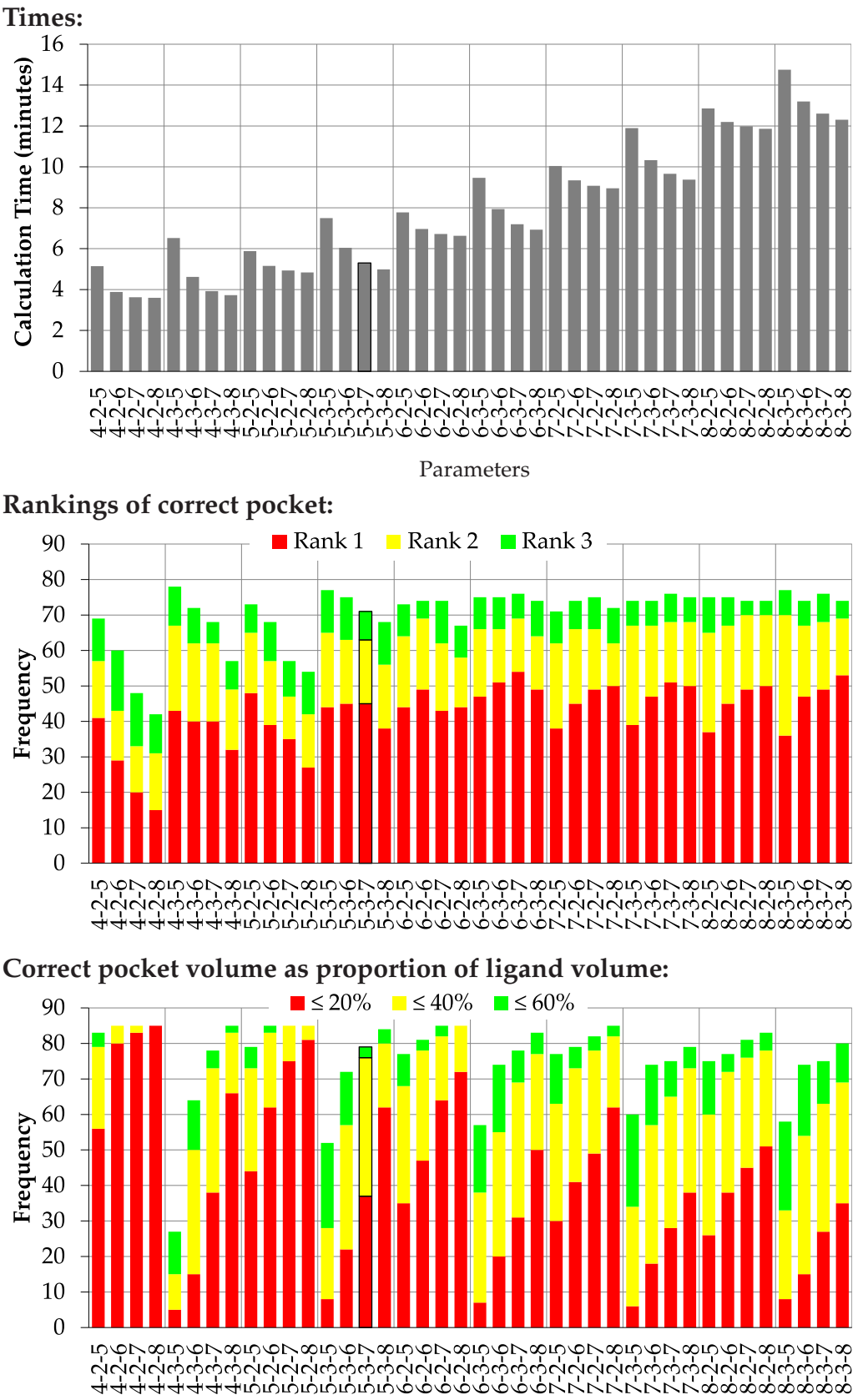
The active site is shown again in Figure 5.4 with more detailed output data: this includes the precise volume (as lattice cells) of the pocket identified and the lines offered for alignment. The yellow markers illustrate the necessity of the spheres in the method: the actual volume of the pocket is quite limited and, although it coincides well with the native pose of the ligand, it does not envelop it. The use of spheres to encroach upon the receptor's surface allows the pocket representation to fill up and provide a larger, more useful space in which to position ligands, while retaining the important details of the shape in a simpler description.

Parameters

The three parameters that are available for user input — the *Cavern* and *Fill* radii and the *Threshold* value — need to be chosen, ideally maximizing the flexibility of the method's application. To assess the variables' effects properly, I ran the *PIES* algorithm for every member of the *Astex* Diverse Set (listed in §E.1.3 (*p*.199)) using 40 different configurations: $Cavern \in \{4,5,6,7,8\}$, $Fill \in \{2,3\}$, and $Threshold \in \{5,6,7,8\}$.

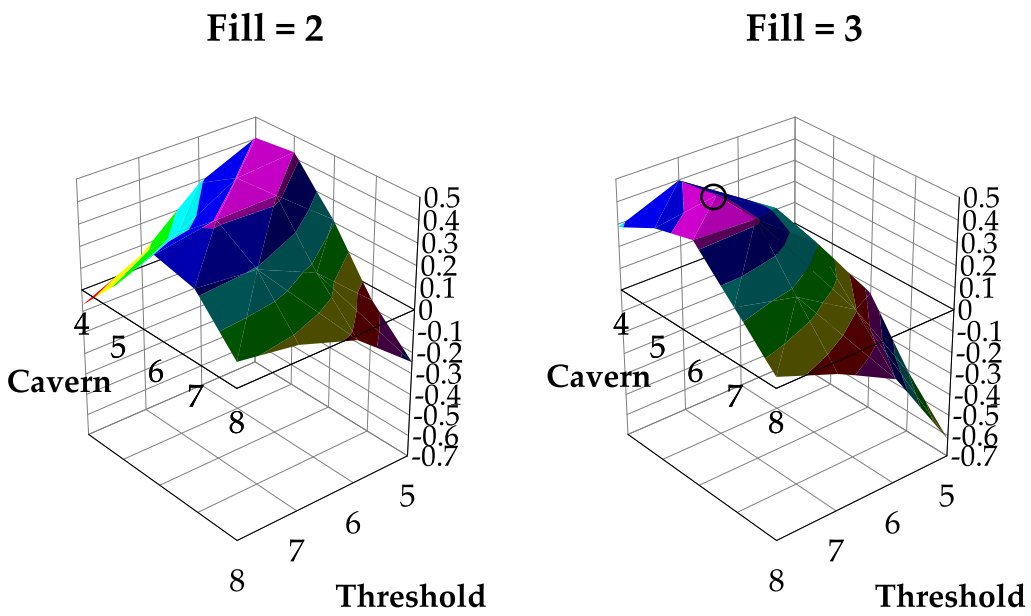
Figure 5.5 compares three important statistics about each of the variations, aggregated over the 85 examples used. The timings show a clear trend: increasing either of the radius values increases the calculation time, and increasing the concavity threshold decreases it. At smaller cavern sizes, the threshold is inversely proportional to the successful rankings of the correct pocket, but with larger caverns this trend is reversed and the top-3 rankings become fairly stable. This should be expected: small caverns are insufficient for the concavity measure to be accurate. The fill radius has little effect on rankings, but does influence the size of the pockets. A larger fill results in the pockets chosen being much less specific than with small fills: they are larger, and indeed there tends to be at least 200% more pocket volume altogether. However, it is preferable to have the pocket volumes up to around half the size of the ligand because this provides some useful shape information for alignment and flexibility in positioning.

Selecting a particular configuration from these statistics is not simple. A choice that minimizes calculation time cost but maintains a high rate of successful and usefully-sized pocket selection is sought, and so I express this requirement as a formula to quantitatively evaluate each set of results. The formula and its graphs are shown in Figure 5.6. The



Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon **A**
(*PIES* calculation only, various parameters)

Figure 5.5: Effects of *PIES* parameters on processing *Astex* Diverse Set, comparing calculation times and accuracy of results, marking the combination selected for normal use



Assessment formula used (for the 85 cases of the *Astex* Diverse Set):

$$\frac{\#(rank = 1) - \#(rank > 3)}{85} + \frac{\#(pocket/ligvol \leq 40\%) - \#(pocket/ligvol > 60\%)}{170} - \frac{minutes}{15}$$

Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon [A]
(*PIES* calculation only, various parameters)

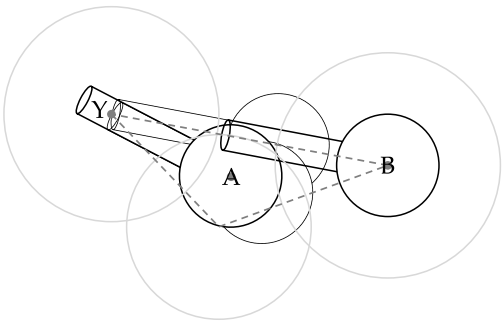
Figure 5.6: Quantitative assessment of *PIES* parameters' effects, showing the best combinations and the selected option

peaks of the function are the favourable configurations, reducing the number of areas to consider enormously. The pattern that emerges is that for the smaller fill radius the cavern size and threshold need to be proportional — increasing one must be matched with an increase to the other. With the larger fill, the best options are smaller caverns with larger thresholds.

Although the best option by this crude rating is *PIES*-6-3-8, this is relatively slow compared with the otherwise similar second and third cases, 5-2-5 and 5-3-7 respectively. The latter produced more pockets between 20% and 40% of ligand volume, which should be beneficial for pre-positioning. Hence, I have used *PIES*-5-3-7 as the standard configuration in this work. The effect of pre-alignment using *PIES* for docking is discussed in §5.2.4 (*p*.99).

PIECE

As a simplifying adaptation of the *PIES* algorithm, the concavity measure was altered to count the voxels within *Cavern*- and *Fill*-half-sided cubes of each surface point, rather



■ Light grey lines: *PIES* pocket clusters
■ Grey dashed lines: *PIES* alignment axes
■ Black outlines: *PASTRY* core and yokes

The core is placed between A, the pocket centroid position, and B, a randomly chosen sphere centre. The rotation directs one yoke at Y, the far end of an alignment axis. The core's translation is then interpolated with the positioning of the chosen yoke's tip at Y (shown as dotted outlines).

Figure 5.7: Diagram showing placement of *PASTRY* onto *PIES*, including the alignments interpolated to generate poses

than spheres. The lines in the *PIES* algorithm (Listing 5.1) that are changed by this are marked with stars. This derivative method is called Pocket Identification by Encroaching Cubes for Efficiency (*PIECE*). To compensate for the greater volumes being considered by the same parameters, I reduced the radii and used *PIECE*-4-2-7 for comparison.

PIECE is quicker to calculate. The entire *Astex* Diverse Set can be analysed in nearly 30% less time for very little change to the algorithm. A comparison of *PIES* and *PIECE* is given in §5.2.3 (*p*.97), evaluating their ability to correctly identify pockets. Although *PIECE* appears much more successful, it is also the case that the pockets it produces are much larger and thus indiscriminate than *PIES*, making it arguably less useful overall. The use of cubic volumes to fill the pockets makes the method dependent on the provided orientation of the molecule to be assessed, which is undesirable. Consequently, I have continued to use *PIES* in this work; the time cost is acceptable for more precise pockets.

Placement with PASTRY

When poses must be generated in a pocket identified by *PIES*, the *PASTRY* shape descriptor (introduced in §5.3.2 (*p*.107)) may be used to assist with positioning. Figure 5.7 illustrates the poses that may be interpolated from four arrangements of a particular random selection of pocket cluster, alignment axis, and yoke. The yokes are chosen with probabilities proportional to their weights, so that larger appendages on a ligand are more likely to be accommodated in the pocket. If a pocket with a single sphere is used, the ligand is simply placed with its core in the centre and oriented at random.

Pre-alignment for docking using *PIES*, *PIECE*, and *PASTRY* is evaluated in §5.2.4 (*p*.99).

5.2.2 PASS

Having developed the *PIES* method, I compared it with some similar algorithms. While a few methods were based on rectilinear lattices, *POCKET* and *LIGSITE* in particular, they employed scan-line techniques for cavity detection. The interior-exterior ratio concavity measure I present does have some similarities with the Putative Active Sites with Spheres (*PASS*) algorithm described by [Brady & Stouten, 2000].

PASS finds every point where a probe sphere may be stacked on the surface of the protein by resting between three atoms. Each probe is assigned a burial score equal to the number of atoms within an 8Å radius. Probes scoring less than a threshold (55 being recommended by the algorithm's authors) are discarded. The process is then repeated, stacking another layer of probe spheres atop those already placed, maintaining the neighbourhood atom count and disallowing any new probes that would overlap with old ones. When no more probes can be added, each is given a weighting determined by its neighbours' burial scores, and the weightiest well-separated probes are selected as active site points (ASPs). If pocket-filling volumes are required, those probes touching an ASP probe are grouped with it, and their neighbours are recursively accumulated. Several parameters are required:

Probe: the radius of the probe spheres,
BCmin: the minimum burial count score for a probe to be retained,
BCrad: the radius within which to count burial counts,
Weed: the minimum separation of probe centres (1Å),
Accretion: a smaller radius for non-primary probe spheres (0.7Å),
ASPrad: the minimum separation of ASPs (8Å),
PWmin: the minimum weighting for ASP acceptance (1100).

The values in brackets are fixed in my implementation based on the recommendations of the original definition. The first three, however, may be varied, and in this work a configuration is written in the form *PASS-BCrad-probe-BCmin*, although I have used the authors' values: *PASS*-8-1.8-55. For full discussion of the behaviour of the method, the reader is referred back to [Brady & Stouten, 2000].

The task of finding triples of atoms that can support a probe sphere is, naively, $\mathcal{O}(N^3)$ in the receptor size, making the overall algorithm cubic in complexity. In practice, this

can be reduced close to $\mathcal{O}(N^2)$ if, for each atom, all sufficiently close atoms are identified and this small list is then used to complete the triples. Assigning the probe burial counts requires a linear scan of the atoms for each (potential) probe, but since a relative minority of the atom triples will result in probes this is an insignificant factor in the execution. Calculating the probe weightings is quadratic in the number of probes retained, which in turn is partly proportional to the size of the molecule. Although this does not necessarily add a large cost to the calculation, for some inputs it could become significant.

My implementation of the algorithm, `PropPlacePASS`, reused the `SphereCluster` class for collecting probes, since this provides an efficient storage method for the pocket volumes and the collection of probes to ASPs.

Spatial Indexing

The use of spatial indexing should reduce the calculation of *PASS* further towards linear complexity, by allowing the identification of near-neighbouring atoms from pre-collected regions rather than the entire molecule. Only one iteration over the atoms is then required, and the number of neighbourhood atoms to consider for each will be approximately constant (proportional to the *Probe* parameter). Building the index is a linearly complex process, as is coating the receptor with probe spheres. Calculating burial scores for the probes can also employ the index for efficiency.

I made these improvements to the implementation by defining a subclass of the `SphereCluster` (see *p*.89) called `IndexedCluster` which maintains a collection of buckets containing the points. Each bucket corresponds to a cube of space; the size of the cubes is defined when the class is constructed. With the use of spatial indexing, one would expect *PASS* to be much better than cubic, between $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$.

5.2.3 Comparison of Pocket Detection Methods

Statistics about the pockets identified and the correct binding site are calculated and recorded by my implementations. The entire *Astex* Diverse Set has been analysed with both *PIES*-5-3-7 and *PASS*-8-1.8-55, and in every case (except two using the indexed *PASS* method) the true binding site was identified amongst the results produced.

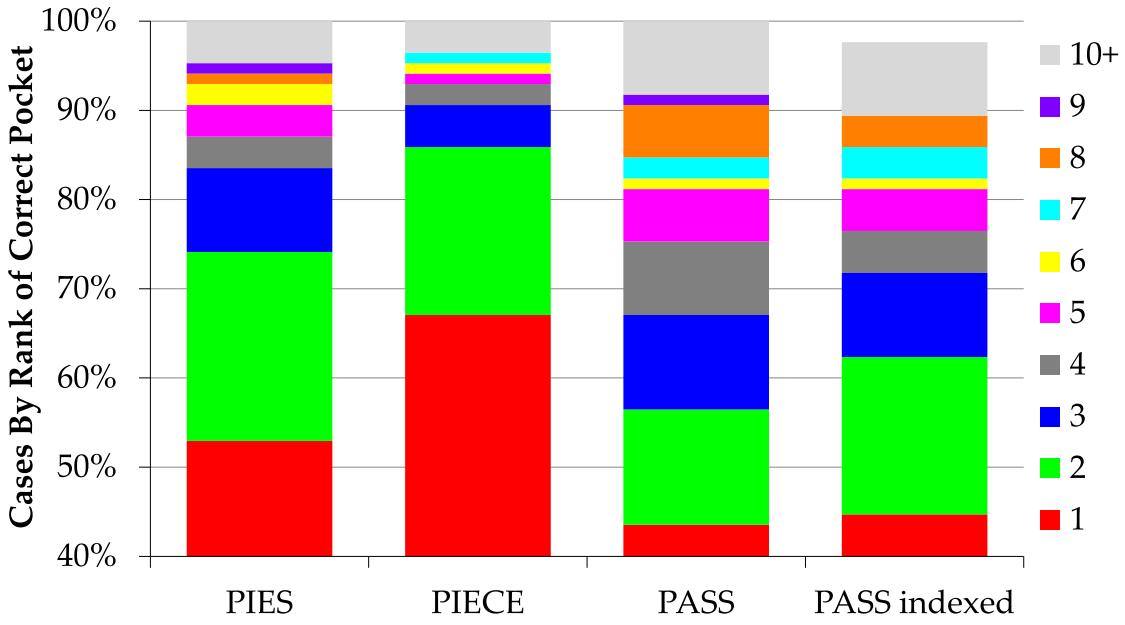
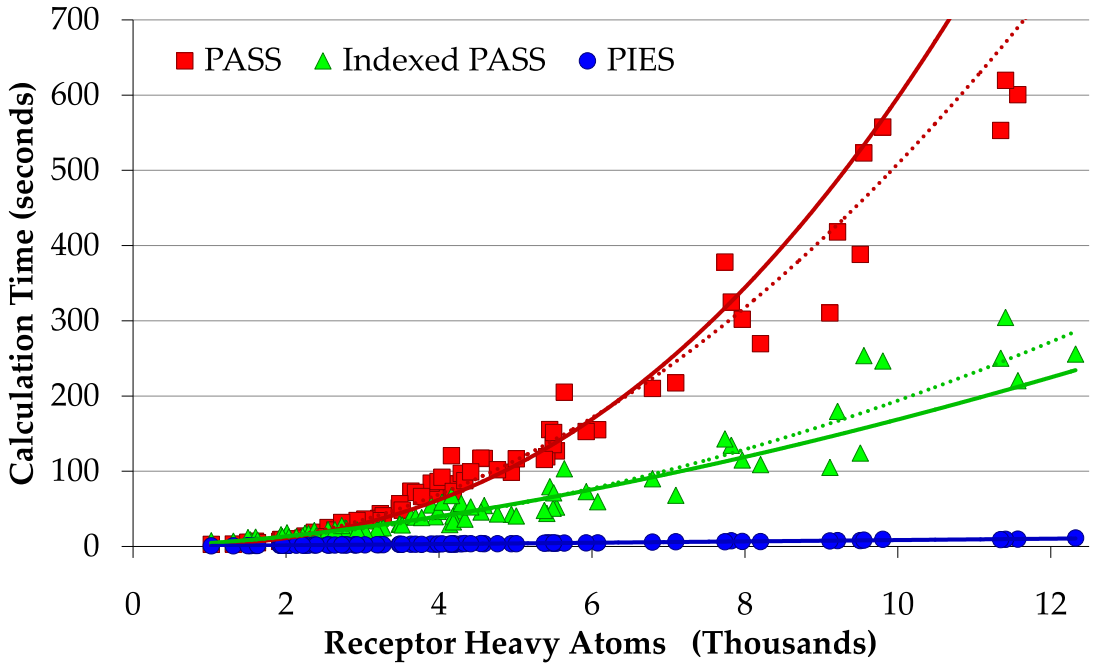


Figure 5.8: Comparison of pocket detection methods' rankings of the correct active site in *Astex* Diverse Set cases, showing the higher success rates of *PIES* and *PIECE*

Figure 5.8 summarizes the results from each method. In more than half (53%) of the cases, the top ranked pocket by *PIES* was the correct one, and 84% had the correct site in the top 3. *PASS* (with indexing) ranked the correct pocket first in 47% of the cases, and in the top 3 for 74%. Comparing these data shows that *PIES* is slightly more successful at selecting the pockets over the diverse set of examples.

The calculation times differ much more. The *PIES* method almost universally produced its results in under ten seconds, mostly under five. The *PASS* algorithm, however, varied much more and had a substantial proportion of cases which took more than two minutes to complete. Figure 5.9 shows the relationship between receptor size (measured by heavy atom count (HAC)) and the pocket detection times. This allows the complexity of the methods to be verified: the lines are least-squares fitted in the form $t = kN^p$ for time t and heavy atom count N. The *PIES* method (blue circles) shows a linear trend — the exponent is almost exactly 1 — whereas *PASS* (red squares) exhibits polynomial behaviour. The dotted lines show the best fits of precisely quadratic curves, reinforcing the conclusion that *PASS* must be $\mathcal{O}(N^2)$. The adapted version of *PASS* that employed spatial indexing (green triangles) was substantially improved, and resulted in an exponent of approximately 1.5. This is much better, and serves to emphasize the



PASS-8-1.8-55: $t = 2.03x^{2.47}$
Indexed *PASS*-8-1.8-55: $t = 4.55x^{1.57}$
PIES-5-3-7: $t = 0.75x^{1.06}$

$x = \frac{N}{1000}$ N is heavy atom count

Dotted lines mark quadratic curves fitted to the data for comparison with powers.
Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon [A]
(*PIES* and *PASS* calculation only)

[1OF6 is an extreme case ($N = 20\,925$, nearly twice the size of the second largest molecule) which has been omitted for clarity in the graph. It did not contradict the trends shown.]

Figure 5.9: Pocket detection calculation times relative to molecule size, validating the complexities of the *PIES* and *PASS* algorithms

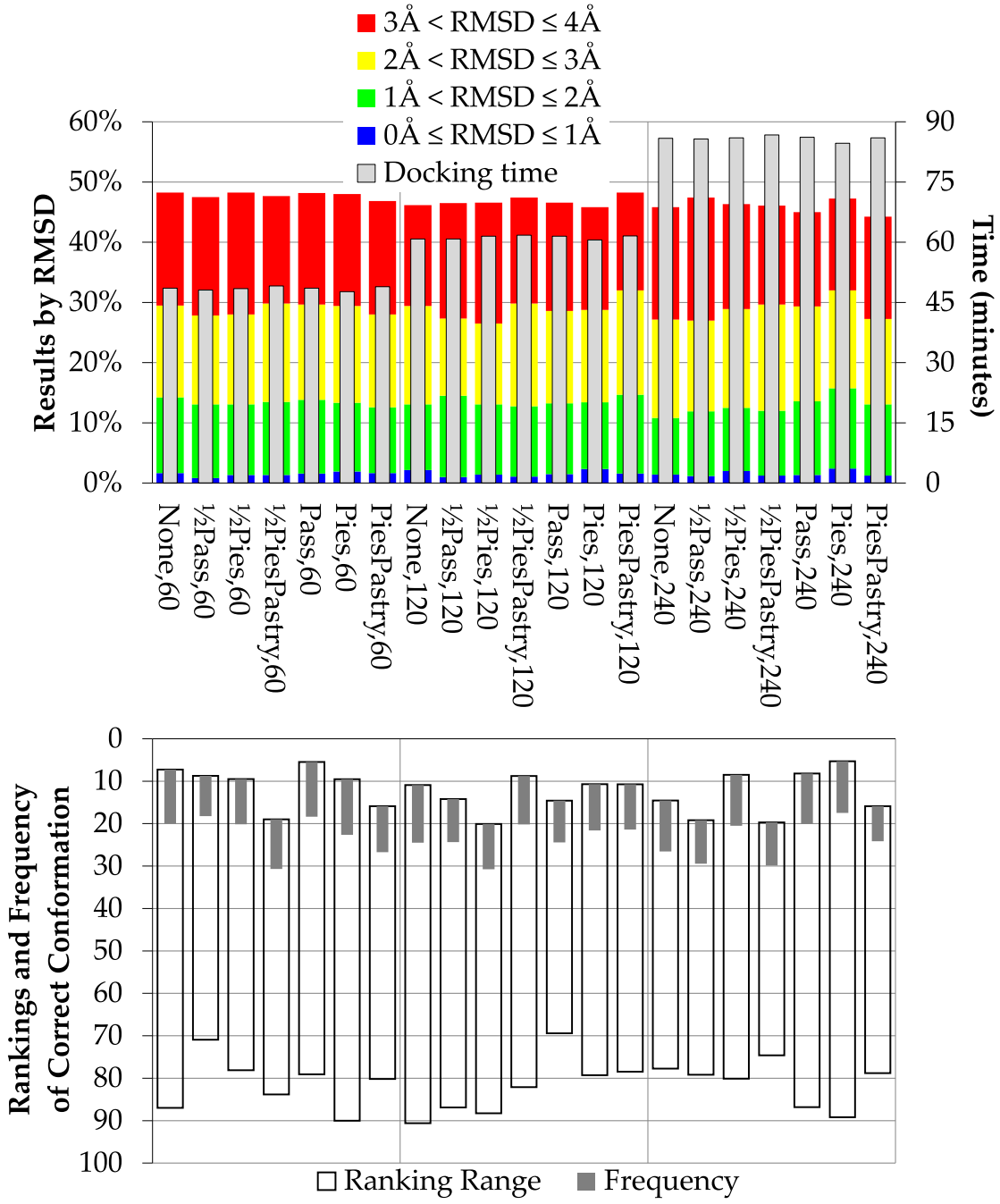
benefits of indexing as an optimization stratagem. Good sorting algorithms are generally $\mathcal{O}(N \log N)$, and so even with efficient data storage for handling the probes it is unlikely that *PASS* could be made linear. Indeed, it is probably hard to improve on $\mathcal{O}(N^{1.5})$.

5.2.4 Pre-Positioned Docking Results

Having established that these methods can identify the correct binding site of a wide range of receptors, the question arises of whether this aids docking searches. To find out, I redocked the *Astex* Mini Set using the [A] edition of *DOX*. This includes all the geometric properties (both pockets and ligand shapes) and allows the `PrePositions` class (see §F.4 (*p*.208)) to recognize the additional `place` option. I tested seven pre-positioning configurations, as listed in Table 5.1: entirely random, or either 50% or 100% of initial poses generated using the top five pockets from either *PIES*-5-3-7, *PIES*-5-3-7 with

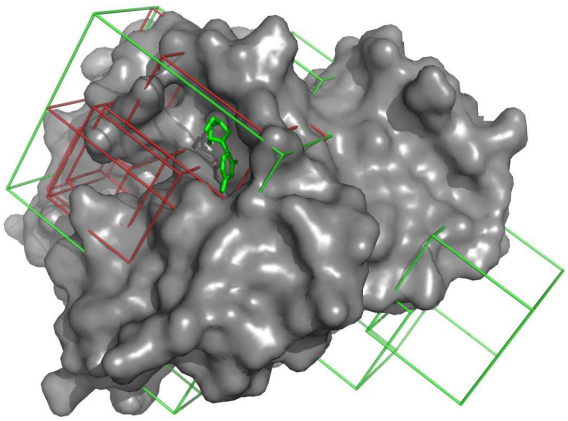
Name	random	place
None	100%	
½Pass	50%	1 5 PlacePASS-8-1.8-55
Pass	0%	1 5 PlacePASS-8-1.8-55
½Pies	50%	1 5 PlacePIES-5-3-7
Pies	0%	1 5 PlacePIES-5-3-7
½PiesPastry	50%	1 5 PlacePIES-5-3-7 AlignPASTRY
PiesPastry	0%	1 5 PlacePIES-5-3-7 AlignPASTRY

Table 5.1: Pre-positioning configurations for pocket placement tests



Test used [see Appendix E]: 1× *Astex* Mini Set on Eurymedon A (various place options)

Figure 5.10: Comparison of configurations from Table 5.1 for pre-positioning ligands in predicted pockets, showing similarity of docking time and results



■ Green: Final boxes chosen and Correct bound ligand pose
■ Red: Predicted boxes before combining

Test used [see Appendix E]: 1× 1AF2 on Eurymedon CA
(search box prediction only, 5 each from *PIES* and *PASS*, maximum overlap 12.5%)

Figure 5.11: Search boxes automatically generated for 1AF2 using *PIES*-5-3-7 and *PASS*-8-1.8-55 in combination

PASTRY, or *PASS*-8-1.8-55. Using the methods in combination together would confuse the results, but permitting the inclusion of random poses alongside those calculated allows for the possibility that analytical pose selection is too precise. I tried the usual 240-generation GA search with each of these, but also 120 and 60 generations, to establish whether the use of pocket detection can replace the early part of a stochastic search.

As Figure 5.10 shows, there is little difference noticeable between the various configurations. With 240 generations the results are fractionally better when using *PIES*, but this is not substantial. Crucially, even without any guidance from pocket predictors, a short search of 60 generations was still capable of identifying a comparable set of poses. These tests were performed using a search box definition based on the known pocket, and so the scope for refinement by a pocket detection algorithm is quite limited. A well-chosen box alone is evidently sufficient to guide a docking.

5.2.5 Automatic Search Extents

The search box must be supplied somehow. Rather than expecting a human to analyse every unknown receptor and provide manual input about which site to test, the pocket detection methods can be used to determine a search box (or boxes) automatically. This in turn can define the LUT extents to be calculated and considered.

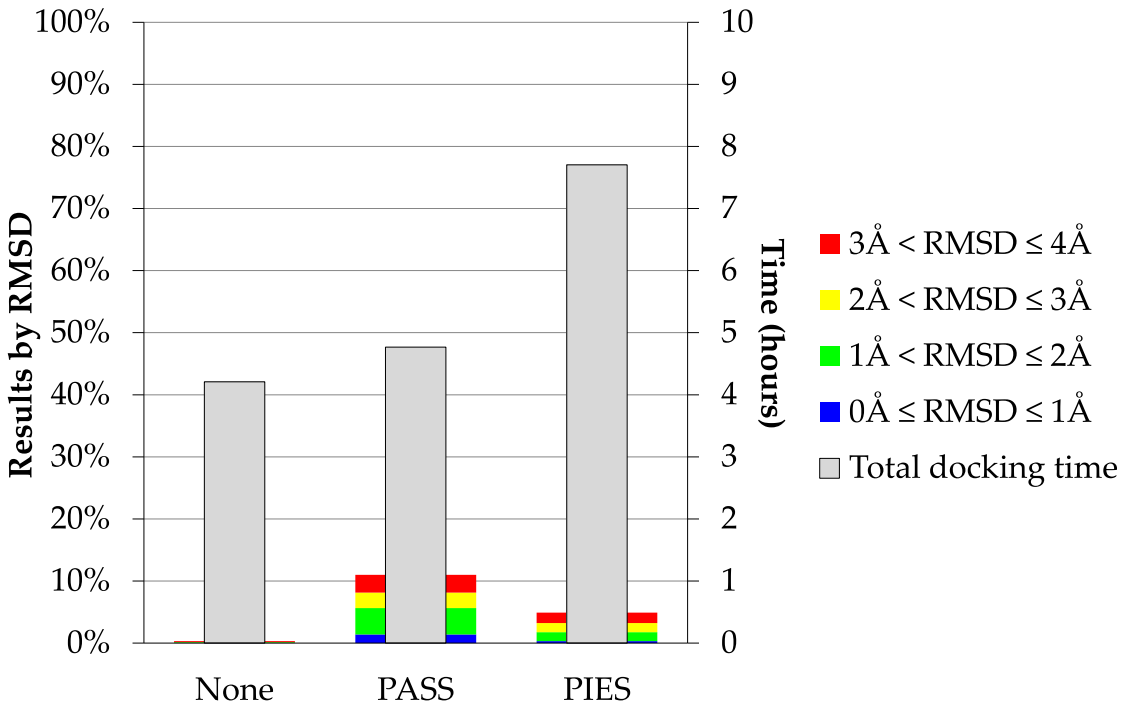
Since it is inconvenient to handle several search regions or LUTs for a single receptor, and since we may find that the active sites chosen by a computational method intersect, the multiple search box situation provides a useful application for the caching LUTs

```
//NarrowEvery and NarrowTo are input parameters
//Prepare boxes...
Create BS, an empty list of boxes
For each place property P do
  Append to BS the search boxes proposed by P
//Prepare searches...
Let SN := count of BS
Construct SN search algorithm instances, S[1..SN]
Initialize each S[i] to explore extents BS[i]
//Perform usual searches...
For Steps := 1 to MaxSteps do
  Begin
  For each search AS in S do
    Begin
    Step AS forward
    //Local optimization and/or elitism here, if necessary
    End
  If (SN > NarrowTo > 0) and (Steps mod NarrowEvery = 0) then
    Begin
    //Squash the least promising box
    Obtain Top[1..SN], the best scores from each search
    Find QI, the index of the worst score in Top
    Terminate search S[QI], and remove it from S
    Decrement SN
    End
  End
Return combined result poses from remaining searches
```

Listing 5.3: Multi-box search narrowing algorithm outline

described in §6.3 (*p*.125). Rather than deferring the calculation of a scoring function for efficiency reasons, the lazy evaluation paradigm allows any irregular part of the receptor's surrounding volume to be considered. To support this, I adapted the `PrePositions` class to allow multiple search boxes to be defined simultaneously. These can be provided directly in the configuration file, or defined by the poses returned from a `place` entry. However, if two boxes overlap substantially (by more than a configurable proportion), they are automatically combined into one for simplicity. Figure 5.11 shows the resulting boxes for 1AF2 using both *PIES* and *PASS*.

Since searching multiple regions takes proportionally more time than one single region, I also added a periodic narrowing heuristic to the GA, as described in Listing 5.3. This abandons the least well-scoring box after every n generations, up to a limiting point, in order to reduce the time taken and eliminate undesirable poses. Multi-box searching may require more time to consider each pocket, but the amount of LUT data that will be calculated is tightly bounded and so the time cost should not be substantially worse than a single unconstrained exploration. Provided that the narrowing algorithm does not



LUT sizes: None: 47 263 178 entries, *PASS*: 11 150 653 entries, *PIES*: 20 083 857 entries
Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon CA
(various search box predictions, crystal conformations only)

Figure 5.12: Automatic search box docking results, demonstrating benefit of pocket prediction when docking without user direction

discard the correct binding site, the selective methods should be able to dock a ligand more accurately, since they will concentrate the search on likely poses.

I ran three of the configurations ('None', 'PASS', and 'PIES') listed in Table 5.1 again using the new CA edition, redocking the *Astex* Diverse Set ligands' crystal structures to their respective proteins. Five search boxes were constructed using the pocket detection algorithm in each case, padded by 5Å to ensure sufficient LUT coverage for the search to explore. The boxes were narrowed every 12 generations until two remained, and the search finished after 120 steps, with a thorough local optimization to conclude (see Listing E.1 (*p.*195)). For comparison purposes, the unguided (None) search was performed using the entire volume of the receptor as one box, also padded by 5Å. In all executions, the look-up tables were initially empty.

As Figure 5.12 shows, using a pocket detection method does improve the accuracy of the docking results. Without any guidance, the search method was generally unable to identify the correct binding poses. The greater number of search algorithm instances necessitated by using several search boxes does increase the time cost, although not

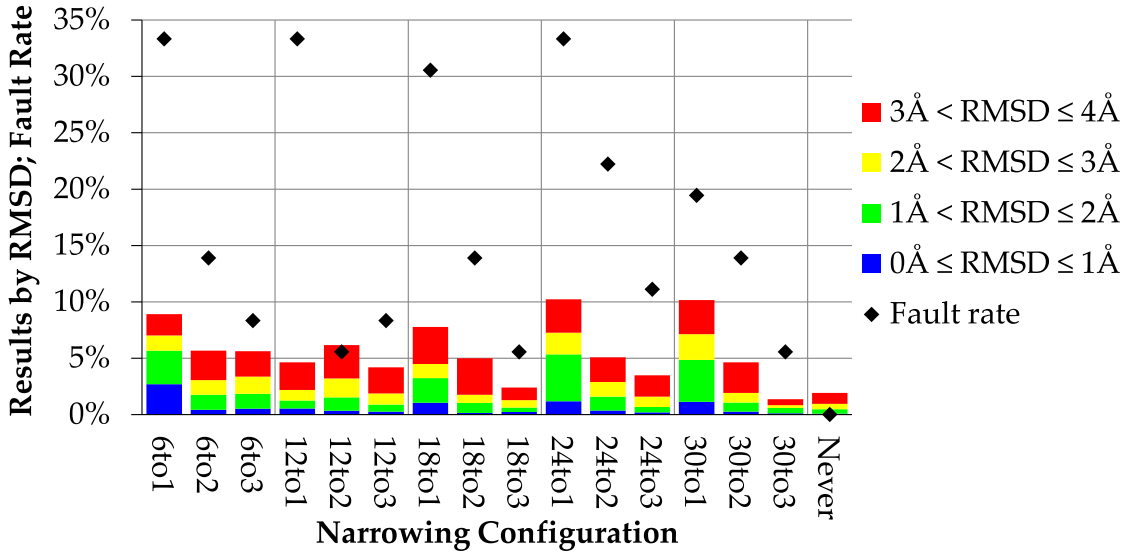
always significantly. The total search volume in the unguided case was more than 4 times that of the *PASS*-prepared method, and so this greater calculation demand balances the multiple search cost. *PIES* tended to generate larger pocket volumes, around 1.8 times those of *PASS* and so more time was spent calculating data in that case. To compensate for this, the LUT padding could probably be reduced when using *PIES*.

The problem of discarding the correct binding site when narrowing should be considered. The *PASS* tests dropped the true pocket in 15% of cases, whereas *PIES* did so in 25%. Consequently, the *PIES* tests were unable to produce as many well-placed poses. From this one can conclude that a method predicting larger pockets requires longer between narrowings to ensure that each pocket is properly evaluated before dismissal.

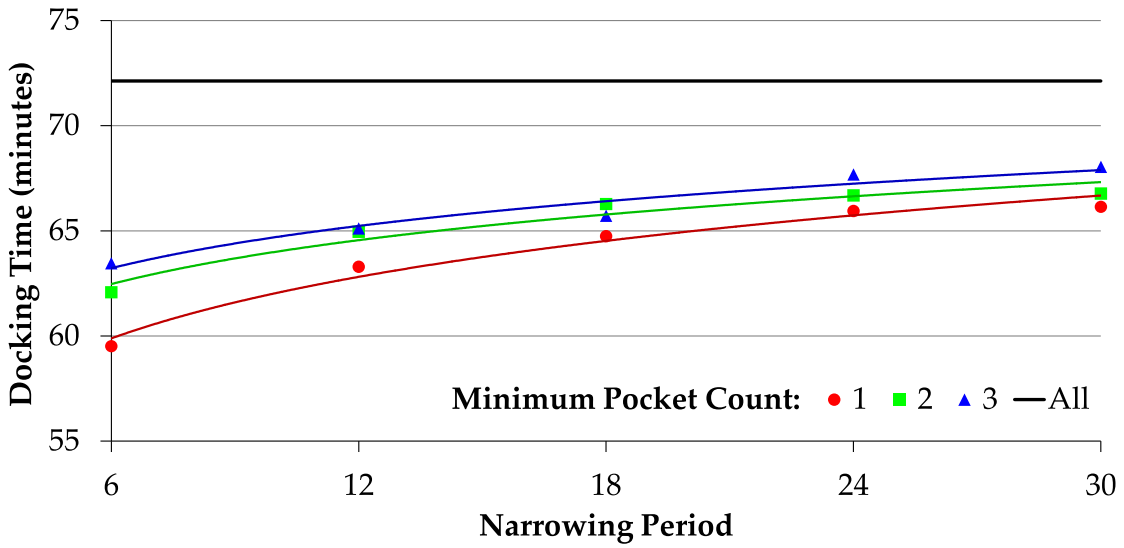
To investigate the effect of the two parameters — narrowing period and the minimum number of search boxes to retain — I used the *Astex* Mini Set with 15 different configurations of the heuristic. The period was varied between 6 and 30 generations, and the limit was 1, 2, or 3 boxes remaining. These combinations are compared in Figure 5.13, where (for example) the configuration labelled '12to2' had a narrowing period of 12 and kept 2 boxes. The docking times were increased by completing more boxes, and therefore searches, and by allowing longer between eliminations. Completing the full 5 box search took about 72 minutes, and using the narrowing technique reduced this by between 6% and 17% for the configurations shown.

The poses found using these automatic searches are not generally as close to the native structures as the results from manually specified single box searches, but this is unsurprising since the predicted pockets provide a much less constrained docking range. Good results within 2Å are still returned, though, and the minimum number of boxes substantially governs how many. Eliminating boxes (assuming they do not contain the correct binding site) prevents a large number of distracting candidates from being present in the output, regardless of their scores. For this reason, the result RMSD distributions on the graph are better for smaller values of *NarrowTo*.

Of course, discarding the correct box will abandon all the best poses. This fault occurred at an approximately constant rate of around 5–10% when retaining three boxes, but when reducing down to a single box the rate was over 30% with all but the slowest period. That particular configuration, '30to1', makes the final narrowing only after



'*p*to*q*': *p*=period, *q*=minimum boxes retained. Fault rate is the proportion of cases and executions in which the correct binding pocket was discarded by narrowing.



Black time line shows the 'Never' case in which no narrowing takes place.
Test used [see Appendix E]: 3× *Astex* Mini Set on Eurymedon CA
(5 boxes from *PIES*, various narrowing periods and limits, crystal conformations only)

Figure 5.13: Comparison of search narrowing parameters, showing shorter run times using quick and maximal use, but better result reliability when retaining more boxes

the last generation of the GA, and so has the best available information with which to choose. However, a greater time reduction with lower fault rate is available using a swift narrowing to two or three boxes — '6to3' and '12to2' being good examples.

In order to maximize the probability of the correct box being selected, it is worth using a consensus of *PIES* and *PASS*: using the top 5 from each, 95% of the *Astex* Diverse Set pockets are identified. To improve the final results, the post-processing could include an additional narrowing stage, where only the best boxes are combined in the output; this

will not significantly alter the time taken, but could filter out unwanted poses. However, it may be more helpful to label each output pose with its box, so that a human assessing the results could select the binding sites of interest conveniently.

5.3 Shape Descriptors

The similarity of molecules, and particularly ligands, can be a useful source of information when docking. Often, this involves reducing the complex definition of a molecular shape (as a union of spheres) to some simpler and more convenient form. Several methods for performing this are mentioned in §2.2 (*p*.28). These techniques can be applied to both the selection of which ligands warrant consideration and where they might be placed.

A learning mechanism is required to fully benefit from such methods, so that there can be accumulated data with which to compare or align later cases. That component is discussed in §7.2 (*p*.148). I discuss two shape descriptors here: the existing *USR* method and my own *PASTRY* algorithm for comparison.

5.3.1 USR

The Ultra-fast Shape Recognition (*USR*) method [Ballester & Richards, 2007] was developed to provide an efficient means of filtering compound databases for ligands matching a given pattern. It does this by calculating the first three statistical moments for the sets of atomic distances from each of four significant points, giving a vector $U = [U_1, \ldots, U_{12}]$ of 12 real numbers altogether. The significant points are defined as:

a. the molecule's centroid

b. the atomic centre closest to the centroid

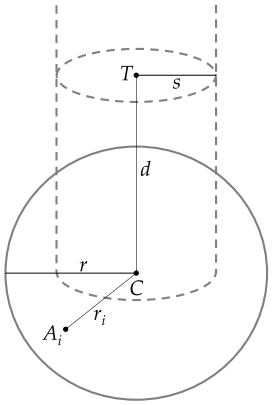
c. the atomic centre furthest from the centroid, and

d. the atomic centre furthest from *c*.

For two *USR* descriptors U_A and U_B, the similarity function is defined as:

$$\textbf{Similarity}_{\textbf{USR}}(U_A, U_B) = \frac{12}{12 + \sum_{i=1}^{12} |U_{A,i} - U_{B,i}|}$$

which gives the desired value of one when $U_A \equiv U_B$ and tends to zero as the descriptors become increasingly different.



For the outermost remaining non-core atom with the core at C of radius r:

1. Let T be the atomic centre and $d = |T - C|$ its distance from the centre.
2. Define a semi-infinite cylinder from C through T with a radius $s = d - r,\ 3 \leq s \leq r$.
3. Redefine T as the centroid of all remaining atoms within this cylinder, and then discard them.
4. Extend the yoke by 1.6Å: T becomes $T + \frac{1.6}{|T-C|}(T - C)$.
5. Add the point T and its weight $d - r$ to the list of yokes.

Figure 5.14: Construction of yokes in *PASTRY* shape descriptor

5.3.2 PASTRY

The Pocket Alignment with Spheres and Thin Radial Yokes (*PASTRY*) description is an alternative representation of ligand shapes I developed for use when pre-positioning ligands into automatically determined active sites (see §5.2.1 (*p*.95)). It is based on the observation that ligands tend to have approximately radial arm structures, converging in a central body. These arms are what determine a good docking: they need to be buried in the necessary crevice of the receptor's surface. The *PASTRY* description reduces a ligand to a core spherical volume and one or more cylindrical yokes (or weighted points) representing the protruding structures.

The core is defined as the sphere centred on the whole molecule's centroid, C, and sized as follows: for the list of all non-hydrogen atomic centres $A_1, \ldots, A_N$ ordered by decreasing distance from C, its radius r extends to the first A_i with $\frac{N-i}{N-1} - \frac{|A_i - C|}{|A_1 - C|} < -\frac{1}{8}$ or A_N if none meet this condition. The left-hand side of the inequality is a crude but efficient measure of an atom's packing based on its proximity to the centroid relative to an evenly distributed expectation (*expected* minus *actual*). A positive value indicates a tightly packed atom, rather than an extended frond, but a margin of $-\frac{1}{8}$ is used because experimentally this produced the most desirable core selections. The atoms within the core are removed from the list to leave the outer points. Groups of atoms are removed from this list and used to define the yokes of protruding structure. Figure 5.14 illustrates how this is done.

A *PASTRY* descriptor with n yokes has the form $P = \{(\boldsymbol{C}, r), (\boldsymbol{T_1}, w_1), \ldots, (\boldsymbol{T_n}, w_n)\}$,

	USR	*PASTRY*	
85 Calculations	112 ms	124 ms	10.5% longer
85^2 Comparisons	8.3 ms	12.4 ms	48.5% longer

Mean total time to process all 85 conformations.
Test used [see Appendix E]: 10× *Astex* Diverse Set on Eurymedon [A] (shape comparison only)

Table 5.2: Comparison of *USR* and *PASTRY* calculation times

and for two cases P_A and P_B the similarity function is defined as:

$$\begin{aligned}
\mathbf{Similarity_{PASTRY}}(P_A, P_B) &= \frac{\mathbf{W}(1) + \sum_{i=2}^{N} \mathbf{W}(i)\mathbf{A}(1,i)\mathbf{A}(2,i)}{N} \\
\text{where} \quad \mathbf{W}(i) &= \frac{\min(|\boldsymbol{V_A}[i]|, |\boldsymbol{V_B}[i]|)}{\max(|\boldsymbol{V_A}[i]|, |\boldsymbol{V_B}[i]|)} \\
\mathbf{A}(i,j) &= 1 - \frac{\left|\arccos(\hat{\boldsymbol{V}}_{\boldsymbol{A}}[i] \cdot \hat{\boldsymbol{V}}_{\boldsymbol{B}}[i]) - \arccos(\hat{\boldsymbol{V}}_{\boldsymbol{A}}[j] \cdot \hat{\boldsymbol{V}}_{\boldsymbol{B}}[j])\right|}{\pi}
\end{aligned}$$

with $\boldsymbol{V_X}[i] = \boldsymbol{P_{X,T_i}} - \boldsymbol{P_{X,C}}$ and $N = \min(\#P_A, \#P_B) - 1$ (i.e. the smaller number of yokes minus one). This gives the desired value of one when $P_A \equiv P_B$, and tends to zero as the lengths or angles between the two examples.

5.3.3 Comparison of Shape Descriptors

Figure 5.15 illustrates both the *USR* and *PASTRY* descriptors as they apply to the *Astex* Mini Set. The four points of *USR* are marked with coloured spheres, and the two or more points of *PASTRY* are connected by yellow lines from the core to the yokes' tips.

To assess the rapidity with which these descriptors are calculated, I added some simple clock-based timing code to the functions. Unfortunately, the (optimistically) millisecond resolution of the clock function was insufficient to register anything for these methods. Instead, I resurrected some old code for rudimentary profiling using the high-resolution performance counter functions in Windows. On the Eurymedon hardware (see §E.2 (*p*.200)), this provided a resolution of nearly 4.5μs. Table 5.2 compares the timings. Although *PASTRY* does show a slightly slower computation, it does arguably provide a greater level of information about the shape of a ligand. Whereas the *USR* points can be used to reduce a ligand to a centre and two extremities, the more sophisticated nature of *PASTRY* can accommodate any number of spokes in the shape of the molecule.

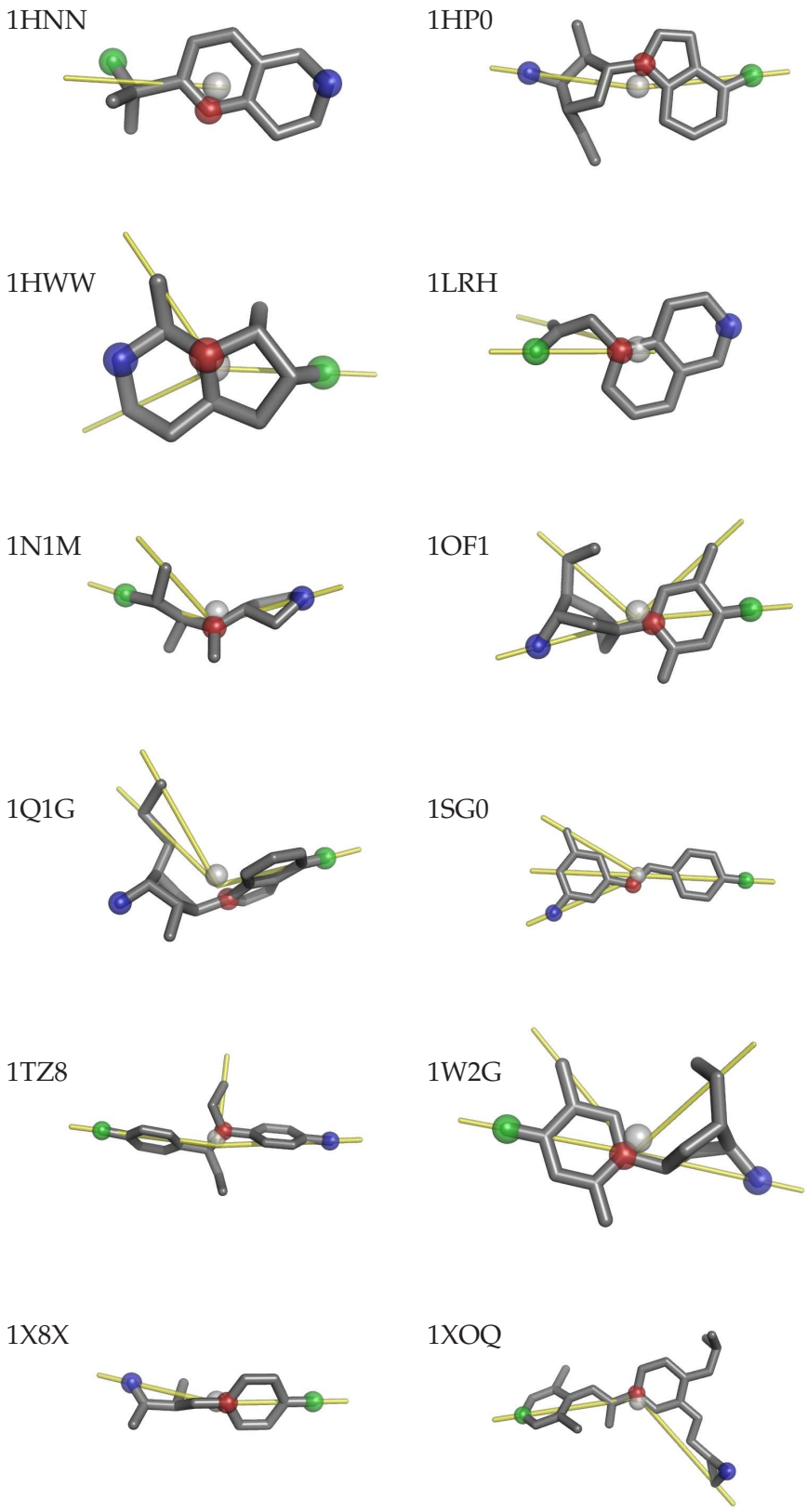
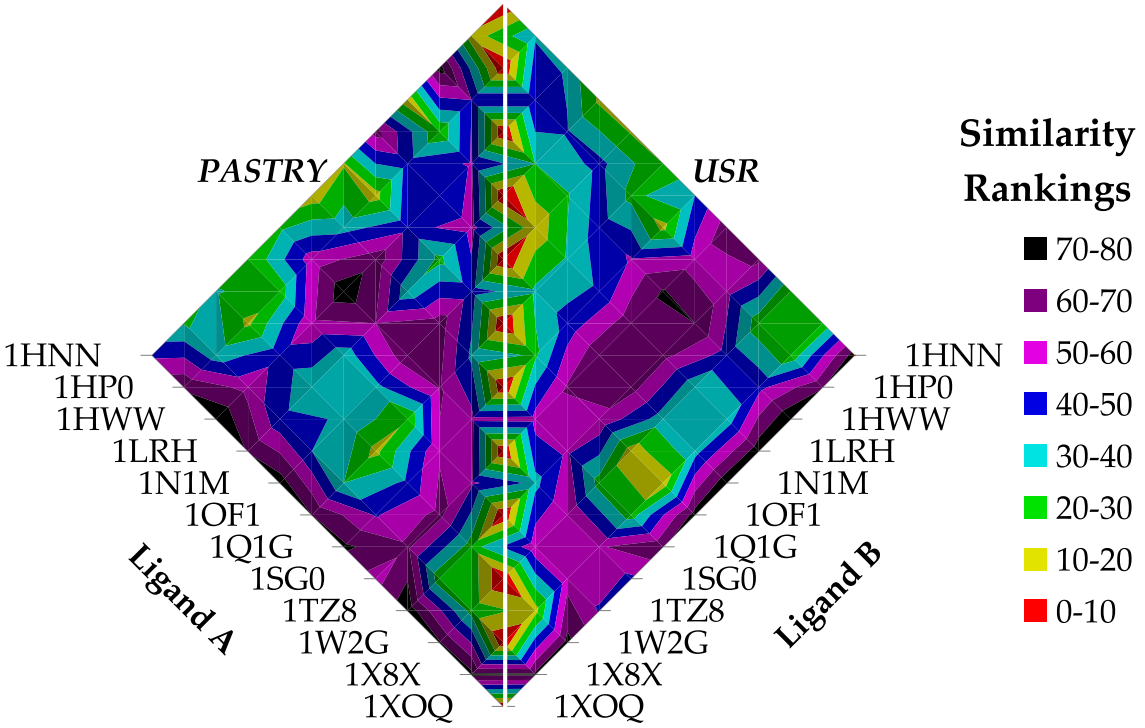


Figure 5.15: Graphical representations of *USR* and *PASTRY* shape descriptors for the ligands of the *Astex* Mini Set



Test used [see Appendix E]: 1× *Astex* Mini Set on Eurymedon A (shape comparison only)

Figure 5.16: Ranking of *Astex* Mini Set pairs by similarity using both *USR* and *PASTRY* demonstrating by symmetry of pattern the agreement between the methods' results

To evaluate the suitability of my new method, Figure 5.16 compares the similarity values produced by both methods for the members of the *Astex* Mini Set with itself. The 78 unique pairs of molecules were scored and ranked, with rank 1 being the most similar (the matching pairs), and these rankings plotted as shown in the graph. The symmetry of the pattern seen across the reflexive cases (the white dividing line through the red points) is a demonstration of the approximate agreement between *USR* and *PASTRY*. Although the similarity values themselves differ quite substantially — *USR* had a mean similarity over the non-identical pairs of 0.07, whereas *PASTRY*'s mean was 0.82 — they do at least provide a comparable ordering on ligand pairs.

The utility of a slower, more complicated shape descriptor is dubious at first consideration, but *PASTRY* was designed for another purpose than solely comparing ligands. It was intended to be used in conjunction with pocket-detection methods such as *PIES* (§5.2.1 (*p*.87)) to allow positioning of ligands in useful sites analytically, and also for pre-aligning one ligand to another to create similar poses for docking (as discussed below). The dual-use nature of the data produced helps to justify a higher time cost.

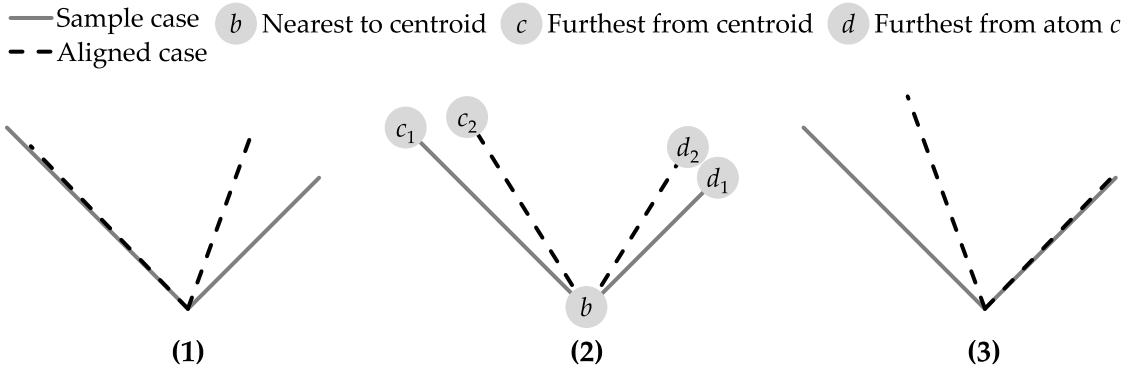


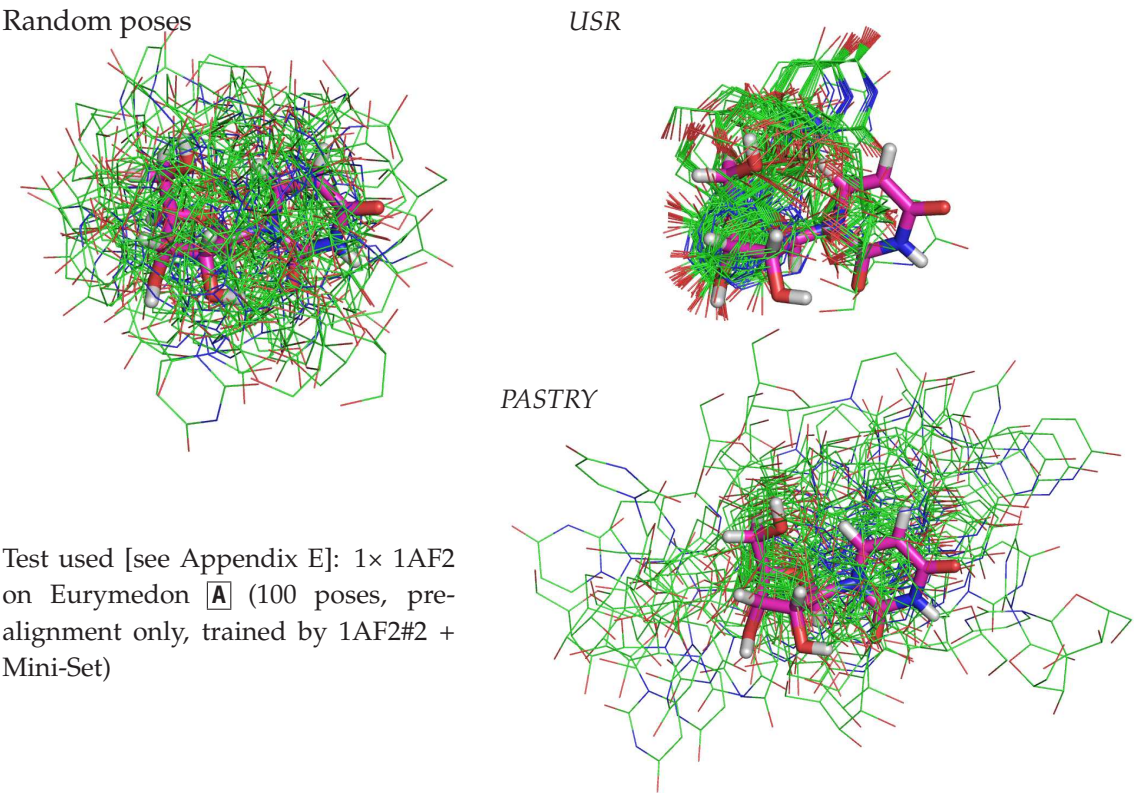
Figure 5.17: Alignment of one *USR* descriptor to another by collimation of extremities

5.3.4 Ligand Alignment Using Known Poses

An important application of shape descriptors is that of estimated pre-alignment, reusing the results of previous dockings. If a ligand with a shape descriptor $\boldsymbol{D}$ has been docked reasonably well, it would be prudent to try any future ligands with shapes similar to $\boldsymbol{D}$ in the same pose first, on the basis that they may well have a corresponding binding mode. To do this, we require the similarity functions discussed above and also a method for generating poses based on a previous arrangement.

Since the *USR* method was not originally designed for this purpose, I devised the derivative property `PropAlignUSR` to work alongside `PropShapeUSR`. The alignment property uses the same definition of the significant points as given in §5.3.1, but then uses the three atomic centre points rather than distance distributions as the data for consideration. This retains the geometry of the molecule as a fan: the angle at the centremost atom (b in the *USR* definition on page 106) between the two external atoms. Figure 5.17 illustrates this diagrammatically. If a particular example has been selected for pre-positioning a new ligand, the translation is chosen to superimpose the apexes, and then rotations are interpolated between the alignments of the two rays (cases (1) and (3) in the figure). If only one pose is required, then only the midpoint of the two rotations is used, as in case (2) in the diagram. To allow more variety in the poses generated, a random number of cases created are aligned with the two rays swapped, meaning that atom d_2 is directed to c_1 and c_2 to d_1.

When aligning *PASTRY* descriptors, a similar procedure is used, but random pairs of yokes (one from each descriptor) for alignment are chosen with probabilities



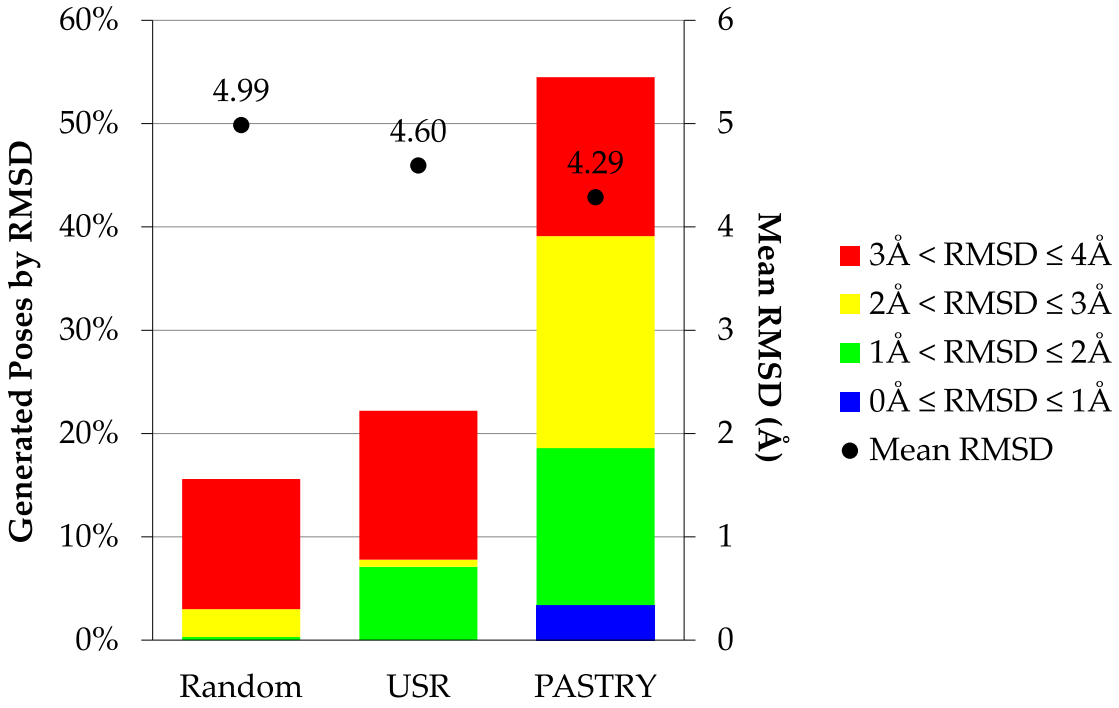
Test used [see Appendix E]: 1× 1AF2 on Eurymedon [A] (100 poses, pre-alignment only, trained by 1AF2#2 + Mini-Set)

Figure 5.18: Ensembles of 100 pre-aligned poses of 1AF2 using *PASTRY* and *USR* showing range and diversity of placements

proportional to their weights' rankings. A second pair of yokes, similarly chosen, is used to orient the ligand by random degrees around the axis defined by the first yoke. The translations are interpolated between the cores or the yokes' tips being superimposed.

When each ligand is introduced in *DOX* and a search is consequently initialized, the number of poses (population size) required is passed to the loaded `PrePositions` object. For each configured alignment property (see §F.4 (*p*.208)), some number of `Opinions` are obtained from the appropriate `Education` entry in the molecular knowledge bases (see §7.2 (*p*.148)). These prior result poses are then used to produce a list of starting points defining the poses for an initial population.

Figure 5.18 shows a set of 100 poses of the 1AF2 ligand generated by each of three alignment methods: *USR*, *PASTRY*, and uniformly random poses. The large, magenta molecule in each picture shows the correct binding pose. These poses were generated based on the education gleaned from a single docking run of the *Astex* Mini Set and the second (i.e. first non-crystal) conformation of the 1AF2 ligand. Although *USR* appears to place the ligand much closer to the correct pose, it tends not to find the right orientation



Test used [see Appendix E]: 10× 1AF2 on Eurymedon A
(100 poses, pre-alignment only, trained by 1AF2#2 + Mini Set)

Figure 5.19: Accuracy of pre-alignment methods when generating poses based on prior knowledge, showing superior proximity to crystal structure using *PASTRY*

very well. *PASTRY* does produce a much broader range of translations, but proved much more successful at rotating the ligand. Figure 5.19 compares statistics about each method's poses, and shows that *PASTRY* achieved a better mean RMSD than *USR*, with significantly more well-placed starting points. Both methods, however, outperformed a purely random selection of translations and rotations.

5.3.5 Pre-Aligned Docking Results

It is possible to use the poses generated by pre-alignment to prior examples as the starting population of a docking search. In order to assess the effect of this, I carried out a similar set of tests to those performed for pocket detection (§5.2.4 (*p*.99)). The *Astex* Mini Set was redocked using half or all the population pre-aligned using *USR* or *PASTRY* — the cases are listed in Table 5.3.

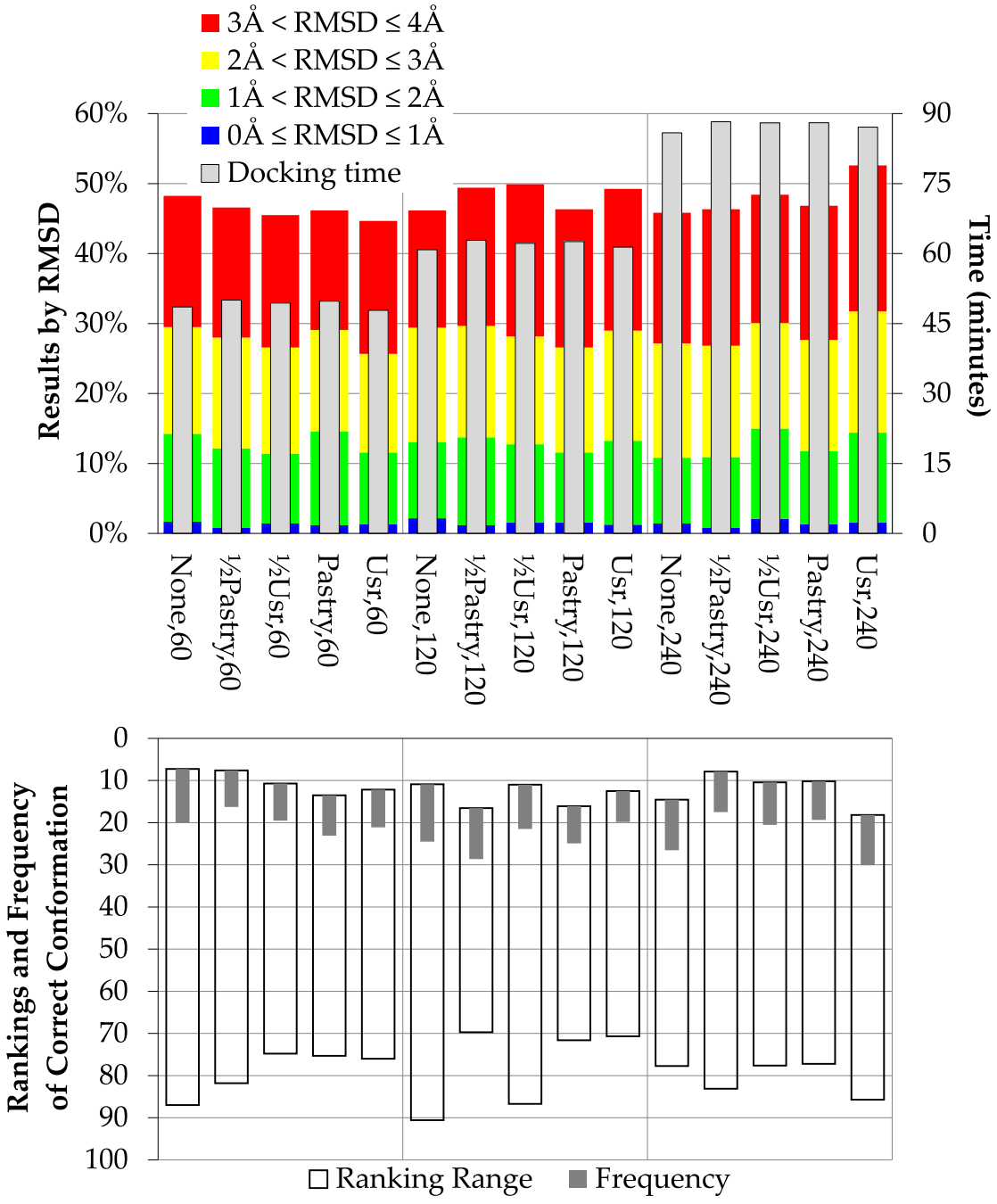
A randomly selected set of 12 other ligands from the *Astex* Diverse Set was used for training: 1GPK, 1JD0, 1MEH, 1N46, 1P62, 1R1H, 1R55, 1SJ0, 1T40, 1T46, 1YGC, 2BR1. Starting with no data at all, these were docked to each receptor of the *Astex* Mini Set and

Name	random	align
None	100%	
1/2Usr	50%	1 5 0.5 AlignUSR=*
Usr	0%	1 5 0.5 AlignUSR=*
1/2Pastry	50%	1 5 0.5 AlignPASTRY=*
Pastry	0%	1 5 0.5 AlignPASTRY=*

Table 5.3: Pre-positioning configurations for pre-alignment tests

the learnt shape suitabilities retained in the receptors' MKBs (see §7.2 (*p*.148) for details of the learning system). This preparatory stage is not counted in the timing results. Again, the searches were tried using 240, 120, and 60 generations.

As occurred when using pocket predictors in §5.2.4 (*p*.99), the effect of pre-aligning within an already constrained search space is limited. Figure 5.20 shows that the poses returned are of similar quality regardless of any estimated positioning, although the time cost is negligible. Predicted alignments do provide an alternative source of box definitions that could be used with the caching LUT search described in §5.2.5 (*p*.101). If pocket detection is already in use, the boxes automatically generated tend to be quite large, and so pre-alignment may help to place initial poses sensibly, accelerating the exploration and thus facilitating quicker search narrowing. Other applications for these kinds of methods are discussed in §8.2.1 (*p*.160).



Test used [see Appendix E]: 1× *Astex* Mini Set on Eurymedon A
(trained by 12 randomly selected cases, various alignment options)

Figure 5.20: Comparison of configurations from Table 5.3 for pre-alignment of ligands based on prior knowledge, showing similarity of docking time and results

Chapter 6

Efficient Exploration

> If Edison had a needle to find in a haystack, he would proceed at once with the diligence of the bee to examine straw after straw until he found the object of his search.
> I was a sorry witness of such doings, knowing that a little theory and calculation would have saved him ninety per cent of his labour.
>
> *Nikola Tesla*

The needle in haystack analogy is representative of the vast combinatorial space to be considered when assessing receptor-ligand interactions for docking. In practice, though, the search is not entirely blind; points can be evaluated on a continuous scale of binding affinities, not simply as right or wrong. Since a full exploration of the entire scope of poses is impractical, any available information that can direct or expedite it should be considered. This chapter discusses how to search thoroughly, but not too thoroughly, in order to find good ligand arrangements efficiently.

6.1 Local Optimization

It is plainly inappropriate to attempt a full exhaustive search of the whole pose space for a ligand. The scale of the search is prohibitive: if we consider a sphere of radius r with surface area $s = 4\pi r^2$, a spherical cap subtending an angle of $\frac{\pi}{180}$ has height $h = r\left(1 - \cos\left(2\frac{\pi}{180}\right)\right) \approx 3.8r \times 10^{-5}$ and surface area $s_c = 2\pi rh$. Then

$$\frac{s}{s_c} = \frac{4\pi r^2}{2\pi 3.8r^2 \times 10^{-5}} \approx 52525$$

orientations must be considered to ensure a maximum difference of 1°, and the range of translations is generally tens of Angstroms in each dimension, resulting in thousands of positions at even a coarse resolution. A 1° rotation is enough to move the outermost atoms of a mid-sized ligand by 0.1Å, a distance which can produce significant variation in close-range interactions. Of course, this also means that a purely stochastic method has

```
Initialize GA population
Evaluate (score) initial population
For G := 1 to NGens do
  Begin
  //Main GA evolution loop
  Generate offspring from current population
  Evaluate population
  If G mod OptFreq = 0 then //If an appropriate generation...
    Call OptimizePopulation //...perform local optimization stage
  End
If OptFinal then
  Call OptimizePopulation //Optimize final generation if required
```

```
Procedure OptimizePopulation
  Begin
  For each genome P in current population do
    Begin
    Create simplex for the pose of P
    Optimize simplex
    Restore optimized pose and score to P
    End
  End
```

Listing 6.1: Lamarckian Genetic Algorithm outline, showing local optimization stage

a vast space to cover, and for this reason a hybrid method of large-scale and small-scale searches is perhaps useful.

A good example of such a design is the Lamarckian Genetic Algorithm [Hart, 1994; Hart *et al.*, 1994], as incorporated into the *AutoDock* program [Morris *et al.*, 1998] amongst others. This is named after Jean Baptiste de Lamarck (1744–1829), who suggested that the effects of both nature and nurture can be inherited by offspring [Lamarck, 1809]. Although, as a biological theory, this has been dismissed, implementations of evolutionary search algorithms are free to incorporate such phenomena as desired. After the offspring have been created from a population, a small-scale exploration of the scoring function is used to find a locally-optimal pose for each. These refined arrangements are then returned to the GA population, before the next generation cycle begins, as outlined in Listing 6.1.

One algorithm for such a local search is the Nelder-Mead Simplex Method [Nelder & Mead, 1965], which finds an optimal interpolation between several sample vertices — generally $n + 1$ for an n-dimensional space. Even with a small search range of ± 3Å in each translation (approximately equal to the diameter of most ligand atoms) and around $\pm 10°$ for each rotation, this method is much slower than one generation of a GA.

```
//For an N-dimensional search space on function F:
Create N+1 vertices, V[0..N]
Initialize V with diverse values in the space
While termination conditions are not met do
  Begin
  Order V by their values in F, best first
  //Reflect worst point through centroid of others:
  Calculate C, the centroid of vertices V[0..N-1]
  Let vertex R := C + 2*(C - V[N])
  If R is better than V[0] then
    Begin
    //New best case, so pursue it
    Let vertex R' := C + 3*(C - V[N])
    Replace V[N] with the better of R and R'
    End
  Else If R is worse than V[N-1] then
    Begin
    //Very bad case, so try a constrained leap...
    Let R := C + 0.5*(C - V[N])
    If R is better than V[N] then
      Replace V[N] with R
    Else
      //...if that didn't work, contract all points towards the best
      For I := 1 to N do
        Replace V[I] with V[I] + (V[0] - V[I])/2
    End
  Else
    //Must be a non-extreme case, so replace the worst
    Replace V[N] with R
  End
Return V[0]
```

Listing 6.2: Nelder-Mead simplex optimization algorithm outline

However, if an approximately correct pose is found, the optimizer will be much more efficient at settling it into the bound state, since it is a deterministic, analytical method. The algorithm is described in Listing 6.2.

The time spent on local optimization of poses is limited to a maximum number of cycles, to avoid seeking gratuitous precision if a pose is not near any minima in the scoring function. Using the simplex in this manner on every generation incurs a substantial time cost: Figure 6.1 illustrates the dramatic rise from minutes to many hours on the second column (note the truncated time bar in the second column). Alternatively, only the final set of poses could be refined, ensuring that if the GA gets close to a good pose it will be valued as such in the output. This option, the final column, shows quite respectable results for a much more acceptable time cost. Indeed, this serves as a validation of the use of optimization, demonstrating the improvement to result quality

possible with the method.

A compromise between time and efficacy might be to refine populations periodically, allowing a longer progression by the GA mutations and crossovers before stopping to survey the localities chosen. This could be a more sensible approach since it should ensure that the populations change significantly between optimizations, avoiding wasted time searching very similar regions and finding the same pose. To test this, I added the facility to specify the frequency of local searches to *DOX* and tried several different values. These results are also shown in Figure 6.1.

Firstly, it is interesting to note that the results offered by more frequent optimization are almost equivalent in some cases (40, 80, and 120), and substantially worse in others (160 and 200). The cases where the optimization period is not a factor of the total number of generations did not optimize the final population, and this would explain the substantial drop in result quality. For the best effect from the overall search, optimization must be applied to the final result set, regardless of whether it has been used at any intermediate stage. Otherwise, the GA will apparently rock the carefully-settled poses out of place. However, it would also seem that optimizing intermediate generations is largely a waste of time. Despite each diversion to refine a generation taking (in this case) more than three minutes, there is little improvement to justify it.

6.1.1 Local Searches Versus GA Generations

Since it is clear that the local search is crucial to finding the best pose from an initial exploration by the GA, perhaps its periodic application would make fewer generations necessary overall. Figure 6.2 shows the time and results from various combinations of optimization periods and GA generations. All the parameters used were chosen so that the final generation would be optimized. Varying the number of generations produces the expected linear effect on docking time, with a gradient proportional to the optimization period. Surprisingly, though, the twelve different configurations had effectively equivalent result quality. The number of good poses (within 1 and 2Å RMSD) varied very little, and the ranking of the correct ligand conformation declined with more generations when optimizing frequently. From these results, one can conclude that, whilst a final local optimization is necessary, repeatedly optimizing the intermediate populations is of no real benefit.

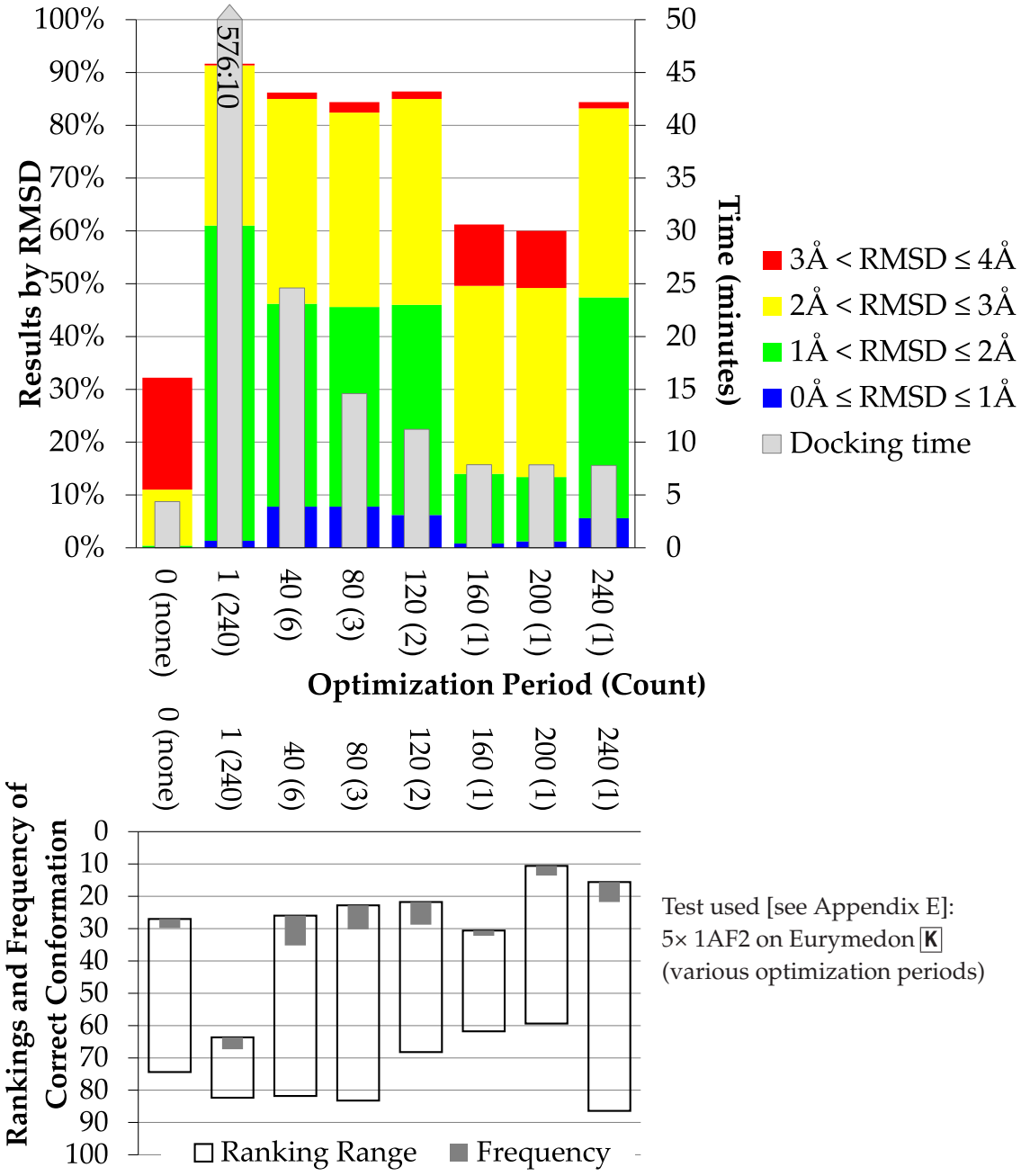
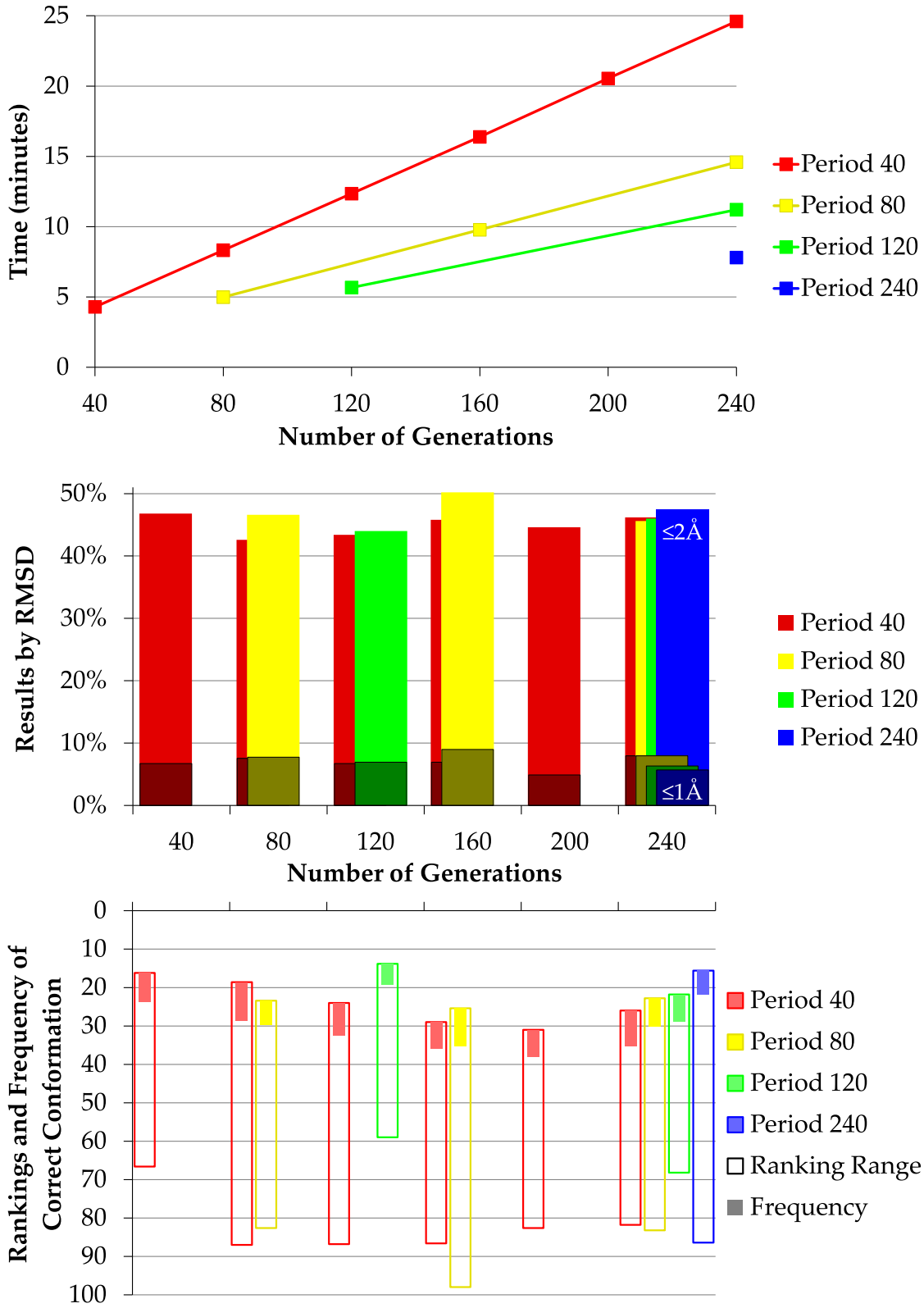


Figure 6.1: Comparison of local optimization periods in a Lamarckian GA, showing the balance between time cost and improved results, and particularly the demerit of not optimizing the final population

6.2 Look-Up Table Interpolation

The values in a look-up table (LUT) may only be calculated at discrete points in the space represented: they cannot be a full record of a function. However, the scoring functions used for docking are inherently continuous. Since ancient times, interpolation has been used to ameliorate this discrepancy, estimating a requested data point from proportions of the nearest available known values [Meijering, 2002]. For an N-dimensional LUT,



Test used [see Appendix E]: 5× 1AF2 on Eurymedon K
(various generation counts and optimization periods)

Figure 6.2: Assessment of frequent local optimization versus more GA generations, showing better selectivity and time of longer searches with fewer optimizations

```
Given requested coordinates X[1..N] on a function F:
Calculate array O[1..N], the floors of X[1..N]
Calculate array D[1..N], equal to abs(X[1..N] - O[1..N])
Create array I[1..N], where I[i] = i
Order I by decreasing values of D[I[i]]
Let array V[1..N] be zeroed
Let R := F(V)
For each i in I do
  Begin
  Let LV := V
  Increment V[i]
  Let R := R + (F(V)-F(LV))*D[i]
  End
Return R
```

Listing 6.3: Piecewise linear interpolation algorithm

assuming a unit resolution, this could be calculated as a piecewise linear interpolation, as described in Listing 6.3. One could also interpolate gradients to calculate the value requested, which would obtain a smoother function than this first-order interpolation, but it would be slower to use and might mask narrow contours in the function.

The practical benefit of interpolating a scoring function is that regions of steep gradient will be better searched by softening the edges around isolated extreme values. Without this, a pose close to a good position might still be scored badly if the LUT rounding falls away towards a nearby bad clash. In addition, two points within a range determined by the LUT's resolution will be evaluated as having the same function value. This could stymie any use of gradients by a search method, masking even a steep region of the function.

Although the *DOX* system used piecewise linear interpolation as part of the LUT implementation, I adapted it to allow this to be selectively enabled. When disallowed, the LUTs simply return the closest available value to the coordinates requested. The first two columns of Figure 6.3 displays the effect of the interpolating LUTs on a docking search by comparing editions `i` and `P`, excluding and including the technique respectively. The result accuracy is worse in the `i` edition — the lack of any interpolation prevents the scoring function from distinguishing between similar poses, and so the proportion of results very close (within 1Å RMSD) to the correct pose is reduced. The selection of the correct conformation is also somewhat worse, because similar molecular shapes will evaluate identically on the discrete scoring grid.

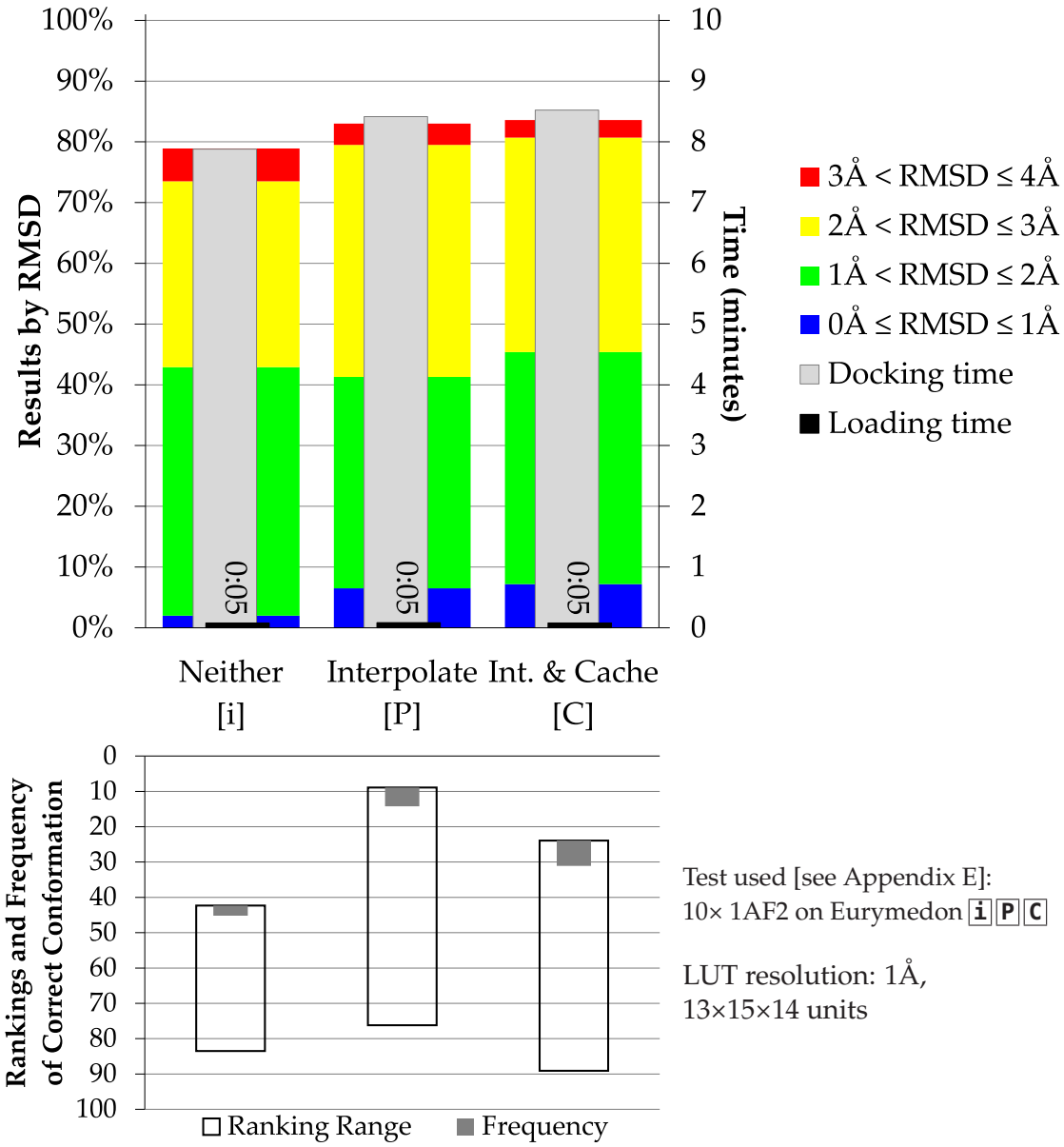


Figure 6.3: Effect of LUT stratagems on docking time and result accuracy, showing benefit of interpolation and little effect of caching

Understandably, the additional arithmetical operations required to interpolate return values did add a time cost of around 7% to the docking process. This will partly be due to the local optimization (§6.1 (*p*.117)) stages taking longer to converge in a smoother function space. Reducing storage requirements with the use of interpolation permits a sensible balance of precision in the LUT generation with the benefits of pre-calculation.

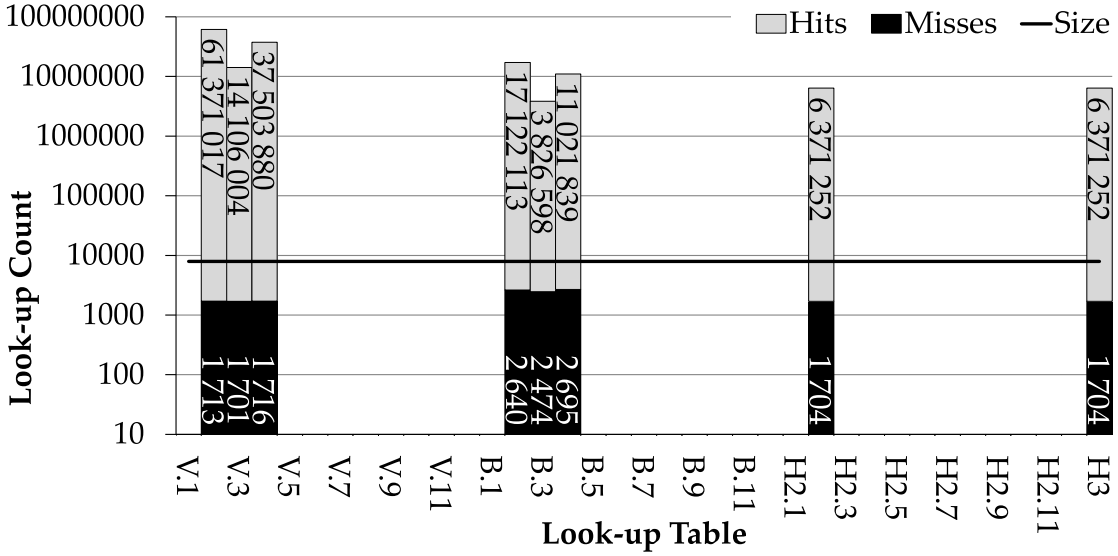
6.3 Lazy Evaluation: Caching Look-Up Tables

The use of lazy evaluation in programming allows complex expressions be computed only when their result becomes necessary for the task at hand. Detailed calculations that later prove irrelevant are avoided; for example, the product $z\mathbf{F}(\cdots)$ will be reduced immediately to zero if z is zero, without reference to even the definition of $\mathbf{F}$. A more familiar example in computer programming is that of short-circuited Boolean operators, where the evaluation of $p \wedge q$ only evaluates q if p is true.

This principle is applicable to docking procedures also. Empirical scoring functions (see Equation 3.1 (*p*.60)) are often quite complex to calculate, and generating look-up tables is a costly process in terms of both time and storage. Although these costs are often acceptable in professional situations, they are still substantial when considered for a high-throughput application. One way to reduce these in a reasonably transparent way and without affecting results is to use lazy evaluation for the LUTs, rather than compiling them in full before docking commences.

To eliminate the overhead time cost of creating a LUT large enough to accommodate any prospective ligand position, I introduced caching LUTs to *DOX*. For the purposes of pre-calculation, these are simply filled with null values (defined according to the individual interaction functions). Then, whenever the table is accessed, the element looked-up is checked against this special value, and replaced with the properly-calculated figure if necessary. This way, rather than evaluating the scoring function exactly once per parameter, it is *at most* once depending on whether the parameter is ever considered.

Implementing this in *DOX* was reasonably straightforward, with a subclass of the `RealWorldArray` data storage class. I made the new class output statistics about the number of hits and misses in each LUT at the program's termination. Figure 6.4 shows what was found: note the logarithmic scale of the vertical axis. The misses — requests for elements not yet calculated — were invariably far fewer than half the size of the table. The number of hits — when results were reused — tended to be exponentially greater than the number of misses. This supports the use of pre-calculation, certainly, but perhaps does not especially warrant the cached implementation. However, if we consider the number of tables that were used (in the *XScore* case), only 8 of 37 were ever



Test used [see Appendix E]: 10× 1AF2 on Eurymedon C

Figure 6.4: Caching statistics showing coverage of LUTs when docking with *XScore*

accessed at all. This is because of the need to allow for various relatively unusual atom types being present in either molecule, but since these occurrences are rare their tables are almost never used. In fact, overall, only 5.6% of all LUT elements were ever calculated. Admittedly the 1AF2 case is a fairly small example, but the LUTs must be allocated with a large margin to allow for all reasonable translations, and it is here that the caching model could provide much efficiency.

The fact that the same calculations are performed, merely at different times, in the caching and non-caching designs means that the results should be unchanged between the two (except for some inevitable variation due to the GA). Figure 6.3 shows that the quality of results is not significantly affected by caching. The fully-calculated version P was run with pre-calculated LUTs, whereas the caching edition C started with no data whatsoever. Initializing an empty LUT took about the same time as loading complete tables, and reading saved data occurs in the caching system as well on all but the first execution with each receptor, so there is no great effect on that preparatory stage.

It would seem that the time saving is primarily in the avoidance of a full pre-calculation. The docking time shown in Figure 6.3 includes the normal docking procedure plus the calculation of 5.6% of the possible LUT data; the overall search time is slightly reduced nonetheless. Calculating data as and when it is required does make effective use of any hardware memory optimizations available in the computer by

		No caching [P]		Caching [C]	
Scaling	Resolution	Loading	Docking	Loading	Docking
2	1.0Å	14.10	595.47	4.70	605.73
3	1.0Å	25.59	612.63	4.84	653.74
2	0.5Å	49.31	606.88	5.78	661.48

All times in seconds. Test used [see Appendix E]: 1× 1AF2 on Eurymedon [C] (various LUT sizes)

Table 6.1: Effect of LUT parameters on calculation and caching times

avoiding a later retrieval step. Given the much greater proportion of hits in the tables, the time should fall further on subsequent dockings to the same receptor.

From the hit/miss statistics, one can estimate that a full LUT calculation would take up to $\frac{1}{5.6\%} \approx 18$ times the cost of LUT misses (although this ignores the likely benefits of compiler loop optimizations and the better use of memory). In fact, a full, uncached pre-calculation took less than 15 seconds using [P], so the cost of cache misses is disproportionately high.

A much larger LUT, either for a larger molecule or using a finer resolution may show a greater divergence when caching is employed. Table 6.1 lists the timings for three different LUT parameter combinations. Reducing the resolution to 0.5Å or increasing the LUT range to triple the reference ligand's extents (rather than double) makes a very small change to the caching initialization time, but does dramatically slow the full pre-calculation. However, the increased cost of cache misses adds at least as much to the docking time as in the pre-calculation edition's loading procedure.

A full LUT implementation is sometimes preferable if well compiled and optimized. The use of caching, whilst theoretically preferable, must be implemented with consideration of the executing hardware's own caching strategies to work well. However, the facility to perform calculations flexibly and on demand will still be beneficial in certain situations: for example, if the region of interest for searching is not predetermined (as in §5.2.5 (*p*.101)).

6.4 Early Rejection

For a processor-intensive task such as docking, it is worth giving some consideration to the information sought from the process — a scoring of the suitability of a ligand

```
Create S and T, both reals equal to zero
For each atom A in ligand do
  For I := 1 to N_int do
    Begin
    Look up T from receptor table I for the atom type of A
    Add T to S
    //The inserted exit point...
    If abs(S) > abs(Threshold) Then Return S
    End
Return S
```

Listing 6.4: Scoring-based early rejection algorithm outline

for binding to a receptor, and some poses that demonstrate the fit. An important observation that should be exploited is the relative vagueness of this requirement. These scores (particularly when using empirical functions) are inevitably inaccurate estimates of physical properties, and so should not be interpreted as having a high degree of precision. The qualitative meaning of a score — the interpretation as being plausibly-docked or not — is often more important than its quantitative value.

6.4.1 Scoring-Based Early Rejection

The values returned by a scoring function fall generally on a (possibly unbounded) scale of real numbers. These represent assessments of a ligand's placement from various degrees of abysmal through to a relatively narrow band of good scores. One can select a threshold on this scale to cap a bottomless pit of values that represent indubitably unsuitable positions — if a ligand is buried or clashes significantly, it does not matter *how* bad its score is, merely that it *is* bad. Sometimes, then, a less accurate value might be equally serviceable and quicker to obtain when this value is needed only to demonstrate the impossibility of a pose [Skone *et al.*, 2009].

It is also worth remembering that suitable binding states form a very small subset of the search space, and so it is a minority of calculations that need a complete and detailed evaluation. Consequently, much time can be wasted on the execution of scoring functions for arrangements that will, upon human assessment or aggregation in a best-n list, be promptly discarded again. It would be efficient to predict whether a score will fall into the pit, and produce an estimated indicative result instead of a fully-calculated value.

In order to avoid spending time and resources pursuing work that will lead nowhere useful, an early exit point is introduced into the computation loops. Specifically, after

the accumulation of each ligand atom's contribution, the running total is used to decide whether it is still possible to reach an appealing final score. This is after each iteration of the outermost summation in the generalized scoring function of Equation 3.6 (*p*.65):

$$\mathbf{Score}(R, L) = \sum_{l \in L_{\text{atoms}}} \sum_{i=1}^{N_{\text{int}}} \mathbf{F}^*{}_{R,i,l_{type}}(l_x, l_y, l_z)$$

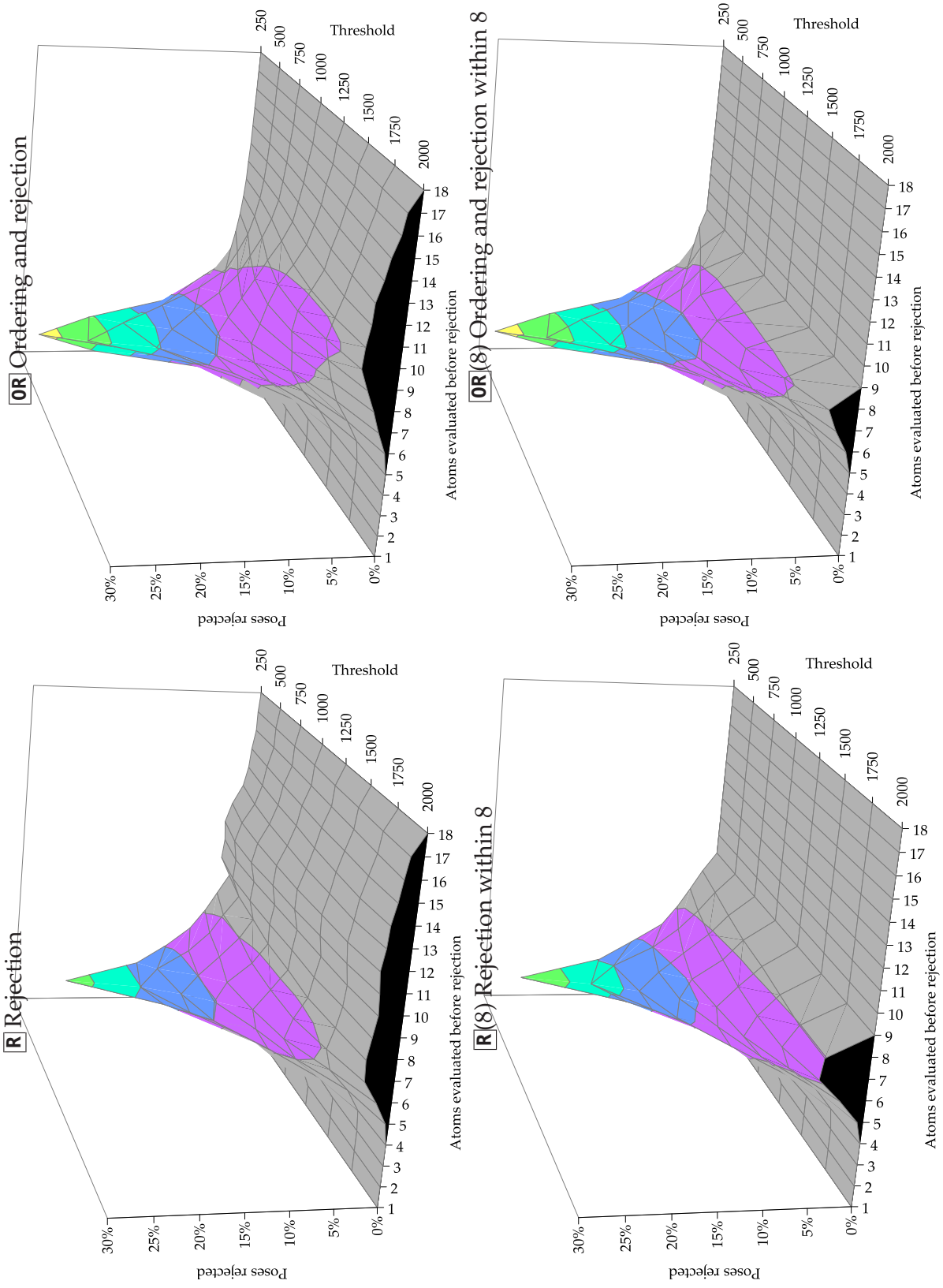
If this incomplete sum has passed a certain threshold which indicates that it is beyond the realm of suitability, then the score is approximated to the current total and returned immediately.

The prospect of abandoning outright some cases without fully evaluating them might seem to jeopardize good results, but provided a sensible rejection threshold is used the method can be justified. Almost all scoring functions include a simple potential term — typically based on Van der Waals radii — and these have by far their largest values at short, clashing distances, as shown in Figure 3.7 (*p*.61). Hence, an unacceptably buried ligand will aggregate quickly a very poor total whose magnitude dwarfs any favourable interactions that might still occur.

Atom Prioritization

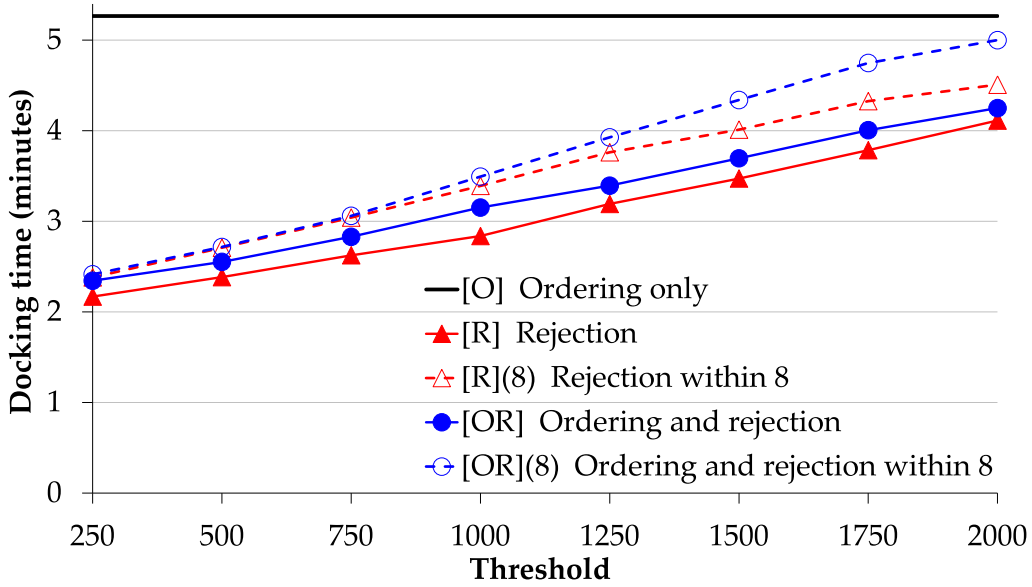
A natural extension to this design is to look for the worst atomic contributions first in the hope that the rejection threshold will be passed as soon as possible. Referring back to the general scoring function above, the commutativity of summation allows us to accumulate the atoms' terms in any order. It would make sense to assess them arranged by the relative likelihood of clashing with a receptor's surface. A simple implementation could use the distance of each atom from the ligand's centroid, ranking the farthest first. It might seem more effective to sort the ligand's atoms by their distance from the receptor's centroid, thus ensuring that those closest to the interaction surface would be seen first, whereas sorting relative to the ligand merely delays assessment of the atoms that are least likely to clash. This more comprehensive approach has one major disadvantage, however: it would necessitate a sorting operation for every pose, rather than every ligand, which is unacceptably expensive.

For efficiency, scoring calculations that are not rejected within the first few atoms might be allowed to complete in full without further interference. This would eliminate



Test used [see Appendix E]: 10× 1AF2 on Eurymedon R OR
(various scoring thresholds, all or 8 atoms)

Figure 6.5: Scoring threshold rejection behaviour, showing the threshold's effect on the number of poses discarded after each ligand atom, the smoothing effect of prioritization, and the truncating effect of limited rejection

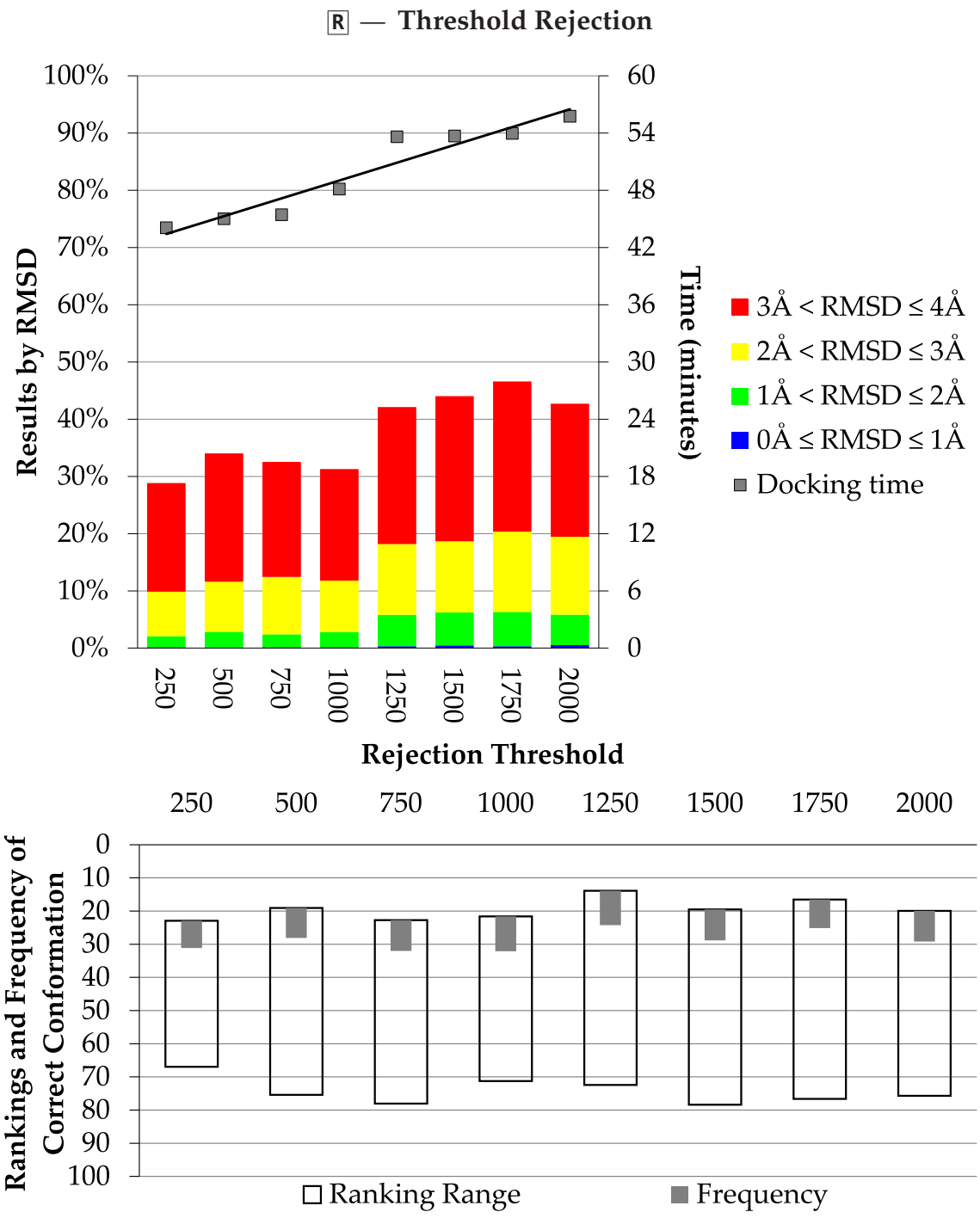


Test used [see Appendix E]: 10× 1AF2 on Eurymedon 0 R OR
(various scoring thresholds, all or 8 atoms)

Figure 6.6: The linearly proportional effect of scoring thresholds on docking time using prioritization and/or rejection

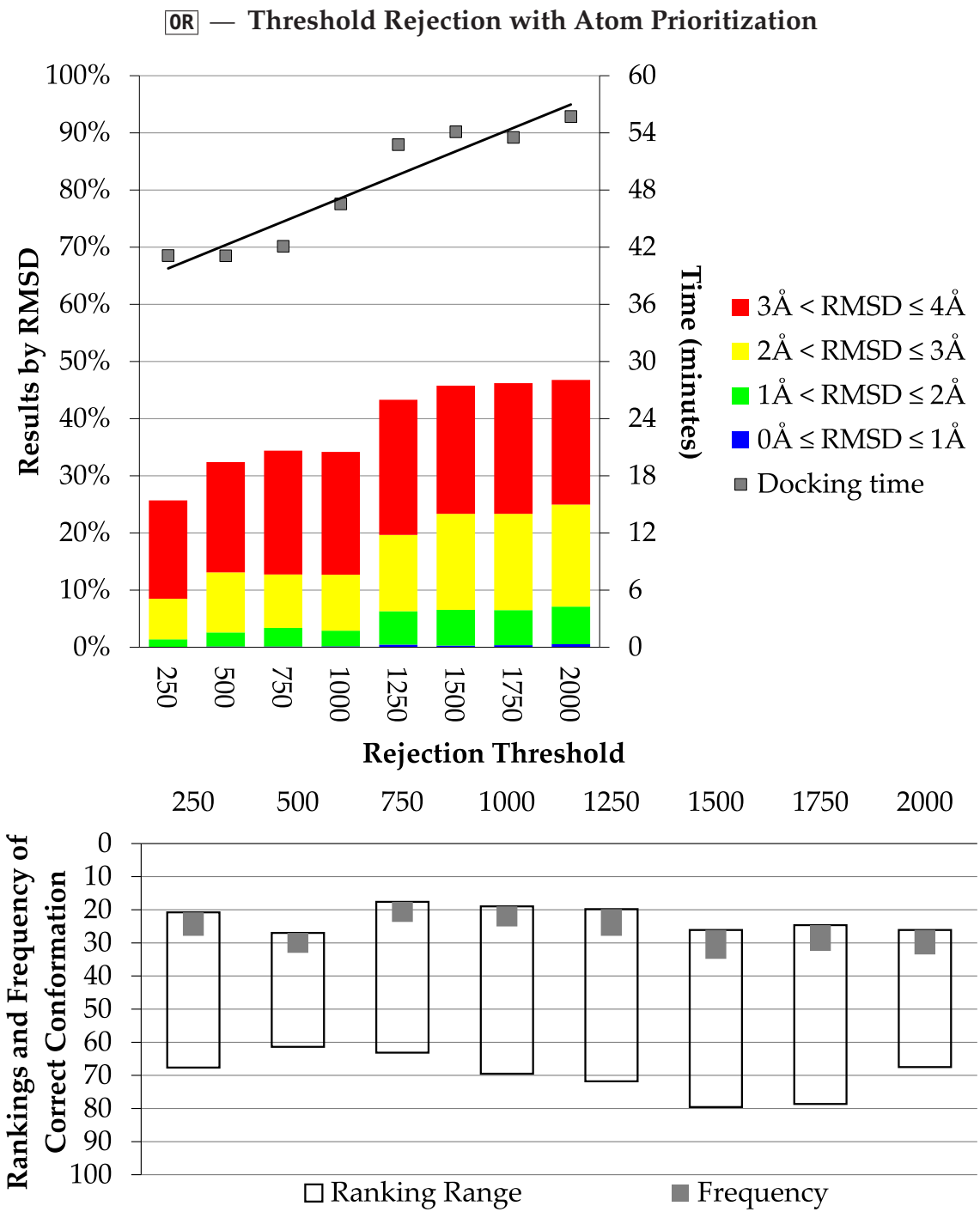
the test required after each atom and might shrink the already small cost of watching for early rejection opportunities. It might, of course, also be insignificant.

To demonstrate the behaviour of these stratagems, I ran the *DOX* program and recorded how many atoms were considered before each score calculation was abandoned at several threshold values. The four cases — unordered R and ordered OR, with and without a limiting atom count — are shown in Figure 6.5. The larger, and thus more relaxed, thresholds rejected fewer poses. Prioritizing the atoms, in this 1AF2 case, increased the number of poses abandoned, as well as smoothing the graph slightly because the likelihood of each atom contributing the fatal term decreases with its index. With the smaller thresholds, around two-thirds of poses generated in the search were being abandoned within three atoms. These translate into roughly proportional time reductions, as shown by Figure 6.6 alongside the constant time for ordering without rejection using edition 0. A clear trend towards longer run times can be seen as the threshold is relaxed, as would be expected from the reduced proportion of cases that will be abandoned. The limited ('(8)') methods are slower than their unlimited counterparts, and the unordered method is slightly faster by what appears to be a constant factor in the unlimited case.



Test used [see Appendix E]: 5× *Astex* Mini Set on Eurymedon R (various scoring thresholds)

Figure 6.7: Effect of early rejection using scoring thresholds on docking time and results, showing improved accuracy and longer time at larger thresholds



Test used [see Appendix E]: 5× *Astex* Mini Set on Eurymedon OR (various scoring thresholds)

Figure 6.8: Effect of early rejection using scoring thresholds and atom prioritization on docking time and results, similar to the unordered examples of Figure 6.7

Astex Mini Set Assessment

To assess whether or not these configurations still produce acceptable results, I redocked the *Astex* Mini Set (see §E.1.4 (*p*.200)) at several rejection thresholds to allow for variation in molecular shapes. Figure 6.7 shows the timings and results for the early rejection method, and Figure 6.8 shows the same statistics when atom prioritization is employed.

The previous observation that docking time is correlated with rejection threshold still holds true here. The accuracy of the results also improves with more relaxed thresholds, particularly at the transition from a cut-off of 1000 to 1250. With close inspection of the algorithm's behaviour, this transpired to be the point at which no evaluations were abandoned after a single atom for any of the 12 test cases. A marked step in run time also occurred at this point, which corresponds to the vastly increased number — hundreds of thousands — of evaluations that are then at least twice as long.

6.4.2 Pose-Based Early Rejection

Any search method based on random variation, such as a GA, suffers from a risk that its candidate results will develop a homogeneity and cease to explore some regions of the search space. If multiple minima/maxima (depending on the goal) exist, this can mean that one or more are invisible to the search. One way of avoiding this, and making more efficient use of time, is to define some measure of similarity between search candidates, and maintain a certain minimum. Hence, if two cases under consideration become sufficiently similar, they are merged into the first, and the second is replaced with a completely new case. The retention of one of the pair ensures that this will not eliminate an easily-found optimum. This should avoid duplicate scoring function evaluations, as well as maintaining a broader exploration.

DOX's search method was adapted to maintain diversity using a very simple method, outlined in Listing 6.5. The similarity measure was defined as the Manhattan distance between translations and absolute value of the cosine of the angle between rotation quaternions: for poses P_1, P_2 with $P_i = (\boldsymbol{T_i}, \boldsymbol{R_i})$ for vector $\boldsymbol{T_i}$ and quaternion $\boldsymbol{R_i}$,

$$\mathbf{Similarity_{Pose}}(P_1, P_2) = |\boldsymbol{T_{1}}_x - \boldsymbol{T_{2}}_x| \ + \ |\boldsymbol{T_{1}}_y - \boldsymbol{T_{2}}_y| \ + \ |\boldsymbol{T_{1}}_z - \boldsymbol{T_{2}}_z| \ + \ |\boldsymbol{R_1} \cdot \boldsymbol{R_2}|$$

```
For each generation do
  Begin
  Get current population P
  Sort P by scores
  For n := 1 to size of P - 1 do
    If comparison of P[n] and P[n+1] < MinDifference then
      Replace P[n] with a new, random pose
  Perform normal search progression step
  End
```

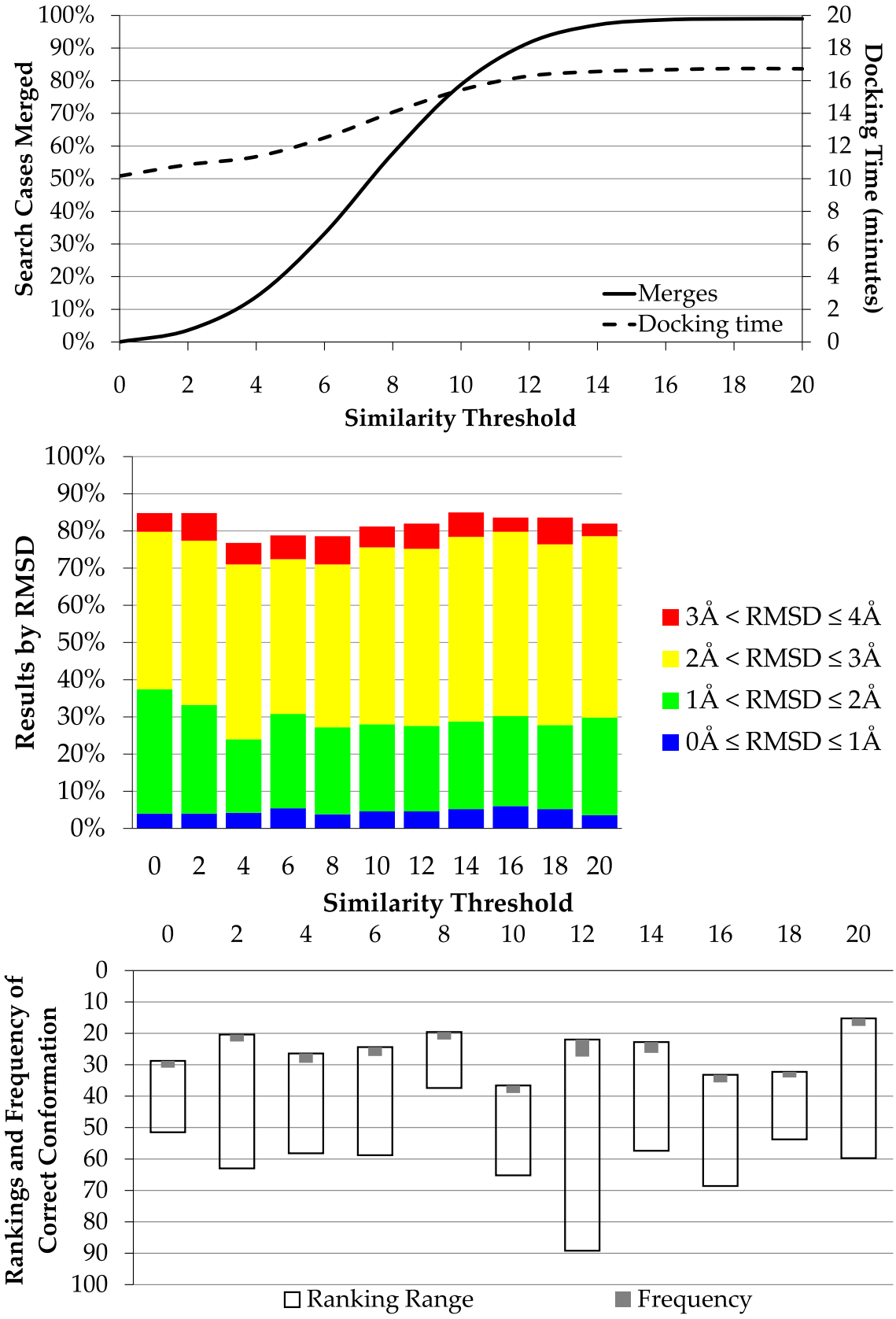
Listing 6.5: Pose merging algorithm outline

At the start of each generation, the population has any easily-discovered converging members merged before the usual genetic operators are applied. The use of sorting and a single comparison scan allows the checking to be done in $\mathcal{O}(N \log N)$ time (N being the population size), rather than the exhaustive $\mathcal{O}(N^2)$ alternative, by assuming that poses sufficiently similar to be merged will have similar scores. Admittedly, this is not especially reliable, but the efficiency consideration is overriding.

For the 1AF2 test case used here (see §E.1 (*p*.197)), 53 conformations each had a population of 96 cases evolved for 240 generations; thus 1 221 120 cases were evaluated altogether (ignoring the simplex optimizer). Figure 6.9 shows the proportion of these cases that were deemed too similar to another and merged at a wide range of threshold measures. There is a limiting point where only one member of each generation remains. In this example, it occurs when 95 members out of each population are discarded (possibly by similarity with a prior randomly-generated replacement), giving a maximum of 53×95×240 = 1 208 400 merges, or 99%. Thresholds above 18 tended towards this limit, corresponding to a maximum variation permitted in the search space.

The rotations cannot differ by more than 1, the maximum absolute value of the cosine function. The translations are constrained to a box of dimensions (5.6, 6.6, 6.5) and thus a maximum variation of 18.7, however this range is determined by the active site in question. From these calculations, an upper bound of approximately 20 can be placed on the similarity threshold for this example.

The effect of the threshold on docking time was proportional to the number of cases merged. Overall, the application of this technique did increase docking time by 21% in comparison with the equivalent [P] edition (see Figure 6.3 (*p*.124)) when using a threshold



Test used [see Appendix E]: 3× 1AF2 on Eurymedon M (various similarity thresholds)

Figure 6.9: Effect of increasing pose merging similarity thresholds in a search method, with the resulting increase to docking time but limited improvement in results

of zero to prevent merges occurring. This is variable depending on the population size, because of the sorting stage.

Considering the result quality, however, it appears that the overall benefit of merging similar poses is somewhat limited. The proportion of results within 2Å is slightly reduced, but the very good poses within 1Å are retained. The selection of the correct conformation was mostly unaffected by the method, but there is a very slight broadening trend in its highest rankings. Increasing the rate of case merging appears to allow the correct conformation to peak slightly higher in the results, but also to drop lower. This suggests that the introduction of random poses can destabilize the search slightly, which is unlikely to be desirable behaviour. Certainly, thresholds above 6, when the proportion of cases being merged is rising dramatically, are not especially helpful and introduce an unjustifiable time cost.

6.4.3 Quota-Based Early Rejection

Having established in §6.4.1 (*p*.128) a means of using quickly-calculated approximate scores to abbreviate steps in a search method, the overall exploration process may also be improved. The same principle of stopping as soon as an unfavourable outcome is predictable can be reapplied here [Skone *et al.*, 2009].

The issue of pose selection has been well-studied already under the umbrella of search algorithms (see §2.3.2 (*p*.32)). However, the selection of a useful termination condition (see Figure 3.8 (*p*.63)) also warrants consideration. In general, this condition is usually the completion of a predefined number of steps or the emergence of some quality or consistency in the results. This kind of stopping point makes the search a probabilistically complete algorithm: unlike a complete algorithm which finds the best result on every occasion in a fixed time, this will usually find the best result given long enough, and the probability increases if more iterations are permitted.

The results will be aggregated with those from every other conformation being docked. If a quota is imposed on the final number of results that may be returned from all cases, then it is possible that most of the results from an individual search will fall off the bottom of the list and be discarded before the final set is recorded. Since the set of results will almost always need further detailed analysis, the quantity of output data is often

Example scored pose lists after ligand conformations $L_1, \ldots, L_4$

L_1		L_2		L_3		L_4	
Pose1a	9	Pose1a	9	Pose1a	9	Pose4a	10
Pose1b	8	Pose1b	8	Pose1b	8	Pose1a	9
Pose1c	4	Pose2a	7	Pose2a	7	Pose1b	8
Pose1d	1	Pose2b	6	Pose2b	6	Pose2a	7
		Pose1c	4	Pose1c	4	Pose2b	6
		Pose2c	**2**	Pose2c	**2**	Pose1c	**4**

Figure 6.10: Illustration of result collection during quota-based early rejection and the definition of the worst-case score

limited to a certain number of docked poses, and so this concept of a quota is justifiable. Indeed, most docking tools expect a limit on the number of results to be given as one of their inputs. Rather than doggedly pursuing a docking of a relatively unsuitable ligand conformation, only to produce results that will ultimately be omitted from any output (or else ignored in later analysis), the termination condition could take into account the results already present in the final list and give up early on unpromising examples. By starting with a widely varying set of cases and progressively reducing that variation, this is an approach with superficial similarities to simulated annealing.

One simple way to do this would be to track the worst score present in the final result list (after its quota has been filled), and abandon searches that have not scored at least as well after some number of cycles. This uses a failure to find any slightly interesting poses after a shorter search as the basis for deducing a dearth of good poses in the whole space. That deduction, although not reliable, may be rational if made only when the search has been afforded opportunity to make a reasonable exploration of the space. It must be assumed that the scores found in each cycle tend to improve over time. This requirement should be acceptable, though, since even a completely random selection method carrying over the best result from each cycle will, in general, maintain a roughly constant quality of poses. The probabilistic completeness of the overall search is thus retained.

Figure 6.10 illustrates this with an artificial example of a search method producing 4 results and using a quota of 6. After conformations L_1 and L_2 were docked, and their results merged into the list, the quota of 6 poses had been filled. Before this point, no quota-based rejection was permitted; there was now a worst-case score of 2. The search with L_3 was abandoned after some specified cycle since no pose had yet been found with

```
//Assuming positive scores are desirable
Create an empty scored pose list, Results
Create a globally-accessible value, W
Create a Boolean flag, Reject
Initialize W to negative infinity
For each conformation C do
  Begin
  Set Reject to false
  Create search population P for C
  Set G to 1
  While G ≤ generations and Reject is false do
    Begin
    Progress P one step using the search method
    If G ≥ MinGens and best score in P < W then
      Set Reject to true
    Increment G
    End
  If Reject is false then
    Merge P into Results
  End
Return Results
```

Listing 6.6: Quota-based rejection algorithm outline

Score$(R, L_3) \geq 2$. After L_4, which did meet this requirement, the worst-case score rose to 4.

It may be helpful to extend the quota specification slightly, by allowing the output count to differ from the quota used to determine the worst case. For example, to use the illustration of Figure 6.10 again, the result list quota of 6 might be cut to the top 4 for the final output. A larger quota than the final result list size could be used to offer the individual conformations a better chance of reaching the worst-case score, and thus not be discarded, while still only returning a manageable batch of results.

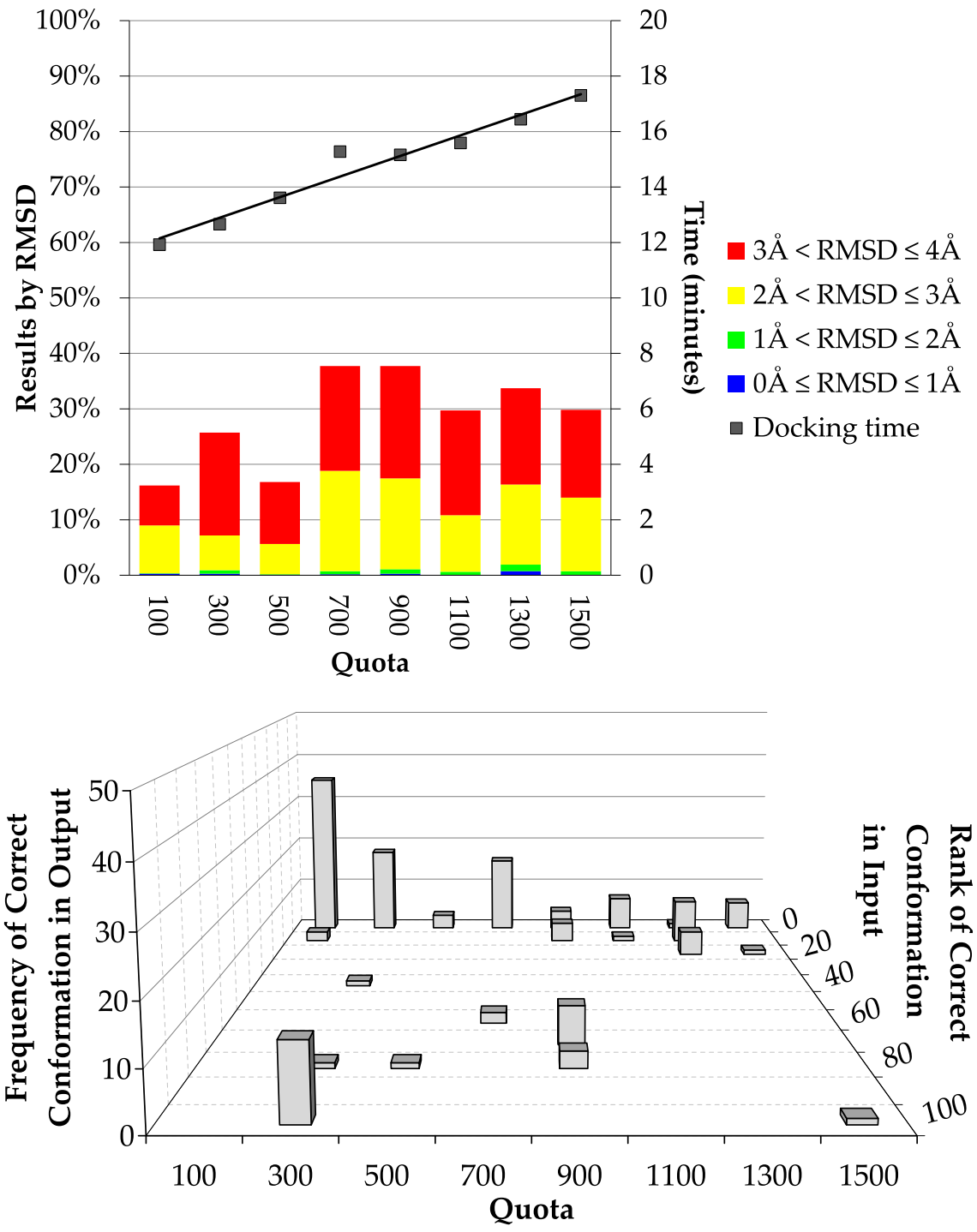
The main loop in *DOX*, which iterates through the ligand conformations, was modified to check whether the result container is full and, if so, report the worst score recorded to the genetic algorithm classes. That code, in turn, had a test inserted at the end of each generation to terminate the procedure if no genome in the population has scored as well as the worst case, provided at least some minimum number of generations have been completed. A summary of this method is shown in Listing 6.6.

If a conformation was rejected in this manner, the subsequent local optimization stage (§6.1 (*p*.117)) would also be skipped; hence, rejecting a conformation after the final generation still reduces the work to be done. Note that if a quota q is used with

a search population of size p, the first $\left\lceil \frac{q}{p} \right\rceil$ conformations in the input ensemble cannot be rejected, since the quota will not have been met when their searches begin.

Eight different quota values (from 100 to 1500) were used for redocking the 1K3U test case (see §E.1.2 (*p*.197)) with eleven different arrangements of the ligand conformation input database. These each had the correct conformation at a different index position, to assess whether using quotas biases the search. As intended, Figure 6.11 shows that the docking time increases linearly with quota size. The effect on result proximity to the correct binding pose is very limited, with some slight improvement of the proportions in each RMSD range as the quota increases, but not a consistent trend.

However, it would seem that quotas do skew the likelihood of ligand conformations appearing in the results by their ranking in the input. The three-dimensional chart in Figure 6.11 shows that the crystal conformation was only reliably returned in the results when it was the first structure presented for docking. Indeed, at small quotas, it was very much preferred (as the safe candidate). At higher quotas — 900 and above — it was still retrieved when near the top of the input, around the top 20 cases. In general, though, once the true binding pose of the ligand was no longer guaranteed retention it was usually lost from the results altogether, indicating that quotas are not an appropriate technique for virtual screening.



Test used [see Appendix E]: 11× 1K3U on Eurymedon N
(various result quotas, minimum 120 of 240 generations)

Figure 6.11: Effect of quota size in quota-based rejection on docking time and accuracy, highlighting conformational bias of smaller quotas

Chapter 7

Properties, Priority, and Parallelization

> *Major Premise:* Sixty men can do a piece of work sixty times as quickly as one man.
> *Minor Premise:* One man can dig a posthole in sixty seconds; therefore —
> *Conclusion:* Sixty men can dig a posthole in one second.
> This may be called the syllogism arithmetical, in which, by combining logic and mathematics, we obtain a double certainty and are twice blessed.
>
> *'The Devil's Dictionary', Ambrose Bierce*

Effectively completing a large batch of molecular docking tasks requires organization. Dividing the work up into distinct, self-contained units allows them to be handled flexibly, executing whenever and wherever appropriate using the available processors. Storing information with a structured but convenient form not only improves data management, but also makes it easier to accumulate and use experience to guide future searches. This chapter discusses possible methods for collecting molecular properties and managing individual docking cases — whether on one computer or many — to coordinate an efficient use of available computing resources.

7.1 Knowledge Bases

To support parallelized docking and the flexible storage of molecules with their associated data, I developed the Molecular Knowledge Base (MKB) interface. Technical details of this structure, its operation and my file-based implementation, are given in §F.6 (*p*.212). The requirements for the design were to:

- Store full structure information for molecules of any size.
- Store and/or calculate arbitrary data entries using identifying names.
- Allow multiple conformations of the same molecule to be kept together.
- Associate data entries with a molecule, and optionally a particular conformation.
- Operate independently of scoring functions, search methods, etc.

- Calculate missing entries automatically when they are requested for the first time.
- Provide notifications of updated entries, allowing clients to retrieve the new data.
- Permit shared access in network environments, merging changes automatically.

These last three points in particular distinguish the MKB from a simple data file, calling for some kind of management code overseeing input and output transactions.

This specification is for a format in which molecular data may be managed; it is not itself a piece of software. It could be fulfilled by a variety of underlying implementations: a database system would be appropriate (especially in some industrial situations), but for this work I have used a directory of files per molecule for simplicity. This is still more adaptable than the commonly-used flat file formats, such as PDB and SDF. The PDB format is generally only used for single conformations and has no structured facility for custom data fields. Structure-Data Format (SDF) is convenient for ensembles of small molecule conformations (fewer than 1000 atoms), and does allow for arbitrary data items to be included provided that the data are formatted as plain text with limited line length.

The previous implementations of *DOX* kept the receptor's structure in a PDB file, while the ligands had to be in SDF format. The stored scoring function data, however, would be in a single file per receptor/function combination, possibly in a different file-system location, named using the molecule's title (not filename) and function's name, and with all LUTs serialized one after another. Although this conveniently holds all the calculated data in one monolithic unit, it does not associate it with any particular structure file, instead requiring users to ensure they provide the correct data file to match the target. More importantly for this work, there is no provision for information not based on scoring function and receptor, such as molecular shape properties for either target or ligand. Besides rectifying this unruliness, an additional convenience of the more flexible method is that fewer command line parameters are required to configure a docking since all data sources are implicitly specified by the choice of molecules.

At the same time as revising all data storage, I took the opportunity to make several other aspects of the code's class hierarchies more clearly delineated and flexibly configurable, including scoring functions, search methods, conformation collections, internal ligand pose models, and search space parameters. Any edition of *DOX* incorporating knowledge base storage has this new architecture.

The overall effect of docking a given ligand/receptor complex should be practically unchanged by the new internal logic. Obtaining data through an MKB interface is schematically no different from parsing the provided structure and data files, but delegates the storage issues to the back-end implementation. The raw data itself — when passed to the scoring functions, search methods, and other processing algorithms — will be identical, and so the output should be unaffected by the design. An added benefit of abstracting data storage is that entries requested can be retained in memory for quick reference during program execution.

In Figure 7.1, the first two columns compare the previous version [P] with the equivalent redesigned system [K]. Rather than showing no difference at all, the redeveloped edition is 7.5% faster and at least equivalently successful in redocking the 1AF2 test case. The speed is an effect of the increased efficiency of the new code design: the elimination of all unnecessary copying and reprocessing of molecular data.

Figure 7.2 compares the two storage methods by redocking the entire *Astex* Diverse Set (see §E.1.3 (*p*.199)), providing a more thorough test with a wide variety of molecule sizes and conformation counts. Using MKBs resulted in a total docking time of just under 58 hours: a 10% reduction relative to the flat file system. Loading pre-calculated data from disk and configuring the docking parameters is around twice as fast, although this is of lesser importance for high-throughput use. Most importantly, the result RMSDs are unaffected (perhaps slightly improved) by the new architecture, reaffirming that the redesign does not interfere with the docking algorithms themselves.

There is some variation in the timing data. Although the overall improvement was a 10% speed-up, a minority of individual cases did take longer in [K]. Ligand 1W1P required nearly triple the time for docking. This is one of the very simplest cases, with only 11 heavy atoms, so I plotted Figure 7.3 to show the relationship between the ligands' sizes and the improvement seen with the MKB edition. This shows that, in general, the cases that slowed down are those that had fewer atoms and therefore conformations. However, for most cases, and in particular those of any complexity, the new design is almost universally better.

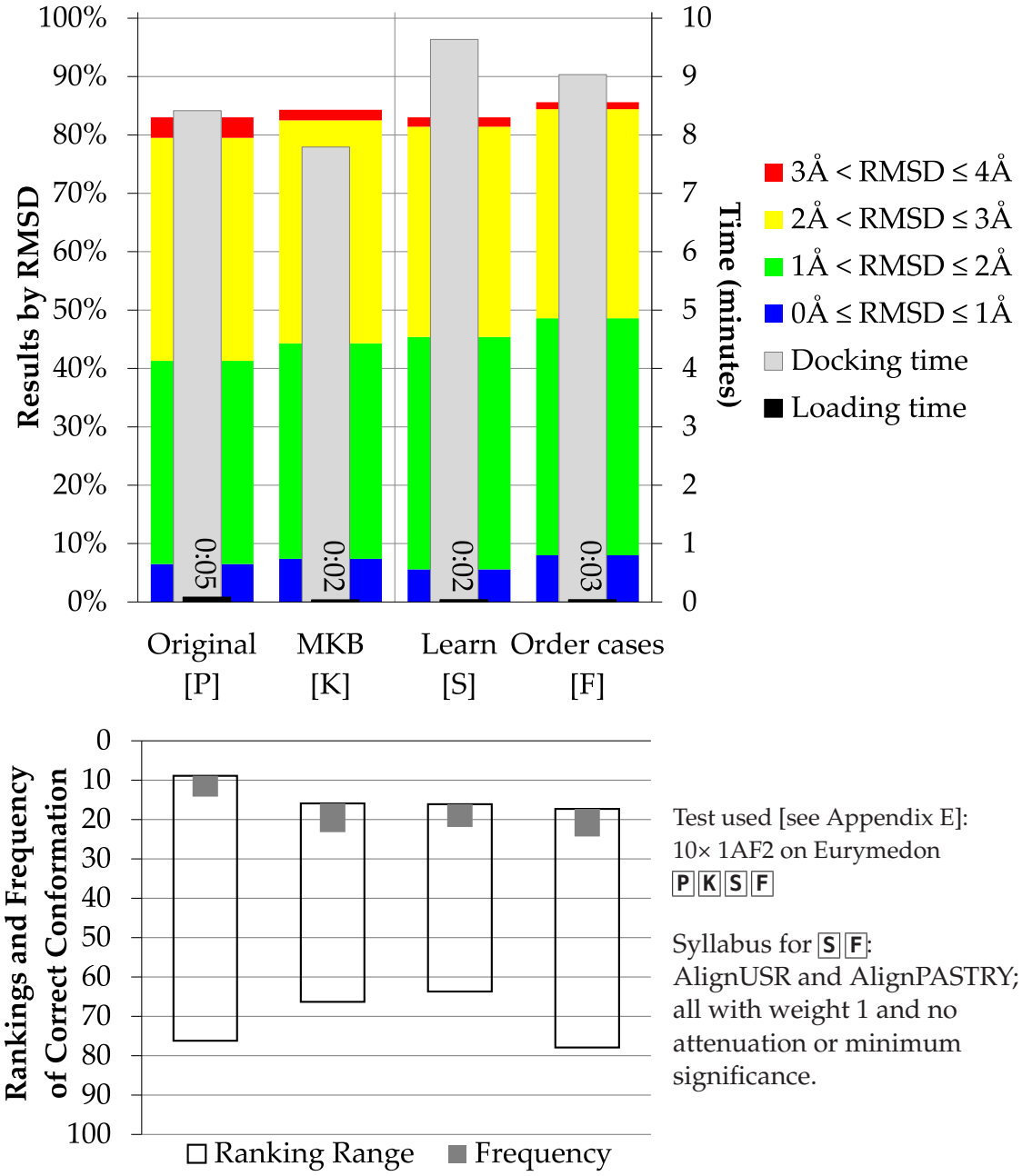
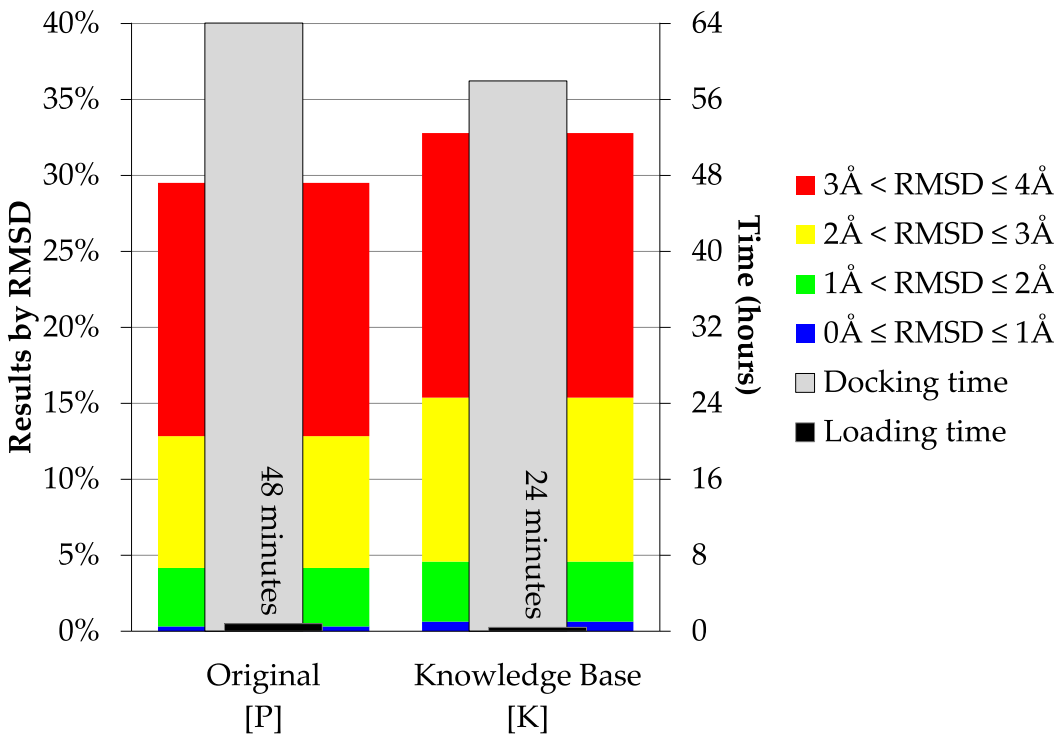


Figure 7.1: Comparison of single file and MKB implementations, and learning procedures, showing improved docking accuracy and time with the new design

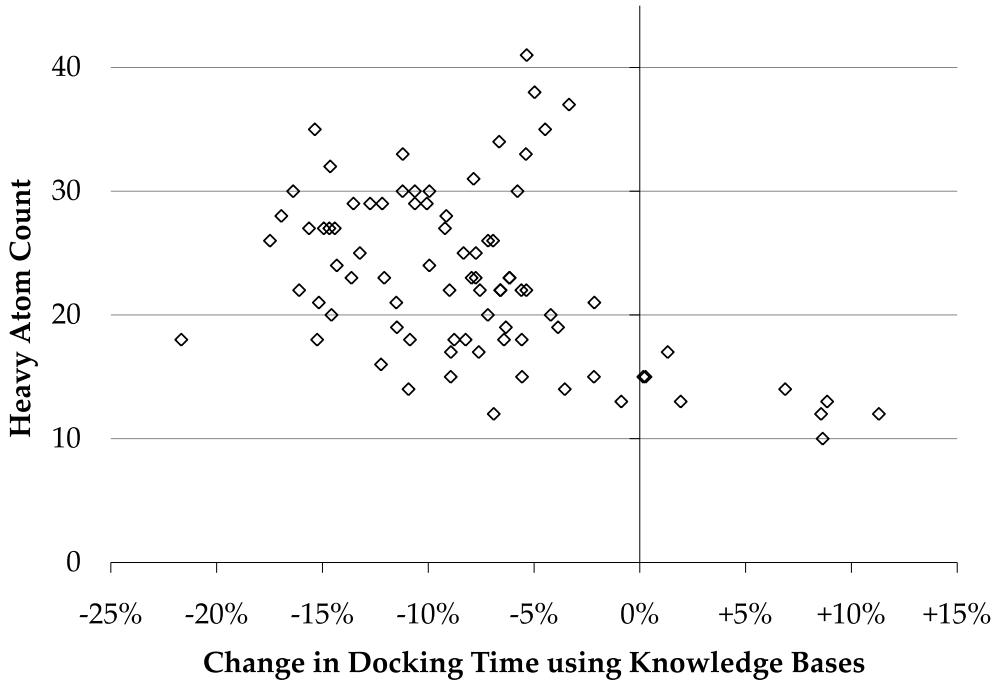
7.1.1 Normalized Scores

One disadvantage of encouraging the use of various interchangeable scoring functions is that they cannot be assumed to provide comparable scores. The values that might be returned often fall in completely different ranges, and may not have a fixed bound at all. To counteract this problem, I introduced the additional measure of a normalized score: one scaled in relation to all other scores produced by a function for a particular receptor. This is enabled by automatically creating an entry in the receptor's MKB called



Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon P K

Figure 7.2: Thorough multi-target comparison of docking time and results for SDF and MKB molecular storage, confirming quicker execution and equivalent results



Case 1W1P omitted for clarity.
Test used [see Appendix E]: 1× *Astex* Diverse Set on Eurymedon P K (crystal structures only)

Figure 7.3: Relationship between ligand size and docking time improvement using MKB storage: time only increased for some of the smallest molecules

'sfRange.*FunctionName*' which tracks the maximum and minimum values produced by the named function. As part of the `IScoringFunction` interface, this behaviour is incorporated automatically into all scoring functions. Normalized scores are calculated thus:

$$\mathbf{F}_{\mathbf{normalized}}(R,L) = 2\left(\frac{\mathbf{F}(R,L) - \text{sfRange}(\mathbf{F})_{\text{worst}}}{\text{sfRange}(\mathbf{F})_{\text{best}} - \text{sfRange}(\mathbf{F})_{\text{worst}}}\right) - 1$$

giving a value in the range $[-1,+1]$. If only one score has been calculated — that is, the best and worst scores are equal — the normalized score is defined to be the neutral value zero. Note that a normalized version of a score is not constant and could change if recalculated; however, as an assessment of relative virtue for such applications as learning, this is an improvement on unscaled values.

7.2 Learnable Properties

Although an MKB entry class may be defined for any purpose and data whatsoever, the primary application is likely to be calculating and storing some property of the molecule. If these attributes have some potential for identifying traits of well-docking ligands, it is desirable to be able to accumulate such knowledge automatically. To enable this, I propose a learning mechanism based on a standardized `LearnableProperty` MKB entry class. A `Syllabus` may be defined, listing learnable property names to be collected for the molecules considered, and used when assessing a new case. `Education` entries are updated using the learnable properties of docking results, and can produce an opinion for any given property value based on the accumulated examples. Interface specifications for the learning system are given in §F.7 (*p*.214).

The learning process uses a supervised case-based reasoning method. Learnable properties are required to provide their value(s) in unidimensional (fixed or variable length) array form, and also a method by which two such values may be compared on the scale $[0,1]$. A method to generate poses and/or search boxes (§5.2.5 (*p*.101)) for a ligand based on a particular property value is also required, but this is free to return fewer than the requested number if appropriate.

When a docking search completes for a conformation, the properties specified in the syllabus are calculated (or loaded from the MKB), their values are adjusted to represent

Name	Kind	Description
AtomCount	L/R	A simple integer property giving the HAC of the molecule.
PlacePIES-*c-f-t*	R	The *PIES* pocket prediction method of §5.2.1 (*p*.87).
PlacePIECE-*c-f-t*	R	The *PIECE* variation on *PIES*.
PlacePASS-*b-p-t*	R	The *PASS* pocket prediction method of §5.2.2 (*p*.96).
ShapeUSR	L	The *USR* shape descriptor from §5.3.1 (*p*.106).
AlignUSR	P	A descendant of the above for pre-alignment purposes.
AlignPASTRY	P	The *PASTRY* descriptor from §5.3.2 (*p*.107).

Kinds: R=receptor property, L=ligand property, P=ligand pose property.

Table 7.1: Learnable molecular properties implemented in *DOX*

each result pose (if necessary), and each case is added to the appropriate education with a weighting determined by its normalized score (§7.1.1 (*p*.146)). Hence, a wealth of examples are aggregated to provide probabilities of high docking affinity for a given ligand descriptor. If a previously-unseen value is presented for assessment, an opinion is formed by interpolation from the most similar examples. If multiple properties are listed in the syllabus, the assessments from each of these are combined in a weighted mean to estimate the expected suitability of the case in question. This is similar to a naive Bayesian classifier, estimating the probability of classifying a ligand conformation as dockable, although the features are combined disjunctively rather than conjunctively. Since a ligand may dock well even if only some properties are suitable, and it is quite possible that multiple correlating features might be used (two shape descriptors, perhaps), I chose to avoid multiplying probabilities in order to maintain reasonable scores for prioritization.

Including the learning process at the end of each conformation's docking search will inevitably increase the overall screening time. Figure 7.1 includes statistics for the **S** edition of *DOX*, which learns about the shape properties of each ligand case, but does nothing with the information accumulated. This learning version produces very similar results — as it should — but increased the time taken substantially. The syllabus used listed the two shape descriptors discussed in §5.3 (*p*.106), *USR* and *PASTRY*, and these required calculating for every result pose. Naturally, expanding this list of properties would add to the delay in the overall process.

I have developed or reimplemented several properties to be used with the learning mechanism, listed in Table 7.1. Admittedly, the 'Place...' properties are not intended for learning, only for pre-alignment, but the `LearnableProperty` interface lends itself

to the situation by enforcing the pose generation method and providing a convenient data storage format. The similarity measure required by the superclass is implemented as the proportional volume of intersection. They all record their pockets as lists of spheres, and so can be treated as learnable values. One could apply them to ligands as excessively sophisticated shape descriptors — there is no technical impediment to such use in *DOX* — and so obtain comparisons of the molecules' relative concavity, but this is unlikely to be useful.

7.2.1 Conformation Prioritization

Using the information accumulated by this learning mechanism, a database of ligand conformations may be appraised to establish which of the presented molecules are more likely to bind to the receptor. If a ligand is seen of similar shape to one that docked well previously, then it may be surmised that it will bind in a similar pose. A set of inputs might then be ordered according to these estimated suitabilities, and cases that appear promising may be considered sooner. The syllabus provides a central point to make such judgements, obtaining an opinion independently from each property and combining these in a weighted mean.

I added a `Sort` method to the `IConfContainer` class, performing a quick-sort on the ligand conformations using any given comparator function, and defined such a function to use the syllabus for prioritization. This sort algorithm is well-known to be generally $\mathcal{O}(N \log N)$ in the number of cases [Hoare, 1962]. Assessing each example (which is done at most once per case by using a cache) requires obtaining an opinion in $\mathcal{O}(PE)$ (but $\mathcal{O}(P \log E)$ if the property value has been seen before), where E is the size of the education data and P is the number of properties being used.

The [F] edition incorporates this pre-processing and Figure 7.1 shows its effect. With the 1AF2 example, there is apparently a small decrease in run time, despite using a syllabus of two properties with approximately 5300 opinions for each. Processing the opinions as part of the sorting stage may cause their data to be pre-arranged in memory, allowing the learning steps to be quicker after each conformation is docked, and thus a net benefit is obtained. The results themselves are unaffected by the different order of docking. This is as it should be, since the individual cases are independent. If, however,

the quota-based rejection stratagem discussed in §6.4.3 (*p*.137) were to be introduced, this independence would be lost.

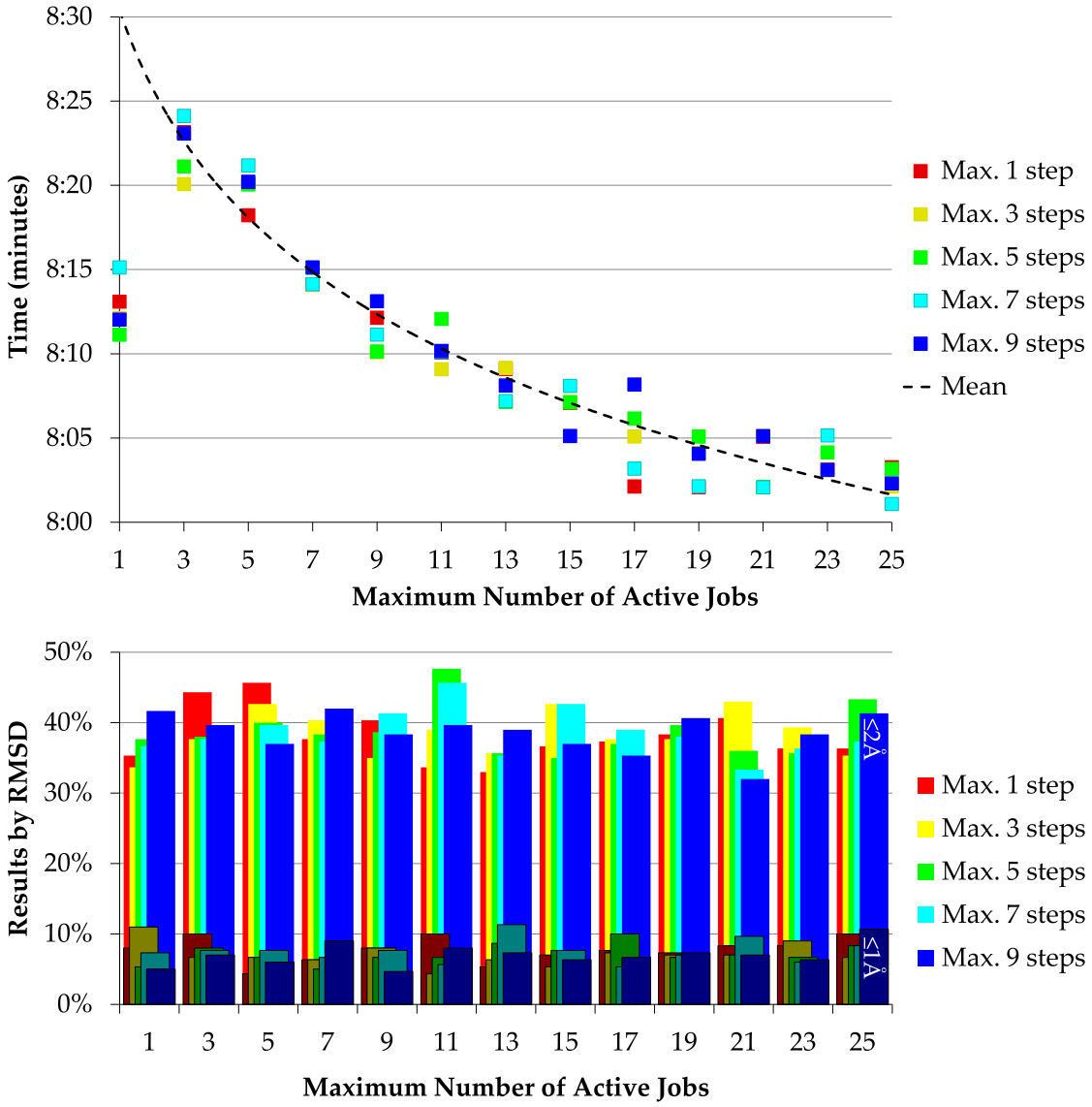
7.3 Job Control

The knowledge base architecture also introduced job control to the system, encapsulating a particular docking search procedure (including receptor, ligand conformation, and parameters) into a `Job` object that is simply started and stopped, with all search operations handled internally. Establishing the concepts of a search, its state, and the cases under consideration not only provides for the suspension, resumption, and abandonment of cases, it will also facilitate the distribution of a virtual screening task over many processors by allocating each a subset of the jobs. The design and implementation details of job-based docking are given in §F.8 (*p*.217). This framework could be adapted easily for many other large-scale parallelizeable search problems.

I implemented one class of priority function in *DOX*, `SimplePriority`, offering parameters to control the maximum number of jobs that may be active concurrently and the maximum absolute value of a job's priority (effectively the number of steps that it may take before another must be considered). The class causes new jobs to be started whenever possible, and adds the normalized scores (§7.1.1 (*p*.146)) of each job's best search cases to their priorities. If the quota-based early rejection stratagem from §6.4.3 (*p*.137) is in use, this priority function also checks the worst-case value; if appropriate, it will reject a job outright.

The [J] edition operates using this job-based system. Although the job-control stratagem automatically incorporates parallel execution capabilities (§7.4 (*p*.153)), it is still possible to run in a single processor mode. Since the jobs are not executed in separate threads here, increasing the number that may be in progress contemporaneously should not speed up the docking search, except by perhaps making better use of available memory. However, we should also see no significant loss of results.

Figure 7.4 compares several configurations of the simple priority function. As predicted, the docking results show no particular trend correlating with either the number of jobs permitted or the maximum number of steps they can take uninterrupted.



Test used [see Appendix E]: 5× 1AF2 on Eurymedon J (various job and step count limits)

Figure 7.4: Effect of simple job priority parameters on docking time and results using a single processor thread, showing the slight speed-up with sufficiently many jobs

Allowing multiple jobs to be under simultaneous consideration slows the docking process at small numbers. It is better to take one complete case at a time than to interleave a few — this is understandable since it does make optimal use of any hardware caching facilities. However, if enough jobs are loaded alongside each other (around 8 in this case), this can reduce the docking time very slightly. Note the scale of the time graph in Figure 7.4, though; the variation in time is proportionally very small — around 2% — and so this is not a significant effect.

7.4 Multi-Processor Distribution

Screening a list of ligands and their conformations for binding poses with a receptor is a task ideal for parallel execution. Genetic algorithms, in general, are regarded as embarrassingly parallel: the evaluation of each member of a population is an independent procedure, and so the search lends itself to being divided across multiple processors. Here, I choose instead to distribute the molecular conformations across execution processes and keep each of their searches on a single thread. Practical difficulties would arise with a multi-threaded design since the *OpenBabel* libraries are not thread-safe, and so the molecular data structures would need redeveloping before the intended work could proceed. In essence, *OrthoDox* merely permits the automatic allocation (and reallocation) of an input ensemble across many instances of a single-processor knowledge base edition. This practice has been used manually in industrial situations; it seems appropriate to mechanize the operation. The configuration and protocol used by the distributed docking system are described in §F.8.1 (*p*.218).

Discussions with experienced modellers in the field suggested a more subtle reason for not attempting a more complicated parallelization: the inter-process communication overheads could outweigh the benefits of a distributed system. The facilitation of interchangeable search methods in the software also precludes any assumptions about their suitability (or mode) of distribution. Whilst a search method is not prevented from multi-threading its own code, the overall *OrthoDox* system has to treat each individual search as an indivisible unit. Thus, the client/server architecture with independent processes treating each conformation as a separate job was chosen. Technically, there is a single executable which may run in either Controller or Job Manager mode, according to its command line invocation. Figure 7.5 gives an overview of the components in this design, illustrating how data passes between the parts of the system.

To investigate the benefits of this collaborative docking system, the Tom cluster of 32 processors (see §E.2 (*p*.200)) was used to execute 32 *OrthoDox* Manager processes. Thirteen different subsets of these were used for redocking all 433 conformations from the *Astex* Mini Set against one of those targets (1LRH). Table 7.2 lists the configurations. No more than 4 Managers were started on each host to avoid having more processes running than processors. Exceeding this level would be detrimental to the system's

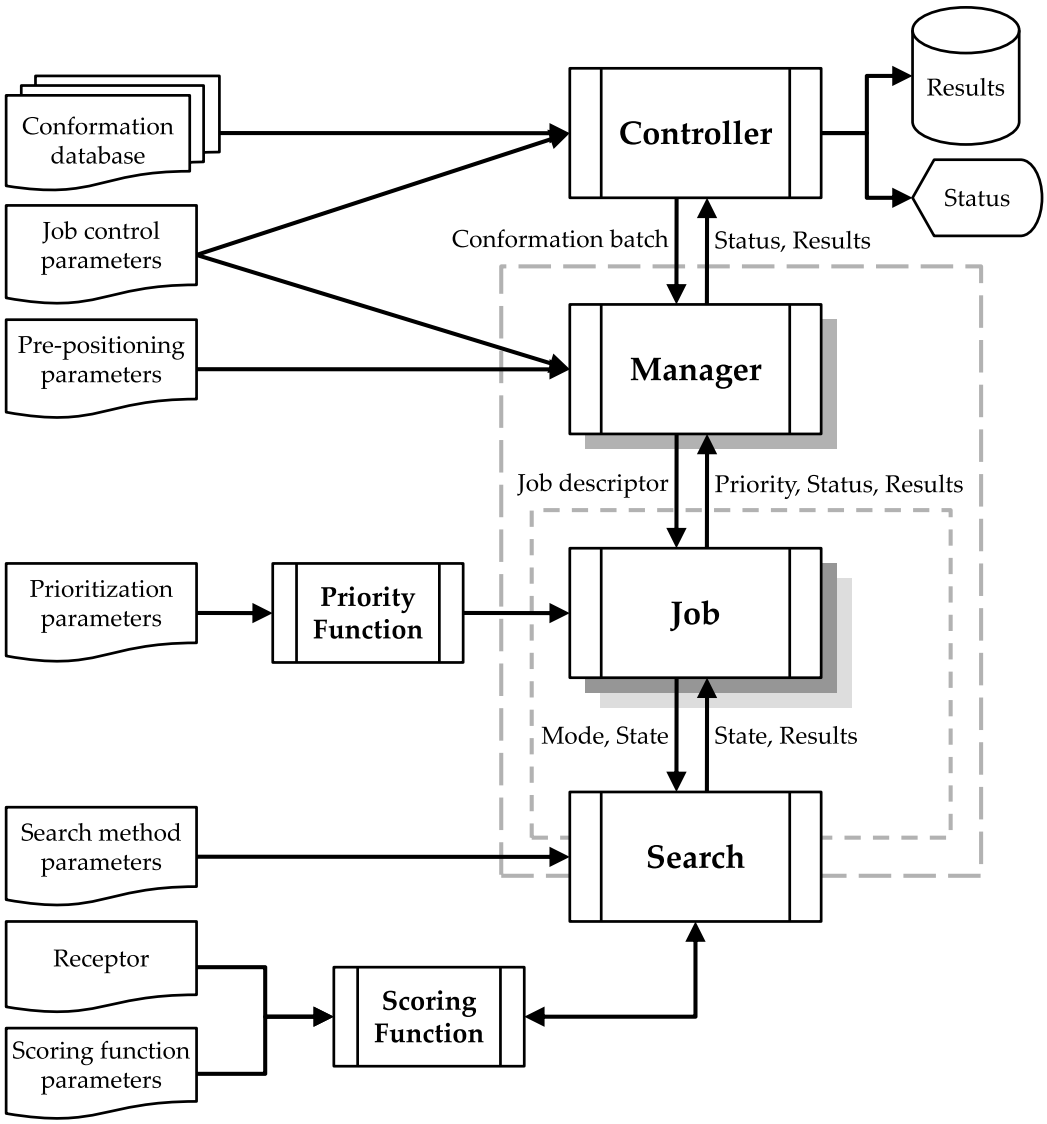
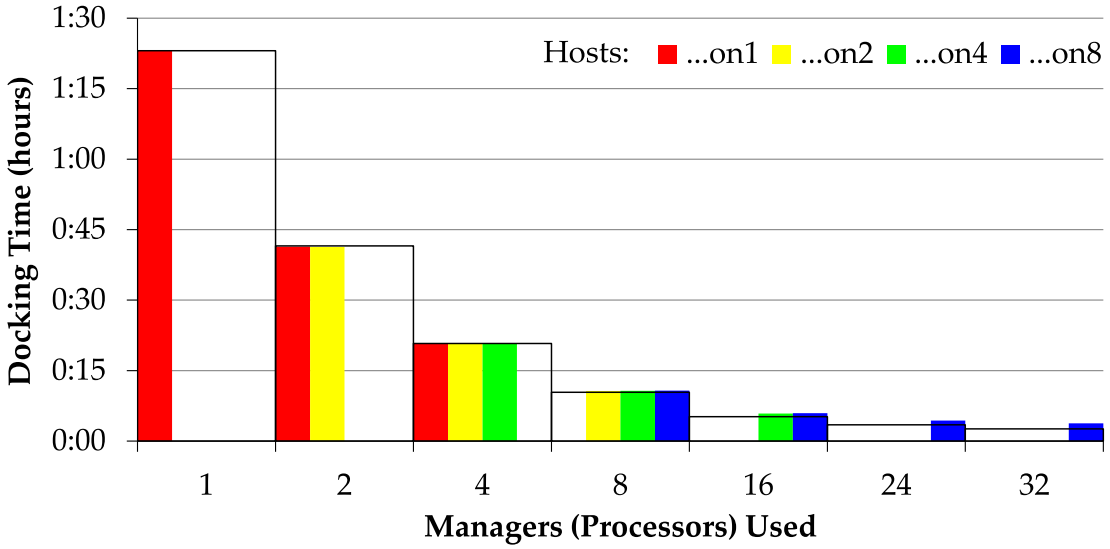


Figure 7.5: Architectural overview of *OrthoDox* parallel docking

Name	Hosts	Managers	Managers per host	Jobs per Manager
1on1	1	1	1	433
2on1	1	2	2	217
2on2	2	2	1	217
4on1	1	4	4	109
4on2	2	4	2	109
4on4	4	4	1	109
8on2	2	8	4	55
8on4	4	8	2	55
8on8	8	8	1	55
16on4	4	16	4	28
16on8	8	16	2	28
24on8	8	24	3	19
32on8	8	32	4	14

Table 7.2: Job Manager configurations for parallel docking tests



Black outlines mark expected times inversely proportional to number of processors.
Test used [see Appendix E]: 10× *Astex* Mini Set to 1LRH on Tom J
(parallel, various Manager configurations)

Figure 7.6: Effect of distributed parallel processing on docking time, using configurations from Table 7.2, confirming the expected inverse proportion to processor count

efficiency, as the operating system would be forced to suspend Managers frequently as it scheduled CPU time between them. The priority function for these tests was set to allow only 8 active jobs on each Manager, with a consecutive step limit of 8. The *XScore* function used a scaling power of 1 to account for the different sizes of the ligands in the input database; i.e. a pose of a ligand with N heavy atoms that scores S is evaluated as $S\left(N^{-1}\right)$ [Pan *et al.*, 2003].

One would expect to see an inverse linear correlation between the number of processors employed and the time taken to dock the conformations in a database. That is, the docking times ought to be in the ratio of the last column in Table 7.2, with the largest team of Managers ('32on8') completing the same task in approximately 3% of the single processor's time. Figure 7.6 shows that this cluster managed around 4.5%, which is reasonable considering that a backup process was competing for the last processor's time. It is also important to note that the timings available are imprecise, owing to a substantial latency in the network communication between Managers and the Controller introducing an error margin of several seconds. In the cases where the same number of processors were used, but distributed between machines differently, the times were practically identical. Evidently, multi-core processors are just as capable as the equivalent number of separate computers.

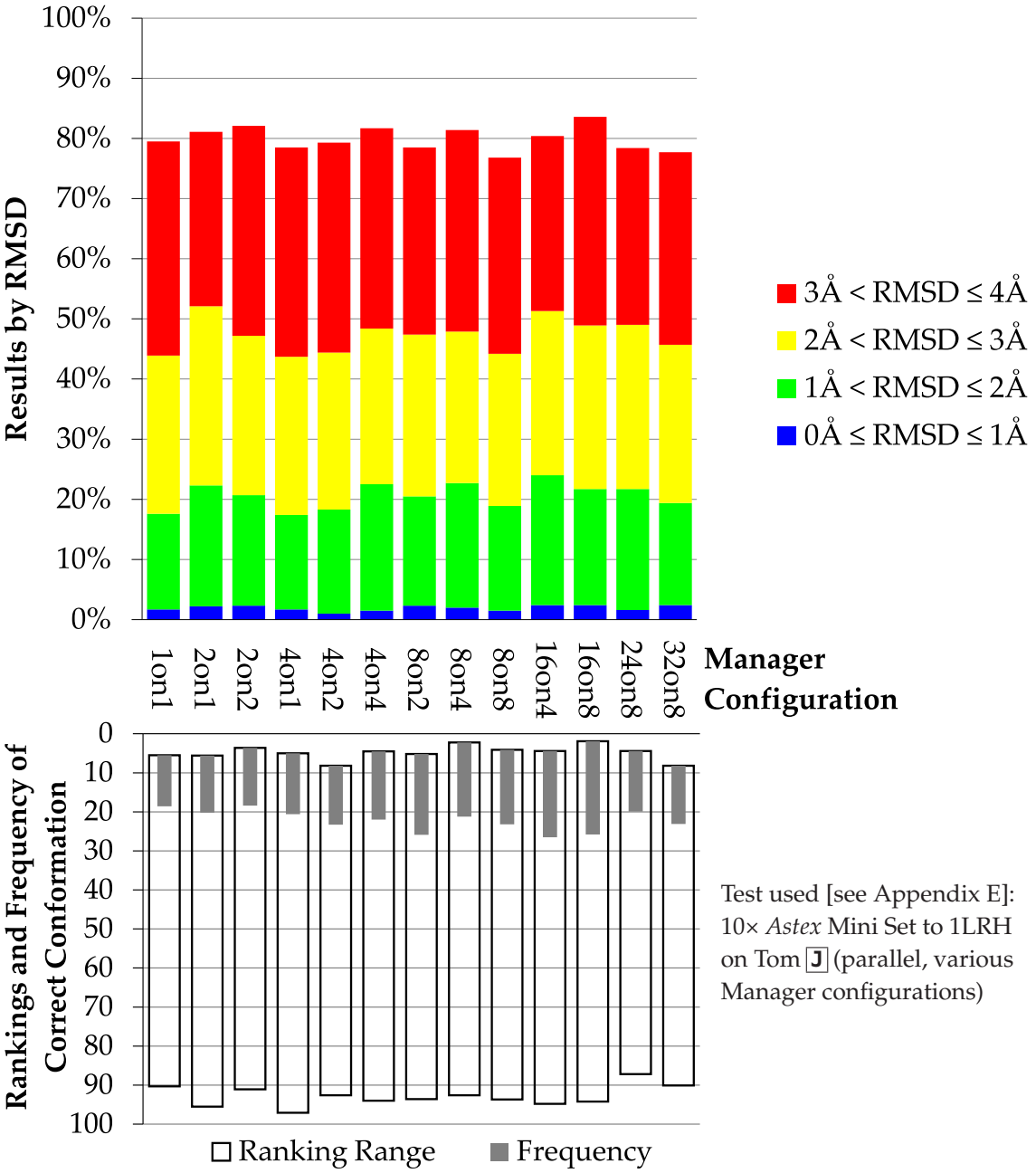


Figure 7.7: Effect of distributed parallel processing on docking results, using configurations from Table 7.2, showing comparable result quality from all arrangements

Importantly, as shown in Figure 7.7, there appears to be very little difference in the quality of results produced by any configuration. Schematically, the distributed docking system is completely equivalent to the previous software design, and so the final collection of poses produced by the controller should not be affected by where the individual conformations were handled.

Chapter 8

Comparisons and Conclusions

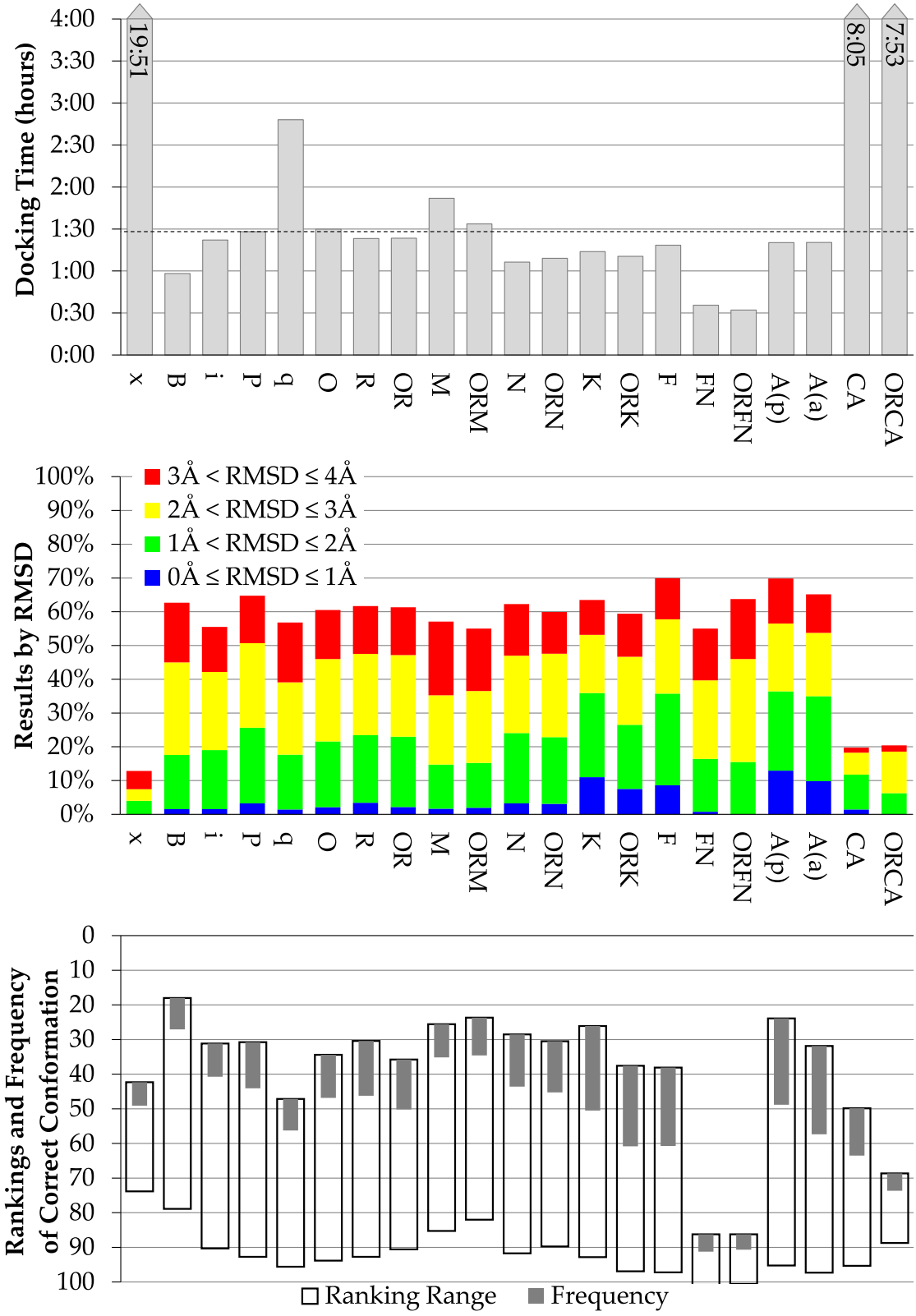
> All knowledge is of itself of some value. There is nothing so minute, or inconsiderable, that I would not rather know it than not.
>
> *Dr. Samuel Johnson*

8.1 Results: Assessment of Stratagems

The combinations of stratagems already discussed, with some others for completeness, are assessed once more here using the 12 test cases in the *Astex* Mini Set (see §E.1.4 (*p*.200)). With each strategy, only one configuration was used: these are listed in Table 8.1 and are based on the results found earlier in this thesis. The total docking time and average results from each design are shown in Figure 8.1. Refer to Appendix E (*p*.195) for details of the editions and their codes. Note that `q` is the original version, from before my investigations began, whereas `x` has none of the discussed stratagems.

Cases	Parameter	Value
All	Maximum results	100
	Population	96
	Generations	240
	Random starting poses	1 part
P	Optimization period	Final generation only
	Optimization range	Translations ±1Å
		Rotation components ±0.5
R	Scoring threshold	1500
M	Similarity threshold	5
N	Result quota	300
	Minimum generations	120
F	Syllabus properties	*USR* and *PASTRY*, equally weighted
C/**A**(p)	Pocket predictions	1 part from top 5 *PASS*-8-1.8-55
		1 part from top 5 *PIES*-5-3-7
(a)	Pose pre-alignment	1 part from top 5 *USR*
		1 part from top 5 *PASTRY*

Table 8.1: Default parameters as used for the large-scale comparison in Figure 8.1



Where tests failed to select the correct conformation in the top 100 results, this has been treated as a highest rank of 101 for statistical purposes.

Test used [see Appendix E]: 1× *Astex* Mini Set on Eurymedon `x` `B` `i` `P` `q` `O` `R` `OR` `M` `ORM` `N` `ORN` `K` `ORK` `F` `FN` `ORFN` `A` `CA` `ORCA` (using default parameters as listed in Table 8.1)

Figure 8.1: Comparison of 20 strategies derived from this thesis, showing their relative merits for speed and accuracy

Applying spatial indexing (B) increases the speed dramatically, especially so in these tests because *XScore* performs surface-based calculations. Adding a final local optimization stage (i) does lengthen the execution, but in conjunction with interpolating the LUT data (P) it also improves the results. Quaternion representation of rotations is substantially faster than Euler angles (q), taking around half the docking time and producing better results.

Ordering atoms for scoring (O) makes little discernible difference, but rejecting using a conservative scoring threshold (R) reduced the time by about 5% — whether or not the atom ordering was applied. The results are hardly damaged, with the proportion of poses within 2Å RMSD dropping from 26% to 23% over this test set. When early rejection for merging of similar poses (M) was used, the threshold chosen caused 14.5% of all cases to be replaced. It is therefore no surprise that this had a detrimental effect on the docking speed, increasing it by 27%. It also produced poorer result accuracy, although its selectivity was effectively equivalent. Combining this with scoring thresholds (ORM) almost restores the original docking time with no further loss of results, however.

Applying a quota to the number of results (N) is fast: it cuts the docking time by a quarter. This comes at a price, however. As discussed in §6.4.3 (*p*.137), the disproportionate preference introduced for the first conformations presented makes the comparably-good results shown here deceptive. Later, when job prioritization means that the correct conformation is no longer guaranteed retention, the results will be lost. Scoring thresholds alongside quotas (ORN) do not appear to have any effect, since their efficiencies are dwarfed by the rejection of entire searches.

The knowledge base redesign of the system (K) made data handling more efficient by minimizing recalculation of even small data structures, and so reduced docking time by 16%. Curiously, the results also improved substantially, which I can only explain by concluding that some subtle behavioural flaws were eliminated in the process. The greater quantity of good results shows clearer the effect of scoring threshold rejection (ORK): there is a drop in results within 2Å for a fairly small gain in speed. Prioritizing conformations (F) makes no difference to accuracy — provided that all cases are processed in full — and only has a very slight time delay for sorting. This makes sense, since the order of docking should be irrelevant to the output. However, introducing the

quota rejection scheme (FN), now that the correct conformation is not guaranteed safety, demonstrates the danger of that technique. It is extremely fast, but too risky for practical use. Combining this with scoring thresholds (ORFN) is at least as bad.

Pre-positioning in predicted pockets (A(p)) does require slightly more time to complete, but is still quicker than the pre-MKB system. When the search is otherwise constrained, though, it makes little difference to the outcome. The same can be said of pre-alignment to prior knowledge (A(a)). Searching a receptor's entire environment is far slower, naturally, and the size of the receptor will determine the time required. However, with automatic selection and elimination of search boxes (CA), a number of good results can still be obtained. Finding 12% of poses within 2Å RMSD is reasonable, especially when considered with the fact that at least two different boxes have been retained in these searches, and these tend to be relatively large. The selection of the correct conformation was tolerable, even if it was not always placed so close to the crystal structure, and filtering the results by search box would improve the ranking substantially. Introducing scoring thresholds as well (ORCA) makes no difference to the large time requirements, and does set back the result accuracy, suggesting that it causes too much approximation in the more expansive search space.

8.2 Future Work: More Stratagems to Consider

Inevitably, there will be many stratagems that have been overlooked by this thesis. It is impossible in one project to cover every conceivable approach to even the relatively specific task of rigid protein-ligand docking using scoring functions. Some ideas emerged too late in the work to be incorporated, while others would have been too diverting to combine with those included here. I introduce now some notable examples that deserve (or are already the subject of) further study.

8.2.1 Pharmacophores and Alignment

Simplified descriptions of a ligand's shape (such as *PASTRY* or *USR*, §5.3 (*p*.106)) can be helpful for alignment, and several methods could be used for this purpose [Kazhdan, 2004]. However, they do ignore many other important factors by hiding the underlying

structure. Molecules can be decomposed into small sections of common chemical motifs, known as pharmacophores [LaValle *et al.*, 1999; Daeyaert *et al.*, 2004]. These typically include various types of ring, charge concentrations, hydrophobic regions, hydrogen bond termini, and suchlike. Tools for annotating a molecule with its pharmacophores already exist, and these could be used to define molecular properties for use with knowledge bases (§7.1 (*p*.143)) — this is another form of structural decomposition, acting on internal volume rather than surface concavities. Later cases could then be pre-aligned using their common pharmacophores as well as overall shape.

If these decompositions are recorded as learnable properties, the significant pharmacophores in a particular receptor might be automatically identified. In turn, this information would be valuable for understanding better the biochemistry of a particular target, and designing ligands accordingly.

8.2.2 Directed Search Heuristics

Applying knowledge garnered from previous executions to picking starting points, as in §5.3.4 (*p*.111), is a relatively limited use of that information. One could also guide the search towards preferred configurations by weighting the exploration steps. This bias must not prevent the search from reaching any part of the space, of course. Obvious gravitational points in this scheme would be the receptor's predicted pockets, pre-aligned poses using shape descriptors, and (if available) pharmacophore tethers. Where an unconstrained search is used with a cached LUT (such as in §5.2.5 (*p*.101)), this could provide an alternative to pockets for box selection.

Alternatively, a simpler method might be to offer suggestions into the search method as it proceeds. For example, each generation of a GA could have one or more population members introduced to replace the worst. These would be generated based on the learnt preferences, or even simple poses that have scored well in the past and been collected in the receptor's knowledge base.

Using my job control design from §7.3 (*p*.151), it would also be quite straightforward for successful poses discovered with one conformation to be injected directly to other jobs for consideration. This would transcend the outer abstract search method layers of Table 3.2 (*p*.64), effectively searching by conformation and pose at the same time.

8.2.3 Search Methods

There are many techniques for exploring a large combinatorial search space. Evolutionary programming methods (genetic algorithms being the major example) have several variations. Other population-based procedures include particle swarms and ant colonies, and the issues discussed in this work would certainly apply to them directly. Tabu methods can also improve upon the behaviour of more stochastic searches by providing a repulsion from known results [Pei *et al.*, 2006].

When using conformation ensembles to model ligand flexibility, the time taken to perform the full docking process is linearly proportional to the number of cases provided. As an alternative, the conformation index could be incorporated into the search as an additional dimension, alongside the usual translation and rotation values. This would require substantially longer to find optimal solutions than a single-conformer search, but could be more efficient than fully docking each case separately.

The principle of appropriate processing, as exemplified by the early rejection work in §6.4 (*p*.127), uses an understanding of what information is relevant to avoid unnecessary precision. If more informative inputs than only a maximum result count are provided to a docking system, then further efficiencies could be made. One example would be a 'good enough' threshold, used to indicate that some representative poses are needed rather than a comprehensive docking. The suggestions generated by pocket prediction and pre-alignment methods could be assessed against the given normalized score (§7.1.1 (*p*.146)). If these poses compare favourably then they would be returned immediately, avoiding a lengthy search procedure, but if insufficient placements can be found using predictive methods then a normal docking would be started and pursued long enough to find some results of that standard.

Another use for the normalized scores, and the learning mechanism that motivated them, would be to intelligently select a cut-off score instead of the fixed quota method discussed in §6.4.3 (*p*.137). The quota system's main flaw is that it heavily prefers early conformations from the input because it cannot reject anything until the result list is full. Using learnt successes from the receptor's knowledge base, the worst-case score could be replaced with a minimum-acceptable suitability — the evaluation of admissibility then based on longer-term data rather than the current task's progress alone. Although this

would make little difference to the very first few cases docked to a new receptor, it may be possible to use it more aggressively without the detrimental biasing effect.

8.2.4 Sphere Tree Representations

Due to time constraints, it was not possible for me to pursue the hierarchical model of molecular structure prototyped in *cSpheres* (§3.2 (*p*.54)), but it has not been dismissed outright. Many aspects of that work could warrant further attention. The construction of intermediate levels in sphere trees automatically from the molecular data would require a clustering algorithm as discussed in §2.1 (*p*.26). Receptors, however, should perhaps have their interiors represented as a solid unit since these cannot generally flex significantly to accommodate a ligand. The atoms near surfaces should be partitioned variably, but some inner core should be treated as constant throughout the tree.

Extending the interaction between sphere trees from hard to soft collisions is necessary. Some form of potential function would have to be used to model electrostatics and other effects, but more subtle is the question of how this is applied to parent nodes in the tree. Simply summing the atoms' contributions would probably produce overestimates, although it may be a reasonable starting point for investigation. Once a useful behaviour is established, however, the ultimate goal would be to perform automatic flexible docking with both ligand and receptor molecules represented in this form. Methods such as roadmap path planning (see §2.3.2 (*p*.45)) suit the hierarchical model for molecular dynamics simulation. However, given some bound pose of the sphere tree, this structural decomposition might also lend itself to the incremental construction of alternative ligands.

8.3 Conclusions

Embarking on this kind of research in such a broad field is like opening Pandora's box: there is a limitless range of questions to answer about the best approaches to the construction of protein-ligand docking tools. In this thesis I have presented a selection of factors and demonstrated their effect in a real-world context.

When handling rotations in a physical system, the algebraic behaviour of quaternions (§5.1 (*p*.83)) befits more complicated processing than alternative systems such as triples of angles, which is necessary if one is to exploit geometric clues when analysing molecular interactions. A case in point is the pre-alignment of ligands to previously-successful arrangements (§5.3 (*p*.106)), for which relative orientations can be calculated directly using quaternions. One of the most important geometric hints available is that of the receptor's surface pockets (§5.2 (*p*.87)). Automatically identifying these can help to focus a search and save time when the correct active site is unknown.

There are many issues to be considered when performing the docking search algorithm itself. Combining large-scale and small-scale explorations (§6.1 (*p*.117)), is useful for optimizing any good results found, but frequent periodic use with Monte Carlo searching is of little benefit compared with one final refinement. Where pre-calculated data are employed, they should be interpolated to support the precision of the positions being sampled (§6.2 (*p*.121)). Although deferring calculations until they are required (§6.3 (*p*.125)) makes relatively little difference to the time cost of constrained docking, it could still be worthwhile when the search space is not defined in advance and will be generated automatically (§5.2.5 (*p*.101)).

Early rejection of unprofitable work is an important principle with many applications. Curtailing the evaluation of scoring functions when their results are predictably bad (§6.4.1 (*p*.128)) is worthwhile, but the enormous efficiency possible must be balanced with the risk of losing results. Maintaining search case diversity (§6.4.2 (*p*.134)) provides a moderate benefit when exploring such a large space as is necessary for this kind of work, but excessive use could introduce a diversion from converging good results. Result quota consideration (§6.4.3 (*p*.137)) is unsuitable for screening conformations sequentially, and of doubtful use with a prioritized job-based design. The technique of eliminating cases that trail behind alternatives is still relevant, though — narrowing a multi-box search (§5.2.5 (*p*.101)) being a useful example. Partial evaluation methods such as these are worth considering, but they are tools whose utility depends on their situation. Their purpose is to allow faster, less accurate searches where the requirement is to find good results, but not necessarily all. Using a stochastic search method implies a similar expectation, however.

Executing searches in parallel across as many processors as possible is an effective way to maximize throughput for screening (§7.4 (*p*.153)). Although the 32-core cluster used limits the demonstration that can be provided here, it is clear that more processors would continue to improve the pace of a docking system — as long as enough inputs are provided to occupy them. It is as much an enhancement of capacity as speed. Job-based processing (§7.3 (*p*.151)) is a necessary feature of such architecture, but allowing deferred and/or prioritized evaluation of conformations makes little other difference to the overall behaviour. A versatile configuration architecture and information storage mechanism (§7.1 (*p*.143)) is needed to underpin much of the above, and certainly can be at least as expeditious as a simpler fixed-content design. Making simulation tools with an inflexible data foundation can unnecessarily hamper any further attempts at sophistication. Using such a knowledge base for the accumulation of experience based on molecular properties (§7.2 (*p*.148)) can identify patterns of preferences and their best poses. Future ligands can then be prioritized using this resource, and pre-aligned to obtain promising lead results quite accurately.

8.3.1 Scoring, Searching, and Screening

Referring back to the roadmap described in §4.3 (*p*.75) and whose illustration is reproduced (at a smaller scale) in Figure 8.2, this thesis has demonstrated that efficient evaluation is both relevant and possible in all three layers of a docking tool. The research workflow (marked by the heavy arrows) has been helpful for investigating the construction of docking software for virtual screening; although it will not apply directly in every situation, I suggest that the basic order of consideration is useful.

Scoring must be done to an appropriate precision, with only as much detail as is required to discern good poses. When docking is required to compare a collection of ligands, the numbers returned as scores are not important in themselves — only their relative values matter — and scoring threshold early rejection provides an effective approximation in this circumstance.

Searching should be guided quickly to the poses most sensible in the context, increasing thoroughness for the most promising cases and eliminating the worst. Features of the receptor, particularly its geometry, determine where a ligand can bind

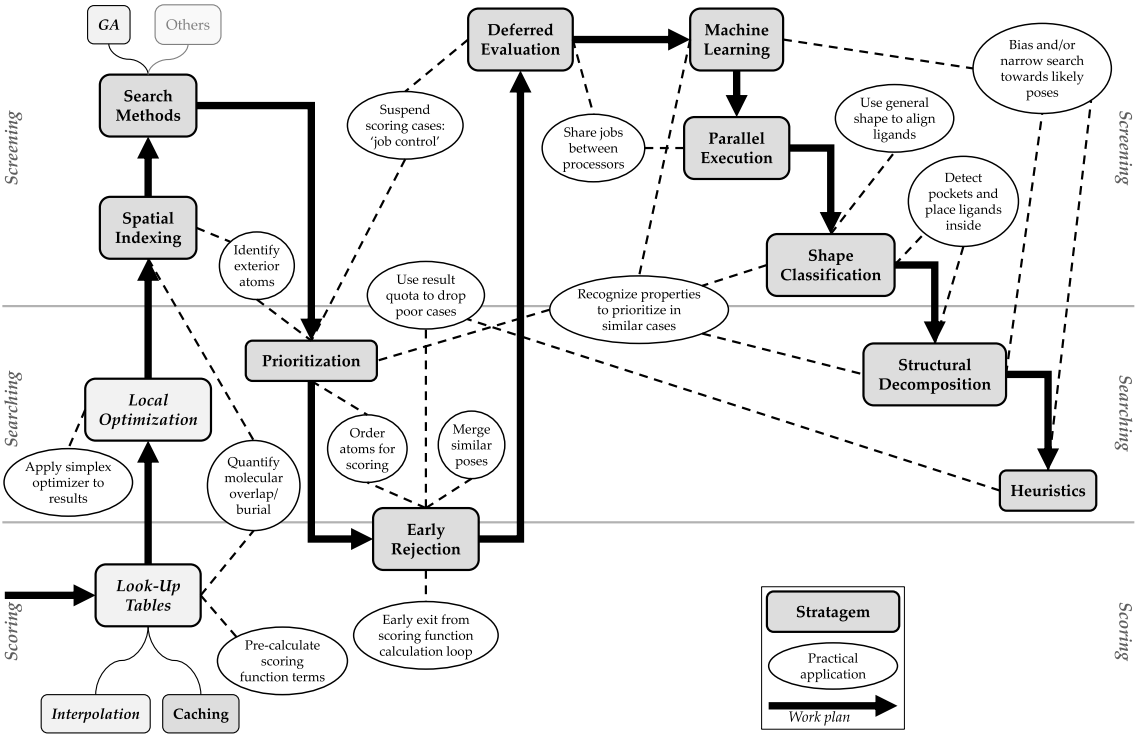


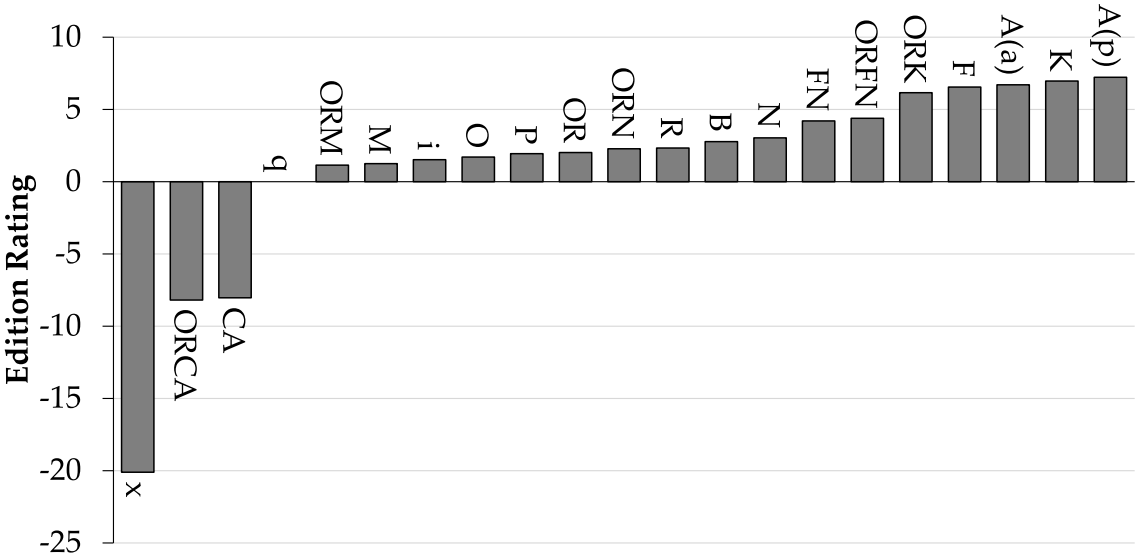
Figure 8.2: The research roadmap of stratagems, as explored by this thesis

and thus which poses a search method ought to assess. Pocket detection algorithms are one good source of this guidance; wherever it comes from, narrowing a search by dismissing unfruitful regions is a useful shortcut for exploration.

Screening can be organized for optimal resource usage, sharing work between parallel computation processes and managing data to facilitate quick retrieval. Information about the molecules, such as structural features, should be indexed in forms suitable to its application, not merely its versatility. Collecting patterns from results provides a means of prioritizing future work and starting searches in the most probable areas. It could also lead to an accumulation of knowledge from which it may be possible to learn about a molecule's general behaviour, contributing to any wider study.

8.3.2 A Strategy

To more clearly evaluate the editions of *DOX* considered, I defined a rating function based on the statistics in Figure 8.1 (*p*.158). This positively scores higher proportions of close (low RMSD) result poses and frequencies of selecting the correct ligand conformation. Penalties are deducted for the mean and top-ranked RMSD values being



Assessment formula used (constant c added to give zero score for [q], positive is good):

$$\frac{within1}{10\%} + \frac{within2}{25\%} + within4 - toprmsd - \frac{meanrmsd}{4} - \frac{ranktrue}{100} + \frac{freqtrue}{25\%} - hours + c$$

Data from Figure 8.1 (*p*.158).

Figure 8.3: Quantitative comparison of strategies, offering a crude preference ranking

large, the correct conformation ranking poorly, and slow execution. The scores assigned are shown in Figure 8.3, relative to the [q] edition that preceded any of this work.

Remembering that the quota-based rejection will have been unduly favoured by this measurement, the ranking offered by this gauge fits with the discussion above. Even so, quotas noticeably worsen the rating, except in comparison with the most simple editions. The MKB designs are all preferable to the older system, except when using the completely self-guided multi-box searches ([CA]/[ORCA]). Pre-alignment methods and scoring-based early rejection are, in the cautious configurations used here, reasonable contributions to the docking algorithms. In the non-MKB systems, using scoring thresholds also receives a similar rating, but pose merging is poor. Allowing for the limited results of multi-box searching, though, it is clear that using quaternions is universally better.

Summary

There are certain stratagems that can improve the efficacy of docking software for virtual screening if incorporated into the algorithms' design and development. To summarize them as a simple strategy, the key recommendations are to:

- Use quaternions to represent angles and rotations.
- Avoid calculating with unwarranted precision, especially scoring functions.
- Use geometric clues, such as pockets and good previous poses, to guide searches.
- Architect the code in an adaptable, modular form to support further improvements.

The topics presented in this thesis should be generally applicable to the continued development of useful tools in the field of computational chemistry. Of course, each potential application will have its own idiosyncrasies that affect the effect of any given factor — with any implementation, your mileage may vary. However, I hope that this smorgasbord of stratagems has contributed some helpful ideas to the already thriving drug discovery community.

Appendices

Appendix A

Fundamentals of Protein Structure

> We may, I believe, anticipate that the chemist of the future who is interested in the structure of proteins, nucleic acids, polysaccharides, and other complex substances with high molecular weight will come to rely upon a new structural chemistry, involving precise geometrical relationships among the atoms in the molecules and the rigorous application of the new structural principles, and that great progress will be made, through this technique, in the attack, by chemical methods, on the problems of biology and medicine.
>
> *Linus Pauling*

This appendix is a modified version of [Skone & Cameron, 2007]

A.1 Introduction

Proteins are of great importance to biological and medical research, and are one of the primary foci of biochemistry. This is largely because of their significance in the function of living cells — indeed, the word protein is derived from the Greek for 'primary importance'. These molecules have a polymeric structure, referred to as a polypeptide. This is constructed from the genetic code stored in DNA by a transcribing process: each set of three genetic nucleotides (called a codon) represents one of twenty amino acids.

Amino acids are small molecules (tens of atoms) that bind together in a chain; the individuals in the chain are called residues. Each has the same basic structure featuring a central alpha carbon atom and a terminal each of COOH and NH_3 (shown in Figure A.1). To construct a protein, two hydrogen atoms and the oxygen ion break off to form a water molecule, then the terminal carbon bonds with the nitrogen of the next residue. This linkage, shown in Figure A.2, is called a peptide bond. The thread of carbons and nitrogens is known as the backbone of the protein.

The twenty common amino acids are identified by either three-letter or single-letter codes, allowing a protein sequence to be written as a simple string. Amino acids have

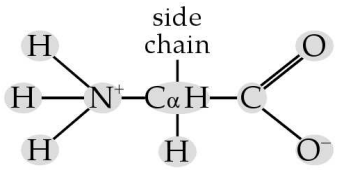


Figure A.1: Amino acid structure

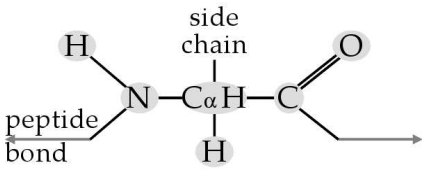


Figure A.2: Backbone linking

Group	Amino Acid	Notation	
Acidic: *Negatively charged, and reactive with water.*			
	Glutamic Acid	Glu	E
	Aspartic Acid	Asp	D
Aliphatic	Alanine: Slightly hydrophobic.	Ala	A
	Valine: More hydrophobic.	Val	V
	Leucine	Leu	L
	Isoleucine	Ile	I
Aromatic: *Large, bulky, and hydrophobic.*			
	Phenylalanine	Phe	F
	Tyrosine	Tyr	Y
	Tryptophan	Trp	W
Basic	Lysine: Side chain is hydrophobic, but overall structure is hydrophilic with positive charge.	Lys	K
	Arginine	Arg	R
	Histidine	His	H
Polar: *Generally on outside of protein, and often on active site of enzymes.*			
	Glutamine	Glu	Q
	Asparagine	Asn	N
	Serine	Ser	S
	Threonine	Thr	T
Sulphur-containing	Methionine	Met	M
	Cysteine: The most reactive amino acid. Pairs of Cysteine can form covalent disulphide bonds between their side chains, adding structural stability to proteins.	Cys	C
Unique conformation: *Not usually found in motifs of secondary structure.*			
	Glycine	Gly	G
	Proline	Pro	P

Table A.1: The alpha amino acids

certain properties, including charge and hydrophobicity, which help to determine their interactions with each other and the protein's environment.

A.1.1 Primary Structure

The alpha amino acids may be grouped under seven headings, shown in Table A.1. Short sequences of amino acids are referred to as peptides; proteins with many similar peptides

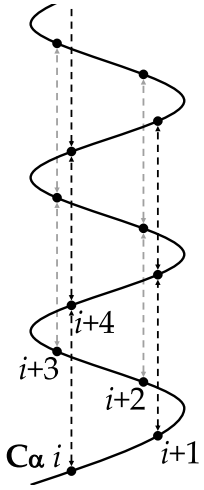


Figure A.3: Alpha helix bonding

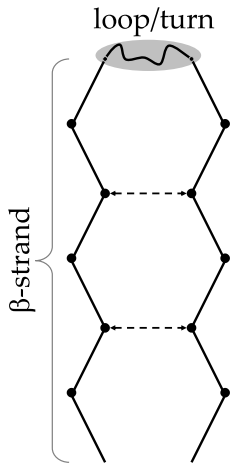


Figure A.4: Beta sheet bonding

in their backbones are described as homologous. The sequence of amino acid residues along a backbone is referred to as the protein's primary structure.

A.1.2 Secondary Structure

The threadlike protein structure is generally arranged in a globular form. The backbone is twisted and bonded together in a stable conformation shortly after the protein is formed. The arrangement taken by a protein (when subjected to no external constraints) is called the native conformation. Certain patterns, or motifs, are seen in most protein structures: two of the most common are alpha helices and beta sheets. These are particularly rigid and provide a great contribution to the structural integrity of the molecule. The alpha helix is a spiral of residues around an axis, with a hydrogen bond formed between each residue and its fourth successor (as shown in Figure A.3).

The beta sheet is a set of approximately straight sections of backbone aligned in a plane, with hydrogen bonds formed between adjacent residues. Sections of backbone that are part of neither a helix nor a sheet are described as loops or turns. Figure A.4 shows the alignment of beta strands in a sheet, with a loop/turn between the strands.

The arrangement of helices, sheets, loops, and turns is collectively referred to as the secondary structure of a protein.

A.1.3 Tertiary Structure

Some proteins exhibit a higher level of conformational grouping: these subsections of the protein with an independently stable arrangement are termed domains. The grouping of large sections of a protein backbone into domains is referred to as its tertiary structure. The largest known protein (Titin) has around 300 domains [Labeit *et al.*, 1997]. Tertiary structure may influence secondary structure by preventing helices or sheets from forming by the intervention of other sections of the backbone.

A.1.4 Quaternary Structure

Certain proteins have multiple backbones or distinct subunits. There are several such arrangements within this quaternary structure class, such as homodimers, oligomers, and concatomers. Homodimers are chains of identical subunits; concatomers are similar, but the subunits may not be identical. Oligomers contain several copies of a protein arranged around each other. Viruses, for example, are icosahedral oligomers.

A.2 Protein Behaviour and Interactions

A.2.1 Folding

Proteins contort into their native conformations in an extremely short time, a phenomenon that is clearly not random. The principal factors governing this process are the laws of thermodynamics, in the context of the electrostatic forces and hydrophobic interactions present between residues. In the body, proteins fold under the influence of their environment, i.e. the cell and other proteins surrounding the newly synthesized amino acid chain [Gething & Sambrook, 1992]. Consequently, it is not always possible for a protein to be denatured (unfolded) and refolded correctly *in vitro*.

A.2.2 Binding

Proteins may bind with other molecules, typically by hydrogen bonding or ionic interaction. These bindings are normally reversible, such that the protein returns to its original form after the second molecule is detached. The general term ligand refers to

a small non-protein molecule bound to a receptor protein. The structure of a protein may incorporate active sites — areas of the molecular surface where ligands tend to bind [Liang *et al.*, 1998]. The protein may deform slightly around or between the active sites when a ligand is present, owing to the forces between the two molecules, particularly by moving the amino acids' side chains or adjusting the relative position of entire domains [Ansari *et al.*, 1994].

A.2.3 Structure Identification

The molecular structure of proteins is of critical importance to understanding their functions and interactions. Two methods are well-established for obtaining data about the atomic arrangement of proteins and other large molecules: X-ray diffraction [Drenth, 1999] and Nuclear Magnetic Resonance (NMR) [Wüthrich, 1986; Wüthrich, 1990].

X-ray diffraction is the older method; it requires that the protein be in a solidified crystalline state, often cryogenically frozen, and mounted in careful alignment with the detecting apparatus. It has been estimated [Abola *et al.*, 2000] that it takes around ten minutes to arrange and obtain data from a single crystal, excluding the time to purify, crystallize, and freeze the sample and then analyse the raw data. X-ray diffraction is a time-consuming process with limited resolution, but it remains important because of its ability to handle larger proteins than NMR with relative ease of execution.

NMR is a more recent development which allows the distances between atoms to be measured with reasonable accuracy, and thus a picture of the entire molecule's structure to be constructed. It requires the use of powerful magnets, and is a similar process to that used in medical MRI scanners. NMR permits proteins to be scanned in solution, rather than a crystalline form, which avoids the possibility of structural deformation as a consequence of solidification. In addition, the conformational flexibility induced by ligand-binding may be investigated. The data produced by NMR is collated by computer, and possibly refined to settle on precise atomic coordinates. A more recent development in this refinement stage is the inclusion of a solvent in the calculations [Linge *et al.*, 2003].

The accuracy of a 3D representation of a protein, whether it be decoded from experimental data or generated by a computer prediction, is often described in terms of the RMSD of the atomic coordinates in the structure from some reference set of values.

However, assessing the accuracy of experimentally determined structures is a difficult and non-standardized issue [Snyder *et al.*, 2005].

A.2.4 Biochemistry

There are numerous properties associated with proteins. The most relevant to computational biochemistry are the Van der Waals atomic radii, charge distribution, overall hydrophobicity, and bond lengths and flexibility. A summary description of many of these may be found in [Richards, 1977], and a detailed exploration of thermodynamics and energy calculations in [Robertson & Murphy, 1997]. Bonds and their elasticity are dealt with in [Sinha & Smith-Gill, 2002], with examples given demonstrating the importance of flexibility in understanding the function and behaviour of some proteins *in vivo*.

For more information regarding protein science, the reader is referred to [Branden & Tooze, 1999] or [Berg *et al.*, 2002].

Appendix B

FFT Tesselation Test

> On two occasions I have been asked, "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?"
> I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.
>
> *Charles Babbage*

B.1 Implementation

This was an implementation of a simple rigid docking method in a graphical Java program. It was used as an early programming and familiarization exercise, with emphasis on good coding practices. See §3.1 (p.53) for the context of this work.

B.1.1 Background

The FFT geometric alignment method for protein-ligand docking seeks the optimal relative translation of two functions representing the spatial occupancy of the molecules [Katchalski-Katzir *et al.*, 1992]. Two lattices, $\boldsymbol{R}$ and $\boldsymbol{L}$, are created for the receptor and pre-rotated ligand, as described in §2.3.2 (p.35). A particular translation (a, b, c) of the ligand may then be scored by summing the pointwise products of $\boldsymbol{R}$ and $\boldsymbol{L}$:

$$\boldsymbol{S}_{a,b,c} = \sum_{x,y,z} \boldsymbol{R}_{x,y,z} \cdot \boldsymbol{L}_{x+a,y+b,z+c}$$

The maximal value in $\boldsymbol{S}$ is then the optimal docking translation. However, this is a costly function to evaluate in full since it involves six nested summations altogether. A more efficient way of calculating $\boldsymbol{S}$ is found by observing that it is a cross-correlation of $\boldsymbol{R}$ and $\boldsymbol{L}$, and applying the cross-correlation theorem [Papoulis, 1962]:

$$\boldsymbol{S} = \boldsymbol{R} \star \boldsymbol{L} = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(\boldsymbol{R})} \cdot \mathcal{F}(\boldsymbol{L})\right) \qquad \text{where } \mathcal{F} \text{ is the Fourier transform.}$$

Since fast, discrete Fourier transforms are well established [Cooley & Tukey, 1965; Elliott & Rao, 1982] and have a complexity of $\mathcal{O}(N \log N)$, this allows the correlation function to be computed in $\mathcal{O}(D^3 \log D^3)$ rather than $\mathcal{O}(D^6)$, where the lattice size is $N = D \times D \times D$.

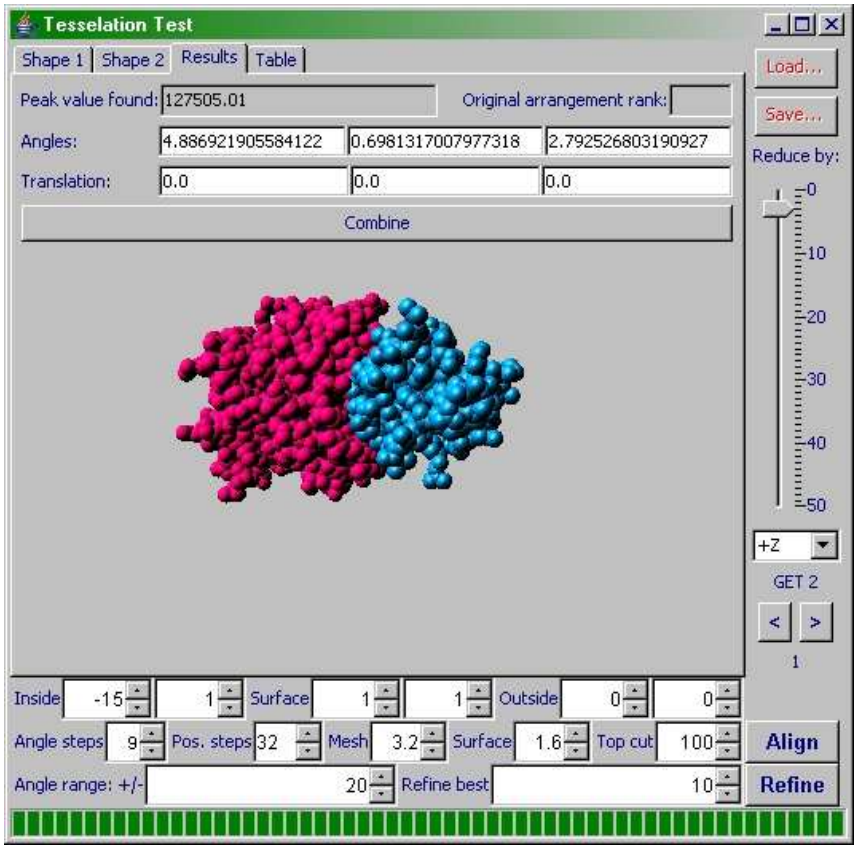
B.1.2 Interface

The Java classes I developed include the user interface, which provides controls for setting all the variables in the algorithm (such as grid resolution and the scoring functions), facilities for loading and viewing a molecule, and displays of the results of a docking — both the complexed structure as an image and the translation/rotation values determined. It also allows for the retrieval and display of the complete matrix of convolution for the most promising orientational arrangements. Figure B.1 shows the program interface after a docking has been performed.

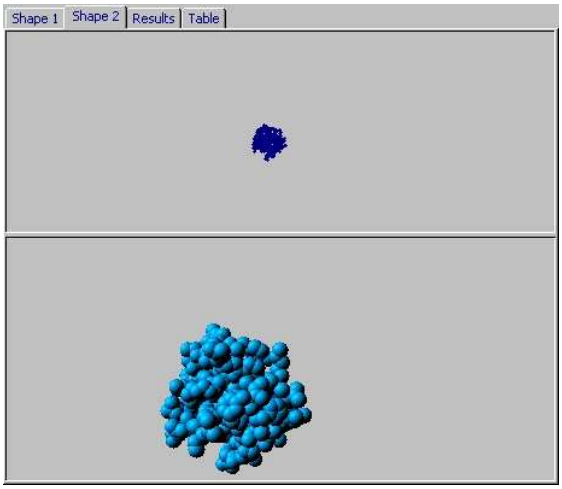
Representation

The molecules are stored as collections of solid spheres. Three classes — `Molecule`, `Residue`, and `Atom` — work together to model a protein (or other molecule if `Residue` is not used), while retaining some of the biochemical data about it. The `Molecule` class is able to read and parse structure files in both the original PDB format and its newer Extensible Markup Language (XML) style. It can also write an XML file to be used later, perhaps when a molecule has been constructed or modified manually. If the `Molecule` represents a protein with multiple chains, it can select either a single chain or all of them together for modelling.

The docking procedure does not use the `Molecule` class directly; it docks two objects of the `Model` class. This is a collection of coloured `SolidParts` (typically `Balls`), which provide spatial classification functions (testing whether a point is inside, outside, or on the surface). A `Model` may be built from a `Molecule` by creating a `Ball` for each `Atom`, coloured according to chain. The `Model` class also provides a method for generating a Java3D [Java3D, 2008] `BranchGroup` object to render a view of the shapes being docked.



Post-docking result display, with full view of controls



Ligand selection: 2D & 3D view

Top results table & matrix listing

■ Red: Receptor, ■ Blue: Ligand.

Figure B.1: Original Java user interface for FFT docking program

Algorithm

The two molecules to be docked are loaded into the program from prepared data files and the corresponding `Models` constructed. As an alternative (primarily for initial testing), the user can manually build a shape by drawing coplanar spheres directly within the interface. The parameters for docking (grid size, scoring functions, molecular surface thickness, and number of angular arrangements to explore) are also adjustable. An object of the `Aligner` class is constructed with the required parameters to initiate the search.

The `Aligner` class takes copies of the geometric `Models` and uses them to produce the matrices of scores to be convolved. I implemented the algorithm in a function that takes a list of orientations (triples of axis rotation angles), the required scale and dimension of the scoring matrix for the ligand, a translation for each matrix relative to its shape's centre, and some parameters concerning the precise method and output required. The outline of the algorithm is given in Listing B.1.

B.1.3 Testing

Initially, all the matrix operation functions contained Java code acting directly on the data, with the exception of the (I)FFT which was in a separate `FDFT` class. Since the FFT code was translated from C source [Press *et al.*, 1992], I conducted an experiment to compare the relative execution speeds of the two languages. I wrote a simple program to forward and then inverse Fourier transform an array of random data, first in Java and then ported almost unchanged into C++. The program repeated the transform cycle ten times, with different data in the source array each time, and reported how many seconds the transforms took to execute. Table B.1 lists these results with the comparison between the two implementations: it shows that, as expected, the Java program is slower, taking around 23% longer to complete the task.

Upon closer inspection, it transpired that the values in the matrix returned after the inverse transformation did not always match the original data used, with errors of the order 10^3 or more. The C++ program did not exhibit this behaviour, whereas the Java code might run a single execution before a series of failures. Since the code was almost identical, it seemed unlikely that the error lay there, so to avoid possible confusion later (and in the light of the unfavourable timings) I abandoned the Java FFT code.

```
//Preparation...
Create A, a 3D lattice, of the given dimension adim
Create B, a 3D lattice, of the given dimension adim
For each element E in A do
  Begin
  Classify point E, offset by aOffset1, on the receptor's volume
  Set E to the receptor score corresponding to E's classification
  End
Apply the Fast Fourier Transform to A
Conjugate each element of A
Create TOPS, an array of scored poses

//Search the rotations...
Let DA equal 2*π/angleSteps
For each rotation R in angleSteps do
  Begin
  Orient the ligand using R
  For each element E in B do
    Begin
    Classify point E, offset by aOffset2, on the ligand's volume
    Set E to the ligand score corresponding to E's classification
    End
  Apply the Fast Fourier Transform to B
  Calculate C, the pointwise product of A and B
  Apply the Inverse Fast Fourier Transform to C
  Find PEAKS, the topcount highest-valued elements in C
  For each element E in PEAKS do
    Begin
    Create a pose, P, with E's value as its score
    Set P's rotation to R
    Set P's translation to E's point offset by aOffset2
    Append P to TOPS
    End
  End
Return TOPS
```

Listing B.1: FFT correlation alignment algorithm

	Java		C++	
	FFT	IFFT	FFT	IFFT
Run time (seconds)	4.15	4.90	3.70	3.67
(Combined)	9.05		7.37	
Comparison with other language	112%	134%	89%	75%
(Combined)	123%		81%	

Test used [see Appendix E]: 10× (I)FFT of 128^3 array on Eurymedon

Table B.1: Timing data for Java vs. C++ FFT execution

	Exterior	Surface	Interior
Ligand	0	1	1
Receptor	0	−15	1

Table B.2: Original scoring function for FFT algorithm

Grid dimension		32	32	64	64	128	128
Grid resolution	Å	1.8	1.8	1.2	1.2	0.6	0.6
Surface tolerance	Å	0.8	0.8	0.8	0.8	0.8	0.8
Angle step	°	40	20	72	40	72	40
Run time	s	60	380	110	590	1500	8190
Rank of correct alignment		>1000	>1000	26	60	18	91

Table B.3: Algorithm parameters and results for Java/C++ FFT implementation

To allow the existing `Aligner` class to use the C++ code for Fourier transforms, I changed all the matrix-handling functions to send instructions to another process, rather than using local data. Overall, the small overhead of inter-process communication did not noticeably slow the docking procedure, and did make the results reliable.

Results

I explored the parameters for the FFT algorithm to establish how to obtain useful results. In the original implementation, the scoring function was proposed as given in Table B.2. I used these unchanged to dock the turkey ovomucoid inhibitor into alpha-chymotrypsin (PDB code 1CHO); parameters and run times are shown in Table B.3.

The results indicate that the method works, but the parameters can make a significant difference to the run time and ordering of results. The test case used provided the protein and ligand in the docked conformation, therefore the correct result was the zero rotation and zero translation. This was easy to identify in the ranking, and was also guaranteed to be one of the arrangements tested by the search algorithm.

B.1.4 Extension

Since it is a straightforward addition, I added a refinement facility to the `Aligner` class. The refinement performed is simply a repeated run of the algorithm, but with a finer grid localized to a potential active site (determined from the results of an initial, coarser run). To enable this, two new functions were added:

```
public double[]
align ( int angleSteps, double angleHalfRange,
        int adim, double ascale,
        double finA, double finB,
        double fonA, double fonB,
        double foffA, double foffB,
        double atol, Point3d aOffset1, Point3d aOffset2,
        int topcount, boolean aspeed )
```

```
public double[]
refine ( double[][] seeds, int angSteps, double angDelta,
         int newDim, double newScale, double newTol,
         Point3d newOffset1, Point3d newOffset2, int topcount )
```

These both call the existing `process` function, having set up the parameters appropriately. The `refine` method's `seeds` parameter is an array of conformations to refine, where each entry is a 7-element array containing the result data returned from a previous alignment run. The refinement explores rotations of $\pm$`angDelta` around each axis from the seed conformation, using new spatial occupancy scoring grids centred at the new offsets. The user interface provides a second button to start a refinement from the results of any completed alignment. The highest-scoring results are used as seeds automatically.

B.2 Gradual Refinement Algorithm

As can be seen from the literature review (§2.3.2 (*p*.32)), there are many approaches to the docking problem. None of these are perfect, but it has been noted [Halperin *et al.*, 2002; Friesner *et al.*, 2004] that there may be much to be gained from using several in sequence or combination. For this reason, I transformed the docking code described above to fit a framework so that other algorithms might be used to refine its results.

B.2.1 Pipelined Architecture

I defined a Java interface for classes implementing docking algorithms. The `Pipelined` interface requires that classes expose methods providing:

- Lists of algorithm parameters (names, types, and default values),
- Descriptions of the values returned by the algorithm,
- A `run` function to initiate an execution of the algorithm, using the supplied parameters, callbacks, and `Environment`,
- A `result` function to create an `Environment` modelling some output data.

The `Environment` class is a container for two `Molecules` — the receptor and ligand — with methods for translating and rotating them. The parameters are all passed in an array, and the callbacks are references to objects implementing either `ProgressCallback` or `ResultCallback` which contract methods for updating a status display or recording reported results.

In addition, I defined an abstract `PipelineStage` class, providing a simple implementation of the `Pipelined` interface suitable for most purposes. This features a constructor which takes several parameters defining the characteristics of the algorithm; these are stored privately and are used to fulfil the interface requirements.

```
PipelineStage ( String name,
                String[] paramNames,
                Object[] paramDefs, Class[] paramTypes,
                String[] resultNames, String[] stageDataNames )
```

This `PipelineStage` class also defines some additional methods to offer a persistent `Environment` and parameter data. These may be set once and are recorded for future executions of the algorithm; parameters may then be changed individually rather than by supplying the complete set. Instantiable subclasses — those performing a docking procedure — need then only override the default zero-parameter constructor to call the new inherited one, and three other functions: `getLabel()` to identify the algorithm, `run()` to implement it, and `result(Object[] resultData)` to return the conformation associated with a result.

B.2.2 Development

I adapted the existing FFT-based algorithm to fit this interface: the `Aligner` class is mostly unchanged except for the addition of the three required methods.

To improve the conformations produced by the grid-based algorithm, which will typically only produce very close placements if the grids and rotation steps happen to coincide on the correct binding site precisely, I implemented a new algorithm in the `Pipelined` interface. This uses a gradient descent to pull the ligand towards the receptor incrementally, aiming to minimize the distances between pairs of surface atoms. This `Potential` class has parameters for the number of atom pairs to consider when calculating a motion step, the thickness of the surface layer from which to consider atoms,

```
Find LC, the ligand's centroid
Find RC, the receptor's centroid
Find C, the midpoint of LC and RC
Find R, half the separation of the centroids (plus a padding value)
Discard atoms not on the surface of either molecule
Discard any atoms not within R from C
Create the Cartesian product of the molecules' remaining atoms, PS
Create a Movement object, M
Let M's translation be an unspecified large value
Let Q be the supplied iteration limit
Let T be the supplied minimum translation length
Let MS be an empty list of Movements
While the magnitude of M's translation > T and Q > 0 do
  Begin
  Sort PS in decreasing order of magnitude using a potential function
  Find N, an integer based on a proportion or count of PS
  Let N' equal N Let V be the zero vector
  While N' > 0 do
    Begin
    Add the ligand-to-receptor vector represented by PS[N'] to V
    Decrement N'
    End
  Divide V by N
  Set M's rotation according to the moment of V about LC
  Set M's translation according to the magnitude of V
  Append M to MS
  Decrement Q
  End
Return MS
```

Listing B.2: Gradient descent pose refinement algorithm

the radius beyond which atoms will be ignored, and stopping conditions (minimum adjustment distance or maximum iteration count). Figure B.2 shows this new program executing: each adjustment made to the conformation is listed above the picture of the new arrangement. The algorithm used in this refinement is described in Listing B.2.

This code exhibits sensible behaviour, i.e. a ligand does not generally penetrate the receptor significantly, and settles (with oscillation) into a small area on the protein's surface.

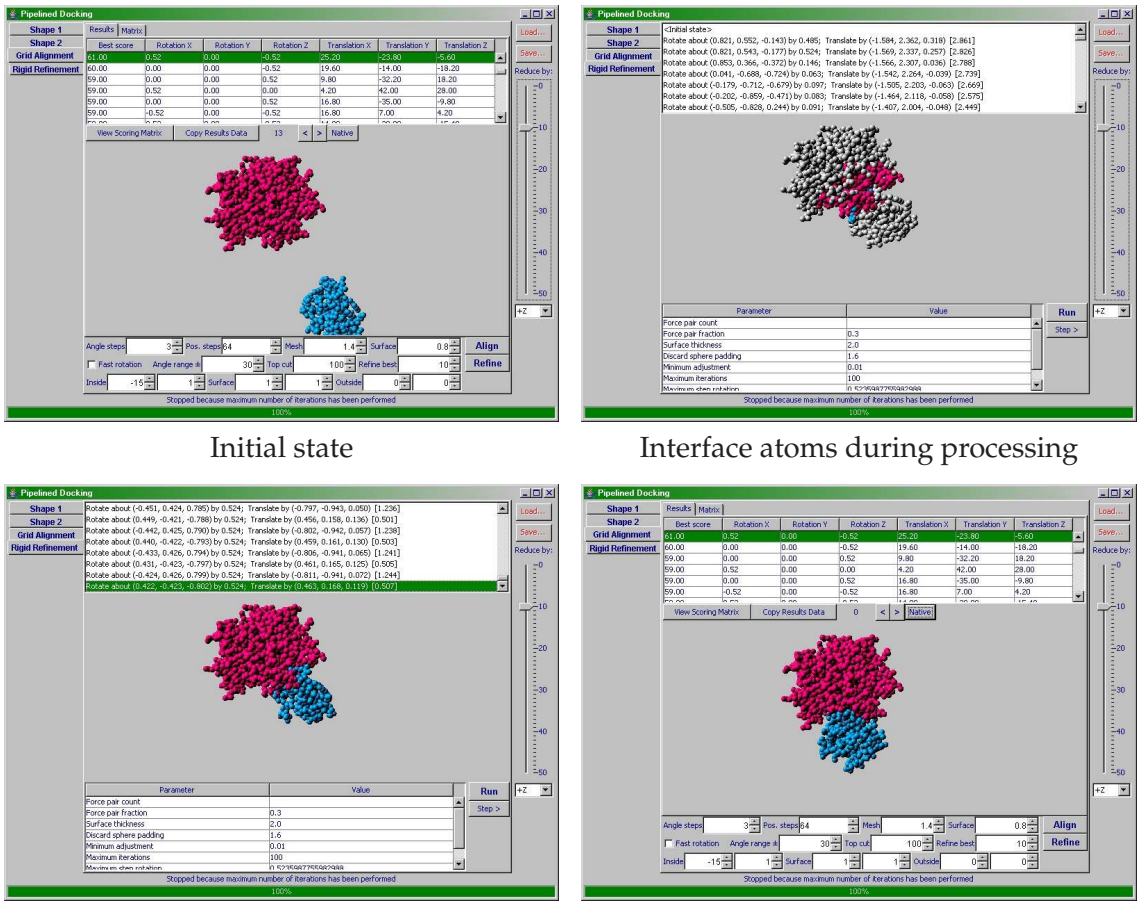


Figure B.2: Updated Java user interface for pipelined docking, showing gradient descent refinement in progress

Appendix C

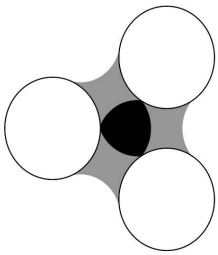
Dotty Surfaces

> Pythagoras as a mater of fact is at the root of all geom. Insted of growing grapes figs dates and other produce of greece Pythagoras aplied himself to triangles and learned some astounding things about them which hav been inflicted on boys ever since.
>
> *'Down With Skool!', Geoffrey Willans & Ronald Searle*

C.1 Motivation

One of the more complex components of the *XScore* function's computation is the calculation of molecular surfaces in the H_1 term (§3.5.1 (*p*.66)).

Methods for real-time construction and display of such surfaces have been established for more than two decades [Bash *et al.*, 1983]. These usually generate a set of points for painting. The points drawn by graphics procedures are not always truly even, especially around the intersection of atomic surfaces. Here, though, the requirement is for a mathematically even distribution of points. To simplify calculations of surface areas, such as the buried surface required by *XScore*, a Monte Carlo method could be used to sample points on surface patches and thus estimate a proportion of the total area.



□ White: Atomic spheres;
■ Grey: Torus holes precisely in contact with spheres;
■ Black: Concave spherical triangle faces of internal volume.

Figure C.1: Molecular volume components

C.2 Method

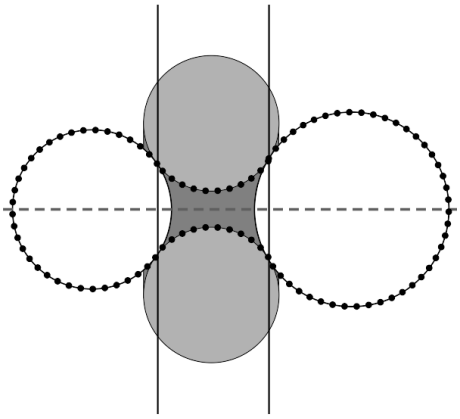
Molecular surface areas and volumes can, of course, be computed analytically as discussed in §2.1 (*p*.23). A common method takes a probe sphere and treats points that it cannot touch without overlapping an atom as being inside the molecule. Equations can be derived from the decomposition of a molecular volume into atomic spheres, the inner faces of tori interposing these spheres, and spherical triangles at the intersection of three such tori. Figure C.1 illustrates these three parts. Full mathematical definitions for the required terms are well-known [Connolly, 1985], while programs to calculate these equations in $\mathcal{O}(N \log N)$ time are available [Sanner *et al.*, 1995].

To be useful, a method such as this must be able to generate points at random with a uniform distribution over a sphere or inward-facing torus surface. Efficient formulae for these and similar distributions exist [Williamson, 1987], and these have been used to add a surface calculation facility to the *cSpheres* program (§3.2 (*p*.54)), as shown in Figure C.3. The displayed 3D plot includes the triangles whose edges are the axes of toroidal patches and whose faces mark areas closed by concave pits.

The algorithm begins by identifying pairs of atoms that are close enough to prevent a probe sphere (of given radius) passing between them. These pairs define edges in a graph of surfaces, which will later be used to form the tori for saddle-shaped patches of dots. Next, for each triple of edges that form a closed loop, a triangular surface is created if the probe sphere could not pass through the loop. The set of all these edges and triangles defines the pale shape visible in Figure C.3, which denotes the enclosed polyhedral volume between atom centres.

To generate the dots, spheres are created centred on the vertices of the surface graph. These have the same radii as the corresponding atoms. Similarly, tori are created for each edge in the graph, such that their boundaries exactly meet those of the spheres on either side. Finally, spheres are created for each triangular face in contact with the three vertex spheres.

Each of these shapes then has dots placed randomly over its exterior in quantity proportional to its total surface area, and then clipped by the planes defined by its intersections with adjacent shapes. For example, the torus between two spheres has two



■ Solid black lines: Clipping planes;
■ Dashed grey line: Torus axis;
■ Black dots: Retained surface regions;
□ White circles: Atomic spheres;
■ Light grey: Probe sphere locus;
■ Dark grey: Enclosed interior space.

Figure C.2: Clipping surfaces between atoms

clipping planes perpendicular to its axis, defined by the circles where each sphere meets the torus' surface (see Figure C.2).

In practice, many molecules produce triangular faces in their surface graphs which are buried within the true surface and so are inaccessible from the outside environment. To eliminate these, a gift-wrapping algorithm to identify reachable triangles has been investigated, but this was unreliable, in particular when very thin triangles occur. Consequently, this surface area work has not been incorporated into the *XScore* function calculation.

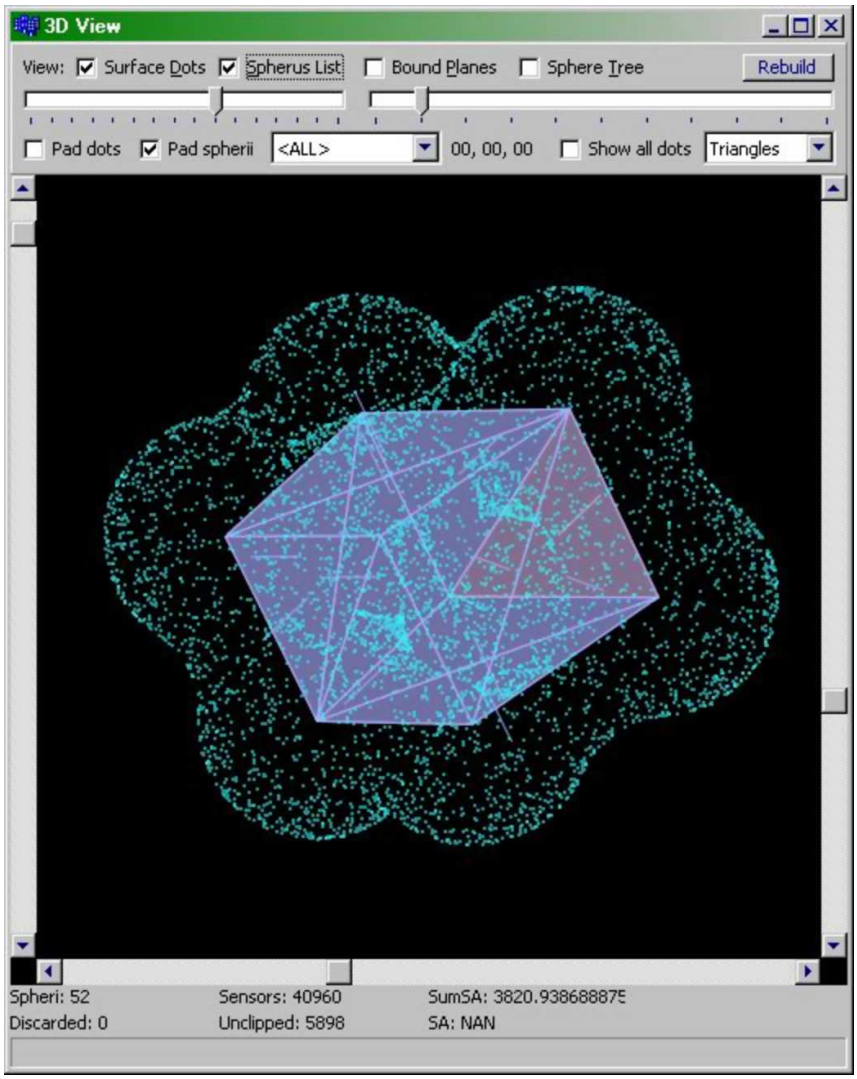


Figure C.3: Random surface dots in *cSpheres*, without interior removal

Appendix D

XScore Calibration

It's clearly a budget. It's got a lot of numbers in it.

George W. Bush

D.1 Training Cases and Function Data

These 128 protein-ligand complexes were used as examples to calculate the term weightings for the *XScore* implementation discussed in §3.5 (*p*.66) by linear regression. The pK_i values are the desired scoring function outputs from experimental data, sourced from the *XScore* definition in [Wang *et al.*, 2002].

PDB Code	pK_i	$\sum T$	$\sum H_1$	$\sum H_2$	$\sum H_3$	$\sum V$	$\sum B$
1A94	8.10	30	1856	213.40	8.18	1538.32	5.02
1AAQ	7.84	18	1567	160.13	5.84	1109.00	5.57
1ACJ	6.33	0	1012	139.76	2.72	684.78	0
1ADF	5.76	11	108	6.12	0.62	1210.88	1.98
1ANF	6.03	4	0	0	0	855.98	3.39
1APU	7.29	16	1640	140.20	5.77	1141.45	4.54
1APV	8.08	15	1535	116.70	5.70	1153.05	5.44
1BAI	7.86	30	1599	181.39	8.18	1565.78	4.38
1BLL	5.22	14	879	60.10	4.23	484.55	3.19
1BXQ	7.90	18	1722	172.12	5.84	1364.89	5.77
1CPS	6.15	4	670	105.89	2.51	592.90	0.35
1CSC	5.34	23	424	36.17	2.06	1108.92	4.87
1CTT	4.50	2	86	9.80	0.28	441.28	1.58
1D3Q	5.92	11	1530	175.47	6.59	880.19	0
1D3T	6.26	11	1407	146.60	5.56	933.83	0
1DBB	6.81	1	1196	152.64	5.03	721.07	0
1DBJ	7.23	0	1097	156.54	4.75	710.65	0
1DBK	6.83	0	1161	148.31	4.47	694.40	0
1DBM	6.65	6	1195	216.83	5.42	931.79	0
1DIH	6.38	13	226	41.43	0.82	1255.45	1.57
1DWB	4.36	1	445	39.10	1.54	337.41	2.36
1DWC	6.46	10	882	113.09	4.01	919.37	3.30
1DWD	7.13	10	1025	178.58	5.32	1074.66	2.92
1EED	8.02	19	2007	200.62	8.06	1372.64	4.40
1ELC	6.24	14	1472	178.22	6.70	1035.04	0.96
1EPO	8.56	17	1397	153.51	6.72	1414.18	6.45
1EPP	7.18	24	1826	178.60	7.02	1453.13	4.30
1ETT	6.58	7	1199	134.65	4.48	1088.09	1.55

PDB Code	pK_i	$\sum T$	$\sum H_1$	$\sum H_2$	$\sum H_3$	$\sum V$	$\sum B$
1FBC	5.80	6	0	0	0	449.61	2.28
1FBF	5.56	6	0	0	0	439.60	2.19
1FBP	4.85	4	0	0	0	503.05	1.12
1FQ4	7.64	16	1304	198.01	5.27	1352.35	2.53
1FQ5	10.56	10	2317	315.96	11.55	1747.50	5.42
1FQ6	7.68	16	1461	216.92	6.90	1163.18	2.06
1FQ8	7.86	17	1426	199.38	6.72	1385.91	1.35
1HBV	6.78	15	1641	195.72	6.52	1242.82	0.92
1HEW	5.07	13	482	26.84	1.54	791.18	2.17
1HPV	6.28	11	1453	165.52	4.40	1120.44	2.20
1HRI	4.95	9	715	141.79	2.68	794.75	0
1HTF	7.43	13	1653	235.19	6.69	1127.75	2.79
1HTG	9.17	18	2121	334.16	8.20	1649.73	2.95
1HVI	9.24	21	1693	270.54	7.46	1520.69	2.64
1HVJ	8.91	21	1733	270.95	7.85	1530.30	2.39
1HVK	9.58	21	1832	298.80	7.46	1633.27	3.31
1HVL	8.75	21	1700	275.27	7.46	1555.57	3.33
1HVS	8.74	21	1658	259.97	7.46	1547.22	3.02
1L82	3.38	5	0	0	0	164.09	0.99
1L83	5.04	0	726	149.68	1.69	267.38	0
1L86	3.43	5	0	0	0	165.64	0.90
1L87	3.35	5	0	0	0	118.09	0.92
1LDM	6.01	11	224	37.02	0.82	1426.62	4.05
1LGR	4.00	4	0	0	0	684.59	0.95
1LYB	8.72	23	1848	182.92	8.21	1312.29	5.51
1MCB	5.51	15	959	147.00	4.00	1140.39	1.21
1MCF	6.02	23	1066	179.54	4.78	1281.39	0.81
1MCH	5.64	23	1122	176.33	4.78	1357.17	2.57
1MCJ	5.29	10	960	129.80	3.22	894.36	0
1MCS	5.86	15	872	149.43	3.61	1209.51	1.47
1MDQ	6.14	4	0	0	0	874.79	4.46
1MFC	5.40	8	453	46.04	1.42	846.86	0.43
1MFE	4.76	6	290	32.80	0.90	734.34	0.17
1NNB	4.74	6	320	25.74	0.79	589.48	1.67
1PGP	4.29	7	0	0	0	465.00	1.74
1PPK	7.13	16	1652	121.41	5.25	1116.70	4.31
1PPL	8.01	17	1795	178.82	6.48	1334.97	5.11
1PPM	7.54	18	1513	162.04	5.33	1353.92	4.33
1PSO	9.03	23	1831	176.62	8.21	1371.09	5.74
1RNE	8.91	20	2157	259.57	9.66	1692.54	2.29
1RPA	4.04	3	0	0	0	241.88	1.32
1RUS	3.53	4	0	0	0	267.08	0.41
1SNC	5.33	6	358	45.22	0.98	547.70	0.75
1TMT	6.90	12	867	123.40	3.30	956.10	5.18
1ULB	3.89	0	0	0	0	409.71	2.15
1XLI	4.81	1	128	16.98	0.65	454.80	3.30
2DBL	7.53	6	1181	200.62	6.17	849.16	0
2DRI	6.41	0	0	0	0	430.17	3.04
2ER0	8.13	26	1945	227.13	8.84	1647.87	3.07
2ER6	7.64	29	1418	213.50	6.59	1918.27	5.25
2ER7	9.16	31	2441	254.96	9.19	2136.21	5.66
2ER9	8.09	26	2089	182.83	7.63	1801.25	3.49
2IFB	6.78	14	1580	232.96	5.97	671.14	0
2LDB	5.96	11	208	39.60	0.82	1164.25	3.54
2MCP	3.61	4	0	0	0	349.97	1.00
2MSB	2.60	15	0	0	0	481.91	2.42
2PHH	5.52	1	332	78.44	1.26	312.94	3.37

PDB Code	pK_i	$\sum T$	$\sum H_1$	$\sum H_2$	$\sum H_3$	$\sum V$	$\sum B$
2R04	6.58	10	1321	220.51	4.14	978.70	0
2RNT	5.27	8	0	0	0	986.50	3.50
2YPI	4.00	3	0	0	0	283.18	1.97
3CSC	5.42	22	547	43.71	2.58	1022.22	2.03
3ER3	8.13	24	2019	218.04	8.64	1736.75	5.25
3PGM	2.95	4	0	0	0	−72.23	5.29
3TS1	5.29	9	519	76.50	1.45	1110.47	3.43
4DFR	6.18	9	479	85.73	2.04	946.73	3.32
4ER1	9.29	20	2336	344.88	11.89	1737.29	2.37
4ER2	7.49	23	1944	151.43	7.31	1337.52	4.53
4ER4	8.02	30	2181	232.98	8.11	2029.98	5.30
4EST	6.42	14	1416	148.37	5.25	885.21	2.47
4FAB	6.93	2	842	146.12	2.66	696.04	1.27
4GR1	1.30	23	94	4.96	1.56	499.46	0
4HVP	8.20	29	1741	209.02	7.49	1482.30	5.40
4MDH	5.61	11	263	36.48	0.82	1368.87	1.24
4PHV	9.49	12	2210	380.17	7.68	1563.97	3.84
4TMN	7.45	14	1471	161.82	5.33	1087.29	5.93
4TS1	4.42	3	526	69.66	1.45	519.46	3.20
5ACN	3.35	4	219	24.49	0.90	397.66	0
5ENL	4.49	3	0	0	0	217.77	3.07
5ER2	8.80	28	2349	236.93	9.79	1938.23	7.30
5HVP	8.33	21	1811	149.10	7.18	1205.83	5.75
5ICD	4.72	4	9	6.99	0.51	311.67	2.08
5LDH	5.42	15	267	50.35	1.48	993.56	2.99
5TIM	3.76	0	0	0	0	133.16	0
5TMN	6.10	14	1205	149.82	4.62	993.50	2.52
5XIA	4.55	4	0	0	0	404.65	3.75
6ACN	3.34	4	213	25.31	0.90	378.02	0
6APR	7.97	23	1819	193.45	8.21	1177.62	4.50
6ENL	4.29	3	0	0	0	235.78	0.10
6TMN	5.91	14	1188	149.68	4.62	970.95	3.25
7ACN	6.08	4	98	23.97	0.51	523.26	2.67
7CAT	5.25	13	101	37.06	0.82	1164.06	2.98
7HVP	8.23	26	1670	210.56	7.65	1643.86	4.94
8ACN	5.76	4	105	14.77	0.39	488.32	2.42
8ATC	6.54	6	96	11.60	0.39	534.05	1.52
8CPA	7.00	13	1282	156.85	3.79	977.12	2.92
8HVP	6.93	28	1527	174.96	8.21	1480.40	5.46
8ICD	4.76	4	59	9.04	0.51	313.44	1.01
9HVP	8.72	20	2115	323.09	8.85	1546.87	2.70
9ICD	5.11	6	0	0	0	622.52	1.91
9RUB	3.14	8	8	5.27	−0.28	412.25	0.38
Correlation (R^2):		0.377	0.759	0.735	0.759	0.701	0.290
Gradient (m):		0.129	0.00201	0.0155	0.474	0.00303	0.519
Weighting (mR^2):		**0.049**	**0.00153**	**0.0114**	**0.360**	**0.00212**	**0.151**
Intercept (c):		4.713	4.406	4.460	4.471	3.370	4.948
Correlated intercept (cR^2):		1.775	3.344	3.278	3.392	2.361	1.437
				Mean correlated intercept:			**2.60**

Appendix E

Test Configurations

> "Is this a question?"
> "If it is, then this might be an answer."
>
> *From an Oxford entrance examination*

Test results presented in this thesis are labelled in the pattern:

Test used: N× *case* on *platform* **Ed** (*parameters*)

which indicates the number of executions (N) where multiple runs were averaged, the choice of target(s), the hardware platform, and program editions used. Unless otherwise specified in the text, all tests used a population size of 96 and 240 generations to search with the *XScore* function and return 100 result poses — the configurations shown in Listings E.1 and E.2. Local simplex optimizations (where used) terminated when either 500 cycles had been performed or the simplex's best and worst vertices' scores converged

```
Genetic
population           *16
generations          *40
mingens              *20
keeptop              0
keepevery            0
keepmax              *10
mutation             0.06
crossover            0.8
irregularcross       0.4
NMgenerations        3
NMmutation           0.5
NMcrossover          0.3
optimizeTol          0.002
optimizeSteps        500
optimizeRange        1.0
optimizeAngleRange   0.5
optimize             final
similarity           2
scoreSaveGens        0
scoreFlushGens       0
```

Listing E.1: GA search method configuration file

In the tests using edition **CA** in §5.2.5 (*p*.101), the following amendments are made:

```
generations          *20
narrowEvery          *2
narrowTo             2
optimizeSteps        1000
optimizeRange        2.0
optimizeAngleRange   1.0
```

In the multiple and larger search extents of those tests, a briefer GA search is used to find poses for a longer local optimization. The number of search boxes is narrowed until only two remain.

```
XScore
const      2.59758140
rotor      0.04855673
hydroS     0.00152764
hydroC     0.01140664
hydroM     0.35982438
vdW        0.00212340
hbond      0.15061649
abandonat  -1500
; Hydrophobic atoms:  These are defined as carbon atoms and halogens
; that do not match one of the specialized definitions below.
type       [#6]              HYDROPHOBIC
type       [#15]             HYDROPHOBIC
type       [#16]             HYDROPHOBIC
type       [F,Cl,Br,I]       HYDROPHOBIC
; Polar atoms:  O and N not matching DONOR or ACCEPTOR definitions.
type       [#7]              POLAR
type       [#8]              POLAR
; Polar atoms are also C atoms attached to heteroatoms.
; Here we define heteroatoms as N, O, P, S and halogen.
type       [#6][#7]          POLAR
type       [#6][#8]          POLAR
type       [#6][#15]         POLAR
type       [#6][#16]         POLAR
type       [#6][F,Cl,Br,I]   POLAR
; Polar atoms are also P or S attached to heteroatoms.
type       [#15][#7]         POLAR
type       [#15][#8]         POLAR
type       [#15][#15]        POLAR
type       [#15][#16]        POLAR
type       [#15][F,Cl,Br,I]  POLAR
type       [#16][#7]         POLAR
type       [#16][#8]         POLAR
type       [#16][#15]        POLAR
type       [#16][#16]        POLAR
type       [#16][F,Cl,Br,I]  POLAR
; Donor atoms:  These are N bearing a hydrogen.
type       [#7D1]-A          DONOR
type       [#7D1]-a          DONOR
type       [#7D1]=C          DONOR
type       [#7D2](-A)-A      DONOR
type       [#7D2](-A)-a      DONOR
type       [#7D2](-a)-a      DONOR
type       [nD2H1](:a):a     DONOR
; Zinc, a special but omitted case highlighted by 1AF2:
type       [#30]             DONOR
; Both atoms:  These are O atoms with a hydrogen.
type       [#8D1]-A          BOTH
type       [#8D1]-a          BOTH
type       [OX2H2]           BOTH
; Acceptor atoms:  These are N and O atoms with no hydrogens.
type       [#7D1]#C          ACCEPTOR
type       [nD2H0](:a):a     ACCEPTOR
type       [#8D1]=*          ACCEPTOR
type       [#8D1]=*[#8D1]    ACCEPTOR
type       [#8D1]*=[#8D1]    ACCEPTOR
type       [#8D2](A)A        ACCEPTOR
type       [#8D2](A)a        ACCEPTOR
type       [#8D2](a)a        ACCEPTOR
```

NB. The atom typing rules (lines beginning 'type') were prepared for me by Daniel Robinson (then of InhibOx) to the specifications of [Wang *et al.*, 2002].

Listing E.2: *XScore* scoring function configuration file

to within 0.2%. This appendix describes all the possible test cases, platforms, and edition strategies. A full list of all tests discussed in this thesis is given on page 245.

E.1 Molecule Test Cases

The tests are all re-dockings — searches for experimentally-determined complexes — to assess whether the correct pose could be found reliably by the docking procedure. Each result was compared with the true crystal structure pose using the root-mean-squared deviation (RMSD) of the ligand's atoms' centres: a value of 2Å RMSD is typically considered close enough to be regarded as successful.

The individual cases 1AF2 and 1K3U were used for docking an ensemble of conformations to a receptor. The *Astex* Diverse Set and *Astex* Mini Set provided collections of ligands for screening, in some instances using multiple conformations of each ligand.

E.1.1 1AF2

The PDB entry 1AF2 is taken from a comparative study of *XScore* by its authors [Wang *et al.*, 2003]. It models cytidine deaminase with its ligand uridine, as shown in Figure E.1. 53 conformations are present in the input ensemble, with the first being the correct crystal structure. This particular complex has been used for many behaviour tests because it is relatively small, making repeated testing tractable, and it has an open, exposed active site allowing much variation in plausible poses.

E.1.2 1K3U

The PDB entry 1K3U is taken from the *Astex* Diverse Set (see below). The receptor is tryptophan synthase and the ligand is the allosteric effector N-[1H-indol-3-yl-acetyl]aspartic acid. 101 conformations are present in the ligand ensemble, and 11 versions have been assembled with the crystal structure at positions 1, 11, 21, etc., respectively. This ligand binds at a completely enclosed site, as shown by the cut-away protein image in Figure E.2. This complex has been used for the quota-based rejection

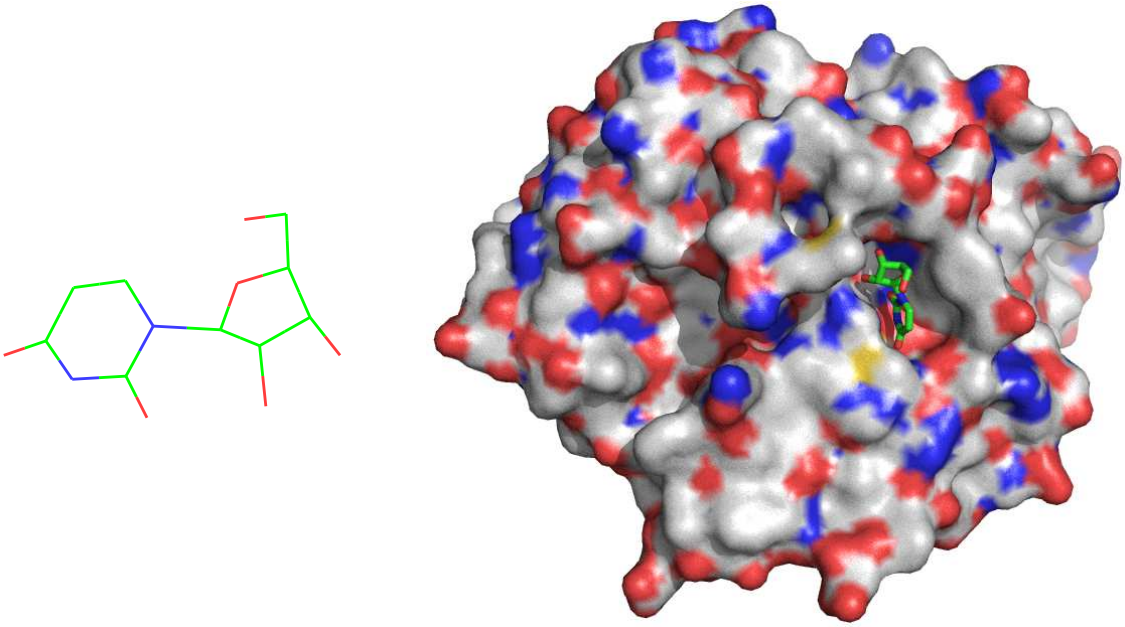


Figure E.1: Ligand and docked structure of 1AF2

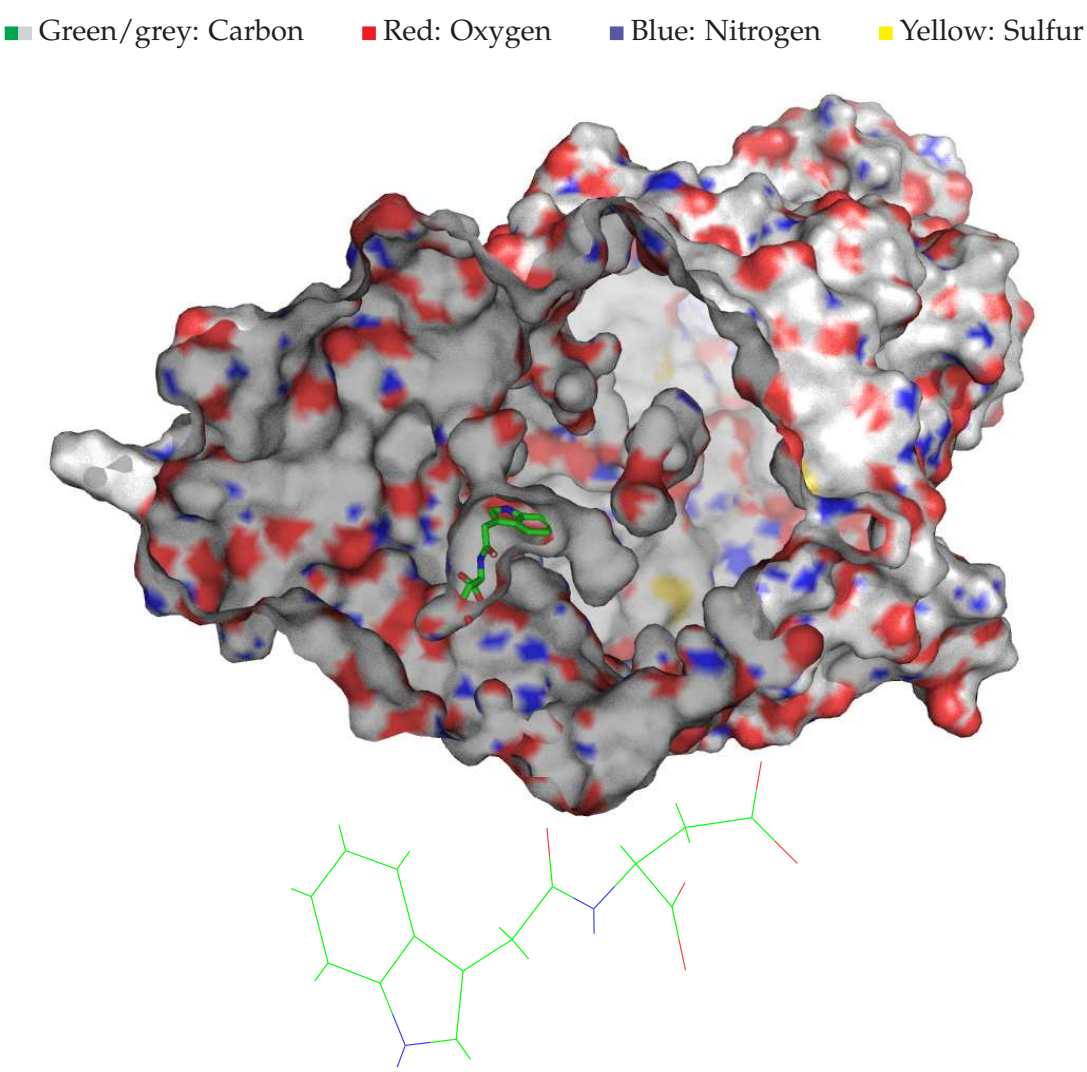


Figure E.2: Ligand and docked structure of 1K3U

PDB code	Conf's	HAC	PDB code	Conf's	HAC	PDB code	Conf's	HAC
1G9V	51	25	1N2V	28	15	1TT1	71	15
1GKC	201	22	1N46	8	27	1TZ8 ⋆	72	20
1GM8	201	24	1NAV	39	21	1U1C	51	20
1GPK	3	18	1OF1 ⋆	37	18	1U4D	3	18
1HNN ⋆	3	14	1OF6	37	13	1UML	101	33
1HP0 ⋆	49	19	1OPK	72	27	1UNL	401	26
1HQ2	4	14	1OQ5	25	26	1UOU	5	15
1HVY	201	32	1OWE	17	22	1V0P	401	29
1HWI	279	30	1OYT	70	30	1V48	51	22
1HWW ⋆	9	12	1P2Y	7	12	1V4S	31	23
1IA1	5	19	1P62	49	18	1VCJ	201	25
1IG3	27	18	1PMN	201	31	1W1P	2	11
1J3J	3	16	1Q1G ⋆	74	20	1W2G ⋆	49	17
1JD0	13	13	1Q41	1	21	1X8X ⋆	37	13
1JJE	201	25	1Q4G	25	17	1XM6	138	19
1JLA	51	27	1R1H	401	29	1XOQ ⋆	51	24
1K3U	101	21	1R55	401	23	1XOZ	7	29
1KE5	281	23	1R58	201	22	1Y6B	201	34
1KZK	401	41	1R9O	25	18	1YGC	201	38
1L2S	101	18	1S19	218	30	1YQY	401	23
1L7F	210	23	1S3V	191	27	1YV3	3	22
1LPZ	51	30	1SG0 ⋆	9	17	1YVF	101	27
1LRH ⋆	7	14	1SJ0	201	33	1YWR	401	35
1M2Z	65	28	1SQ5	101	15	1Z95	101	29
1MEH	101	23	1SQN	3	22	2BM2	101	30
1MMV	401	15	1T40	51	28	2BR1	101	29
1MZC	401	35	1T46	101	37	2BSM	51	26
1N1M ⋆	36	12	1T9B	136	22	**Total: 85**	**9596**	**1933**
1N2J	25	10	1TOW	51	19	All conf's:		**255621**

Conf's = number of conformations in ligand ensemble
Starred cases are the members of the mini-set (see text)

Table E.1: The *Astex* Diverse Set of receptor-ligand complexes

testing in §6.4.3 (*p*.137) because it has sufficiently many atoms to generate a reasonable number of conformations. The quota method required a larger range of input ensemble arrangements than would have been available with 1AF2.

E.1.3 The Astex Diverse Set

The Cambridge Crystallographic Data Centre and Astex Therapeutics collaborated on the development of a test suite of relevant and varied receptor-ligand complexes, originally for the assessment of *GOLD* [Verdonk *et al.*, 2003]. This *Astex* Diverse Set [Hartshorn *et al.*, 2007] contains 85 examples of pharmaceutical or agrochemical targets with drug-like ligands, all of which have fairly accurate structural data and are quite

Name	Processor	Clock	Cores	L2 Cache	RAM	OS
Eurymedon	Athlon64	2.20 GHz	1	0.5 MB	1 GB	Windows
Tom 8×	Opteron x64	1.95 GHz	2 dual = 4	2 MB	4 GB	Linux

RAM = random-access memory size, per machine. OS = operating system.

Table E.2: Hardware configurations used for testing

different from each other. The members of this set are listed in Table E.1, along with an indication of their ligands' sizes. The number of conformations of each ligand, generated using the online *FROG* tool [Bohme Leite *et al.*, 2007], is also shown. In each case, the correct crystal structure has been included as the first conformation.

E.1.4 The Astex Mini Set

I chose a subset of twelve cases from the *Astex* Diverse Set, to be used for quicker trials when multiple molecules are still necessary, referred to as the *Astex* Mini Set in this document. The complexes in this set are starred in Table E.1, and were chosen as those for which the basic *XScore* implementation in *DOX* edition P was able to find more than 5% of results within 1Å RMSD. They vary in size from 12 to 24 heavy atoms, and have ensembles ranging from 3 to 74 conformations, 433 in all. The total heavy atom count for all conformations, to give a relative scale of the docking task, is 7888 compared with 255621 for the full Diverse Set.

E.2 Hardware Platforms

Different hardware systems were used for certain test executions, depending on the practical requirements. Specifically, parallelized docking (§7.4 (*p*.153)) had to be trialled in a multi-processor cluster environment. The computers used are denoted by name, as listed in Table E.2. Eurymedon is a fairly typical desktop machine, operating as a 32-bit system, on which *DOX* is run as any other console program. Tom is the head node of a cluster with 7 additional working computation nodes, all of which have the same specification. These are rack-mounted servers with four logical processors each, connected via gigabit Ethernet and remotely-accessible from Tom. On both systems, *DOX* has been compiled as a statically-linked native executable.

E.3 DOX Editions: Strategies and Codes

The *DOX* system (introduced in §3.4 (*p*.64)) was used as an example with which to demonstrate the effect of the stratagems in various combinations. This was achieved by wrapping all new programming code in preprocessor directives that enabled or disabled the changes according to definitions in the compiler options, each one corresponding to a particular stratagem. Hence, a vast range of editions can be built as needed. Those editions that have been compiled and used for tests in this thesis are shown in Table E.3.

Some stratagems are dependent on others to work: these requirements are automatically enforced as follows:

Extensive local optimization	⇒	Local optimization
Predicted pocket placement	⇒	Quaternions *and* Learnable properties
Conformation prioritization	⇒	Learnable properties
Shape descriptors	⇒	Learnable properties
Learnable properties	⇒	Knowledge base storage
Jobs & parallelization	⇒	Knowledge base storage
Directed search	⇒	Quaternions *and* knowledge base storage

The editions featuring knowledge base storage had a substantial redesign of all inputs and components into a more flexible architecture. Consequently, some tests (such as the optimization frequencies in §6.1 (*p*.117)) had to be performed using the newer design, although the `K` edition is abstractly the same system as `P`.

Edition `q` was the original form of the *DOX* code before my investigations began. I created three cases (`x`, `B`, `i`) with features excluded in order to test their effects.

Edition		Quaternions §5.1	Spatial indexing §3.5.1	LUT interpolation §6.2	Local optimization §6.1	LUT caching §6.3	Atom prioritization §6.4.1	Scoring-based rejection §6.4.1	Pose-based rejection §6.4.2	Quota-based rejection §6.4.3	Knowledge base storage §7.1	Jobs & parallelization §7.4	Learnable properties §7.2	Shape descriptors §5.3	Conformation prioritization §7.2.1	Predicted pocket placement §5.2
		Stratagem Applications														
Subsets of existing code:																
x	*Nothing*															
B	*Basic*	■	■	■												
i	*No interpolation*	■	■		■											
q	*No quaternions*		■	■	■											
Function evaluation stratagems:																
P	*Optimize*	■	■	■	■											
C	*Cache*	■	■	■	■	■										
O	*Order atoms*	■	■	■	■		■									
R	*Rejection threshold*	■	■	■	■			■								
OR	*Order+Reject*	■	■	■	■		■	■								
M	*Merge*	■	■	■	■				■							
ORM	*OR+Merge*	■	■	■	■		■	■	■							
N	*Quota (number)*	■	■	■	■					■						
ORN	*OR+Quota*	■	■	■	■		■	■		■						
Job control and geometric properties:																
K	*Knowledge base*	■	■	■	■						■					
ORK	*OR+Knowledge base*	■	■	■	■		■	■			■					
J	*Jobs*	■	■	■	■						□	■				
S	*Learn shapes*	■	■	■	■						□		□	■		
F	*Prioritize jobs (first)*	■	■	■	■						□		□	■	■	
FN	*Prioritize+Quota*	■	■	■	■					■	□		□	■	■	
ORFN	*OR+FN*	■	■	■	■		■	■		■	□		□	■	■	
A	*Pre-align*	□	■	■	■						□		□	■		■
CA	*Cache+Align*	□	■	■	■	■					□		□	■		■
ORCA	*OR+CA*	□	■	■	■	■	■	■			□		□	■		■

Stratagems implicitly included as prerequisites of others are marked with hollow squares.

Table E.3: The editions of *DOX* and the stratagems enabled in each

Appendix F

Technical Implementation Details

> I like work: it fascinates me. I can sit and look at it for hours.
>
> *'Three Men in a Boat', Jerome K. Jerome*

This appendix presents a summary of the *OrthoDOX* software's architecture, as refined to test the stratagems. It was developed from an existing docking tool, called *DOXGA*, introduced in §3.4 (*p*.64). Figure F.1 shows the major classes and their relationships, coloured by the parts of the system to which they belong.

F.1 Operational Overview

DOX is a console program: it provides all its status displays on the standard output and error text streams, and once started requires no interaction from the user. All the necessary input is provided by the initial command line (as described by Table F.1) and some plain text files. A sample transcript from an execution is reproduced in Listing F.1, starting with the command line used and showing the output as appears on screen (minus some spacing).

The `ApplicationContext` class is used to create a static global object containing the objects needed to perform the docking task. It parses the program's command line to obtain values for the execution parameters, loads the knowledge bases, and reads the specified configuration files to construct and set up the necessary objects. The use of files allows related settings to be kept in logical groups for easy reuse. These files define:

- Search methods (`ISearchMethod`)
- Scoring functions (`IScoringFunction`)
- Search extents and pre-positioning (`PrePositions`)
- Prioritization functions (`IPriorityFunction`)
- Properties for learning (`Syllabus`)
- Managers for parallel processing (`ManagerLinks` or a `JobManager`)

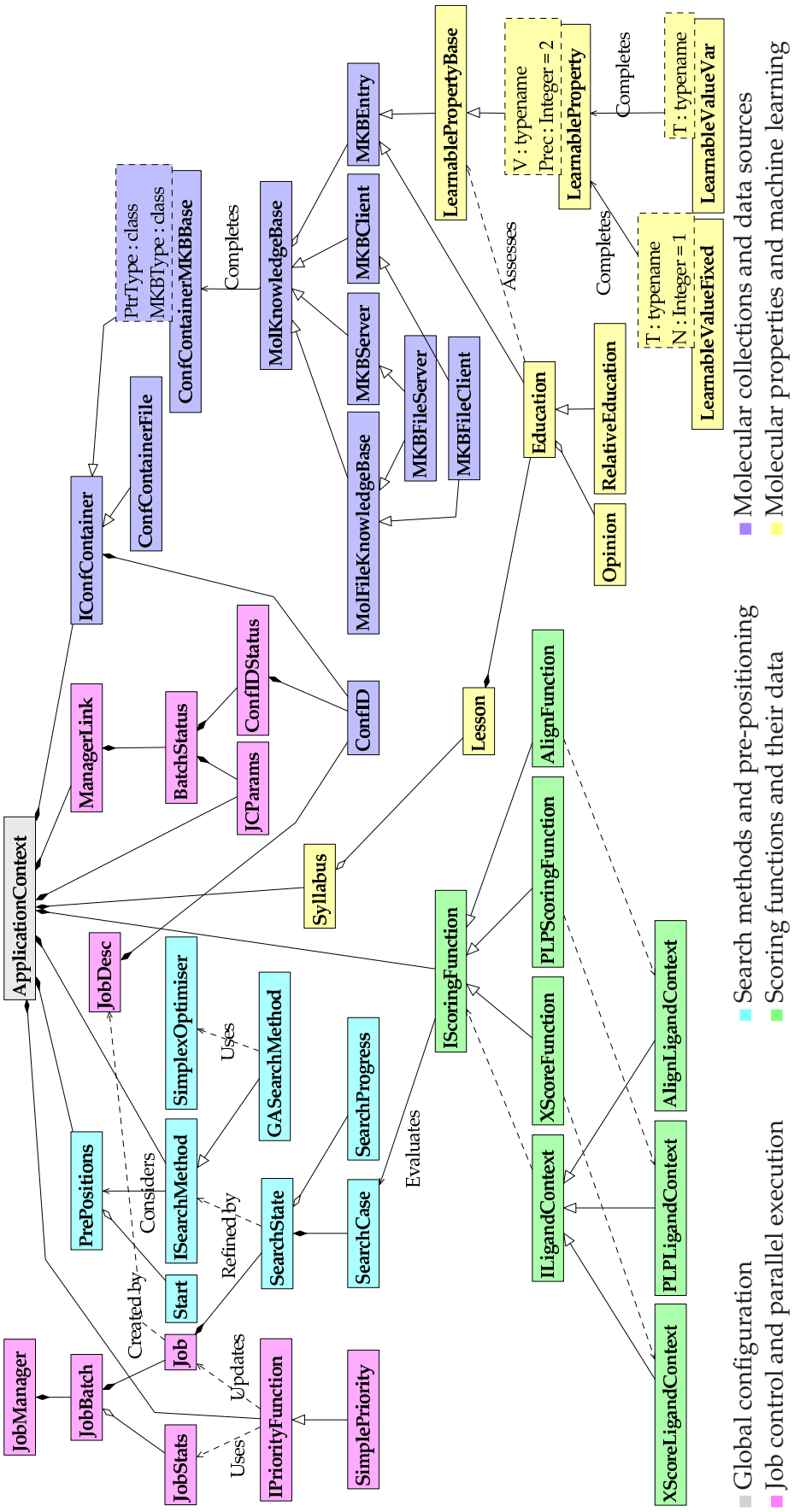


Figure F.1: UML overview of all major classes in the *DOX* system

Parameter	Default	Description
`jobconfig`	⋆	The configuration file name for the job control system, listing Managers and their ID numbers.
`id`	0	The unique positive ID for a Manager process. Specify zero or omit to run as the Controller.
`receptor`	⋆	Specifies the MKB of the master protein. Append a hash symbol and number to specify a particular conformation to use, e.g. '`mkbpath#3`'. The first conformation is assumed otherwise.
`points`	⋆	Specifies the path to a starting point file, listing points and optionally rotations to use as initial poses and range limits when searching.
`ligands`	⋆	Specifies the MKB of the ligand to be docked, or else a plain text file listing MKBs and conformation masks.
`output`	⋆	Specifies the SDF file in which to record the results.
`maxresults`	100	Specifies the maximum number of results to return. Setting this to zero ensures that all results are saved to the output.
`commonkb`		Specifies an MKB used for storing receptor- and ligand-independent data.
`function`	⋆	Specifies the scoring function configuration file to be used. This lists the name of the scoring function, and any parameters it requires.
`loadgrid`	true	Specifies whether to use a pre-calculated LUT. If this argument is false, the program recalculates all scoring function data.
`savegrid`	true	Specifies whether to save any new LUT data at the end of execution.
`method`		Specifies the search method configuration file to use for docking. If omitted, the ligand conformations supplied are scored without any translation/rotation.
`priority`		Specifies the prioritization configuration file to use for job control on the Managers. If not specified, the 'simple' priority is used, with its default parameters.
`stop`		Terminates Manager processes remotely. Ignores all but the `jobconfig` option, since no docking is performed once the Managers have ended. Specify zero as the argument to terminate all Managers, or an ID to terminate only one.
`auto`	0	Specifies the ID for a local Manager process to automatically spawn, use, and then terminate. This must appear in the job configuration file. The default value of zero does not create a local Manager.
`test`		Tests the communication between Controller and the Managers by attempting to connect, displaying the responses, but not running any further.

⋆ Starred parameters are required as a minimum configuration for docking.
When starting a Manager process, only `jobconfig` and a non-zero `id` are needed.

Table F.1: Command line parameters for *OrthoDOX* with summary descriptions.

```
orthodox --receptor rec-1lrh.mkb --ligands 1lrh --points empty.box
--output Results/test1lrh-ORK1.sdf --function xscore.sf --method ga.sm
Program execution started at: 6/11/09  2:12:26
==================
     OrthoDOX
==================
Version 1.3.0       (Stratagem development edition)
Copyright (c) 2007- InhibOx Ltd.
 Portions (c) 2008- The University of Oxford.
         Written by Gwyn Skone
               with Daniel Robinson
                and Romesh Ranawana
STRATAGEMS: LUTs (interpolation), Quaternions, Optimization, Indexing, Atom ordering,
Early rejection (thresholds)
> Processing command line options and loading inputs.
  Data path: D:/Molecules/Astex
  Loading receptor knowledge base (rec-1lrh.mkb).
  Loading ligand conformations (1lrh).
  Loading search space extents and points (empty.box).
  Preparing scoring function (xscore.sf) "XScore".
    Copying protein data...
    Identifying atom types...
  Preparing search method (ga.sm) "Genetic Algorithm".
> Preparing look-up tables.
  Loading data...
    Loading vdW function...
    Loading hydrogen bonding function...
    Loading hydrophobic potential function...
    Loading hydrophobic environment function...
    Loading spatial occupancy mesh...
> Evaluating database.  6/11/09  2:14:18
  7 conformations from 1 ligand.
    #      1  ( 0.0%)    0,  1   02:14:18   Got 96+0.   -8.7464    0.8600  *
    #      2  (14.3%)    0,  2   02:14:26   Got 93+0.   -8.6460    0.8510
    #      3  (28.6%)    0,  3   02:14:33   Got 95+0.   -8.5987    0.8467
    #      4  (42.9%)    0,  4   02:14:40   Got 94+0.   -8.5750    0.8446
    #      5  (57.1%)    0,  5   02:14:47   Got 90+0.   -8.5531    0.8426
    #      6  (71.4%)    0,  6   02:14:54   Got 93+0.   -8.4455    0.8329
    #      7  (85.7%)    0,  7   02:15:02   Got 94+0.   -8.4025    0.8291
    >        (100.0%)            02:15:09  Best score   -8.7464
                                             Normalized    0.8600
> Post processing.  6/11/09  2:15:09
  Writing results...
    Data table written to Results/test1lrh-ORK1-data.txt
    Search results summary: (where RMSDs were calculable)
      <=1A:  21  21.0%  (21.0%)
      <=2A:  62  62.0%  (83.0%)
      <=3A:  15  15.0%  (98.0%)
      <=4A:   2   2.0%  (100.0%)
      Mean: 1.52 angstroms from 100 cases
    100 results written to Results/test1lrh-ORK1.sdf
SF counters:  174521 0 0 0 3400 3461 7333 3384 2858 2306 2234 741 599 301 300 170 167
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Program execution started at  : 6/11/09  2:12:26  taking  112.609  0:01:52.61
Database evaluation started at: 6/11/09  2:14:18  taking   50.125  0:00:50.13
Database evaluation ended at  : 6/11/09  2:15:09
Program execution finished at : 6/11/09  2:15:09
```

Optional diagnostic/debugging outputs printed in grey.

Listing F.1: Transcript of *DOX* edition **ORK** output when redocking 1LRH

The search methods, scoring functions, and prioritization functions use abstract interfaces for interchangeable docking components. Their configuration files each have a name as the first line; the `ApplicationContext` translates this name by means of a factory function into an object of the appropriate subclass. Any subsequent lines in the files contain name-value pairs which are passed to that object for interpretation.

F.2 Ligand Conformation Containers

In older editions of *DOX*, ligands are loaded into a `ConfContainerFile` object from a multi-molecule SDF file. With the MKB design, more than one source can be used by listing their paths in a text file, and each ligand's conformations can be filtered to a specific subset. These are collected in a `ConfContainerMKBBase` object for use as one database.

The conformation container interface imposes methods for both sequential and random access of loaded molecules. The `FirstConf` and `NextConf` methods allow iteration, returning a `ConfID` structure for each case which specifies the ligand and conformation indices, and the absolute position in the entire list. Molecular structures (as *OpenBabel* `OBMol` objects) can be obtained using these data. If case prioritization is in use, the members of a container can be sorted in preference order. Each member is assessed using the loaded syllabus, and those ratings used to sort the list of indices for iteration.

F.3 Search Methods

Only one search method is currently implemented: the genetic algorithm. However, the search method interface is designed to support at least any population-based algorithm. It incorporates a `SearchState`, which is a collection of active and set-aside `SearchCase` objects with some overall status data. The search cases represent individual poses under consideration, with a score (from the scoring function) and flags to mark particularly good or bad cases. Both of these classes, as well as the overall `ISearchMethod`, provide non-specific data fields for associating the necessary structures for the particular

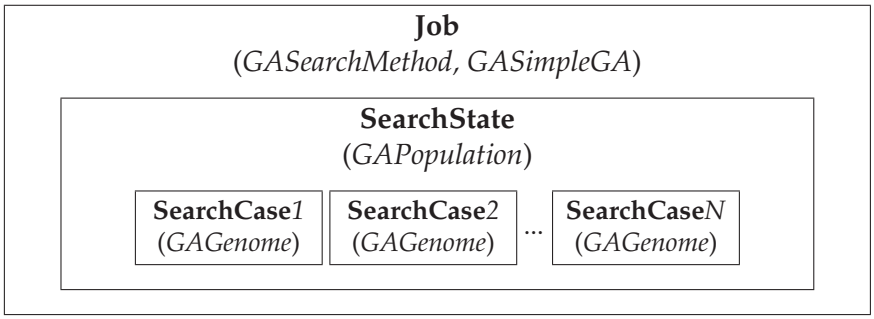


Figure F.2: Ownership relationship between classes used by search methods, and the corresponding GA classes used in their implementation

algorithm being performed. The implementations are required to provide several additional functions:

`Init` is called to prepare any associated data objects that might be required.
`InitState` should create and initialize `SearchCase` objects using the provided `PrePositions` object and add them to the provided `SearchState`.
`FreeState` can clean-up any associated data objects created for a search once it has terminated.
`Step` should perform some cycle of the search algorithm. In the GA, this is one generation, however it can be any procedure using the data in the search cases.
`Progress` must return an indication of whether the search has finished (from its termination conditions) or should be rejected, and how many results are currently available. In the GA, this is merely a generation count test.
`Results` should append the best poses found to a provided result list.

The `SearchState` belongs to a `Job` object (see §F.8 (*p*.217)) which provides the molecular data to be used, as illustrated in Figure F.2. It is the `Job` that calls the appropriate methods as necessary to drive the docking and collate the results with those from other conformations.

F.4 Pre-Positioning

The geometry of the search environment is defined using boxes: two points in space marking the minimum and maximum x, y, and z axis extents respectively. The pre-positioning configuration, parsed and stored in a `PrePositions` object, specifies the

```
reach x1 y1 z1 x2 y2 z2
box ( ( MKBpath | *L | *R | *C )#conf | sdFile )
reference ( ( MKBpath | *L | *R | *C )#conf | sdFile )
pad margin
scale factor
res resolution
overlap ( proportion | pc% )
try ( x y z | centre ) [ rX rY rZ ]
random [define] ( ratio | pc% )
place [box] ratio cases ReceptorProperty[*] [params]
if rating EducationSpec
align ratio cases similarity EducationSpec [params]
```

Keywords are in bold; square brackets mark optional parts; round brackets and vertical bars indicate a selection. Education specifications are listed in Table F.2 (*p*.216).

Listing F.2: Pre-positioning configuration options

box(es) for a search method to explore, and the size of any look-up tables should they be required. In addition, a list of `Start` objects is generated upon request, each representing a pose that can be used by a search method's `InitState` function to seed the initial candidates. If no search is requested (no `method` option is given on the command line), then only the `Start` poses will be scored and output.

The pre-positioning configuration is a list of command-like lines of text, each matching one of the patterns in Listing F.2. Some entries are only recognized by certain editions of *DOX*.

`Reach` entries specify the two points that define a box to add to the search extents. `Box` entries use either an MKB conformation or an individual molecule file to define a box; the MKB name can be `*L`, `*R`, or `*C` to use the ligand, receptor, or common knowledge base respectively. `Pad` and `scale` add to or multiply the dimensions of the hull of all search boxes to define the LUT sizes. `Res` defines the resolution to use if LUTs are calculated. A `reference` entry can be used to load a molecule with which to calculate result RMSDs, but without allowing it to affect the search. The first `box` entry is used as a reference if no others are provided. An `overlap` sets the proportional intersection (from 0 to 1, and 0.125 by default) above which two boxes will be merged to reduce the search complexity. This is particularly relevant when using multiple `place` entries, since at least some of their suggestions should match.

`Try` entries specify translations, and optionally rotations, to include in the list of `Start` poses. `Try centre` uses the midpoint of the search extents. A `random` entry controls the

minimum proportion of search cases that should be initialized at random, without pre-alignment. It may specify either a ratio or a percentage; it defaults to 100% when no `align` or `place` lines are present and 0% otherwise. If `define` is specified, then the pre-positioning system explicitly generates random `Start` poses, otherwise their selection is left to the search method in use.

`Place` entries use a receptor property to generate `Start` poses. If the `box` flag is included, they also define additional search boxes. These are typically active site prediction methods, and are normally ligand-independent. Appending an asterisk indicates that the property is independent of the receptor conformation, although this is unlikely to be the case. A limit on how many poses should be generated is specified as a ratio relative to other sources, along with the number of places to consider if the property predicts multiple pockets. Parameters can be added to the end of the line if required by the method chosen. The line `place 1 4 PlacePIES-5-3-7` will place a single share of poses in the top 4 pockets identified by the Place*PIES*-5-3-7 receptor property.

`If` entries may be included to control which pre-alignment properties are used: a minimum opinion rating for the ligand's specified property must be met if any subsequent `align` entries are to be used for that case. These conditions are not cumulative, and any `align` entries before an `if` line are always used. `Align` entries specify the education data (see §F.7 (*p*.214)) to use when generating `Start` poses. The cases and similarity values are used to select the set of opinions from the knowledge base on which the poses will be based. Again, the proportion of all poses to generate, and parameters for their creation, may be specified in the line. The line `align 2 3 0.4 AlignUSR=*` will use the top 3 previous Align*USR* values with a similarity of at least 0.4 to the new ligand conformation's Align*USR* value to generate a double helping of poses.

F.5 Scoring Functions and Ligand Contexts

The scoring function superclass contracts several functions from its implementations:

`GetSFcomplexity` returns a number indicating an approximate relative complexity, compared with a single smooth interaction potential, that may be used to adjust search parameters accordingly. For example, *PLP* returns 2 but *XScore* returns 6.

`CreateLigandContext` must construct and return an empty `ILigandContext` object of the type used by the scoring function.
`DoCalculateScore` is the scoring function itself: a ligand context is provided as the argument, and its floating-point evaluation should be returned.
`NormalizeScore` can take a score and its ligand context and adjust it onto the range $[-1, +1]$ representing the worst and best possible values. If not implemented, scoring function ranges are automatically used, as described in §7.1.1 (*p*.146).
`HighIsGood` returns a Boolean value indicating whether to maximize the scoring function in the search.
`Calc` should generate any LUTs required with the dimensions provided as arguments.
`LoadMKB` should read the relevant LUT entries from the appropriate MKB.
`SaveMKB` should commit any changes to the LUTs back to the knowledge bases.

I adapted the existing *PLP* code to fit this interface — a fairly trivial problem — and implemented *XScore* for it also. In addition, I created an alignment scoring function, `AlignFunction`, which expects to be given a receptor with the same number of atoms as the ligands docked, and scores poses simply using their RMSD. This allows the docking system to be used for superposition by RMSD minimization, a utility briefly helpful for some early test case analysis.

Ligand contexts are wrappers around an `OBMol` object that handle all pose transformations internally. They store the atom positions for both the original source data and a current translation and rotation, and provide `setTransform` methods to allow a search method to update the pose. The atomic centres are listed in an `AtomPositionContainer` object, which may be ordered or unordered depending on whether atom prioritization (§6.4.1 (*p*.129)) has been enabled. In either case, the external behaviour of the class is the same; the order in which atomic vectors will be presented for iteration is undefined for the rest of the system.

For each scoring function, a subclass of `ILigandContext` should be defined, supplying the `prep_ligand` method to perform any necessary processing on ligands, such as stripping or adding hydrogens or identifying and assigning atom types. Thus, to evaluate a pose, the ligand context's `setTransform` function is called to specify the geometry, and then that object is passed to the scoring function's `CalculateScore` method.

F.6 Molecular Knowledge Bases

The molecular knowledge base design requires client and server interfaces. Both of these must be able to store and retrieve multiple conformations associated with integer identifiers. The client contracts methods to retrieve entries from and record them back to the knowledge base, calculating them upon request if they are not available. It also collects recently-loaded entries for quick access should they be requested again. The server only has to process new data: it must identify newly-committed entries and store them in the knowledge base, resolving any clashing updates if necessary.

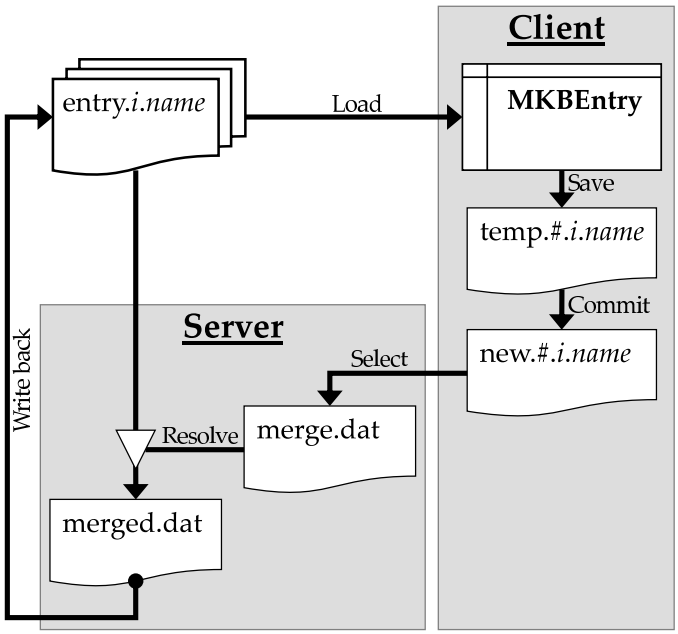
MKB entries are accessed by name and conformation number (zero if they apply to the ligand in general, not any particular shape). They also have an internal revision number, incremented by the server whenever a new version is committed, to identify any clashes between multiple clients' edits. The standard base class, `MKBEntry`, handles all the semantics of interacting with the client or server. Each possible entry type (molecular properties, scoring function LUTs, etc.) must complete four additional functions:

`IsPersistent` returns a Boolean indicating whether the value should be recorded in the MKB, or recalculated whenever it is requested.
`Calc` is the calculation function which should retrieve the molecule conformation and generate the appropriate data for the entry.
`Load/Save` should read/write the entry's value from/to the provided data stream (for transfer with the data repository). These may do nothing if `IsPersistent` returns false.

In addition, a static `EntryFactory` object must also be defined to convert the entry name strings to the appropriate `MKBEntry` classes for instantiation.

F.6.1 File-Based Implementation

I used a basic file structure to keep the implementation simple. Figure F.3 illustrates the files used in the design: a file-based MKB is a single file-system directory, and all data are stored together in that location. It might be preferred in some situations to use a database management system for the back-end storage, and this could be transparently switched at a later stage.



Filenames: # denotes the client's ID; *i* is the entry's associated conformation index.

Figure F.3: File-based implementation of molecular knowledge bases, showing the stages involved in the entry update cycle

The molecular conformations are recorded as a conventional MOL2 file. I chose this in preference to the SDF because it permits more than 1000 atoms per structure, a necessary feature for use with receptors. The conformation indices are assigned sequentially by default, but standard MOL2 format comment lines can be used to override the numbering.

Data entries are stored in individual files, named using the entry name and conformation index. When a client commits an entry, it writes the new value to a file prefixed 'temp' first, and when this is complete it is renamed to a prefix of 'new'. These filenames include a number unique to the particular client, so that multiple clients could update the same entry concurrently. The server periodically checks for these new data files. When one is available, its name is parsed to identify the entry being updated, and then it is renamed to 'merge.dat' to avoid further updates from the same client colliding with the resolution process. The revision numbers of the current entry file and the new data are compared to check for conflicting updates. If they differ, and a resolver function is provided by the entry type's `EntryFactory`, then that function is given both versions of the data to merge. Otherwise, the new data is used as-is. The new value is written to a 'merged.dat' file, and when this is complete it replaces the proper entry. The server then calls a notification function so that the change can be reported back to clients.

F.6.2 Ideogen: Standalone Server Utility

Since the existing *DOX* programs are not expected to be their own MKB servers, I created a simple program to provide that functionality on any computer. This program, *Ideogen*, also acted as a test implementation for development, including a utility for converting existing LUT data rather than recalculating it all. It requires an MKB path as a parameter to start, and then provides a command-driven interface to create and manage that knowledge base. A transcript of a typical execution is shown in Listing F.3; at the end of the listing the program continues to run, periodically checking for newly-committed entries until interrupted.

F.7 Learnable Properties and Syllabi

Learnable properties, a class of MKB entry, may have one or more values. These must be either `LearnableValueFixed` or `LearnableValueVar` objects, depending on whether their length is constant. If the values' elements are not a fundamental data type, a templated function is required to provide conversions of individual elements to and from strings.

The superclass contains the remaining logic needed to encode each value as a single string, and decode such a string back into an approximation of the original data. This may be an imperfect conversion, since decoding is required only for estimating the similarity of a dissimilar value. The `Load` and `Save` methods from `MKBEntry` simply read and write these strings, but can be overridden to record additional precision or supporting data. The individual properties are still required to provide a `Calc` function which should construct the appropriate value objects and pass them to the `AddValue` method. They must also define a `Similarity` method which, for a given value of the same type, should return a real number in $[0, 1]$ to be used when estimating suitabilities of unfamiliar values.

Since geometric properties, particularly those concerned with alignment, will change their values according to a ligand's pose, the `Docked` method is invoked whenever post-docking values are required. It is given the updated ligand context, and provides an opportunity for the property to revise its values accordingly. The `LearnableValue...` classes can contain both pre- and post-docking values, but only the undocked case is

```
ideogen rec-1af2.mkb

IDEOGEN: rec-1af2.mkb
  Opening knowledge base...
  Reading MKB conformations...  1
::help
  COMMANDS:
  - HELP
  - MINIO [OFF]
  - TITLE [newtitle]
  - MERGE [mergefile]
  - DROP [mergefile]
  - CLEAN
  - START [checkdelay]
  - EXIT
  Conformations:
  - SETMOL filename
  - ADDMOL filename
  - MOLCOUNT
  - MOLTOTAL
  - SORTMOL confindex ordering
  - PRINTMOL confindex [format]
  - EXPORT filename [format [confindex]]
  Entries:
  - LIST
  - ADD entryname confindex filename
  - HAS entryname [confindex]
  - PRINT entryname confindex
  Utility:
  - GRIDS scoringfunction gridfile
  - OLDGRIDS scoringfunction gridfile
::list
Conf.#  Rev.  Name
        2     education.AlignPASTRY
        2     education.AlignUSR
        48    sfRange.XScore
        6     XScoreLUThbond
        6     XScoreLUThydenv
        6     XScoreLUThydpot
        6     XScoreLUTvdw
        6     XScoreSpatOcc
1       3     PrePosSuggestions
9 entries.
::clean
  Cleaning up MKB...
    Discarding MKB entry education.AlignPASTRY from ID 15944...  succeeded.
    Discarding MKB entry education.AlignUSR from ID 15944...  succeeded.
    Discarding MKB entry sfRange.XScore from ID 16173...  succeeded.
    Deleting temporary file temp.26888.0.XScoreLUTvdw...
    Deleting merge file...
    Deleting resolution file...
  Done.
::start
  Monitoring MKB for committed data (^C to stop)...
    Merging MKB entry sfRange.XScore...  storing new entry...  succeeded.
```

Listing F.3: Example transcript of *Ideogen* MKB server program

Kind	Specification Pattern	MKB	Purpose
Specific	`ligProp[*]=[*]`	Receptor	Learns about ligands that dock well to a particular receptor.
Common	`ligProp[*]=+`	Common	Learns about ligands that dock well in general.
Relative	`ligProp[*]=recProp[*]`	Common	Learns about the relationship between a receptor property and ligand property. The opinions contain pairs of values, one from each property.

Square brackets mark optional parts. Bold symbols are literal. Asterisk suffixes indicate conformation-independence of the properties and/or learning.

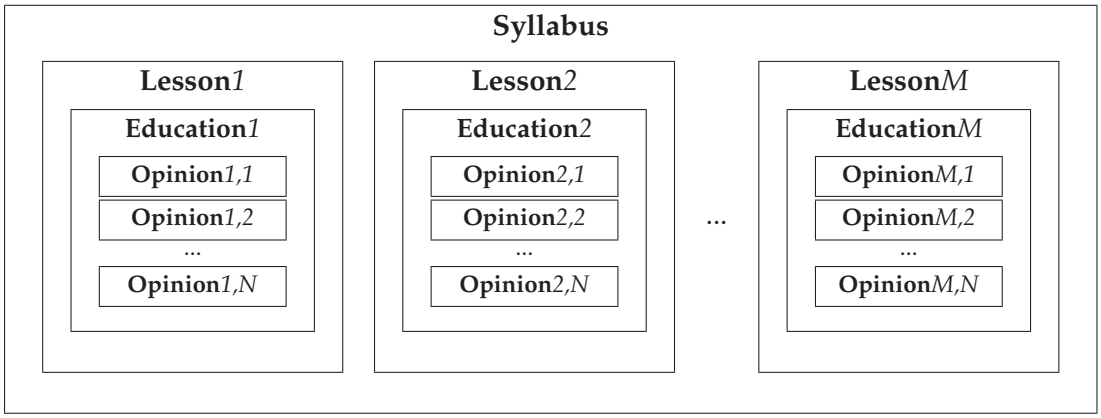
Table F.2: Education kinds and specification string patterns for use in syllabus files and pre-positioning '`align`' entries

recorded in the knowledge base since this is the one that matches the stored source structure.

Learnable properties also have `GeneratePoses` and `GenerateBoxes` methods, which do nothing in the base class but are designed to support the pre-positioning system (§F.4 (*p*.208)). `GenerateBoxes` can return a list of boxes in response to a '`place`' entry. `GeneratePoses` receives a ligand context and a number of poses requested by an '`align`' entry, and can return a list of `Start` objects with suggested alignments to try.

The learning mechanism collects good and bad poses from docking searches by associating property values with normalized scores in `Opinion` structures, and then accumulating these in an `Education` MKB entry. Each `Education` object is the set of all such information about a particular learnable property's past successes. Three kinds of Education are possible, distinguished by their names and the knowledge base in which they are stored. Table F.2 lists these kinds, together with their corresponding string patterns used for loading `Education` entries in a syllabus (see below).

The `Education` class provides an `Opine` function to obtain an `Opinion` for a given property value (or values, in the case of a relative education). If the case has not been seen before, an assessment is calculated based on comparable values using their `Similarity` methods for a weighted interpolation. The `Teach` function takes a post-docking `LearnableProperty` object and its normalized score, and creates or amends the appropriate `Opinion` record accordingly. The `Guess` function obtains the top-rated opinions from an `Education`, optionally subject to a minimum similarity to a given property value.



Lessons also hold the property names to use when obtaining new values to learn.

Figure F.4: Containment relationships between the classes used by the learning system

F.7.1 Syllabus Configuration and Use

As illustrated by Figure F.4, a `Syllabus` object contains a list of `Lesson` objects, each of which refers to a particular `Education` and controls how it should be used for assessing and learning from properties. Each line of the `Syllabus` configuration contains a education specification string (see Table F.2), and also provides weighting, attenuation, and minimum significance values for the resulting `Lesson`.

To learn from a docking result, its ligand context is passed to the `Syllabus` object's `Learn` method, which passes it in turn to each `Lesson`. If its normalized score's absolute value meets the minimum significance level, then the necessary `LearnableProperty` entries are obtained and passed to the education's `Teach` method with the attenuated score. An attenuation parameter of a scales the nth best score from a set of c results by $\max\left(0,\ 1 - a\left(\frac{n-1}{c-1}\right)\right)$. Consequently, if $a > 1$ then only the top $\frac{1}{a}$ of the cases will be considered.

For the prioritization of ligand conformations, each is passed to the `Assess` method of the `Syllabus` for evaluation. This calculates a weighted average of the `Opinion` scores obtained from each `Lesson` using its `Education`. Those values are used to impose an ordering on the cases in the conformation container.

F.8 Job Control and Parallel Execution

The job control design takes a complete docking task — its receptor, ligand, search method, and scoring function — and encapsulates it in a single object. This `Job` provides

only a few essential public methods. It is constructed using a job descriptor structure and an `ISearchMethod` object. The job descriptor provides the `ConfID` of the ligand, as obtained from a conformation container, and an appropriate ligand context to use. The search method will already have references to the scoring function and receptor.

The `Job` class provides a `Resume` method which causes the search to be progressed some number of steps, determined by the job's priority. This returns a Boolean indicating whether the search still has more work to do. The first time `Resume` is called on a particular object, the `Job` allocates a `SearchState` and passes it to the search method for initialization with `SearchCase` objects. The `Job` also provides methods to retrieve the search results, both intermediate and confirmed.

The priority can be obtained and changed using `GetPriority` and `SetPriority`: the floating-point positive or negative infinity values immediately finish or reject the search respectively. A `GetProgress` method returns a structure indicating whether the `Job` has been started, finished, or rejected yet, and whether the search method considers its state rejectable (in the GA, this is when a certain number of generations have completed). An `IPriorityFunction` may use this progress data as part of its implementation, which must provide two methods:

`Prioritize` should consider a `Job`, alongside some additional statistics about all jobs, and return a new floating-point priority. Again, the special $\pm\infty$ values can be used to end a job immediately with or without rejection.
`Start` receives only the overall statistics, and must return a Boolean value indicating whether another `Job` should be started. Note that this function must not always return false, otherwise execution will deadlock.

The `SimplePriority` function takes the best score in the search state, normalizes it, and adds it to the current `Job` priority (with a capping value). Its `Start` method simply returns true whenever fewer than a certain number of jobs are active.

F.8.1 Distributed Job Processing

There are two modes in which the parallelized *OrthoDOX* system can run:

Controller: invoked by a user with all the usual docking parameters, and outwardly equivalent to the single-processor version.

Manager: one of many instances silently running as a docking processor node, awaiting instructions from a Controller to load and execute `Jobs`.

A job configuration file is used to list the Managers available in an environment with their unique integer identifiers. Each is also given a relative load capacity, so that faster computers can be allocated a greater share of docking tasks automatically.

The Controller program does not perform any docking, but is responsible for reconciling data updates produced by the Managers as they work. Hence, it uses `MKBServer` objects to access the molecules being studied. As its `ApplicationContext` parses the docking parameters, it records them in a `JCParams` structure. The job configuration file is then used to create a `ManagerLink` object for each Manager that should be available — an internal representation of the state and behaviour of the remote process through which all communication is transparently handled. Once a connection is established with the Managers, those that are responding are sent the `JCParams` data and a list of ligand conformations to dock. The Controller then monitors the status reports being returned, merging new MKB entries as appropriate, until all the conformations have been processed. If one Manager finishes ahead of the others, the Controller will give it a number of additional jobs taken from another Manager, thus dynamically balancing the workload. Finally, it combines all the results received into the final output files.

The Managers are started using no command line parameters other than the job configuration filename and a Manager ID. The `ApplicationContext` is supplemented by a single `JobManager` class, which processes any instructions received from a Controller to create and manage `JobBatch` objects. Each set of jobs received is treated as a batch so that multiple Controllers might be able to use the same set of Managers concurrently (although this is not yet fully implemented). To start work, a `JobBatch` parses the `JCParams` data received, loads the appropriate MKB clients, sets up all the objects previously handled by the `ApplicationContext` alone (see §F.1 (*p*.203)), and constructs the individual `Job` objects for the ligand conformations. These are then sequentially started and resumed, with status reports and results sent back when appropriate. The batch is not unloaded, even when completed, until requested by the Controller. Managers do not terminate themselves unless a break signal (Ctrl+C) is received at the console or a Controller is invoked with the `stop` command line parameter.

hello	Requests acknowledgement to check whether the Manager is live and communicating successfully.
reset	Instructs the Manager to drop all unstarted and reject all active jobs, clear all receptor data, and zero all statistics. Generally sent when the Controller starts up, after `hello`.
chatter [**on**]	Enables or disables extra status reports from the Manager. Normally only job state transitions are reported — starting, finishing, etc. — but after `chatter on` all cycles are reported with the new job priorities.
mkbchange `entry/conf kb`	Notifies the Manager of a changed entry in an MKB. The string `kb` is the MKB's relative path.
start `batchID`	Indicates that a new conformation batch has been created, and instructs the Manager to load the appropriate input file and start docking.
add `batchID filename`	Specifies that batch `batchID` should be extended using the conformations listed in the given file.
report [`jobID`]	Requests a status report. If `jobID` is omitted, requests a summary of statistics for each batch; if a job ID is given, then only the status and priority of that job is required.
finish [`jobID`]	Instructs the Manager to immediately treat the specified job as finished, recording the best available results and stopping its search. If no job ID is given, all active jobs should be finished.
reject [`jobID`]	Instructs the Manager to immediately reject the specified job, stopping its search without recording results. If no job ID is given, all active jobs should be rejected.
drop `count` [`batchID`]	Requests that the Manager should relinquish up to `count` unstarted jobs, either from the specified batch or any if none is given. The dropped jobs must be reported to the Controller.
worst `score batchID`	Informs the Manager that only results with a score of at least `score` will be retained in the output file, because the quota of results has been reached and this is the worst of those.
stop	Requests that the Manager process should terminate. After this, any further instructions may not be received.

Table F.3: Instructions that a Controller can send to Managers

Communication Protocol

I wanted to make the communication between the Controller and Managers simple, traceable, and cross-platform compatible. Since the file-based MKBs must be accessible to all Managers, it can be assumed that some file-system directory will be available for shared access.

hello	Acknowledges a received `hello` instruction, or indicates readiness to receive.
started `jobID`	Reports an unstarted job taking its first search step. It cannot be dropped after this point.
progress `steps priority jobID`	Reports a job's status: number of steps executed and current priority.
finished `outFile best jobID`	Reports a job finishing, its best score, and the name of its results file.
rejected `bestScore jobID`	Reports a job as having been rejected.
done	Reports that the Manager has no jobs waiting or active in any batch, and has thus gone idle. It is still possible to extend those batches.
status `ws as fs rs batchID`	Reports statistics about the jobs in a batch. Sent when a batch is loaded before starting the first job, and also whenever `report` is received. `ws` = the number of jobs waiting and droppable, `as` = number started (active), `fs` = number finished, `rs` = number rejected.
dropped `[jobID]`	Informs the Controller of jobs that have been dropped. When a Manager receives a `drop` instruction, it sends a series of `dropped jobid` messages followed by a lone `dropped`.
failed `batchID`	Reports that an error occurred during the loading of a new batch. If this is sent, the batch will not be processed.
crashed `message`	Reports an exception. If this message is sent to the Controller, the Manager will have automatically stopped docking and reset itself, but should then be able to start new batches.
farewell	Indicates that the Manager is terminating, and will not respond to any further instructions.

Table F.4: Messages that Managers can send to the Controller

Each Manager is configured with a unique communication path, typically a subdirectory of a shared directory, to which the Controller and Manager processes must both have full read/write access. Their contents may be safely deleted when neither part of the system is running, however. The job configuration file lists these directories, with their paths as seen by each component of the system. Two files are created per Manager, one for communication in each direction, with additional files as required for starting batches and reporting results. These all contain plain text, using line-based instructions to pass information between the programs.

To start a batch of jobs on a Manager, the Controller creates a file with a listing of

the parameters defining the task, named using the batch number and the suffix '.in'. The contents of relevant configuration files (scoring function, search method, etc.) are inserted in the appropriate places, rather than being transmitted separately. All MKB paths are made relative to a standard data directory. The final part of the file lists the ligand conformations to be docked. If more jobs should be given to a Manager, for load balancing, another file is created with only this last section, named with the suffix '.in2'. Once these files are ready, an instruction is sent through the normal channel to notify the Manager of the filename it should process.

When jobs finish, their results are recorded in a file named according to the batch and job numbers with the suffix '.out'. The file contains a block of lines for each result, in the following pattern (ending with a blank line):

```
jobid
score scoreVal normalScore SFname
trans (tx ty tz)
rotate [ra rx ry rz]
data value name
```

where `jobid` has the form `batch*ligand#conformation` containing index numbers defined by the Controller and MKBs. The `data` lines are determined by the scoring function, and may be several. When such a file is written, the Manager notifies the Controller of its availability as part of the message reporting the job finishing.

Table F.3 lists the messages that may be sent from the Controller to Managers, and Table F.4 lists the response and status messages that Managers can send back. Periodically, each program sends a `hello` signal to the other, regardless of any other activity, to confirm that the system is still working. The following section demonstrates the operation of this design in the context of a simple docking execution.

F.8.2 Sample Execution Sequence

The following pages describe the sequence of events in a hypothetical docking execution. It is assumed that the ligand conformations are divided between two Managers, but one of these is shown in full here for clarity. Only the steps for job redistribution are included for Manager 2. The priority function allows only two jobs active on each Manager, and quota-based rejection is enabled. Text in shaded boxes shows communication *sent* to the other process; the Controller's messages are preceded by the recipient Manager number.

Controller	**Manager 1**	**Manager 2**
Invoked by user at command line.	*Running on each computer available for use.*	
Loads job configuration file.		
1: `hello`		
2: `hello`		
	`hello`	`hello`
Live Manager count incremented twice, establishing connections.		
Loads target data, including MKB servers, and writes 1.in file.		
1: `start 1`		
2: `start 1`	Reads 1.in file and loads target data, including MKB clients, as a batch.	
	Creates one job per conformation, 8 in all.	Also begins docking.
	`status 8 0 0 0 1`	
Initializes the Manager Link's batch status with 8 waiting jobs and no others.	Checks priority function for permission to start a job. Granted.	
	Picks and starts job 1.	
	`started 1*0#1`	
Updates batch status: 7 waiting, 1 active.	Checks priority function for permission to start a job. Granted.	
	Picks and starts job 2.	
	`started 1*0#2`	
Updates batch status: 6 waiting, 2 active.	Checks priority function for permission to start a job. Refused.	
	Selects and resumes job 1.	
	Updates priority of job 1.	
	Checks priority function for permission to start a job. Refused.	
	Selects and resumes job 2.	

Controller	Manager 1	Manager 2
	Updates priority of job 2.	
	Checks priority function for permission to start a job. Refused.	
	Selects and resumes job 1, which finishes.	
	Writes the results from job 1 to file 1-1.out.	
	`finished 1-1.out -3.141 1*0#1`	
Updates batch status: 6 waiting, 1 active, 1 finished.	Checks priority function for permission to start a job. Granted.	
Parses results from file 1-1.out and inserts them into the main result list.	Picks and starts job 3.	
Checks the quota, which is not yet filled.	`started 1*0#3`	
Updates batch status: 5 waiting, 2 active, 1 finished.	Checks priority function for permission to start a job. Refused.	
	Selects and resumes job 2, which finishes.	
	Writes the results from job 2 to file 1-2.out.	
	`finished 1-2.out -2.718 1*0#2`	
Updates batch status: 5 waiting, 1 active, 2 finished.	Passes results to Syllabus for learning.	
Parses results from file 1-2.out and inserts them into the main result list.	Commits changed education.X MKB entry.	
Checks the quota, which is now filled with a worst-case score of –1.414.	Checks priority function for permission to start a job. Granted.	
1: `worst -1.414 1`	Picks and starts job 4.	
Resolves new data to MKB(s).	`started 1*0#4`	
2: `mkbchange education.X/0 rec`	Notes the worst-case score of –1.414 in the statistics for batch 1.	
Updates batch status: 4 waiting, 2 active, 2 finished.	Checks priority function for permission to start a job. Refused.	Marks education.X as dirty.
	Selects and resumes job 3.	
	Updates priority of job 3; it is rejected using the worst-case score.	
	`rejected -1.250 1*0#3`	
Updates batch status: 4 waiting, 1 active, 2 finished, 1 rejected.	Checks priority function for permission to start a job. Granted.	
	Picks and starts job 5.	

Controller	Manager 1	Manager 2
	`started 1*0#5`	Completes last job.
Updates batch status: 3 waiting, 2 active, 2 finished, 1 rejected.	Checks priority function for permission to start a job. Refused.	`done`
Identifies the slowest Manager from which to reallocate jobs: Manager 1.	Selects and resumes job 4.	
1: `drop 2 1`	Updates priority of job 4.	
Prepares a list of additional jobs for Manager 2.	Finds 2 waiting jobs to drop: jobs 7 and 8.	
	`dropped 1*0#7`	
Appends 1*0#7 to the additional job list.	`dropped 1*0#8`	
Appends 1*0#8 to the additional job list.	`dropped`	
Writes the additional list to the 1a.in2 file.	Checks priority function for permission to start a job. Refused.	
2: `add 1 1a.in2`	Selects and resumes job 5.	
	Updates priority of job 5.	Loads extra jobs from 1a.in2.
. . . . .	. . . . .	. . . . .
	Remaining jobs finish.	
Results are collected.	`done`	`done`
. . . . .	. . . . .	. . . . .
1: `reset`		
2: `reset`		
Writes user output files: poses in SDF format and data table in plain text.	Unloads batch and all associated data objects.	
Terminates.	*Continues running for future access.*	

References

[Abola *et al.*, 2000] Abola, E., Kuhn, P., Earnest, T., & Stevens, R.C. 2000. **Automation of X-ray crystallography**. *Nature Struct. & Mol. Biol.*, **7**, 973–977. Nature Publishing. (ref. *p*.175)

[Adcock & McCammon, 2006] Adcock, S.A., & McCammon, J.A. 2006. **Molecular Dynamics: Survey of Methods for Simulating the Activity of Proteins**. *Chem. Rev.*, **106**, 1589–1615. American Chemical Society. (ref. *p*.39, 48)

[Alard & Wodak, 1991] Alard, P., & Wodak, S.J. 1991. **Detection of Cavities in a Set of Interpenetrating Spheres**. *J. Comput. Chem.*, **12**, 918–922. Wiley Periodicals. (ref. *p*.35)

[Alonso *et al.*, 2006] Alonso, H., Bliznyuk, A.A., & Gready, J.E. 2006. **Combining docking and molecular dynamic simulations in drug design**. *Med. Res. Rev.*, **26**, 531–568. Wiley Periodicals. (ref. *p*.39)

[Ansari *et al.*, 1994] Ansari, A., Jones, C.M., Henry, E.R., Hofrichter, J., & Eaton, W.A. 1994. **Conformational Relaxation and Ligand Binding in Myoglobin**. *Biochemistry*, **33**, 5128–5145. American Chemical Society. (ref. *p*.175)

[Apaydin, 2004] Apaydin, M.S. 2004. ***Stochastic Roadmap Simulation: An Efficient Representation And Algorithm For Analyzing Molecular Motion***. Ph.D. thesis, Stanford University. (ref. *p*.46)

[Apaydin *et al.*, 2001] Apaydin, M.S., Singh, A.P., Brutlag, D.L., & Latombe, J-C. 2001. **Capturing Molecular Energy Landscapes with Probabilistic Conformational Roadmaps**. *Pages 932–939 of: Proc. IEEE Int. Conf. Rob. Autom.*, vol. 1. IEEE. (ref. *p*.46)

[Apaydin *et al.*, 2002] Apaydin, M.S., Guestrin, C.E., Varma, C., Brutlag, D.L., & Latombe, J-C. 2002. **Stochastic roadmap simulation for the study of ligand-protein interactions**. *Bioinformatics*, **18**, S18–S26. Oxford University Press. (ref. *p*.46)

[Apaydin *et al.*, 2004] Apaydin, M.S., Brutlag, D.L., Guestrin, C., Hsu, D., & Latombe, J-C. 2004. **Stochastic Conformational Roadmaps for Computing Ensemble Properties of Molecular Motion**. *Pages 131–147 of: Proc. Int. Workshop Algorithmic Foundations of Robotics*, vol. V. Springer. (ref. *p*.46)

[Åqvist *et al.*, 2002] Åqvist, J., Luzhkov, V.B., & Brandsdal, B.O. 2002. **Ligand Binding Affinities from MD Simulations**. *Acc. Chem. Res.*, **35**, 358–365. American Chemical Society. (ref. *p*.39)

[Baker, 2000] Baker, D. 2000. **A surprising simplicity to protein folding**. *Nature*, **405**, 39–42. Macmillan Magazines. (ref. *p*.31)

[Baldridge *et al.*, 2002] Baldridge, K.K., Greenberg, J.P., Elbert, S.T., Mock, S., & Papadopoulos, P. 2002. **QMView and GAMESS: Integration into the World Wide Computational Grid**. *Pages 64–88 of: Proc. Supercomputing (ACM/IEEE 2002 Conference)*. IEEE. (ref. *p*.28)

[Ballester & Richards, 2007] Ballester, P.J., & Richards, W.G. 2007. **Ultrafast Shape Recognition to Search Compound Databases for Similar Molecular Shapes**. *J. Comput. Chem.*, **28**, 1711–1723. Wiley Periodicals. (ref. *p*.29, 106)

[Bash *et al.*, 1983] Bash, P.A., Pattabiraman, N., Huang, C., Ferrin, T.E., & Langridge, R. 1983. **Van Der Waals Surfaces in Molecular Modeling: Implementation with Real-Time Computer Graphics**. *Science*, **222**, 1325–1327. American Association for the Advancement of Science. (ref. *p*.187)

[Bayazit, 2003] Bayazit, O.B. 2003. ***Solving Motion Planning Problems By Iterative Relaxation Of Constraints***. Ph.D. thesis, Texas A&M University. (ref. *p*.46)

[Bayazit *et al.*, 2000] Bayazit, O.B., Song, G., & Amato, N.M. 2000. ***Ligand Binding with OBPRM and Haptic User Input: Enhancing Automatic Motion Planning with Virtual Touch***. Tech. rept. TR00-025. Texas A&M University. (ref. *p*.46)

[Berg *et al.*, 2002] Berg, J.M., Tymoczko, J.L., & Stryer, L. 2002. ***Biochemistry***. 5th edn. W.H. Freeman. ISBN 0-716-74684-0. (ref. *p*.17, 176)

[Berman *et al.*, 2000] Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., & Bourne, P.E. 2000. **The Protein Data Bank**. *Nucleic Acids Research*, **28**, 235–242. Oxford University Press. (ref. *p*.29)

[Bernstein *et al.*, 1977] Bernstein, F.C., Koetzle, T.F., Williams, G.J., Meyer, Jr., E.F., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., & Tasumi, M. 1977. **The Protein Data Bank. A computer-based archival file for macromolecular structures**. *J. Mol. Biol.*, **112**, 535–542. Academic Press. (ref. *p*.19, 29)

[Billeter *et al.*, 1987] Billeter, M., Havel, T.F., & Kuntz, I.D. 1987. **A new approach to the problem of docking two molecules: The ellipsoid algorithm**. *Biopolymers*, **26**, 777–793. John Wiley & Sons. (ref. *p*.35)

[Böhm, 1994] 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. ESCOM Science. (ref. *p*.60, 66)

[Böhm & Stahl, 2000] Böhm, H-J., & Stahl, M. 2000. **Structure-based library design: molecular modelling merges with combinatorial chemistry**. *Curr. Opin. Chem. Biol.*, **4**, 283–286. Elsevier Science. (ref. *p*.61)

[Bohme Leite *et al.*, 2007] Bohme Leite, T., Gomes, D., Miteva, M.A., Chomilier, J., Villoutreix, B.O., & Tufféry, P. 2007. **Frog: A FRee Online druG 3D conformation generator**. *Nucleic Acids Research*, **35**, W568–W572. Oxford University Press. (ref. *p*.200)

[Boost, 2009] Boost. 2009. ***Boost C++ Libraries***. `http://www.boost.org` (ref. *p*.64)

[Bourne & Weissig, 2003] Bourne, P.E., & Weissig, H. (eds). 2003. ***Structural Bioinformatics***. Wiley-Liss. ISBN 0-471-20199-5. (ref. *p*.19)

[Brady & Stouten, 2000] Brady, Jr., G.P., & Stouten, P.F.W. 2000. **Fast prediction and visualization of protein binding pockets with PASS**. *J. Comput. Aided Mol. Des.*, **14**, 383–401. Kluwer Academic Publishers. (ref. *p*.35, 96)

[Branden & Tooze, 1999] Branden, C., & Tooze, J. 1999. ***Introduction to Protein Structure***. 2nd edn. Garland. ISBN 0-815-32305-0. (ref. *p*.17, 176)

[B-Rao *et al.*, 2009] B-Rao, C., Subramanian, J., & Sharma, S.D. 2009. **Managing protein flexibility in docking and its applications**. *Drug Discovery Today*, **14**, 394–400. Elsevier Science. (ref. *p*.42, 51)

[Brooks *et al.*, 1983] Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S., & Karplus, M. 1983. **CHARMM: A program for macromolecular energy, minimization, and dynamics calculations**. *J. Comput. Chem.*, **4**, 187–217. Wiley Periodicals. (ref. *p*.26)

[Brooks *et al.*, 2009] Brooks, B.R., Brooks III, C.L., Mackerell, Jr., A.D., Nilsson, L., Petrella, R.J., Roux, B., Won, Y., Archontis, G., Bartels, C., Boresch, S., Caflisch, A., Caves, L., Cui, Q., Dinner, A.R., Feig, M., Fischer, S., Gao, J., Hodoscek, M., Im, W., Kuczera, K., Lazaridis, T., Ma, J., Ovchinnikov, V., Paci, E., Pastor, R.W., Post, C.B., Pu, J.Z., Schaefer, M., Tidor, B., Venable, R.M., Woodcock, H.L., Wu, X., Yang, W., York, D.M., & Karplus, M. 2009. **CHARMM: The Biomolecular Simulation Program**. *J. Comput. Chem.*, **30**, 1545–1614. Wiley Periodicals. (ref. *p*.26, 39)

[Bryngelson & Wolynes, 1987] Bryngelson, J.D., & Wolynes, P.G. 1987. **Spin glasses and the statistical mechanics of protein folding**. *Proc. Natl. Acad. Sci. USA*, **84**, 7524–7528. National Academy of Sciences. (ref. *p*.31)

[Burley, 2000] Burley, S.K. 2000. **An overview of structural genomics**. *Nature America*, 932–934. Macmillan Magazines. (ref. *p*.30)

[Busetta *et al.*, 1983] Busetta, B., Tickle, I.J., & Blundell, T.L. 1983. **DOCKER, an interactive program for simulating protein receptor and substrate interactions**. *J. Appl. Cryst.*, **16**, 432–437. International Union of Crystallography. (ref. *p*.33)

[Campbell *et al.*, 2003] Campbell, S.J., Gold, N.D., Jackson, R.M., & Westheady, D.R. 2003. **Ligand binding: functional site location, similarity and docking**. *Curr. Opin. Struct. Biol.*, **13**, 389–395. Elsevier Science. (ref. *p*.34, 51)

[Carlson, 2002] Carlson, H.A. 2002. **Protein flexibility and drug design: how to hit a moving target**. *Curr. Opin. Chem. Biol.*, **6**, 447–452. Elsevier Science. (ref. *p*.50)

[Carlson & McCammon, 2000] Carlson, H.A., & McCammon, J.A. 2000. **Accommodating Protein Flexibility in Computational Drug Design**. *Mol. Pharmacol.*, **57**, 213–218. American Society for Pharmacology and Experimental Therapeutics. (ref. *p*.41)

[Carpin & Pillonetto, 2005] Carpin, S., & Pillonetto, G. 2005. **Motion Planning Using Adaptive Random Walks**. *Transactions on Robotics*, **21**, 129–136. IEEE. (ref. *p*.46)

[Case *et al.*, 2005] Case, D.A., Cheatham III, T.E., Darden, T., Gohlke, H., Luo, R., Merz, Jr., K.M., Onufriev, A., Simmerling, C., Wang, B., & Woods, R.J. 2005. **The Amber Biomolecular Simulation Programs**. *J. Comput. Chem.*, **26**, 1668–1688. Wiley Periodicals. (ref. *p*.26)

[Cavasotto & Abagyan, 2004] Cavasotto, C.N., & Abagyan, R.A. 2004. **Protein Flexibility in Ligand Docking and Virtual Screening to Protein Kinases**. *J. Mol. Biol.*, **337**, 209–225. Elsevier Science. (ref. *p*.42, 81)

[Cavasotto & Orry, 2007] Cavasotto, C.N., & Orry, A.J.W. 2007. **Ligand Docking and Structure-based Virtual Screening in Drug Discovery**. *Curr. Topics Med. Chem.*, **7**, 1006–1014. Bentham. (ref. *p*.40, 51)

[Chen *et al.*, 2007] Chen, M.E., Cang, H.X., & Nymeyer, H. 2007. ***NOC***. `http://noch.sourceforge.net` (ref. *p*.253)

[Chen *et al.*, 2003] Chen, R., Li, L., & Weng, Z. 2003. **ZDOCK: An Initial-Stage Protein-Docking Algorithm**. *Proteins*, **52**, 80–87. Wiley-Liss. (ref. *p*.38)

[Chen & Shoichet, 2009] Chen, Y., & Shoichet, B.K. 2009. **Molecular docking and ligand specificity in fragment-based inhibitor discovery**. *Nature Chemical Biology*, **5**, 358–364. Nature Publishing. (ref. *p*.45)

[Chen *et al.*, 2008] Chen, Y., Ding, F., Nie, H., Serohijos, A.W., Sharma, S., Wilcox, K.C., Yin, S., & Dokholyan, N.V. 2008. **Protein folding: Then and now**. *Arch. Biochem. Biophys.*, **469**, 4–19. Elsevier Science. (ref. *p*.31, 51)

[Chikenji *et al.*, 2006] Chikenji, G., Fujitsuka, Y., & Takada, S. 2006. **Shaping up the protein folding funnel by local interaction: Lesson from a structure prediction study**. *Proc. Natl. Acad. Sci. USA*, **103**, 3141–3146. National Academy of Sciences. (ref. *p*.47)

[Claußen *et al.*, 2001] Claußen, H., Buning, C., Rarey, M., & Lengauer, T. 2001. **FLEXE — Efficient Molecular Docking Considering Protein Structure Variations**. *J. Mol. Biol.*, **308**, 377–395. Academic Press. (ref. *p*.45)

[Cohen *et al.*, 1990] Cohen, N.C., Blaney, J.M., Humblet, C., Gund, P., & Barry, D.C. 1990. **Molecular Modeling Software and Methods for Medicinal Chemistry**. *J. Med. Chem.*, **33**, 883–894. American Chemical Society. (ref. *p*.26)

[Collins & McKusick, 2001] Collins, F.S., & McKusick, V.A. 2001. **Implications of the Human Genome Project for Medical Science**. *JAMA*, **285**, 540–544. American Medical Association. (ref. *p*.30)

[Connolly, 1983a] Connolly, M.L. 1983a. **Analytical Molecular Surface Calculation**. *J. Appl. Cryst.*, **16**, 548–558. International Union of Crystallography. (ref. *p*.24)

[Connolly, 1983b] Connolly, M.L. 1983b. **Solvent-Accessible Surfaces of Proteins and Nucleic Acids**. *Science*, **221**, 709–713. American Association for the Advancement of Science. (ref. *p*.24)

[Connolly, 1985] Connolly, M.L. 1985. **Computation of Molecular Volume**. *J. Am. Chem. Soc.*, **107**, 1118–1124. American Chemical Society. (ref. *p*.188)

[Connolly, 1996] Connolly, M.L. 1996. ***Molecular Surfaces: A Review***. Network Science. `http://www.netsci.org/Science/Compchem/feature14.html` (ref. *p*.24)

[Cooley & Tukey, 1965] Cooley, J.W., & Tukey, J.W. 1965. **An Algorithm for the Machine Calculation of Complex Fourier Series**. *Math. Comput.*, **19**, 297–301. American Mathematical Society. (ref. *p*.178)

[Cortés *et al.*, 2005] Cortés, J., Siméon, T., Ruiz de Angulo, V., Guieysse, D., Remaud-Siméon, M., & Tran, V. 2005. **A path planning approach for computing large-amplitude motions of flexible molecules**. *Bioinformatics*, **21**, i116–i125. Oxford University Press. (ref. *p*.46)

[Cozzini *et al.*, 2008] Cozzini, P., Kellogg, G.E., Spyrakis, F., Abraham, D.J., Costantino, G., Emerson, A., Fanelli, F., Gohlke, H., Kuhn, L.A., Morris, G.M., Orozco, M., Pertinhez, T.A., Rizzi, M., & Sotriffer, C.A. 2008. **Target Flexibility: An Emerging Consideration in Drug Discovery and Design**. *J. Med. Chem.*, **51**, 6237–6255. American Chemical Society. (ref. *p*.42, 51)

[Daeyaert *et al.*, 2004] Daeyaert, F., de Jonge, M., Heeres, J., Koymans, L., Lewi, P., Vinkers, M.H., & Janssen, P.A.J. 2004. **A Pharmacophore Docking Algorithm and its Application to the Cross-Docking of 18 HIV-NNRTIs in their Binding Pockets**. *Proteins*, **54**, 526–533. Wiley-Liss. (ref. *p*.161)

[David *et al.*, 2001] David, L., Luo, R., & Gilson, M.K. 2001. **Ligand-receptor docking with the Mining Minima optimizer**. *J. Comput. Aided Mol. Des.*, **15**, 157–171. Kluwer Academic Publishers. (ref. *p*.38)

[Davis & Baker, 2009] Davis, I.W., & Baker, D. 2009. **RosettaLigand Docking with Full Ligand and Receptor Flexibility**. *J. Mol. Biol.*, **385**, 381–392. Elsevier Science. (ref. *p*.40)

[de Azevedo & Dias, 2008] de Azevedo, Jr., W.F., & Dias, R. 2008. **Computational Methods for Calculation of Ligand-Binding Affinity**. *Curr. Drug Targets*, **9**, 1031–1039. Bentham. (ref. *p*.51)

[DeLano, 2006] DeLano, W. 2006. ***PyMOL***. `http://www.pymol.org` (ref. *p*.253)

[Dill *et al.*, 2007] Dill, K.A., Ozkan, S.B., Weikl, T.R., Chodera, J.D., & Voelz, V.A. 2007. **The protein folding problem: when will it be solved?** *Curr. Opin. Struct. Biol.*, **17**, 342–346. Elsevier Science. (ref. *p*.31, 51)

[Dill *et al.*, 2008] Dill, K.A., Ozkan, S.B., Shell, M.S., & Weikl, T.R. 2008. **The Protein Folding Problem**. *Ann. Rev. Biophys.*, **37**, 289–316. Annual Reviews. (ref. *p*.31, 51)

[Diller & Merz, 2001] Diller, D.J., & Merz, Jr., K.M. 2001. **High Throughput Docking for Library Design and Library Prioritization**. *Proteins*, **43**, 113–124. Wiley-Liss. (ref. *p*.41, 42)

[Diller & Verlinde, 1999] Diller, D.J., & Verlinde, C.L.M.J. 1999. **Optimization Algorithms for the Purpose of Molecular Docking**. *J. Comput. Chem.*, **20**, 1740–1751. Wiley Periodicals. (ref. *p*.41)

[Dobson & Karplus, 1999] Dobson, C.M., & Karplus, M. 1999. **The fundamentals of protein folding: bringing together theory and experiment**. *Curr. Opin. Struct. Biol.*, **9**, 92–101. Elsevier Science. (ref. *p*.31)

[Dobson & Ptitsyn, 1999] Dobson, C.M., & Ptitsyn, O.B. 1999. **Folding and Binding: The Biological Consequences of Physical Principles**. *Curr. Opin. Struct. Biol.*, **9**, 89–91. Elsevier Science. (ref. *p*.31)

[Dominguez *et al.*, 2003] Dominguez, C., Boelens, R., & Bonvin, A.M.J.J. 2003. **HADDOCK: a protein-protein docking approach based on biochemical or biophysical data**. *J. Am. Chem. Soc.*, **125**, 1731–1737. American Chemical Society. (ref. *p*.38, 48)

[DOX, 2009] DOX. 2009. ***Ligand-Protein Docking***. InhibOx Ltd. `http://www.inhibox.com/docking` (ref. *p*.64)

[Drenth, 1999] Drenth, J. 1999. ***Principles of Protein X-Ray Crystallography***. Springer-Verlag. ISBN 0-387-98587-5. (ref. *p*.175)

[Eckert & Bajorath, 2007] Eckert, H., & Bajorath, J. 2007. **Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches**. *Drug Discovery Today*, **12**, 225–233. Elsevier Science. (ref. *p*.28, 51)

[Edelsbrunner & Mücke, 1994] Edelsbrunner, H., & Mücke, E.P. 1994. **Three dimensional alpha-shapes**. *ACM Transact. Graph.*, **13**, 43–72. ACM Press. (ref. *p*.24)

[Eldridge *et al.*, 1997] Eldridge, M.D., Murray, C.W., Auton, T.R., Paolini, G.V., & 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. Springer. (ref. *p*.60)

[Elliott & Rao, 1982] Elliott, D.F., & Rao, K.R. 1982. ***Fast Transforms: Algorithms, Analyses, Applications***. Academic Press. ISBN 0-122-37080-5. (ref. *p*.178)

[Ellis & Hartl, 1999] Ellis, R.J., & Hartl, F.U. 1999. **Principles of protein folding in the cellular environment**. *Curr. Opin. Struct. Biol.*, **9**, 102–110. Elsevier Science. (ref. *p*.31)

[Erickson *et al.*, 2004] Erickson, J.A., Jalaie, M., Robertson, D.H., Lewis, R.A., & Vieth, M. 2004. **Lessons in Molecular Recognition: The Effects of Ligand and Protein Flexibility on Molecular Docking Accuracy**. *J. Med. Chem.*, **47**, 45–55. American Chemical Society. (ref. *p*.40, 81)

[Ewing & Kuntz, 1997] Ewing, T.J.A., & Kuntz, I.D. 1997. **Critical Evaluation of Search Algorithms for Automated Molecular Docking and Database Screening**. *J. Comput. Chem.*, **18**, 1175–1189. Wiley Periodicals. (ref. *p*.35)

[Feig *et al.*, 2004] Feig, M., Karanicolas, J., & Brooks III, C.L. 2004. **MMTSB Tool Set: enhanced sampling and multiscale modeling methods for applications in structural biology**. *J. Mol. Graph. Model.*, **22**, 377–395. Elsevier Science. (ref. *p*.27)

[Finkelstein, 1997] Finkelstein, A.V. 1997. **Protein structure: what is it possible to predict now?** *Curr. Opin. Struct. Biol.*, **7**, 60–71. Current Biology. (ref. *p*.31)

[Finn & Kavraki, 1999] Finn, P.W., & Kavraki, L.E. 1999. **Computational Approaches to Drug Design**. *Algorithmica*, **25**, 347–371. Springer-Verlag. (ref. *p*.51)

[Fischer *et al.*, 1993a] Fischer, D., Norel, R., Nussinov, R., & Wolfson, H.J. 1993a. **3D Docking of Protein Molecules: Combinatorial Pattern Matching**. *Lect. Notes. Comput. Sci.*, **684**, 20–34. Springer-Verlag. (ref. *p*.36)

[Fischer *et al.*, 1993b] Fischer, D., Norel, R., Wolfson, H.J., & Nussinov, R. 1993b. **Surface Motifs by a Computer Vision Technique: Searches, detection, and implication for Protein-Ligand recognition**. *Proteins*, **16**, 278–292. Wiley-Liss. (ref. *p*.28, 38)

[Foreman *et al.*, 1999] Foreman, K.W., Phillips, A.T., Rosen, J.B., & Dill, K.A. 1999. **Comparing Search Strategies for Finding Global Optima on Energy Landscapes**. *J. Comput. Chem.*, **20**, 1527–1532. Wiley Periodicals. (ref. *p*.26)

[Fradera *et al.*, 2000] Fradera, X., Knegtel, R.M.A., & Mestres, J. 2000. **Similarity-Driven Flexible Ligand Docking**. *Proteins*, **40**, 623–636. Wiley-Liss. (ref. *p*.42)

[Fradera *et al.*, 2002] Fradera, X., De la Cruz, X., Silva, C.H., Gelpi, J.L., Luque, F.J., & Orozco, M. 2002. **Ligand-induced changes in the binding sites of proteins**. *Bioinformatics*, **18**, 939–948. Oxford University Press. (ref. *p*.40)

[Fradera *et al.*, 2004] Fradera, X., Kaur, J., & Mestres, J. 2004. **Unsupervised guided docking of covalently bound ligands**. *J. Comput. Aided Mol. Des.*, **18**, 635–650. Springer. (ref. *p*.43)

[Freskos *et al.*, 2007] Freskos, J.N., Fobian, Y.M., Benson, T.E., Moon, J.B., Bienkowski, M.J., Brown, D.L., Emmons, T.L., Heintz, R., Laborde, A., McDonald, J.J., Mischke, B.V., Molyneaux, J.M., Mullins, P.B., Prince, D.B., Paddock, D.J., Tomassellia, A.G., & Winterrowd, G. 2007. **Design of potent inhibitors of human β-secretase. Part 2**. *Bioorg. Med. Chem. Lett.*, **17**, 78–81. Elsevier Science. (ref. *p*.18)

[Friesner *et al.*, 2004] 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., & 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. American Chemical Society. (ref. *p*.50, 183)

[Friesner *et al.*, 2006] Friesner, R.A., Murphy, R.B., Repasky, M.P., & Sherman, B.W. 2006. **Use of the Glide extra precision methodology for docking and scoring**. *In: 232nd ACS Nat. Meet.* American Chemical Society. (ref. *p*.50)

[Gabb *et al.*, 1997] Gabb, H.A., Jackson, R.M., & Sternberg, M.J.E. 1997. **Modelling Protein Docking using Shape Complementarity, Electrostatics and Biochemical Information**. *J. Mol. Biol.*, **272**, 106–120. Academic Press. (ref. *p*.37, 38, 48)

[Gardiner *et al.*, 2001] Gardiner, E.J., Willett, P., & Artymiuk, Peter J. 2001. **Protein Docking Using a Genetic Algorithm**. *Proteins*, **44**, 44–56. Wiley-Liss. (ref. *p*.37)

[Gehlhaar *et al.*, 1995] Gehlhaar, D.K., Verkhivker, G.M., Rejto, P.A., Sherman, C.J., Fogel, D.B., Fogel, L.J., & Freer, S.T. 1995. **Molecular recognition of the inhibitor AC-1343 by HIV-1 protease: Conformationally flexible docking by evolutionary programming**. *Chemistry & Biology*, **2**, 317–324. Current Biology. (ref. *p*.43, 61)

[Gething & Sambrook, 1992] Gething, M.J., & Sambrook, J. 1992. **Protein folding in the cell**. *Nature*, **355**, 33–45. Nature Publishing. (ref. *p*.174)

[Glick *et al.*, 2002a] Glick, M., Grant, G.H., & Richards, W.G. 2002a. **Docking of Flexible Molecules Using Multiscale Ligand Representations**. *J. Med. Chem.*, **45**, 4639–4646. American Chemical Society. (ref. *p*.26, 59)

[Glick *et al.*, 2002b] Glick, M., Robinson, D.D., Grant, G.H., & Richards, W.G. 2002b. **Identification of Ligand Binding Sites on Proteins Using a Multi-Scale Approach**. *J. Am. Chem. Soc.*, **124**, 2337–2344. American Chemical Society. (ref. *p*.26)

[Gohlke & Thorpe, 2006] Gohlke, H., & Thorpe, M.F. 2006. **A Natural Coarse Graining for Simulating Large Biomolecular Motion**. *Biophys. Journal*, **91**, 2115–2120. Biophysical Society. (ref. *p*.41)

[Goldman & Wipke, 2000a] Goldman, B.B., & Wipke, W.T. 2000a. **Quadratic Shape Descriptors. 1. Rapid Superposition of Dissimilar Molecules Using Geometrically Invariant Surface Descriptors**. *J. Chem. Inf. Comput. Sci.*, **40**, 644–658. American Chemical Society. (ref. *p*.36)

[Goldman & Wipke, 2000b] Goldman, B.B., & Wipke, W.T. 2000b. **Quadratic Shape Descriptors. 2. Molecular Docking Using Quadratic Shape Descriptors (QSDock)**. *Proteins*, **38**, 79–94. Wiley-Liss. (ref. *p*.36, 48)

[Grant, 2009] Grant, M.A. 2009. **Protein structure prediction in structure-based ligand design and virtual screening**. *Comb. Chem. High Throughput Screening*, **12**, 940–960. Bentham. (ref. *p*.30, 51)

[GraphicsMagick, 2003] GraphicsMagick. 2003. ***GraphicsMagick***. `http://www.graphicsmagick.org` (ref. *p*.253)

[Gräter *et al.*, 2005] Gräter, F., Schwarzl, S.M., Dejaegere, A., Fischer, S., & Smith, J.C. 2005. **Protein-Ligand Binding Free Energies Calculated with Quantum Mechanics/Molecular Mechanics**. *J. Phys. Chem.*, **109**, 10474–10483. American Chemical Society. (ref. *p*.40)

[Gschwend *et al.*, 1996] Gschwend, D.A., Good, A.C., & Kuntz, I.D. 1996. **Molecular Docking towards Drug Discovery**. *J. Mol. Recog.*, **9**, 175–186. John Wiley & Sons. (ref. *p*.50)

[Halgren *et al.*, 2004] Halgren, T.A., Murphy, R.B., Friesner, R.A., Beard, H.S., Frye, L.L., Pollard, W.T., & Banks, J.L. 2004. **Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening**. *J. Med. Chem.*, **47**, 1750–1759. American Chemical Society. (ref. *p*.50)

[Halperin *et al.*, 2002] Halperin, I., Ma, B., Wolfson, H., & Nussinov, R. 2002. **Principles of Docking: An Overview of Search Algorithms and a Guide to Scoring Functions**. *Proteins*, **47**, 409–443. Wiley-Liss. (ref. *p*.33, 60, 183)

[Hamilton, 1967] Hamilton, W.R. 1967. **On Quaternions; or On a New System of Imaginaries in Algebra**. *In:* Halberstam, H., & Ingram, R.E. (eds), *The Mathematical Papers of Sir William Rowan Hamilton*. Cambridge University Press. (ref. *p*.84)

[Hardesty *et al.*, 1999] Hardesty, B., Tsalkove, T., & Kramer, G. 1999. **Co-translational Folding**. *Curr. Opin. Struct. Biol.*, **9**, 111–114. Elsevier Science. (ref. *p*.31)

[Hart, 1994] Hart, W.E. 1994. ***Adaptive Global Optimization with Local Search***. Ph.D. thesis, University of California, San Diego. (ref. *p*.72, 118)

[Hart *et al.*, 1994] Hart, W.E., Kammeyer, T.E., & Belew, R.K. 1994. ***The Role of Development in Genetic Algorithms***. Tech. rept. CS94-394. University of California, San Diego. (ref. *p*.118)

[Hartshorn *et al.*, 2007] Hartshorn, M.J., Verdonk, M.L., Chessari, G., Brewerton, S.C., Mooij, W.T.M., Mortenson, P.N., & Murray, C.W. 2007. **Diverse, High-Quality Test Set for the Validation of Protein-Ligand Docking Performance**. *J. Med. Chem.*, **50**, 726–741. American Chemical Society. (ref. *p*.199)

[Hawkins *et al.*, 2008] Hawkins, P.C.D., Warren, G.L., Skillman, A.G., & Nicholls, A. 2008. **How to do an evaluation: pitfalls and traps**. *J. Comput. Aided Mol. Des.*, **22**, 179–190. Springer. (ref. *p*.43)

[Head *et al.*, 1997] Head, M.S., Given, J.A., & Gilson, M.K. 1997. **Mining Minima — Direct Computation of Conformational Free Energy**. *J. Phys. Chem.*, **101**, 1609–1618. American Chemical Society. (ref. *p*.26, 38)

[Hendlich *et al.*, 1997] Hendlich, M., Rippmann, F., & Barnickel, G. 1997. **LIGSITE: Automatic and efficient detection of potential small molecule binding sites in proteins**. *J. Mol. Graph. Model.*, **15**, 359–363. Elsevier Science. (ref. *p*.35)

[Hoare, 1962] Hoare, C.A.R. 1962. **Quicksort**. *Comput. J.*, **5**, 10–16. Brit. Comput. Soc. (ref. *p*.150)

[Holm & Park, 2000] Holm, L., & Park, J. 2000. **DaliLite workbench for protein structure comparison**. *Bioinformatics*, **16**, 566–567. Oxford University Press. (ref. *p*.29)

[Holm & Sander, 1993] Holm, L., & Sander, C. 1993. **Protein Structure Comparison by Alignment of Distance Matrices**. *J. Mol. Biol.*, **233**, 123–138. Academic Press. (ref. *p*.29)

[Holm & Sander, 1995] Holm, L., & Sander, C. 1995. **DALI: a network tool for protein structure comparison**. *Trends Biochem. Sci.*, **20**, 478–480. Elsevier Science. (ref. *p*.29)

[Holm *et al.*, 1992] Holm, L., Ouzounis, C., Sander, C., Tuparev, G., & Vriend, G. 1992. **A database of protein structure families with common folding motifs**. *Protein Science*, **1**, 1691–1698. Cold Spring Harbor Laboratory Press. (ref. *p*.29)

[Holm *et al.*, 2008] Holm, L., Kääriäinen, S., Rosenström, P., & Schenkel, A. 2008. **Searching protein structure databases with DaliLite v.3**. *Bioinformatics*, **24**, 2780–2781. Oxford University Press. (ref. *p*.29)

[Hom, 2005] Hom, G.K. 2005. ***Advances in computational protein design: Development of more efficient search algorithms and their application to the full-sequence design of larger proteins***. Ph.D. thesis, California Institute of Technology. (ref. *p*.31)

[Honig, 1999] Honig, B. 1999. **Protein Folding: From the Levinthal Paradox to Structure Prediction**. *J. Mol. Biol.*, **293**, 283–293. Academic Press. (ref. *p*.31)

[Hubbard, 1996] Hubbard, P.M. 1996. **Approximating Polyhedra with Spheres for Time-Critical Collision Detection**. *ACM Transact. Graph.*, **15**, 179–210. ACM Press. (ref. *p*.54)

[Irwin & Shoichet, 2005] Irwin, J.J., & Shoichet, B.K. 2005. **ZINC — A Free Database of Commercially Available Compounds for Virtual Screening**. *J. Chem. Inf. Model.*, **45**, 177–182. American Chemical Society. (ref. *p*.19)

[Isto, 2003] Isto, P. 2003. ***Adaptive Probabilistic Roadmap Construction With Multi-Heuristic Local Planning***. Ph.D. thesis, Helsinki University of Technology. (ref. *p*.46)

[Jacobs *et al.*, 1999] Jacobs, D.J., Kuhn, L.A., & Thorpe, M.F. 1999. **Flexible and rigid regions in proteins**. *Pages 357–384 of:* Thorpe, M.F., & Duxbury, P.M. (eds), *Rigidity Theory and Applications*. Kluwer Academic/Plenum Publishers. ISBN 0-306-46115-3. (ref. *p*.26, 41)

[Jacobs *et al.*, 2001] Jacobs, D.J., Rader, A.J., Kuhn, L.A., & Thorpe, M.F. 2001. **Protein Flexibility Predictions Using Graph Theory**. *Proteins*, **44**, 150–165. Wiley-Liss. (ref. *p*.41)

[Jaillet *et al.*, 2005] Jaillet, L., Yershova, A., LaValle, S.M., & Simeon, T. 2005. **Adaptive tuning of the sampling domain for dynamic-domain RRTs**. *Pages 2851–2856 of: Proc. Int. Conf. Intelligent Robots and Systems*. IEEE. (ref. *p*.46)

[Java3D, 2008] Java3D. 2008. ***Java 3D API***. Sun Microsystems Inc. `http://java.sun.com/javase/technologies/desktop/java3d` (ref. *p*.178)

[Jones & Willett, 1995] Jones, G., & Willett, P. 1995. **Docking small-molecule ligands into active sites**. *Curr. Opin. Biotechnol.*, **6**, 652–656. Current Biology. (ref. *p*.32)

[Jones *et al.*, 1997] Jones, G., Willett, P., Glen, R.C., Leach, A.R., & R.Taylor. 1997. **Development and Validation of a Genetic Algorithm for Flexible Docking**. *J. Mol. Biol.*, **267**, 727–748. Academic Press. (ref. *p*.43)

[Jones, 1924] Jones, J.E. 1924. **On the Determination of Molecular Fields. II. From the Equation of State of a Gas**. *Pages 463–477 of: Proc. R. Soc. Lond. A*, vol. 106. Royal Society. (ref. *p*.61)

[Jones & Thornton, 1996] Jones, S., & Thornton, J.M. 1996. **Principles of protein-protein interactions**. *Proc. Natl. Acad. Sci. USA*, **93**, 13–20. National Academy of Sciences. (ref. *p*.33)

[Kairys & Gilson, 2002] Kairys, V., & Gilson, M.K. 2002. **Enhanced Docking with the Mining Minima Optimizer: Acceleration and Side-Chain Flexibility**. *J. Comput. Chem.*, **23**, 1656–1670. Wiley Periodicals. (ref. *p*.38)

[Kan *et al.*, 1996] Kan, W.K., SiuI, Y.T., & But, P.P.H. 1996. **TCMD: An On-line Database of Traditional Chinese Medicine**. *Page 821 of: AMIA Annual Fall Symp.* American Medical Informatics Association. (ref. *p*.19)

[Karplus *et al.*, 1998] Karplus, K., Sjölander, K., Barrett, C., Cline, M., Haussler, D., Hughey, R., Holm, L., & Sander, C. 1998. **Predicting Protein Structure Using Hidden Markov Models**. *Proteins*, **Suppl. 1**, 134–139. Wiley-Liss. (ref. *p*.31)

[Karplus *et al.*, 1999] Karplus, K., Barrett, C., Cline, M., Diekhans, M., Grate, L., & Hughey, R. 1999. **Predicting Protein Structure Using Only Sequence Information**. *Proteins*, **3**, 121–125. Wiley-Liss. (ref. *p*.31)

[Katchalski-Katzir *et al.*, 1992] Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A.A., Aflalo, C., & Vakser, I.A. 1992. **Molecular Surface Recognition: Determination of geometric fit between proteins and their ligands by correlation techniques**. *Proc. Natl. Acad. Sci. USA*, **89**, 2195–2199. National Academy of Sciences. (ref. *p*.36, 37, 53, 177)

[Kazhdan, 2004] Kazhdan, M.M. 2004. ***Shape Representations and Algorithms for 3D Model Retrieval***. Ph.D. thesis, Princeton University. (ref. *p*.29, 51, 160)

[Kihara *et al.*, 2009] Kihara, D., Chen, H., & Yang, Y.D. 2009. **Quality Assessment of Protein Structure Models**. *Curr. Protein Pept. Sci.*, **10**, 216–228. Bentham. (ref. *p*.30, 51)

[Kinnings & Jackson, 2009] Kinnings, S.L., & Jackson, R.M. 2009. **LigMatch: A Multiple Structure-Based Ligand Matching Method for 3D Virtual Screening**. *J. Chem. Inf. Model.*, **49**, 2056–2066. American Chemical Society. (ref. *p*.28)

[Kitchen *et al.*, 2004] Kitchen, D.B., Decornez, H., Furr, J.R., & Bajorath, J. 2004. **Docking And Scoring In Virtual Screening For Drug Discovery: Methods And Applications**. *Nature Reviews (Drug Discovery)*, **3**, 935–949. Nature Publishing. (ref. *p*.33, 51, 60)

[Kolb *et al.*, 2009] Kolb, P., Ferreira, R.S., Irwin, J.J., & Shoichet, B.K. 2009. **Docking and chemoinformatic screens for new ligands and targets**. *Curr. Opin. Biotechnol.*, **20**, 429–436. Elsevier Science. (ref. *p*.51)

[Kolinski & Skolnick, 1994] Kolinski, A., & Skolnick, J. 1994. **Monte Carlo Simulations of Protein Folding. I. Lattice Model and Interaction Scheme**. *Proteins*, **18**, 338–352. Wiley-Liss. (ref. *p*.31)

[Kong *et al.*, 2000] Kong, J., C.A.White, Krylov, A.I., Sherrill, D., Adamson, R.D., Furlani, T.R., Lee, M.S., Lee, A.M., Gwaltney, S.R., Adams, T.R., Ochsenfeld, C., Gilbert, A.T.B., Kedziora, G.S., Rassolov, V.A., Maurice, D.R., Nair, N., Shao, Y., Besley, N.A., Maslen, P.E., Dombroski, J.P., Daschel, H., Zhang, W., Korambath, P.P., Baker, J., Byrd, E.F.C., Van Voorhis, T., Oumi, M., Hirata, S., Hsu, C-P., Ishikawa, N., Florian, J., Warshel, A., Johnson, B.G., Gill, P.M.W., Head-Gordon, M., & Pople, J.A. 2000. **Q-Chem 2 — A High-Performance *Ab Initio* Electronic Structure Program**. *J. Comput. Chem.*, **21**, 1532–1548. Wiley Periodicals. (ref. *p*.27)

[Kontoyianni *et al.*, 2004] Kontoyianni, M., McClellan, L.M., & Sokol, G.S. 2004. **Evaluation of Docking Performance: Comparative Data on Docking Algorithms**. *J. Med. Chem.*, **47**, 558–565. American Chemical Society. (ref. *p*.32)

[Kozakov *et al.*, 2006] Kozakov, D., Brenke, R., Comeau, S.R., & Vajda, S. 2006. **PIPER: An FFT-Based Protein Docking Program with Pairwise Potentials**. *Proteins*, **65**, 392–406. Wiley-Liss. (ref. *p*.38, 60)

[Kramer *et al.*, 1999] Kramer, B., Rarey, M., & Lengauer, T. 1999. **Evaluation of the FLEXX Incremental Construction Algorithm for Protein-Ligand Docking**. *Proteins*, **37**, 228–241. Wiley-Liss. (ref. *p*.45)

[Kuffner & LaValle, 2000] Kuffner, Jr., J.J., & LaValle, S.M. 2000. **RRT-Connect: An Efficient Approach to Single-Query Path Planning**. *Pages 995–1001 of: Proc. IEEE Int. Conf. Rob. Autom.*, vol. 2. IEEE. (ref. *p*.46)

[Kuhl *et al.*, 1986] Kuhl, F.S., Crippen, G.M., & Friesen, D.K. 1986. **A combinatorial algorithm for calculating ligand binding**. *J. Comput. Chem.*, **5**, 24–34. Wiley Periodicals. (ref. *p*.35)

[Kuntz *et al.*, 1982] Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R., & Ferrin, T. 1982. **A geometric approach to macromolecule-ligand interactions**. *J. Mol. Biol.*, **161**, 269–288. Academic Press. (ref. *p*.34, 35)

[Labeit *et al.*, 1997] Labeit, S., Kolmerer, B., & Linke, W.A. 1997. **The Giant Protein Titin: Emerging Roles in Physiology and Pathophysiology**. *Circ. Res.*, **80**, 290–294. American Heart Association. (ref. *p*.174)

[Lamarck, 1809] Lamarck, J.B. 1809. ***Philosophie Zoologique***. Translation by Elliot, H., Macmillan, 1914. (ref. *p*.118)

[Lang *et al.*, 2007] Lang, P.T., Aynechi, T., Moustakas, D., Shoichet, B., Kuntz, I.D., Brooijmans, N., & Oshiro, C.M. 2007. **Molecular Docking and Structure-Based Design**. *Pages 3–23 of:* Huang, Z. (ed), *Drug Discovery Research: New Frontiers in the Post-Genomic Era*. John Wiley & Sons. ISBN 978-0-471-67200-5. (ref. *p*.33, 51)

[Lang *et al.*, 1988] Lang, R., *et al.* 1988. ***GhostScript***. `http://pages.cs.wisc.edu/~ghost` (ref. *p*.253)

[Laskowski, 1995] Laskowski, R.A. 1995. **SURFNET: A program for visualizing molecular surfaces, cavities, and intermolecular interactions**. *J. Mol. Graph.*, **13**, 323–330. Elsevier Science. (ref. *p*.35)

[Laskowski *et al.*, 1996] Laskowski, R.A., Luscombe, N.M., Swindells, M.B., & Thornton, J.M. 1996. **Protein clefts in molecular recognition and function**. *Protein Science*, **5**, 2438–2452. Cold Spring Harbor Laboratory Press. (ref. *p*.35, 87)

[Laurie & Jackson, 2006] Laurie, A.T., & Jackson, R.M. 2006. **Methods for the prediction of protein-ligand binding sites for structure-based drug design and virtual ligand screening**. *Curr. Protein Pept. Sci.*, **7**, 395–406. Bentham. (ref. *p*.34, 51)

[LaValle & Kuffner, 1999] LaValle, S.M., & Kuffner, Jr., J.J. 1999. **Randomized Kinodynamic Planning**. *Pages 473–479 of: Proc. IEEE Int. Conf. Rob. Autom.*, vol. 1. IEEE. (ref. *p*.46)

[LaValle *et al.*, 1999] LaValle, S.M., Finn, P.W., Kavraki, L.E., & Latombe, J-C. 1999. **Efficient Database Screening for Rational Drug Design Using Pharmacophore-Constrained Conformational Search**. *Pages 250–260 of: Proc. Int. Conf. Comput. Mol. Biol.* ACM Press. (ref. *p*.47, 161)

[Leach *et al.*, 2006] Leach, A.R., Shoichet, B.K., & Peishoff, C.E. 2006. **Prediction of Protein-Ligand Interactions. Docking and Scoring: Successes and Gaps**. *J. Med. Chem.*, **49**, 5851–5855. American Chemical Society. (ref. *p*.20, 32, 51, 60)

[Leach *et al.*, 2010] Leach, A.R., Gillet, V.J., Lewis, R.A., & Taylor, R. 2010. **Three-Dimensional Pharmacophore Methods in Drug Discovery**. *J. Med. Chem.*, **53**, 539–558. American Chemical Society. (ref. *p*.24, 51)

[Lei *et al.*, 2004] Lei, M., Kuhn, L.A., Zavodsky, M.I., & Thorpe, M.F. 2004. **Sampling protein conformations and pathways**. *J. Comput. Chem.*, **25**, 1133–1148. Wiley Periodicals. (ref. *p*.41)

[Lengauer & Rarey, 1996] Lengauer, T., & Rarey, M. 1996. **Computational methods for biomolecular docking**. *Curr. Opin. Struct. Biol.*, **6**, 402–406. Current Biology. (ref. *p*.32)

[Levitt & Banaszak, 1992] Levitt, D.G., & Banaszak, L.J. 1992. **POCKET: A computer graphics method for identifying and displaying protein cavities and their surrounding amino acids**. *J. Mol. Graph.*, **10**, 229–234. Elsevier Science. (ref. *p*.35)

[Levitt & Lifson, 1969] Levitt, M., & Lifson, S. 1969. **Refinement of Protein Conformations using a Macromolecular Energy Minimization Procedure**. *J. Mol. Biol.*, **46**, 269–279. Elsevier Science. (ref. *p*.26)

[Li *et al.*, 2003] Li, L., Chen, R., & Weng, Z. 2003. **RDOCK: Refinement of Rigid-body Protein Docking**. *Proteins*, **53**, 693–707. Wiley-Liss. (ref. *p*.39)

[Liang *et al.*, 1998] Liang, J., Edelsbrunner, H., & Woodward, C. 1998. **Anatomy of protein pockets and cavities: Measurement of binding site geometry and implications for ligand design**. *Protein Science*, **7**, 1884–1897. Cold Spring Harbor Laboratory Press. (ref. *p*.175)

[Lichtarge & Sowa, 2002] Lichtarge, O., & Sowa, M.E. 2002. **Evolutionary predictions of binding surfaces and interactions**. *Curr. Opin. Struct. Biol.*, **12**, 21–27. Elsevier Science. (ref. *p*.34)

[Lin *et al.*, 1994] Lin, S.L., Nussinov, R., Fischer, D., & Wolfson, H.J. 1994. **Molecular surface representations by sparse critical points**. *Proteins*, **18**, 94–101. Wiley-Liss. (ref. *p*.24)

[Linge *et al.*, 2003] Linge, J.P., Williams, M.A., Spronk, C.A.E.M., Bonvin, A.M.J.J., & Nilges, M. 2003. **Refinement of protein structures in explicit solvent**. *Proteins*, **50**, 496–506. Wiley-Liss. (ref. *p*.175)

[Lorber & Shoichet, 1998] Lorber, D.M., & Shoichet, B.K. 1998. **Flexible ligand docking using conformational ensembles**. *Protein Science*, **7**, 938–950. Protein Society / Cambridge University Press. (ref. *p*.35, 42)

[Lorber & Shoichet, 2005] Lorber, D.M., & Shoichet, B.K. 2005. **Hierarchical docking of databases of multiple ligand conformations**. *Curr. Topics Med. Chem.*, **5**, 739–749. Bentham. (ref. *p*.43)

[Lunney, 2001] Lunney, E.A. 2001. **Computing in Drug Discovery — the design phase**. *Comput. Sci. Eng.*, **3**, 105–108. IEEE Computer Society / American Institute of Physics. (ref. *p*.19, 20, 49)

[Lybrand, 1995] Lybrand, T.P. 1995. **Ligand-protein docking and rational drug design**. *Curr. Opin. Struct. Biol.*, **5**, 224–228. Current Biology. (ref. *p*.18, 51)

[Maiorov & Abagyan, 1997] Maiorov, V., & Abagyan, R. 1997. **A New Method for Modeling Large-Scale Rearrangements of Protein Domains**. *Proteins*, **27**, 410–424. Wiley-Liss. (ref. *p*.26)

[Mancera *et al.*, 2004] Mancera, R.L., Källblad, P., & Todorov, N.P. 2004. **Ligand-Protein Docking using a Quantum Stochastic Tunnelling Optimization**. *J. Comput. Chem.*, **25**, 858–864. Wiley Periodicals. (ref. *p*.39)

[Mandell *et al.*, 2001] Mandell, J.G., Roberts, V.A., Pique, M.E., Kotlovyi, V., Mitchell, J.C., Nelson, E., Tsigelny, I., & Eyck, L.F. Ten. 2001. **Protein docking using continuum electrostatics and geometric fit**. *Protein Eng.*, **14**, 105–113. Oxford University Press. (ref. *p*.37, 60)

[Mangoni *et al.*, 1999] Mangoni, M., Roccatano, D., & Di Nola, A. 1999. **Docking of Flexible Ligands to Flexible Receptors in Solution by Molecular Dynamics Simulation**. *Proteins*, **35**, 153–162. Wiley-Liss. (ref. *p*.44)

[Marcia *et al.*, 2005] Marcia, R.F., Mitchell, J.C., & Rosen, J.B. 2005. **Iterative Convex Quadratic Approximation for Global Optimization in Protein Docking**. *Comput. Optim. Appl.*, **32**, 285–297. Springer. (ref. *p*.26)

[Mavridis *et al.*, 2007] Mavridis, L., Hudson, B.D., & Ritchie, D.W. 2007. **Toward High Throughput 3D Virtual Screening Using Spherical Harmonic Surface Representations**. *J. Chem. Inf. Model.*, **47**, 1787–1796. American Chemical Society. (ref. *p*.29)

[May *et al.*, 2003] May, A., Eisenhardt, S., Schmidt-Ehrenberg, J., & Cordes, F. 2003. ***Rigid Body Docking for Virtual Screening***. Tech. rept. ZIB-Report 03-47. Konrad-Zuse-Zentrum, Berlin. (ref. *p*.41)

[McCammon, 2005] McCammon, J.A. 2005. **Target flexibility in molecular recognition**. *Biochim. Biophys. Act.*, **1754**, 221–224. Elsevier Science. (ref. *p*.42)

[McGaughey *et al.*, 2007] McGaughey, G.B., Sheridan, R.P., Bayly, C.I., Culberson, J.C., Kreatsoulas, C., Lindsley, S., Maiorov, V., Truchon, J-F., & Cornell, W.D. 2007. **Comparison of Topological, Shape, and Docking Methods in Virtual Screening**. *J. Chem. Inf. Model.*, **47**, 1504–1519. American Chemical Society. (ref. *p*.20)

[McMartin & Bohacek, 1997] McMartin, C., & Bohacek, R.S. 1997. **QXP: Powerful, rapid computer algorithms for structure-based drug design**. *J. Comput. Aided Mol. Des.*, **11**, 333–344. Kluwer Academic Publishers. (ref. *p*.42)

[Meijering, 2002] Meijering, E.H.W. 2002. **A chronology of interpolation: From ancient astronomy to modern signal and image processing**. *Proc. IEEE*, **90**, 319–342. IEEE. (ref. *p*.121)

[Meiler & Baker, 2006] Meiler, J., & Baker, D. 2006. **ROSETTALIGAND: Protein-Small Molecule Docking with Full Side-Chain Flexibility**. *Proteins*, **65**, 538–548. Wiley-Liss. (ref. *p*.40, 81)

[Miller *et al.*, 1994] Miller, M.D., Kearsley, S.K., Underwood, D.J., & Sheridan, R.P. 1994. **FLOG: A system to select quasi-flexible ligands complementary to a receptor of known three-dimensional structure**. *J. Comput. Aided Mol. Des.*, **8**, 153–174. Springer. (ref. *p*.42)

[Miller, 2005] Miller, W.H. 2005. **Quantum Dynamics of Complex Molecular Systems**. *Proc. Natl. Acad. Sci. USA*, **102**, 6660–6664. National Academy of Sciences. (ref. *p*.27, 48, 51)

[Mizuguchi *et al.*, 1998] Mizuguchi, K., Deane, C.M., Blundell, T.L., Johnson, M.S., & Overington, J.P. 1998. **JOY — sequence-structure representation and analysis**. *Bioinformatics*, **14**, 617–623. Oxford University Press. (ref. *p*.25)

[Morozov *et al.*, 2004] Morozov, A.V., Kortemme, T., Tsemekhman, K., & Baker, D. 2004. **Close agreement between the orientation dependence of hydrogen bonds and QM calculations**. *Proc. Natl. Acad. Sci. USA*, **101**, 6946–6951. National Academy of Sciences. (ref. *p*.27)

[Morris & Lim-Wilby, 2008] Morris, G.M., & Lim-Wilby, M. 2008. **Molecular Docking**. *Pages 365–382 of:* Kukol, A. (ed), *Molecular Modeling of Proteins*. Humana Press. ISBN 1-588-29864-7. (ref. *p*.33, 51)

[Morris *et al.*, 1998] Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., Belew, R.K., & Olson, A.J. 1998. **Automated Docking Using a Lamarckian GA and an Empirical Binding Free Energy Function**. *J. Comput. Chem.*, **19**, 1639–1662. Wiley Periodicals. (ref. *p*.37, 44, 48, 60, 72, 118)

[Morris *et al.*, 2005] Morris, R.J., Najmonovich, R.J., Kahraman, A., & Thornton, J.M. 2005. **Real spherical harmonic expansion coefficients as 3D shape descriptors for protein binding pocket and ligand comparisons**. *Bioinformatics*, **21**, 2347–2355. Oxford University Press. (ref. *p*.29)

[Nagata *et al.*, 2002] Nagata, H., Mizushima, H., & Tanaka, H. 2002. **Concept and prototype of protein-ligand docking simulator with force feedback technology**. *Bioinformatics*, **18**, 140–146. Oxford University Press. (ref. *p*.33)

[Nelder & Mead, 1965] Nelder, J.A., & Mead, R. 1965. **A Simplex Method for Function Minimization**. *Comput. J.*, **7**, 308–313. Brit. Comput. Soc. (ref. *p*.78, 118)

[OpenBabel, 2008] OpenBabel. 2008. ***OpenBabel Tools and Libraries***. `http://www.openbabel.org` (ref. *p*.64)

[Orengo *et al.*, 1999] Orengo, C.A., Todd, A.E., & Thornton, J.M. 1999. **From protein structure to function**. *Curr. Opin. Struct. Biol.*, **9**, 374–382. Elsevier Science. (ref. *p*.28)

[Österberg *et al.*, 2002] Österberg, F., Morris, G.M., & Sanner, M.F. 2002. **Automated Docking to Multiple Target Structures: Incorporation of Protein Mobility and Structural Water Heterogeneity in AutoDock**. *Proteins*, **46**, 34–40. Wiley-Liss. (ref. *p*.44)

[Palma *et al.*, 2000] Palma, P.N., Krippahl, L., Wampler, J.E., & Moura, J.J.G. 2000. **BiGGER: A New (Soft) Docking Algorithm for Predicting Protein Interactions**. *Proteins*, **39**, 372–384. Wiley-Liss. (ref. *p*.38)

[Pan *et al.*, 2003] Pan, Y., Huang, N., Cho, S., & MacKerell, A.D. 2003. **Consideration of Molecular Weight during Compound Selection in Virtual Target-Based Database Screening**. *J. Chem. Inf. Comput. Sci.*, **43**, 267–272. American Chemical Society. (ref. *p*.155)

[Papoulis, 1962] Papoulis, A. 1962. ***The Fourier Integral and Its Applications***. McGraw-Hill. ISBN 0-070-48447-3. (ref. *p*.36, 177)

[Parsons & Canny, 1994] Parsons, D., & Canny, J. 1994. **Geometric Problems in Molecular Biology and Robotics**. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **2**, 322–330. Am. Assoc. Artificial Intelligence. (ref. *p*.20)

[Pattabiraman *et al.*, 1985] Pattabiraman, N., Levitt, M., Ferrin, T.E., & Langridge, R. 1985. **Computer Graphics in Real-time Docking with Energy Calculation and Minimization**. *J. Comput. Chem.*, **6**, 432–436. Wiley Periodicals. (ref. *p*.33)

[PDB, 1977] PDB. 1977. ***Protein Data Bank***. `http://www.pdb.org`

[Pei *et al.*, 2006] Pei, J., Wang, Q., Liu, Z., Li, Q., Yang, K., & Lai, L. 2006. **PSI-DOCK: Towards Highly Efficient and Accurate Flexible Ligand Docking**. *Proteins*, **62**, 934–946. Wiley-Liss. (ref. *p*.48, 162)

[Perola *et al.*, 2004] Perola, E., Walters, W.P., & Charifson, P.S. 2004. **A Detailed Comparison of Current Docking and Scoring Methods on Systems of Pharmaceutical Relevance**. *Proteins*, **56**, 235–249. Wiley-Liss. (ref. *p*.32)

[Peters *et al.*, 1996] Peters, K.P., Fauck, J., & Frömmel, C. 1996. **The Automatic Search for Ligand Binding Sites in Proteins of Known Three-dimensional Structure Using only Geometric Criteria**. *J. Mol. Biol.*, **256**, 201–213. Academic Press. (ref. *p*.34)

[Peters *et al.*, 2006] Peters, M.B., Raha, K., & Merz, Jr., K.M. 2006. **Quantum mechanics in structure-based drug design**. *Curr. Opin. Drug. Discov. Devel.*, **9**, 370–379. Thomson Reuters. (ref. *p*.27)

[Phillips *et al.*, 2005] Phillips, J.C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R.D., Kale, L., , & Schulten, K. 2005. **Scalable Molecular Dynamics with NAMD**. *J. Comput. Chem.*, **26**, 1781–1802. Wiley Periodicals. (ref. *p*.26)

[Pitt-Francis & Featherstone, 1998] Pitt-Francis, J., & Featherstone, R. 1998. **Automatic Generation of Sphere Hierarchies from CAD Data**. *Pages 324–329 of: Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1. IEEE. (ref. *p*.55)

[Poirrette *et al.*, 1997] Poirrette, A.R., Artymiuk, P.J., Rice, D.W., & Willett, P. 1997. **Comparison of protein surfaces using a genetic algorithm**. *J. Comput. Aided Mol. Des.*, **11**, 557–569. Springer. (ref. *p*.29, 37, 60)

[Ponder & Richards, 1987] Ponder, J.W., & Richards, F.M. 1987. **An efficient Newton-like method for molecular mechanics energy minimization of large molecules**. *J. Comput. Chem.*, **8**, 1016–1024. Wiley Periodicals. (ref. *p*.26)

[Press *et al.*, 1992] Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. 1992. ***Numerical Recipes in C: The Art of Scientific Computing***. 2nd edn. Cambridge University Press. ISBN 0-521-43720-2. (ref. *p*.180)

[Putta & Beroza, 2007] Putta, S., & Beroza, P. 2007. **Shapes of Things: Computer Modeling of Molecular Shape in Drug Discovery**. *Curr. Topics Med. Chem.*, **7**, 1514–1524. Bentham. (ref. *p*.24, 51)

[Rader *et al.*, 2002] Rader, A.J., Hespenheide, B.M., Kuhn, L.A., & Thorpe, M.F. 2002. **Protein unfolding: rigidity lost**. *Proc. Natl. Acad. Sci. USA*, **99**, 3540–3545. National Academy of Sciences. (ref. *p*.41)

[Raha *et al.*, 2007] Raha, K., Peters, M.B., Wang, B., Yu, N., Wollacott, A.M., Westerhoff, L.M., & Merz. Jr., K.M. 2007. **The role of quantum mechanics in structure-based drug design**. *Drug Discovery Today*, **12**, 725–731. Elsevier Science. (ref. *p*.27, 51)

[Rangwala *et al.*, 2006] Rangwala, H., Deronne, K., & Karypis, G. 2006. ***Protein Structure Prediction using String Kernels***. Tech. rept. DTC Research Report. University of Minnesota. (ref. *p*.31)

[Richards, 1977] Richards, F.M. 1977. **Areas, Volumes, Packing, and Protein Structure**. *Ann. Rev. Biophys. Bioeng.*, **6**, 151–176. Annual Reviews. (ref. *p*.176)

[Richards, 2002] Richards, W.G. 2002. **Virtual screening using grid computing — the screensaver project**. *Nature Reviews (Drug Discovery)*, **1**, 551–555. Nature Publishing. (ref. *p*.20, 50)

[Richards, 2007] Richards, W.G. 2007. **From Diatomics to Drugs and Dividends**. *J. Mol. Graph. Model.*, **26**, 596–601. Elsevier Science. (ref. *p*.19)

[Ritchie, 1998] Ritchie, D.W. 1998. ***Parametric Protein Shape Recognition***. Ph.D. thesis, University of Aberdeen. (ref. *p*.29)

[Ritchie & Kemp, 1999] Ritchie, D.W., & Kemp, G.J.L. 1999. **Fast Computation, Rotation and Comparison of Low Resolution Spherical Harmonic Molecular Surfaces**. *J. Comput. Chem.*, **20**, 383–395. Wiley Periodicals. (ref. *p*.29)

[Ritchie & Kemp, 2000] Ritchie, D.W., & Kemp, G.J.L. 2000. **Protein Docking using Spherical Polar Fourier Correlations**. *Proteins*, **39**, 178–194. Wiley-Liss. (ref. *p*.38)

[Robertson & Murphy, 1997] Robertson, A.D., & Murphy, K.P. 1997. **Protein Structure and the Energetics of Protein Stability**. *Chem. Rev.*, **97**, 1251–1267. American Chemical Society. (ref. *p*.176)

[Rocchia *et al.*, 2002] Rocchia, W., Sridharan, S., Nicholls, A., Alexov, E., Chiabrera, A., & Honig, B. 2002. **Rapid Grid-Based Construction of the Molecular Surface and the Use of Induced Surface Charge to Calculate Reaction Field Energies: Applications to the Molecular Systems and Geometric Objects**. *J. Comput. Chem.*, **23**, 128–137. Wiley Periodicals. (ref. *p*.60)

[Rosales-Hernandez *et al.*, 2009] Rosales-Hernandez, M.C., Bermúdez-Lugo, J., Garcia, J., Trujillo-Ferrara, J., & Correa-Basurto, J. 2009. **Molecular Modeling Applied to Anti-Cancer Drug Development**. *Anti-Cancer Agents Med. Chem.*, **9**, 230–238. Bentham. (ref. *p*.27)

[Ruppert *et al.*, 1997] Ruppert, J., Welch, W., & Jain, A.N. 1997. **Automatic identification and representation of protein binding sites for molecular docking**. *Protein Science*, **6**, 524–533. Cambridge University Press. (ref. *p*.35)

[Sánchez & Šali, 1997] Sánchez, R., & Šali, A. 1997. **Advances in Comparative Protein-Structure Modelling**. *Curr. Opin. Struct. Biol.*, **7**, 206–214. Current Biology. (ref. *p*.28)

[Sandak *et al.*, 1998] Sandak, B., Wolfson, H.J., & Nussinov, R. 1998. **Flexible Docking Allowing Induced Fit in Proteins: Insights From an Open to Closed Conformational Isomers**. *Proteins*, **32**, 159–174. Wiley-Liss. (ref. *p*.47)

[Sanner *et al.*, 1995] Sanner, M.F., Olson, A.J., & Spehner, J-C. 1995. **Fast and Robust Computation of Molecular Surfaces**. *Pages 406–407 of: Proc. 11th Ann. Symp. Comp. Geom.* ACM Press. (ref. *p*.188)

[Schenk *et al.*, 2000] Schenk, C., *et al.* 2000. ***MiKTeX***. `http://www.miktex.org` (ref. *p*.253)

[Schlessinger *et al.*, 2006] Schlessinger, A., Yachdav, G., & Rost, B. 2006. **PROFbval — predict flexible and rigid residues in proteins**. *Bioinformatics*, **22**, 891–893. Oxford University Press. (ref. *p*.41)

[Schmidt *et al.*, 1993] Schmidt, M., Baldridge, K.K., Boatz, J.A., Elbert, S., M. Gordon, J.H. Jenson, Koeski, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., & Montgomery, J.A. 1993. **The General Atomic and Molecular Electronic Structure System**. *J. Comput. Chem.*, **14**, 1347–1363. Wiley Periodicals. (ref. *p*.28)

[Schnecke & Kuhn, 2000] Schnecke, V., & Kuhn, L.A. 2000. **Virtual screening with solvation and ligand-induced complementarity**. *Perspect. Drug Discov. Des.*, **20**, 171–190. Kluwer Academic Publishers. (ref. *p*.60)

[Schneidman-Duhovny *et al.*, 2003] Schneidman-Duhovny, D., Inbar, Y., Polak, V., Shatsky, M., Halperin, I., Benyamini, H., Barzilai, A., Dror, O., Haspel, N., Nussinov, R., & Wolfson, H.J. 2003. **Taking Geometry to Its Edge: Fast Unbound Rigid (and Hinge-Bent) Docking**. *Proteins*, **52**, 107–112. Wiley-Liss. (ref. *p*.40)

[Scott *et al.*, 2006] Scott, K.A., Alonso, D.O.V., Pan, Y., & Daggett, V. 2006. **Importance of Context in Protein Folding: Secondary Structural Propensities versus Tertiary Contact-Assisted Secondary Structure Formation**. *Biochemistry*, **45**, 4153–4163. American Chemical Society. (ref. *p*.47)

[Sharf & Shamir, 2004] Sharf, A., & Shamir, A. 2004. **Feature-sensitive 3D Shape Matching**. *Pages 596–599 of: Proc. Comp. Graph. Int.* IEEE. (ref. *p*.27)

[Sherwood, 2000] Sherwood, P. 2000. **Hybrid Quantum and Molecular Mechanics Approaches**. *Pages 285–305 of: Proc. Modern Methods and Algorithms of Quantum Chemistry*, vol. 3. John von Neumann Institute for Computing. (ref. *p*.27)

[Shikama & Matsuoka, 2003] Shikama, K., & Matsuoka, A. 2003. **Human haemoglobin: A new paradigm for oxygen binding involving two types of $\alpha\beta$ contacts**. *Eur. J. Biochem.*, **270**, 4041–4051. FEBS. (ref. *p*.32)

[Shoemake, 1992] Shoemake, K. 1992. **Uniform Random Rotations**. *Pages 124–132 of:* Kirk, D. (ed), *Graphics Gems III*. Academic Press. ISBN 0-124-09673-5. (ref. *p*.85)

[Singh *et al.*, 1999] Singh, A.P., Latombe, J-C., & Brutlag, D.L. 1999. **A Motion Planning Approach to Flexible Ligand Binding**. *Pages 252–261 of: Proc. Int. Conf. Intell. Syst. Mol. Biol.* Am. Assoc. Artificial Intelligence. (ref. *p*.46)

[Sinha & Smith-Gill, 2002] Sinha, N., & Smith-Gill, S.J. 2002. **Protein Structure to Function via Dynamics**. *Protein Pept. Lett.*, **9**, 367–377. Bentham. (ref. *p*.176)

[Skeel *et al.*, 2002] Skeel, R.D., Tezcan, I., & Hardy, D.J. 2002. **Multiple Grid Methods for Classical Molecular Dynamics**. *J. Comput. Chem.*, **23**, 673–684. Wiley Periodicals. (ref. *p*.27)

[Skjevik *et al.*, 2009] Skjevik, Å.A., Teigen, K., & Martinez, A. 2009. **Overview of computational methods employed in early-stage drug discovery**. *Fut. Med. Chem.*, **1**, 49–63. Future Science. (ref. *p*.39, 48, 51)

[Skolnick & Brylinski, 2009] Skolnick, J., & Brylinski, M. 2009. **FINDSITE: a combined evolution/structure-based approach to protein function prediction**. *Brief. Bioinform.*, **10**, 378–391. Oxford University Press. (ref. *p*.34)

[Skolnick *et al.*, 1997] Skolnick, J., Kolinski, A., & Ortiz, A.R. 1997. **MONSSTER — A Method for Folding Globular Proteins with a Small Number of Distance Restraints**. *J. Mol. Biol.*, **265**, 217–241. Academic Press. (ref. *p*.31)

[Skone, 2010] Skone, G.S. 2010. ***Stratagems for Effective Function Evaluation in Computational Chemistry***. Ph.D. thesis, University of Oxford. (ref. *p*.253)

[Skone & Cameron, 2007] Skone, G.S., & Cameron, S.A. 2007. **Protein Structure Computation**. *Pages 135–140 of: Proc. FBIT 2007*. IEEE. (ref. *p*.171)

[Skone *et al.*, 2009] Skone, G.S., Voiculescu, I., & Cameron, S.A. 2009. **Knowing When To Give Up: Early-Rejection Stratagems in Ligand Docking**. *J. Comput. Aided Mol. Des.*, **23**, 715–724. Springer. (ref. *p*.128, 137)

[Skórczyński & Deorowicz, 2005] Skórczyński, A., & Deorowicz, S. 2005. ***LaTeX Editor***. `http://www.latexeditor.org` (ref. *p*.253)

[Smith & Sternberg, 2002] Smith, G.R., & Sternberg, M.J.E. 2002. **Prediction of protein-protein interactions by docking methods**. *Curr. Opin. Struct. Biol.*, **12**, 28–35. Elsevier Science. (ref. *p*.32)

[Snow, 2008] Snow, C.D. 2008. **Hunting for predictive computational drug-discovery models**. *Expert Rev. Anti Infect. Ther.*, **6**, 291–293. Expert Reviews. (ref. *p*.30, 51)

[Snyder *et al.*, 2005] Snyder, D.A., Bhattacharya, A., Huang, Y.J., & Montelione, G.T. 2005. **Assessing Precision and Accuracy of Protein Structures Derived From NMR Data**. *Proteins*, **59**, 655–661. Wiley-Liss. (ref. *p*.176)

[Sobolev *et al.*, 1996] Sobolev, V., Wade, R.C., Vriend, G., & Edelman, M. 1996. **Molecular Docking Using Surface Complementarity**. *Proteins*, **25**, 120–129. Wiley-Liss. (ref. *p*.37)

[Song *et al.*, 2009] Song, C.M., Lim, S.J., & Tong, J.C. 2009. **Recent advances in computer-aided drug design**. *Brief. Bioinform.*, **10**, 579–591. Oxford University Press. (ref. *p*.33, 51)

[Soto, 2001] Soto, C. 2001. **Protein misfolding and disease; protein refolding and therapy**. *FEBS Letters*, **498**, 204–207. Elsevier Science. (ref. *p*.17, 30)

[Sotriffer & Klebe, 2002] Sotriffer, C., & Klebe, G. 2002. **Identification and mapping of small-molecule binding sites in proteins: computational tools for structure-based drug design**. *Il Farmaco*, **57**, 243–251. Elsevier Science. (ref. *p*.34, 51, 87)

[Sotriffer *et al.*, 2002] Sotriffer, C.A., Gohlke, H., & Klebe, G. 2002. **Docking into Knowledge-Based Potential Fields: A Comparative Evaluation of DrugScore**. *J. Med. Chem.*, **45**, 1967–1970. American Chemical Society. (ref. *p*.44)

[StarUML, 2005] StarUML. 2005. ***StarUML***. `http://staruml.sourceforge.net` (ref. *p*.253)

[Symyx, 1982-2009] Symyx. 1982-2009. ***Available Chemicals Directory***. Symyx Technologies, Inc. `http://www.symyx.com/products/databases/sourcing/acd` (ref. *p*.50)

[Szustakowski & Weng, 2000] Szustakowski, J.D., & Weng, Z. 2000. **Protein Structure Alignment Using a Genetic Algorithm**. *Proteins*, **38**, 428–440. Wiley-Liss. (ref. *p*.37)

[Taft *et al.*, 2008] Taft, C.A., Semighini, E.P., & Silva, C.H.T.P. 2008. **Applications of quantum mechanics to drug design**. *Pages 1–32 of:* Taft, C.A., & Silva, C.H.T.P. (eds), *Current Methods in Medicinal Chemistry and Biological Physics*. Research Signpost. ISBN 978-81-308-0292-3. (ref. *p*.27, 51)

[Tatsumi *et al.*, 2004] Tatsumi, R., Fukunishi, Y., & Nakamura, H. 2004. **A Hybrid Method of Molecular Dynamics and Harmonic Dynamics for Docking of Flexible Ligand to Flexible Receptor**. *J. Comput. Chem.*, **25**, 1995–2005. Wiley Periodicals. (ref. *p*.44)

[Taylor & Burnett, 2000] Taylor, J.S., & Burnett, R.M. 2000. **DARWIN — A Program for Docking Flexible Molecules**. *Proteins*, **41**, 173–191. Wiley-Liss. (ref. *p*.44)

[Taylor *et al.*, 2002] Taylor, R.D., Jewsbury, P.J., & Essex, J.W. 2002. **A review of protein-small molecule docking methods**. *J. Comput. Aided Mol. Des.*, **16**, 151–166. Kluwer Academic Publishers. (ref. *p*.32, 51)

[Teague, 2003] Teague, S.J. 2003. **Implications of Protein Flexibility for Drug Discovery**. *Nat. Rev. Drug Discovery*, **2**, 527–541. Nature Publishing. (ref. *p*.40, 51)

[Teodoro & Kavraki, 2003] Teodoro, M.L., & Kavraki, L.E. 2003. **Conformational Flexibility Models for the Receptor in Structure Based Drug Design**. *Curr. Pharm. Des.*, **9**, 1635–1648. Bentham. (ref. *p*.44, 51)

[Teodoro *et al.*, 2001] Teodoro, M.L., Phillips, Jr., G.N., & Kavraki, L.E. 2001. **Molecular Docking: A Problem with Thousands Of Degrees Of Freedom**. *Pages 960–965 of: Proc. IEEE Int. Conf. Rob. Autom.*, vol. 1. IEEE. (ref. *p*.41, 47)

[Teodoro *et al.*, 2002] Teodoro, M.L., Phillips, Jr., G.N., & Kavraki, L.E. 2002. **A Dimensionality Reduction Approach to Modeling Protein Flexibility**. *Pages 299–308 of: Proc. Int. Conf. Comput. Biol.*, vol. 1. ACM Press. (ref. *p*.47)

[Todorov *et al.*, 2003] Todorov, N.P., Mancera, R.L., & Monthoux, P.H. 2003. **A new quantum stochastic tunnelling optimisation method for protein-ligand docking**. *Chem. Phys. Lett.*, **369**, 257–263. Elsevier Science. (ref. *p*.39)

[Tompa *et al.*, 2005] Tompa, M., Li, N., Bailey, T.L., Church, G.M., de Moor, B., Eskin, E., Favorov, A.V., Frith, M.C., Fu, Y., Kent, W.J., Makeev, V.J., Mironov, A.A., Noble, W.S., Pavesi, G., Pesole, G., Régnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C., & Zhu, Z. 2005. **Assessing computational tools for the discovery of transcription factor binding sites**. *Nature Biotechnology*, **23**, 137–144. Nature Publishing. (ref. *p*.51)

[Totrov & Abagyan, 2008] Totrov, M., & Abagyan, R. 2008. **Flexible ligand docking to multiple receptor conformations: a practical alternative**. *Curr. Opin. Struct. Biol.*, **18**, 178–184. Elsevier Science. (ref. *p*.42)

[Tovchigrechko *et al.*, 2002] Tovchigrechko, A., Wells, C.A., & Vakser, I.A. 2002. **Docking of Protein Models**. *Protein Science*, **11**, 1888–1896. Cold Spring Harbor Laboratory Press. (ref. *p*.36)

[Vakser, 1996] Vakser, I.A. 1996. **Low-Resolution Docking — Prediction of Complexes for Underdetermined Structures**. *Biopolymers*, **39**, 455–464. John Wiley & Sons. (ref. *p*.36)

[Vakser *et al.*, 1999] Vakser, I.A., Matar, O.G., & Lam, C.F. 1999. **A systematic study of low-resolution recognition in protein-protein complexes**. *Proc. Natl. Acad. Sci. USA*, **96**, 8477–8482. National Academy of Sciences. (ref. *p*.36)

[Venkatachalam *et al.*, 2003] Venkatachalam, C.M., Jiang, X., Oldfield, T., & Waldman, M. 2003. **LigandFit: a novel method for the shape-directed rapid docking of ligands to protein active sites**. *J. Mol. Graph. Model.*, **21**, 289–307. Elsevier Science. (ref. *p*.35, 39)

[Verdonk *et al.*, 2003] Verdonk, M.L., Cole, J.C., Hartshorn, M.J., Murray, C.W., & Taylor, R.D. 2003. **Improved Protein-Ligand Docking Using GOLD**. *Proteins*, **52**, 609–623. Wiley-Liss. (ref. *p*.43, 199)

[Villoutreix *et al.*, 2007] Villoutreix, B.O., Renault, N., Lagorce, D., Sperandio, O., Montes, M., & Miteva, M.A. 2007. **Free resources to assist structure-based virtual ligand screening experiments**. *Curr. Protein Pept. Sci.*, **8**, 381–411. Bentham. (ref. *p*.32, 51)

[Villoutreix *et al.*, 2009] Villoutreix, B.O., Eudes, R., & Miteva, M.A. 2009. **Structure-based virtual ligand screening: recent success stories**. *Comb. Chem. High Throughput Screening*, **12**, 1000–1016. Bentham. (ref. *p*.33, 51)

[von Itzstein *et al.*, 1993] von Itzstein, M., Wu, W.Y., Kok, G.B., Pegg, M.S., Dyason, J.C., Jin, B., Van Phan, T., Smythe, M.L., White, H.F., Oliver, S.W., Colman, P.M., Varghese, J.N., Ryan, D.M., Woods, J.M., Bethell, R.C., Hotham, V.J., Cameron, J.M., & Penn, C.R. 1993. **Rational design of potent sialidase-based inhibitors of influenza virus replication**. *Nature*, **363**, 418–423. Nature Publishing. (ref. *p*.19)

[Wall, 1996] Wall, M. 1996. ***GAlib***. Massachusetts Institute of Technology. `http://lancet.mit.edu/ga` (ref. *p*.65)

[Wang, 2003] Wang, R. 2003. ***X-Score***. University of Michigan. `http://sw16.im.med.umich.edu/software/xtool` (ref. *p*.61)

[Wang *et al.*, 2000] Wang, R., Gao, Y., & Lai, L. 2000. **Calculating partition coefficient by atom-additive method**. *Perspect. Drug Discov. Des.*, **19**, 47–66. Kluwer Academic Publishers. (ref. *p*.63)

[Wang *et al.*, 2002] Wang, R., Lai, L., & 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. Kluwer Academic Publishers. (ref. *p*.60, 61, 62, 191, 196)

[Wang *et al.*, 2003] Wang, R., Lu, Y., & Wang, S. 2003. **Comparative Evaluation of 11 Scoring Functions for Molecular Docking**. *J. Med. Chem.*, **46**, 2287–2303. American Chemical Society. (ref. *p*.60, 61, 197)

[Warren *et al.*, 2006] Warren, G.L., Webster Andrews, C., 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., & Head, M.S. 2006. **A Critical Assessment of Docking Programs and Scoring Functions**. *J. Med. Chem.*, **49**, 5912–5931. American Chemical Society. (ref. *p*.51, 60)

[Wass & Sternberg, 2009] Wass, M.N., & Sternberg, M.J.E. 2009. **Prediction of ligand binding sites using homologous structures and conservation at CASP8**. *Proteins*, **77**, 147–151. Wiley-Liss. (ref. *p*.34)

[Wei *et al.*, 2002] Wei, B.Q., Baase, W.A., Weaver, L.H., Matthews, B.W., & Shoichet, B.K. 2002. **A Model Binding Site for Testing Scoring Functions in Molecular Docking**. *J. Mol. Biol.*, **322**, 339–355. Academic Press. (ref. *p*.35)

[Wells *et al.*, 2005] Wells, S., Menor, S., Hespenheide, B., & Thorpe, M.F. 2005. **Constrained geometric simulation of diffusive motion in proteins**. *Physical Biology*, **2**, 127–136. Institute of Physics. (ref. *p*.41)

[Williamson, 1987] Williamson, J.F. 1987. **Random selection of points distributed on curved surfaces**. *Phys. Med. Biol.*, **32**, 1311–1319. Institute of Physics. (ref. *p*.188)

[Wilson & Madsen, 2001] Wilson, P.R., & Madsen, L. 2001. ***The Memoir Class for Configurable Typesetting***. `http://tug.ctan.org/tex-archive/macros/latex/contrib/memoir` (ref. *p*.253)

[Wodak & Méndez, 2004] Wodak, S.J., & Méndez, R. 2004. **Prediction of protein-protein interactions: the CAPRI experiment, its evaluation and implications**. *Curr. Opin. Struct. Biol.*, **14**, 242–249. Elsevier Science. (ref. *p*.30)

[Wüthrich, 1986] Wüthrich, K. 1986. ***NMR of Proteins and Nucleic Acids***. John Wiley & Sons. ISBN 0-471-82893-9. (ref. *p*.175)

[Wüthrich, 1990] Wüthrich, K. 1990. **Protein Structure Determination in Solution by NMR Spectroscopy**. *J. Biol. Chem.*, **265**, 22059–22062. American Society for Biochemistry and Molecular Biology. (ref. *p*.175)

[Xu *et al.*, 2000] Xu, D., Xu, Y., & Uberbacher, E.C. 2000. **Computational Tools For Protein Modelling**. *Curr. Protein Pept. Sci.*, **1**, 1–21. Bentham. (ref. *p*.30)

[Yao *et al.*, 2003] Yao, H., Kristensen, D.M., Mihalek, I., Sowa, M.E., Shaw, C., Kimmel, M., Kavraki, L., & Lichtarge, O. 2003. **An Accurate, Sensitive, and Scalable Method to Identify Functional Sites**. *J. Mol. Biol.*, **326**, 255–261. Academic Press. (ref. *p*.34)

[Yershova & LaValle, 2006] Yershova, A., & LaValle, S.M. 2006. **Improving motion planning algorithms by efficient nearest-neighbor searching**. *Transactions on Robotics*, **22**. IEEE. (ref. *p*.46)

[Yershova *et al.*, 2005] Yershova, A., Jaillet, L., Siméon, T., & LaValle, S.M. 2005. **Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain**. *Pages 3856–3861 of: Proc. IEEE Int. Conf. Rob. Autom.* IEEE. (ref. *p*.46)

[Zavodszky *et al.*, 2004] Zavodszky, M.I., Lei, M., Thorpe, M.F., Day, A.R., & Kuhn, L.A. 2004. **Modelling Correlated Main-Chain Motions in Proteins for Flexible Molecular Recognition**. *Proteins*, **57**, 243–261. Wiley-Liss. (ref. *p*.41)

[Zeng, 2000] Zeng, J. 2000. **Computational Structure-Based Design of Inhibitors that Target Protein Surfaces**. *Comb. Chem. High Throughput Screening*, **3**, 355–362. Bentham. (ref. *p*.50)

[ZINC, 2005] ZINC. 2005. ***ZINC Is Not Commercial***. `http://zinc.docking.org`

[Zsoldos *et al.*, 2003] Zsoldos, Z., Szabo, I., Szabo, Z., & Johnson, A.P. 2003. **Software tools for structure based rational drug design**. *J. Mol. Struct. (Theochem)*, **666**, 659–665. Elsevier Science. (ref. *p*.45)

All websites checked on 1st May 2010.

List of Tests Performed

For explanations of the notation, see Appendix E (*p*.195).

Index

Program names are listed in *italics*;
Implementation classes, functions, and inputs are printed in `monospaced` type.

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

Colophon

This document was typeset using LaTeX, MiKTeX [Schenk *et al.*, 2000], the Memoir package [Wilson & Madsen, 2001], and much wrangling [Skone, 2010]. The text font is URW Palladio, a Palatino clone, with additional shapes from Pazo Math. The headings and captions are set in URW Classico (Optima), and the monospaced code font is Bera Mono, based on Bitstream Vera. The source files were written using LaTeX Editor (LEd) [Skórczyński & Deorowicz, 2005], and the index was compiled with the help of the MakeIndex tool and my own program Tin. Bibliographic references were managed using Microsoft Excel and Access, and then Word was used to convert that data into a BibTeX source list. Matilda would have been helpful, but unfortunately arrived too late.

The graphs were produced by Excel; their data was extracted from *DOX*'s outputs using utilities I created in Borland (now Embarcadero) Delphi. Most of the illustrations were drawn in PowerPoint, and the UML diagram in Figure F.1 was constructed with StarUML [StarUML, 2005]. Where necessary, PrimoPDF rendered these graphics into Portable Document Format (PDF); GhostScript [Lang *et al.*, 1988] was then used to generate Encapsulated PostScript (EPS) files for printing. Molecular line diagrams were produced with NOC [Chen *et al.*, 2007]. Three-dimensional molecular images were ray-traced using PyMOL [DeLano, 2006] and saved as Portable Network Graphics (PNG) files. These and screen output bitmaps were converted to EPS by GraphicsMagick [GraphicsMagick, 2003].

GSS. Oxford, Andover, and Seattle. 2005–2010.

end.