The Dissertation Committee for Md. Muhibur Rasheed
certifies that this is the approved version of the following dissertation:

Predicting Multibody Assembly of Proteins

Committee:

Chandrajit Bajaj, Supervisor

Ron Elber

Kristen Grauman

Adam Klivans

Greg Plaxton

Predicting Multibody Assembly of Proteins

by

Md. Muhibur Rasheed, B.S.C.S.E, M.S.C.S.E

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2014

Dedicated to my parents

Acknowledgments

I wish to thank the multitudes of people who have supported me throughout my life and specially during the years in graduate school, in so many ways.

First of all, I thank Prof. Bajaj, my supervisor, not just for introducing me to many exciting problems, guiding me, correcting me, and giving me ideas, but also for showing me what it truly means to be dedicated to research, to be open to new concepts and topics, and to keep inquiring. I would also like to convey my heartfelt gratitude to my committee members Professors Ron Elber, Greg Plaxton, Adam Klivans and Kristen Grauman, all of whom were available and responsive whenever I needed them and their questions and comments have definitely helped make improve this thesis both in content and in presentation.

I was blessed to be part of a vibrant group of people in our research lab. Among the post-doctoral fellows in the lab, I would first like to thank Dr. Rezaul Alam Chowdhury, now a faculty member at SUNY Stony Brook, who immensely helped me to adjust to a new country and culture and also to get familiar with the research at the lab. We worked together on several projects including neighborhood data structure and protein-protein docking. I would also like to thank Dr. Alex Rand and Dr. Qin Zhang both of whom helped me with projects including the dynamic surface and modeling of large molecular complexes. I thank Dr. Deukhyun Cha for helping me with software implementation issues. I also thank Professor Paolo

Guidugli and Andrea Michelleti from University of Rome for discussions and ideas regarding the mechanical aspects of virus assembly.

Among my fellow graduate students, I am most indebted to Dr. Radhakrishna Bettadapura, for the time when we pair-programmed flexibility modeling code for proteins, and for all the other code written by him, that I ended up using for my research. I am also grateful to Dr. John Edwards, Maysam Moussalem, Eric Daniel, and James Delfeld for many discussions on their and my research which often gave me new ideas and solutions, and always increased my knowledge. I am grateful to Nathan Clement with whom I have recently been developing uncertainty quantification methods for optimization and modeling problems.

While discovery and development are exciting, the path to them can sometimes be quite tedious. The people who helped me most in this regard were my wife Fayeza and my children Nawar and Abraar. They never failed to lift me up when a particularly difficult proof, or some programming bug conspired to keep me down. They make me whole as a person and gives purpose to everything I do.

Finally, I want to thank my parents, without whose guidance, patience and encouragement I would be where I am now. I have never met a more honest, honorable and hard-working person than my father, and a more patient, caring and selfless person than my mother. I can only wish that I can be at least fractionally as good a parent for my own children.

Predicting Multibody Assembly of Proteins

Publication No. _____

Md. Muhibur Rasheed, Ph.D.
The University of Texas at Austin, 2014

Supervisor: Chandrajit Bajaj

This thesis addresses the multi-body assembly (MBA) problem in the context of protein assemblies. MBA can be formally defined as a geometric optimization problem as follows- given a set of objects $\mathbf{M} = \{M_1, \ldots, M_n\}$ whose configurational space is defined as $\mathcal{T}^{\mathbf{M}}$, and a scoring function $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T}) : \mathcal{T}^{\mathbf{M}} \mapsto \mathbb{R}$, computationally predict the configuration $\mathbf{T} \in \mathcal{T}^{\mathbf{M}}$ which maximizes $\mathcal{F}^*_{\mathbf{M}}$.

A solution to MBA requires design and calibration of $\mathcal{F}^*_{\mathbf{M}}$ such that for any $\mathbf{M}$, the maximum of $\mathcal{F}^*_{\mathbf{M}}$ corresponds to the true solution; efficient algorithms for evaluation of $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T})$ for any given configuration $\mathbf{T}$; and, an efficient and accurate sampling and search scheme for exploring $\mathcal{T}^{\mathbf{M}}$ and finding the maximum of $\mathcal{F}^*_{\mathbf{M}}$. Note that problems of the same flavor appear in image stitching, 2D-to-3D reconstruction, global registration, jigsaw puzzle solving etc., and the solution for the above three issues depend on the context. In this thesis, we chose the protein assembly domain because accurate and reliable computational modeling, simulation and prediction of such assemblies would clearly accelerate discoveries in understanding

of the complexities of metabolic pathways, identifying the molecular basis for normal health and diseases, and in the designing of new drugs and other therapautics.

It is well known that even a simple 2D jigsaw puzzle has a combinatorial space of $O(n!)$ and is NP-hard. The multi-body assembly of proteins is even harder. First, while checking the feasibility of a particular assembly is trivial for the puzzle pieces, it is extremely difficult to design a function which can robustly discriminate between feasible and infeasible protein assemblies, because of the diversity of proteins and the highly complex physico-chemical interactions that stabilizes protein assemblies. Second, while there are only a constant number of ways to assemble two puzzle pieces, parameterization of the configurational space of two proteins requires 6 continuous degrees of freedom (or more if flexibility of the proteins are taken into account).

We address the first challenge by developing F^2Dock (Fast Fourier Docking) which includes a multi-term function which includes both a statistical thermodynamic approximation of molecular free energy as well as several of knowledge-based terms. Parameters of the scoring model were learned based on a large set of positive/negative examples, and when tested on 176 protein complexes of various types, showed excellent accuracy in ranking correct configurations higher (F^2Dock ranks the correct solution as the top ranked one in 22/176 cases, which is better than other unsupervised prediction software on the same benchmark).

Most of the protein-protein interaction scoring terms can be expressed as integrals over the occupied volume, boundary, or a set of discrete points (atom locations), of distance dependent decaying kernels. We developed a dynamic adaptive

grid (DAG) data structure which computes smooth surface and volumetric representations of a protein complex in $O(m \log m)$ time, where m is the number of atoms assuming that the smallest feature size h is $\theta(r_{max})$ where r_{max} is the radius of the largest atom; updates in $O(\log m)$ time; and uses $O(m)$ memory. We also developed the dynamic packing grids (DPG) data structure which supports quasi-constant time updates ($O(\log w)$) and spherical neighborhood queries ($O(\log \log w)$), where w is the word-size in the RAM. DPG and DAG together results in $O(k)$ time approximation of scoring terms where $k << m$ is the size of the contact region between proteins.

On searching the configurational space $\mathcal{T}^{\mathbf{M}}$, we show that to provide theoretical guarantee that at least one sample configuration $\mathbf{T}$ falls within a given distance threshold (δ) from the true solution $\mathbf{T}^*$, one must bound the dispersion of the samples in a space isomorphic to $\mathbb{R}^{6(n-1)}$, where n is the number of proteins. However, depending on n and δ, such sampling may become impractical, in which case further information like symmetry or prior knowledge about binding partners is needed to make the problem more tractable. We consider two such cases in this thesis.

First we consider the symmetric spherical shell assembly case, where multiple copies of identical proteins tile the surface of a sphere. Though this is a restricted subclass of MBA, it is an important one since it would accelerate developement of drugs and antibodies to prevent viruses from forming capsids, which have such spherical symmetry in nature. We proved that it is possible to characterize the space of possible symmetric spherical layouts using a small number of representative local arrangements (called tiles), and their global configurations (tiling).

We further show that the tilings, and the mapping of proteins to tilings on arbitrary sized shells is parameterized by 3 discrete parameters and 6 continuous degrees of freedom; and the 3 discrete DOF can be restricted to a constant number of cases if the size of the shell is known (in terms of the number of protein n).

We also consider the case where a coarse model of the whole complex of proteins are available. We show that even when such coarse models do not show atomic positions, they can be sufficient to identify a general location for each protein and its neighbors, and thereby restricts the configurational space. We developed an iterative refinement search protocol that leverages such multi-resolution structural data to predict accurate high resolution model of protein complexes, and successfully applied the protocol to model gp120, a protein on the spike of HIV and currently the most feasible target for anti-HIV drug design.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

We consider the multibody assembly problem. The problem, in simple terms, is to computationally predict the structure of the complex formed by a set of components whose individual structural models are known. Figure 1.1 provides examples where respectively 7 and 60 components must be put together with the aim of predicting their true assembled complexes.

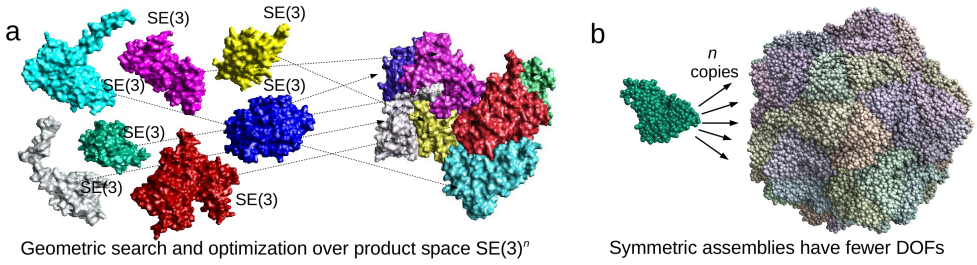


Figure 1.1: (a) We are given multiple disjoint objects which are components of a complex or assembly. The challenge is to search the space of possible arrangments and find the optimal. Typically, for objects in 3D, the problem can be mapped to a geometric search over $SE(3)^n$, where n is the number of components and $SE(3)$ is the group of rigid body motions, to find the optimum of a scoring function. Typically, $SE(3)$ is represented using 6 continuous degrees of freedom. (b) In this thesis we show that for special cases like the symmetric assembly shown in the figure, the total number of degrees of freedom is typically fewer than n, and the space of possible motions is also not the entire product space $SE(3)^n$, but rather a symmtric subgroup of it.

This problem appears in many areas of computational geometry, image pro-

cessing, and computational biology. For example, reconstruction of a 3D scene from 2D images, image stitching, reconstruction of broken pieces of an ancient vase or a dinosaur bone, etc. All of these problems can be reduced to a geometric optimization problem as discussed below.

1.1 Problems

Problem 1: Multibody Assembly

Given a set of objects $\mathbf{M} = \{M_1, \ldots, M_n\}$ whose configurational space is defined as $\mathcal{T}^{\mathbf{M}}$, and a scoring function $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T}) : \mathcal{T}^{\mathbf{M}} \mapsto \mathbb{R}$, report-

$argmax_{\mathbf{T} \in \mathcal{T}^{\mathbf{M}}} \mathcal{F}^*_{\mathbf{M}}(\mathbf{T})$

We detail the notations and definitions used in the problem statement in Section 2.1.

A solution to the problem must address, in the least, the following three issues. First, one needs to design and calibrate a scoring function whose maximization actually leads to the true arrangment or configuration of any set of given objects. Second, the space of configurations need to be characterized and then algorithms to search/sample it need to be developed. Third, data structures and algorithms for efficient representation and maintenance of the object models, and fast computation of the scoring function must be developed.

Sub problem 1: Scoring Function Design

Designing a scoring function $\mathcal{F}_{\mathbf{M}}(\mathbf{T})$ such that, for any set of objects $\mathbf{M}$, we

have $argmax_{\mathbf{T}\in\mathcal{T}^{\mathbf{M}}}\mathcal{F}^{*}_{\mathbf{M}}(\mathbf{T}) = \mathbf{T}^{*}$, where $\mathbf{T}^{*}$ is the true configuratiom of the complex.

Sub problem 2: Configurational Space Characterization and Search

Mathematically characterize and parameterize the space of possible configuration $\mathcal{T}^{\mathbf{M}}$. Develop search algorithms which provides guaranteed convergence in the following sense-

$\exists_{\mathbf{T}\in\mathcal{T}_D}(dist(\mathbf{T},\mathbf{T}^{*}) \leq \delta)$,

where $\mathcal{T}_D$ is a discrete subset of $\mathcal{T}^{\mathbf{M}}$ explored by the search algorithm, $\mathbf{T}^{*}$ is the true configuration, $dist$ is a distance metric and δ is a user defined threshold.

Sub problem 3: Model Representation and Efficient Evaluation of Score

Design computationally efficient representations of the objects, develop data structures for maintenance and rapid update of the models, and algorithms that leverage the data structures to evaluate $\mathcal{F}$ for a given configuration.

Note that, though the multibody assembly problem and subproblems are formulated in a generic fashion, actual realizations or solutions of the problems must be tailored depending on the application domain. In this thesis, we primarily focus on molecular, specifically protein, assemblies.

Proteins are the fundamental functional units of biological systems. From structural functions like forming shells, membranes, muscle fibers etc, to metabolic

functions, cellular and neuronal signaling, and defense against invaders, are all carried out by proteins interacting with other proteins, RNAs or other molecules. Proteins interact by binding with each other and forming complexes. An atomic resolution understanding of such interactions is necessary for designing target-specific therapautics, new drug-delivery mechanisms, identifying the molecular level basis/cause metabolic or neural disorders, and many more applications, and is the central topic in the area of structural molecular biophysics. Computational modeling, simulations and predictions that we develop in this thesis promise to be powerful assistive tools to wet lab experiments in gaining a molecular level understanding of the structural, thermodynamic and kinematic properties of such complexes and interactions.

Specific versions of the subproblems in the context of protein assembly is described in Chapter 2.2 and Section 1 provides a brief overview of protein structure, their interactions and complexes. In the next section, we summarize our main contibutions and provide forward references to the chapters where further detail are provided.

1.2 Contributions

The main contributions of this dissertation are listed below-

1.2.1 Dynamic Packing Grid Data Structure for Efficient Maintenance of Point Collections and Neighborhood Queries

We developed the “Dynamic Packing Grid" (DPG), a neighborhood data structure with applications in maintaining and manipulating flexible molecules and assemblies, efficient computation of binding affinities in drug design or in molecular dynamics calculations. DPG can efficiently maintain sets of points in 3D using only linear space and supports quasi-constant time insertion, deletion and movement (i.e., updates) of weighted points or groups of points. DPG also supports constant time neighborhood queries from arbitrary points. Applications of DPG in maintenance of molecular surface and polarization energy computations using exhibit marked improvement in time and space requirements over other spatial data structures. Finally, DPG was implemented as a modular library which can be easily applied to speed up neighborhood dependent integration in any problem domain. Details of DPG is presented in Chapter 3.

Publications

- C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. *In Proceedings of the ACM Symposium on Solid and Physical Modeling*, pages 259–270, 2009.

- C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. *Bioinformatics*, 27(1):55–62, 2011.

Software DPG can be downloaded for free under an academic use license from `http://www.cs.utexas.edu/~bajaj/cvc/software/DPG.shtml`.

1.2.2 Dynamic Adaptive Grid Data Structure for Computing Volumetric and Smooth Surface Representation of Molecules

We report the Dynamic Adaptive Grid data structure [207] which adaptively subdivides space to provide higher resolution sampling near the boundary (surface) of a molecule with n atoms in $O(n \log n)$ time. It can support any surface approximation as long as it is expressed as a level set of a volumetric function. Our implementation includes the van der Waals surface, the solvent accessible surface (SAS), and the solvent excluded surface (SES) all of which requires the evaluation of an anlytical signed distance function at each gridpoint, and a faster Gaussian integration based surface approximation. This grid-based approach prodives simultaneous maintenance of molecules in atomic, smooth surface and volumetric representations, making it applicable for a wide range of applications. Finally, when an atom is moved (added or removed), the surface can be dynamically and locally updated by our algorithm in $O(\log n)$ time and hence for many applications like docking and structure refinement, where only a fraction of the atoms need to be updated frequently, the dynamic algorithm is beneficial. Details of Dynamic Adaptive Grid is presented in Chapter 4.

Publications

- M. Rasheed, A. Rand, and C. Bajaj. Maintaining flexible molecular sur-

faces using augmented dynamic octrees. Technical Report 14-23, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA 78712., 2014.

Software Dynamic Adaptive Grid is implemented as a library of the MolSurf software package, and is available freely for academic users at `http://www.cs.utexas.edu/~bajaj/cvc/software/molsurf.shtml`.

1.2.3 Design and Calibration of Scoring Functions for Predicting of Protein-Protein Assemblies

We augmented the F^2Dock protocol [26] for predicting 2-body assembly of proteins, with new knowledge-based scoring terms, and a solvation energy (GBSA) based reranking. The new protocol dubbed F^2Dock 2.0 + GB-rerank [65], was trained [208] using large sets of protein-protein interaction decoys of different classes. The improved scoring functions showed superior performance compared to their traditional counterparts in finding correct docking poses at higher ranks. We found that the new filters and the GBSA based reranking individually and in combination significantly improve the accuracy of docking predictions with only minor increase in computation time. We compared F^2Dock 2.0 with ZDock 3.0.2 [179] and found improvements over it, specifically among 176 complexes in ZLab Benchmark 4.0 [129], F^2Dock 2.0 finds a near-native solution as the top prediction for 22 complexes; where ZDock 3.0.2 does so for 13 complexes. Details of F^2Dock is presented in Chapter 5.

Publications

- M. Rasheed, Q. Yuan, and C. Bajaj. Learning optimized scoring models for protein-protein docking. Technical Report 14-25, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA 78712., 2012.

- R. A. Chowdhury, M. Rasheed, D. Keidel, M. Moussalem, A. Olson, M. Sanner, and C. Bajaj. Protein-protein docking with f2dock 2.0 and gb-rerank. *PLoS ONE*, 8(3):e51307, 2013.

Software F^2Dock was implemented as a command line tool which can be used standalone or as a server, with a graphical user interface (TexMol [15]) acting as the client/frontend. A web-based client will be also be released in August 2014. Both the client and server for F^2Dock are available from the following `http://www.cs.utexas.edu/~bajaj/cvc/software/f2dock.shtml`.

1.2.4 Characterization, Enumeration and Construction of *Almost-regular* Polyhedra

We have characterized a new family of polyhedra [204] with regular faces such that it is isotoxal, isohedral, and have exactly 2 types of vertices; as well as a dual family which is isogonal, isotoxal and have exactly 2 types of regular faces. We have shown that both of polyhedrons of these families generated by unfolding a regular polyhedron onto a lattice in a compatible way, thereby allowing the lattice vertices, edges and faces to etch out a tiling on the unfolded polyhedron, and finally

folding it back again. Further, the compatible ways are specified using only a couple of integer parameters. We also provided a deterministic and efficient algorithm for generating such polyhedra of any size (determined by the two parameters). We have proved that our construction covers all possible polyhedron which satisfies the stated properties. Our theoretical groundwork would greatly support computational techniques for exploring and automatically predicting possible nano-structures that can be formed symmetrically by one type of building block. Details of this contribution is presented in Chapter 6.

Publications

- M. Rasheed and C. Bajaj. Characterization, enumeration and construction of *almost-regular* polyhedra. Technical Report 14-23, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA 78712., 2014.

Software The algorithm for generating symmetric tilings is available as library called TilingGen and also as part of TexMol. `http://www.cs.utexas.edu/~bajaj/cvc/software/texmol.shtml`.

1.2.5 Predicting Symmetric Spherical Assemblies Templated on Extended Families of Regular Polyhedra

We use the characterizion of tiled, symmetric shell structures, and address the computational problem of predicting their assembly from primitive 3D building

blocks. We developed an efficient polynomial time, shell assembly approximation solution (based on a combinatorial and motion-space search, and complementarity scoring) to an otherwise NP-hard geometric optimization problem. Our 3D shell assembly prediction, first uses a small set of tiles to generate either periodic or aperiodic surface tilings, parameterized by a small set of inter-tile matching rules. Then to form a 3D shell, we decorate each tile using appropriate number of copies of the given 3D blocks such that symmetry constraints are met and inter-tile interfaces are favorable. We have successfully applied this procedure to the prediction of spherical protein shells of biological viruses of different sizes, all of which exhibit icosahedral symmetry [205]. Our symmetry characterization also provides a natural hierarchical representation of the mechanical degrees of freedom available for a complete viral capsid, and enables automated normal mode analysis under sticks-and-springs model. Simulations of the vibrationals modes derived from this automated model matched independent simulations and experiments [27]. Details of these contributions are presented in Chapter 7.

Publications

- M. Rasheed and C. Bajaj. Predicting symmetric spherical shell assemblies. Technical Report 14-24, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA 78712., 2014.
- C. Bajaj, A. Favata, A. Micheletti, P. Podio-Guidugli, and M. Rasheed. A multiscale comparative analysis of viral capsids' normal modes. *Manuscript*, 2014

Software The algorithm viral capsid assembly has been implemented in a client/server settings similar to F^{2}Dockand is available through TexMol `http://www.cs.utexas.edu/~bajaj/cvc/software/texmol.shtml`.

1.2.6 Multi-resolution and Data-driven Modeling and Validation of Molecular Assemblies

We developed and validated a search and scoring protocol which, using available EM map, partial atomic structures and information about binding interfaces, produce a refined and complete high resolution model of a protein complex. The protocol uses a combination of pairwise docking and flexible fitting [41] to optimize a scoring term that not only accounts for protein-protein interaction potentials, but also the quality of fit with the EM map. We found that the method was extremely successful in predicting near-native configurations for a large set of protein-protein complexes involving the HIV spike protein gp120 and various antibodies. In 18 out of 32 cases, our method picked the lowest RMSD solution as the top solution. Further, gp120 have several variable regions which are known to play an important role in binding to CD4 and then CCR5 which acts as a precursor to infection. But the precise structure and configuration of these variable regions' interaction with CD4 and CCR5 have not been resolved yet by x-ray crystallography. We showed that our validated scoring model can be combined with off-the-shelf threading and energy minimization algorithms to produce reliable models of the variable regions [206]. Details of these contributions are presented in Chapter 8.

Publications

- M. Rasheed, R. Bettadapura, and C. Bajaj. Structure of HIV spike protein gp120 including all variable loops in complex with CD4 and 17b through computational modeling, fitting and validation. *Submitted*, 2014

- R. Bettadapura, M. Rasheed, A. Volrath, and C. Bajaj. PF^{2}Fit: Polar fast fourier alignment of atomistic structures with 3D electron microscopy. *Submitted*, 2014

Software Our flexible fitting and docking software are available at `http://www.cs.utexas.edu/~bajaj/cvcwp/?page_id=2366` and `http://www.cs.utexas.edu/~bajaj/cvc/software/f2dock.shtml`, and an automated script to integrate different stages of the protocol is under construction.

1.3 Organization of the Dissertion

Chapter 2 first provides an overview of the notations used in the problem statements, and in the rest of the dissertation. Then it provides a brief introduction to the assembly problem specifically for the case of molecular assemblies to highlight the unique challenges in this domain, and wraps up with a discussion on prior work in generic and symmetric multi-body assembly. This chapter may be skipped by reader familiar with the molecular domain and prior work.

Chapters 3 to 8 discuss the main contributions of this thesis in the same order they are listed in the previous section.

Chapter 3 details the Dynamic Packing Grids (DPG) data structure and de-

scribes how it can be used to efficiently compute properties and functions defined over sets of points (or atoms). Then Chapter 4 presents a dynamic octree data structure augmented with DPG to support efficient and simultaneous dynamic maintenance of level set surfaces, collection of points, and volumetric functions defined over the set of points.

Chapters 5 and 8 focus on the scoring part of the problem. In Chapter 5, we introduce several physically-based and knowledge-based scoring terms for evaluating protein-protein interfaces, train a combination of the terms to effectively predict correct interfaces. In Chapter 8, additional scoring terms to measure the internal structure of the components, as well as to measure the correlation of a predicted assembly to a coarse-resolution template are introduced. These new terms in particular can use data from various modalities to effectively refine structural models involving multiple components.

Chapters 6 and 7 deal with characterizing the restricted configuration (search) space for symmetric assemblies and then apply it to predict structures of virus capsids. Chapter 6 in particular provides a characterization, and necessary and sufficient conditions of generating a class of highly symmetric polyhedra which can be tiled using a single type of building block. This construction is used to generate families of layouts on which proteins are placed to produce thick shell structures (viral capsids) in Chapter 7.

Finally, in Chapter 9, we summarize our contributions and then discuss a few interesting and important problems which are still open, and discuss some possible solutions.

All the chapters are pretty self contained, and include description of related prior work, any new notations that is used in the chapter, as well as relevant experimental results. Also when results from a different chapter are used, they are accompanied with a brief review and a pointer to the corresponding chapter. Hence, the chapters can technically be read in any order, however we believe it would be best if the reader followed the order in which the chapters are organized.

Chapter 2

Background and Prior Work

2.1 Notations and Definitions

We introduce some key notations and definitions here which will be used throughout the dissertation.

2.1.1 Object

Anything that occupies volume in 2D or 3D space is considered an object in the context of this dissertation. For example, an object can be a molecule, a jigsaw puzzle piece, or a broken piece of furniture. The object must have a structural model, and can have additionally have functional and kinematic models associated to it.

Definition 2.1.1 (Structural Model)**.** We define a structure model as a geometric description of a physical object as a set of points, a scalar-valued function in 3D, a surface or volumetric mesh, or any other format. Note that each of these representations can be mapped to each other and hence, without loss of generality, we shall assume that the structure model is simply a set of points $M : \{a_1, \ldots, a_n\}$ and the corresponding boundary and occupancy are expressed as $S^M(x) : \mathbb{R}^3 \mapsto [0, 1]$ and $V^M(x) : \mathbb{R}^3 \mapsto \mathbb{R}$ respectively. Surface and Volume meshes are simply discretized

evaluations of S^M and V^M.

Even though, M is strictly speaking only the structural description of an object, we shall often use M to denote the object itself.

Definition 2.1.2 (Functional Model)**.** The functional model of an object are attributes of the object. We shall denote each attribute as functions which may be used as part of mathematical expressions. For example, if M is the structural model of a puzzle piece, $Color^M(x)$ specifies the color assigned to each point x. Also, if it is clear from the context (or explained in the related text), then we would also use shorthands like c_x instead of $Color^M(x)$.

Definition 2.1.3 (Kinematic Model)**.** In case an object is flexible, a kinematic model describes how different parts of an object move with respect to each other. We shall assume that the kinematic model is expressed as a motion graph $G^M(V, E)$, where the nodes $v \in V$ correspond to disjoint subsets, called subdomains, $M_i \subseteq M$ of the object, and each edge $(u, v) \in E$ indicates that the movement of the subdomains are constrained by each other. Each edge is annotated with a parameterization of the type and range of motions allowed for the two subdomains with respect to each other.

2.1.2 Transformation, Configuration and Complex

Definition 2.1.4 (Rigid Body Transformation)**.** We define a transformation as an isometry, or a map $A : \mathbb{R}^3 \mapsto \mathbb{R}^3$ in Euclidean space which preserves distances

between any pair of points it is applied to. Such transformations for the Euclidean space includes combinations of translation, rotations and reflections, but not scaling or shear. We shall assume that T is expressed as a matrix of the form-

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

where R is an orthogonal 3×3 matrix representing a rotation around an arbitrary axis through the origin in 3D, and t is a 1×3 column vector representing a translation.

Definition 2.1.5 (Flexible Transformation)**.** We define a flexible transformation T applied to an object model M with k subdomains (according to a specific kinematic model of the object), as a set of k rigid body transformations $\{A_1, \ldots, A_k\}$, such that each transformation A_i is to be applied to each point a_j in subdomain $M_i \in M$. If all the individual A_i are the same, then the object undergoes a rigid body motion.

Note that functional models of the object may be changed by such transformations and need to be re-evaluated.

Definition 2.1.6 (Configurational Space of an Object)**.** The configurational space $\mathcal{T}^M$ of an object M with k subdomains, is a subset of $SE(3)^k$, where $SE(3)$ is the Euclidean group. Depending on the kinematic model of the object, this product space may have different structure. For example, for flexible molecules with torsion angle degrees of freedom (see Section 1.1 for details), the space is $(S^1)^k$ where S^1 is the 1D rotation group. Note that for rigid body objects, $k = 1$.

Definition 2.1.7 (Complex or Arrangment)**.** A complex $\mathbf{C} =< \mathbf{M}, \mathbf{T} >$ is an object formed by compositions of a set of objects $\mathbf{M} = \{M_1, \ldots, M_n\}$ such that the configuration of each component is $T_i(M_i)$. The model for $\mathbf{C}$ is hence defined as $M_{\mathbf{C}} = \cup_i T_i(M_i)$.

Definition 2.1.8 (Configurational Space of a Set of Objects)**.** The configurational space $\mathcal{T}^{\mathbf{M}}$ of a set of n objects $\{M_1, \ldots, M_n\}$, is a subset of product space $\prod \mathcal{T}^{M_i}$. If all objects are rigid body, then the space is $SE(3)^n$.

2.1.3 Scoring Functions and Quality of a Complex

Definition 2.1.9 (Scoring function)**.** A scoring function is defined as $\mathcal{F}_{\mathbf{M}}(\mathbf{T}) : \mathcal{T} \mapsto (R)$, where $\mathcal{T}$ is the space of possible transformations for the set of components $\mathbf{M}$. We shall also use $\mathcal{F}(\mathbf{C})$ to represent the same function.

Now, let us assume that the true configuration $\mathbf{C}^*$ for a complex of the n objects $\mathbf{M} = \{M_1, \ldots, M_n\}$, is given to us by some oracle. Now, we would like to compare it with some other configuration $\mathbf{C}$ generated or predicted by an algorithm. For example, we want to verify that the jigsaw puzzle put together by someone is actually the desired/correct arrangement of the pieces. This concept is formalized below-

Definition 2.1.10 (Correctness of a Complex)**.** Given a predicted complex $\mathbf{C}(\mathbf{M}, \mathbf{T})$, and a ground truth arrangment $\mathbf{C}^*(\mathbf{M}, \mathbf{T}^*)$ produced by an oracle, we define the error in the predicted arrangment as the distance $dist(\mathbf{C}, \mathbf{C}^*)$. We say that $\mathbf{C}$ is correct if $dist(\mathbf{C}, \mathbf{C}^*)$ is less than some threshold ε.

Since, both $\mathbf{C}$ and $\mathbf{C}^*$ are defined over the same set of componenets $\mathbf{M}$, let us assume that for each point a_i in predicted arrangment, the corresponding true position is a_i^*. Then a simple distance norm can be defined as follows-

$$dist(\mathbf{C}, \mathbf{C}^*) = \min_{T \in SE(3)} \sum_{a_i \in \mathbf{M}} \delta(T(a_i), a_i^*)$$

where $\delta(T(a_i), a_i^*)$ is a metric, usually Euclidean distance. Popular variations include RMSD and Hausdorff distance.

Note that, we shall use $dist_{\mathbf{M}}(\mathbf{T}, \mathbf{T}^*)$ and $dist(\mathbf{C}, \mathbf{C}^*)$ interchangeably.

Now we define the scoring function, which tries to measure the quality of a complex, without access to the oracle.

Definition 2.1.11 (Scoring oracle)**.** A scoring oracle $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T}) : \mathcal{T} \mapsto (R)$ is a scoring function which behaves like the oracle such that $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T})$ is negatively correlated with $dist(\mathbf{T}, \mathbf{T}^*)$.

Now, we are ready to formally define the assembly problem.

2.1.4 Problem Statement

We had previously expressed The multibody assembly problem in simple terms, as computationally predicting the structure of the complex formed by a set of objects whose individual models are known. Below, we formalize it.

Definition 2.1.12 (Multibody Assembly)**.** Given a set of objects $\mathbf{M} = \{M_1, \ldots, M_n\}$ whose configurational space is defined as $\mathcal{T}^{\mathbf{M}}$, and a scoring oracle $\mathcal{F}^*_{\mathbf{M}}(\mathbf{T}) : \mathcal{T}^{\mathbf{M}} \mapsto \mathbb{R}$, report-

$$argmax_{\mathbf{T}\in\mathcal{T}^{\mathbf{M}}}\mathcal{F}^{*}_{\mathbf{M}}(\mathbf{T}) \tag{2.1.1}$$

This leads us directly to three separate subproblems-

1. Efficient computational representation of object models $\mathbf{M} = \{M_1, \ldots, M_n\}$. Introduced in Section 2.2.3 and detailed in Chapter 3.

2. Designing a scoring function $\mathcal{F}_{\mathbf{M}}(\mathbf{T})$ which behaves like an oracle, even without access to the true solution. Introduced in Section 2.2.2 and detailed in Chapter 5.

3. Characterizing $\mathcal{T}^{\mathbf{M}}$ and developing sampling and search techniques to find the best configuration. Introduced in Section 2.2.1 and detailed in Chapters 7 and 8.

2.2 A More Detailed Look into the Subproblems

2.2.1 Configurational Space Characterization and Search

In this section we shall first characterize the configurational space for the most general setting, discuss some simplifications which still does not make the problem tractable. Then, we present two special cases, where the configurational space can be bounded.

The search space for the *Multibody Assembly* problem is $SO(3)^N$ where N is the number of rigid subdomains among all the set the n proteins $M_1, \ldots, M_n$ that need to be assembled. This is an extremely high dimensional optimization problem

where the scoring function is multimodal. We shall show in Chapter 9 that in principle it is possible to produce low discrepancy point samples in this product space which would guarantee that at least one sample would be within a user-defined distance threshold ε from the true configuration [186]. However, the guarantee depends on the local variability of the scoring function and the number of samples required to provide practical values of ε are infeasible. A popular alternative is to use Markov Chain Monte Carlo sampling [80, 117, 247, 265] which has usually has better convergence in practice, but no theoretical guarantees. A relaxation of a similar multi-registration problem into a semidefinite program was recently proposed in [57].

Now, if we assume that simultaneous assembly is highly unlikely and proteins assemble into pairs first, and then gradually larger assemblies, then we can use a distributed assembly approach (similar to solving a jigsaw puzzle).

2.2.1.1 Multi-body Assembly as a Combinatorial Optimization Problem

We first solve the 2-body assembly problem defined as follow-

Definition 2.2.1 (2-body Assembly)**.** $argmax_{\mathbf{T}\in\mathfrak{I}^{\mathbf{M}}}\mathcal{F}^{*}_{\mathbf{M}}(\mathbf{T})$ where $\mathbf{M} = \{M_1, M_2\}$

In practice, due to uncertainties/error introduced by sampling and primarily due to the scoring function not being an oracle, we predict a series of k complexes ordered by their scores and expect the optimal solution to be ranked between $[1, ..., k]$. If we further assume that the proteins are rigid, then the configuration can be expressed as transformation applied to M_2, while keeping M_1 fixed. We shall

refer to this as a relative transformation. The outcome of the 2-body assembly is hence a list of k predicted relative transformations.

Definition 2.2.2 (Assembly graph)**.** Define a multi-graph $G(V, E)$ such that each node $v \in V$ correspond to an object in M and each edge $(u, v, i) \in E$ correspond the i-th relative transformation T^i between the objects u and v as predicted by the 2-body assembly solution where $1 \leq i \leq k$.

It is easy to see that the assembly graph has the following properties-

- It is a complete graph.
- A simple edge between a pair of nodes correspond to a relative pose between them.
- A simple subtree of m nodes is sufficeint to specify the relative poses among all ${}^mC_2$ pairs of nodes on the tree.
- A simple spanning tree is sufficient to specify a complex.

Definition 2.2.3 (Graph Multi-body Assembly)**.** The multi-body assembly problem reduces to finding a simple spanning tree S of G such that $\mathcal{F}_G(S) = \mathcal{F}_{\mathbf{M}}(\mathbf{T})$ is maximized. Note that $\mathbf{T}$ can be defined based on the relative transformations without any ambiguity or conflicts if S is a spanning tree.

There are $O(n^{n-2} * K^{n-1})$ possible simple spanning trees of V. And this problem was shown to be NP-hard [130]. We provide a sketch of the hardness proof below.

Graph Multi-body Assembly is NP-hard Monkey puzzle is a known NP-hard problem. The problem is to find the solution to a rectangular puzzle (the solution makes a rectangle of known dimensions) with square pieces with straight line edges such that each piece has the picture of half a monkey (either head or tail) on each of its edge. The pieces must be matched such that along any interface, a complete monkey is formed (by a head and a tail of same colors coming together). This problem can easily be reduced to the assembly problem where each puzzle piece represent a node and there are at most 16 possible edges between each pair. Now if we define the scoring function such that it adds 1 for each whole monkey and subtracts 1 for each mismatch, then the solution to the assembly graph problem is exactly the solution to the monkey puzzle problem.

So, even this simplified problem is NP-hard. There are several algorithms which models the problem in this fashion and then provide greedy or heuristics-based algorithms. For example, Papaioannou et al. [191, 192] provide a simulated annealing algorithm for assembling fractered geometric objects, Huang et al. [127] developed avariant of forward serach algorithm for the same application. Both of them perform well mainly because the scoring function for purely geometric objects are much more reliable than those for proteins. Inbar et al. [130] applied a greedy Prim's minimum spanning tree algorithm and was quite successful for protein complexes with upto 7 proteins. However, their experiment was a redocking experiment, where the proteins are in their correct shape and rigid body motion alone is sufficient to produce perfect solutions. Finally, Esquivel-Rodriguez, Yang and Kihara developed a genetic algorithm to explore the space of spanning trees

[90].

2.2.1.2 Symmetric Assembly

Many macromolecular assemblies are symmetric (see the discussion in Section 1.3 for example symmetric macromolecules).

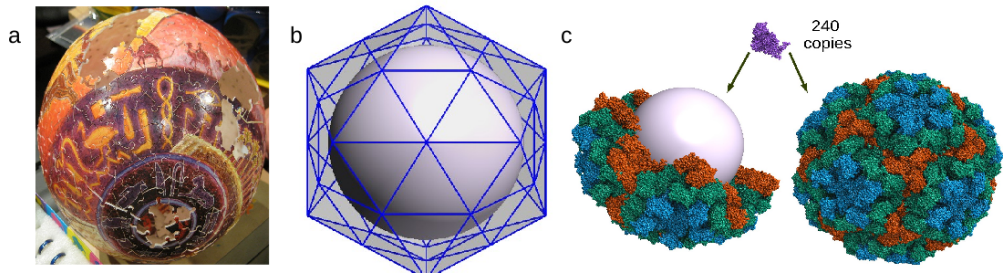


Figure 2.1: (a) A jigsaw puzzle covering the surface of a sphere. (b) Symmetries of this puzzle can be represented as the symmetry of the icosahedron whose faces are subdivided symmetrically into smaller triangles. (c) Decorating each of those faces using proteins arranged in cyclic symmetry produces a symmetric shell assembly. The figure shows the assembly predicted by our algorithm for the Nudaurelia Carpensis Virus.

The most common are the cyclic and dihedral symmetries, and there are multiple prior work [7, 35, 36, 69, 88, 139, 198, 226] where such assemblies have been characterized and computationally predicted. In most cases, the algorithms do not leverage the symmetry to bound the search space, but rather only use it as apost-processing step. On the other hand, a few like the approach of Pierce, Tong and Weng [198], provide a characterization of the the space of cyclic symmetric configurations and shows that the DOFs are less than a 2-body problem. However, such characterizations for other symmetric spaces are not known in the context of assembly prediction yet.

In this thesis, we specifically focus on the symmetric spherical shells (see Figure 2.1) as such shells are the predominant structure for virus capsids, and an understanding of such capsids and their assembly would help in designing drugs to stop the formation/dissolution of the capsids, or to produce atomic resolution models of capsids for which only the sequence or a coarse model is available, or to design novel protein cages to act as targeted drug delivery packages.

Sub problem 1: Characterize the space of symmetric spherical shell topologies. Find the minimal set of parameters, or degrees of freedom that need to be sampled to explore the space.

Sub problem 2: Develop efficient algorithms that can predict 3D shells by arranging multiple copies of an object symemtrically based on the symmetric shell topology.

2.2.1.3 Assembly Guided by Coarse Constraints

Combinatorial complexity of the general multi-body assembly problem can be subverted if prior knowledge regarding the complex is available. For example, wet lab experiments like spectroscopy or Isothermal Titration Calorimetry (ITC) [200] can be used to identify protein stoichiometry data, i.e. the likelihood of a pair of proteins forming a complex, or binding to each other. If this data is available, then the assembly graph can be pruned. Also, for many of the macromolecular assemblies, coarse resolution structural information of the entire complex is available

in the form of cryo Electron Microscopy images [97]. EM captures the electron scattered by a specimen as a 2D image. In single particle EM, thousands of copies of the same molecule is imaged at the same time and with the assumption that the image captures a 2D projection of multiple randomly oriented molecules, a complete 3D volumetric representation can be reconstructed, and be used as a template of guide for searching the configurational space (cf. Figure 2.2).

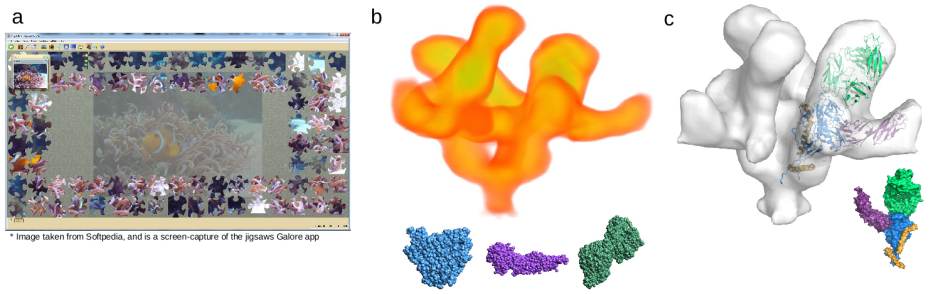


Figure 2.2: (a) Solving a jigsaw puzzle is easier when we are given hints about the complete picture. (b) Coarse resolution volumetric model (3DEM map) of the HIV spike protein gp120 in complex with proteins CD4 and 17b can be used to guide the search for generating a atomic model of the same complex (c).

Recently a series of papers [5, 155, 218, 227, 253, 267] have been published where 3DEM maps were segmented and then individual proteins were fit into different segments, and finally small local motions between neighboring proteins were used to predict the atomic resolution structure of the complex. However, success of such methods relies heavily on the quality of the 3DEM map and the segmentation tool. In this thesis, instead of decoupling the search into segmentation, fitting (optimizing configurations of the proteins with respect to the EM) and docking (optimizing configurations of the proteins with respect to each other), we use the entire

EM map to define a coarse boundary for the sampling, and develop a modified scoring function whose optimization leads to both good fitting and docking.

Sub problem 3: Characterize the restricted space of configurations for a set of proteins, when a coarse 3DEM map is available. And develop coupled scoring and search algorithm that optimizes the configuration of the proteins with respect to each other and the EM map simultaneously.

2.2.2 Scoring Function Design

We would like to define a scoring function which would be negatively correlated with $dist(\mathbf{T}, \mathbf{T}^*)$. However, it is quite improbable since $dist(\mathbf{C}, \mathbf{C}^*)$ itself is not always monotomincally related to changes in $\mathcal{T}$, so instead of requiring that the scoring function mimics the oracle over the entire configurational space, we simply require it to be able to identify the optimum.

Definition 2.2.4 (Accuracy of a Scoring Function)**.** A scoring function $\mathcal{F}$ is accurate for a set $\mathbf{M}$ if,

$$dist_{\mathbf{M}}(\mathbf{T}^m, \mathbf{T}^*) \leq \varepsilon$$

where, $\mathbf{T}^*$ is the true arrangment, and,

$$\mathbf{T}^m = argmax_{\mathbf{T} \in \mathcal{T}} \mathcal{F}_{\mathbf{M}}(\mathbf{T})$$

Secondly, due to discrete sampling and the presence of multiple local maxima, sometimes the highest scoring sample may not be close to the true maxima. So in this discrete setting, we use a weaker definition of accuracy.

Definition 2.2.5 (Weaker Accuracy of a Scoring Function)**.** A scoring function $\mathcal{F}$ is accurate for a set $\mathbf{M}$ if,

$$\exists_{1 \leq i \leq k} dist_{\mathbf{M}}(\mathbf{T}^i, \mathbf{T}^*) \leq \varepsilon$$

where, $\mathbf{T}^*$ is the true arrangment, and,

$\mathbf{T}^1, \ldots, \mathbf{T}^k$ are the k highest scoring samples in a discrete sampling of $\mathcal{T}$.

Finally, we formally define the problem of scoring function design.

Definition 2.2.6 (Scoring function design)**.** Design a function $\mathcal{F}$ such that for *any* set $\mathbf{M}$, $\mathcal{F}$ is accurate according Definition 2.2.5.

Sub problem 4: Design a function $\mathcal{F}$ such that for *any* set $\mathbf{M}$ of proteins, $\mathcal{F}$ is accurate according Definition 2.2.5.

We have found that scoring functions which combine multiple affinity terms defined based on different biophysical and statistical potentials has a better probability of being successful over diverse set of protein-protein interactions. Chapter 5 details the new scoring terms we developed, a few terms that we have improved, a machine learning model to calibrate the parameters of the function, and detailed analysis of the accuracy of the scoring function (with different k) on a benchmark [129].

2.2.3 Model Representation and Efficient Evaluation of Score

The most common representation for molecules is as a collection of atoms represented as hard spheres, with radii equal to their van der Waals radii [47]. Such

a representation along with partial charges assigned to each atom based on a statistical thermodynamics [183, 260, 268] or quantaum mechanical force field [81] can be sufficient for the computation of many scoring functions including the van der Waals energy under the Lennard-Jones model, electrostatics (Coulombic) energy, statistical terms like residue-residue contact potentials etc. (please see details of these terms in Chapter 5).

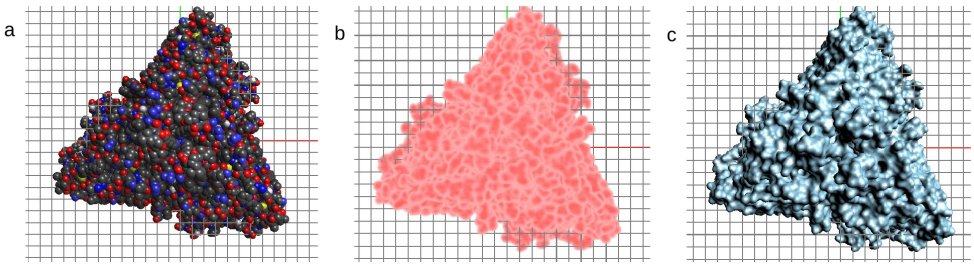


Figure 2.3: (a) An union of balls model. Each atom in the figure is colored by its atom type, and the radii are equal to their van der Waals radii. Note that the spacing between the gridlines are 5 angstroms. (b) A volumetric model of the same molecule. Here a function $f(x)$ is defined over the Euclidean space as the electron density in the neighborhood of x. The function is visualized by setting up a color mapping for the function values. (c) A boundary representation of the molecule. The boundary was computed as a level set of a signed distance function, and represented as a triangulated mesh.

Some scoring functions, for solvation energy computed under the GBSA model [245] requires integration over the boundary of the molecule. The boundary of the union of the hard spheres is known as the van der Waals (vdW) surface. Another model, introduced by Lee and Richards [157], called the solvent accessible surface (SAS) is defined as the boundary of the union of hard spheres with their radii extended by r_s, the radius of solvent (water) molecules. This surface represents the locus of the center of a ball of radius r_s on the vdW surface. However, both

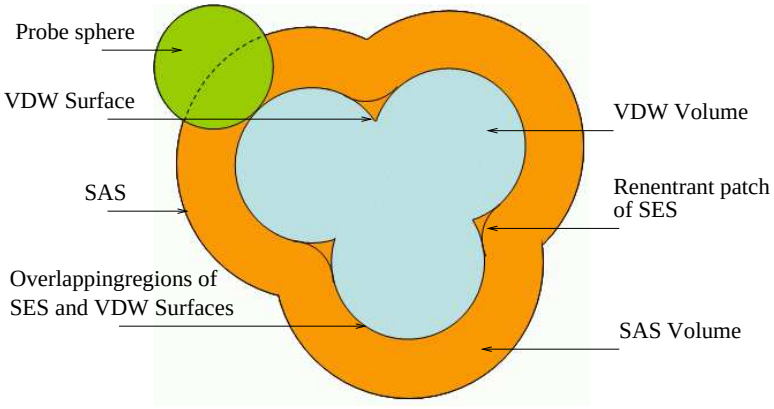


Figure 2.4: The different molecular surfaces and regions are shown for a 3 atom model in 2D. The SAS surface is the locus of the center of the rolling probe sphere. The VDW surface is the exposed union of spheres representing atoms with their van der Waals radii and contains the VDW volume. The lower side of the rolling probe defines the smooth SES which contains parts of the VDW surface and reentrant patches. We also define the SAS volume as the region between the SAS and SES. The region between the SAS and VDW volumes is later refered to as the SES volume.

the vdW and SAS surfaces are not C^1 continuous and unsuitable for integration. Richards then gave a more commonly used definition for molecular surface [213] as the footprint of a probe solvent sphere, rolling over the atoms of a protein defines a region in which none of its points pass through. This surface is composed of convex patches where the probe touches the atom surfaces, concave spherical patches when the probe touches more than 2 atoms simultaneously and toroidal patches when the probe rolls between two atoms. Connolly called this as an alternative definition of the SAS surface in [71], but is now commonly known as the Solvent Contact Surface (SCS), or Solvent Excluded Surface (SES) or simply the Molecular Surface. These surfaces, for a 3 atom example is shown as a 2D cross section in figure 2.4.

Finally, volumetric representations of the surface as functions defined over the Euclidean space, but having non-trivial values only within or near the region occupied by the molecule is also useful, not only as an intermediate step towars computing the boundary or surface, but also for defining scoring functions. For example, alginment of two proteins can be defined as the convolution of their volumeteric representations.

Hence, it essential to develope data structures and algorithms to generate and maintain all three types of representations, based on a union of ball input. Further, to account for the flexibility of proteins the algorithms must be able to perform dynamic and local updates efficiently. Efficiency is very important here since these algorithms will be at the innermost loop of any assembly search protocol and be invoked the maximum number of times.

Sub problem 5: Design efficient spatial data structures for maintaining mixed representations of molecules. Develop algorithms for efficient computation and dynamic update of volumetric and surface representations from the union of balls representations.

2.3 Prior Work

2.3.1 General Multi-body Assembly

Prior work has addressed this problem from the perspective of reconstructing or reassembling historic 3D artifacts or statues from fractured 3D pieces recovered by archaeologists. For example, [127] and [191] both assume that the neigh-

boring fractured pieces have good shape complementarity and use shape descriptor based matching to find possible interfaces between each pair of components. The remaining challenge is then to select a subset of those interfaces which are consistent, and optimize a global assembly. The approach is equivalent to computing the best simple spanning tree from a multi-graph where each graph node represent a p-tile, and k edges between two nodes represent k possible interface configurations. The number of simple spanning trees is exponential in the number of components, i.e. $O(n^{n-2}k^{n-1})$. Furthermore, its quite easy to reduce the known NP-hard 2D Monkey Puzzle problem to the more difficult 3D assembly problem. One can show that a polynomial time approximation scheme ($PTAS$) can be designed if the number of parallel edges between each pair of nodes is at most 2. However, for all practical cases, $PTAS$s do not exist. The paper [127] proposed a greedy forward search algorithm and [191] used genetic programming to solve this combinatorial problem. A simpler greedy algorithm was used by [130] for predicting the assembly of multiple proteins. The 2D counterpart is a well-studied problem, for both texture or image driven matching [62, 219] as well as based purely on the shape [107].

Though the combinatorics of these multi-piece matching approaches is essentially similar to the multi-protein docking problem, but the nature of the interface and rules of interaction are vastly different. The most obvious difference is that the affinity between two fractured pieces depend only on their shapes, with only a non-penetration constraint. There is no attractive-repulsive forces and/or different factors counterbalancing each other. So, the challenge in designing a good scoring function is often restricted to finding an efficient one. The search space can also

be reduced by factors such edge the presence of sharp edges which provide good starting positions as well as a mode of verification for matching, and of course there are no flexibility in the pieces.

2.3.2 Multi-Protein Assembly

Almost half of the complexes in the protein data bank are complexes involving more that two proteins. However, due to the combinatorial complexity of docking multiple proteins, most researchers tried to solve the problem of pairwise docking first. After the pairwise docking problem matured a little, and more importantly after the CAPRI challenge featured complexes with cyclic symmetries, the interest in multi-protein docking increased, specially for complexes with cyclic and dihedral symmetries. Most of the existing techniques are only applicable to these two classes of multi-protein complexes. A few techniques can also dock several proteins with no symmetry. However, they either depend on the availability of prior knowledge about the relative positions of the proteins and only optimizes the positions (registration), or makes unrealistic assumptions to reduce the combinatorial complexity. Among the existing multiprotein docking research, we identify the following as most relevant.

- mMolFit by Eisenstein et al. [35, 36, 88]. Based on MolFit [37, 122, 141].
- mZDock by Pierce, Tong and Weng [198]. Based on ZDock [59, 61, 179].
- mClusPro by Comeau and Camacho [69]. Based on DOT [170] and ClusPro [70].

- mRosetta by Andre et al. [7]. Based on ROSETTA [73].
- mHaddock by Karaca et al. [139]. Based on HADDOCK [80].
- MultiFit by Lasker, Dali, Wolfson [154, 155, 249, 253].
- CombDock by Inbar et al. [130]. Based on PatchDock [84].
- SymmDock by Schneidman-Duhovny et al. [226]. Based on PatchDock [84].

In this section, we briefly present and compare these techniques in terms of their search space, search technique and scoring function in Table 2.1 (continued to Table 2.2). Then we identify the open issues and indicate how we address those.

2.3.2.1 Limitations

First of all, the search space of most of the techniques is restricted to only cyclic and dihedral symmetries. Only CombDock can dock heterogeneous proteins into a nonsymmetric assembly without any prior knowledge about the structure of the complex (like MultiFit) or user-defined interfaces between the proteins (like mHaddock). However, the algorithm in CombDock assumes that the sum of the weights on the edges of the spanning tree is the score of the complex. Formally, given a multigraph $G(V, E)$ with n nodes and k edges between each pair of nodes; the weight of a spanning tree T is defined as $w_T = \sum_{e \in T} w_e$. This idea is fallacious even in the simplest cases. For example, consider the case of a trimer of a virus (each protein looks like a trapezoid), and let the score is based on shape complementarity. Let us assume that the pairwise docking reported only two principal

METHOD	SEARCH SPACE	SEARCH TECHNIQUE	SCORING TERMS
Berchanski et al. (MolFit)	• Rigid Body. • C_n or D_n. • Single type of protein. • Search Space: R^6	• Sample 3D rotational space. 3D FFT for each rotation to score each translation. • Select transformations which result in symmetry.	• Shape complementarity and electrostatics
Weng et al. (M-ZDock)	• Rigid Body. • C_n or D_n. • Single type of protein. • Search Space: R^4	• Due to symmetry, sampling 2D rotational space is sufficient. • 2D FFT for each rotation to score each translation. • Select transformations which result in symmetry.	• Shape complementarity, electrostatics and desolvation energy
Comeau and Camacho (DOT+ClusPro)	• Rigid Body. • Mixed C_n, D_n. • Single type of protein. • Search Space: R^6	• Sample 3D rotational space and use 3D FFT to score each translation. • Cluster similar poses • Select transformations which result in symmetry.	• Only shape complementarity is used in the FFT • Electrostatics, solvation energy and entropy for reranking
Andre et al. (ROSETTA)	• Flexible backbone and side chain. • C_n, D_n, helical and icosahedral. • Single type of protein. • Search Space: R^6 for each rigid part	• Set up the proteins in symmetric positions • Apply symmetric steps in Monte Carlo minimization	• Molecular mechanical energy, solvation energy, hydrogen bonds and statistical interface propensity
Schneidman-Duhovny et al. (SymmDock)	• Flexible backbone with domain decomposition. • C_n symmetry. • Single type of protein. • Search Space: R^6	• The protein is decomposed into a few domains. domain movements are defined. • Each domain docked with itself. Docking is done by matching surface patches. Only symmetries transformations are used. • The results of the symmetric docking of each domain is combined such that the difference of the transformations are within the amount of movement allowed at the inter-domain interface.	• Scoring is based on curvature based shape complementarity.

Table 2.1: Comparison of the search space, sampling and search methods and scoring terms of current multi-protein docking algorithms

modes of binding- one where the long edge of the trapezoid attaches with dihedral symmetry and has a weight of 10 and another where the non-parallel edges meet with a weight of 8 (see Figure 2.5). Clearly, the best configuration is the trimer where 3 pairwise interfaces exist for a total score of 24. However, there are only 4

METHOD	SEARCH SPACE	SEARCH TECHNIQUE	SCORING TERMS
Karaca et al. (HADDOCK)	• Rigid body and local flexibility near binding interface. • C_n, D_n, and asymmetric. • Different proteins. • Search Space: mR^6, for m proteins.	• User specified restraints to set up initial pose. Symmetry is also used to set up restraints. • Monte Carlo minimization is done in 3 stages. First stage is rigid body search. Stages 2 and 3 considers the interface residues to be flexible.	• Stage 1,2: shape complementarity, molecular mechanical energy and approximate solvation energy. • Stage 3: shape complementarity, molecular mechanical energy and explicit solvation energy.
Lasker et al. (MultiFit)	• Rigid body. • Asymmetric. • Different proteins. • Search Space: m^2R^6, for m proteins.	• EM density of the complex is segmented. Each protein is fit to a segment. • Pairs of proteins are docked to improve the binding interface.	• Scoring term for fitting is based on inclusion/exclusion. • Scoring for docking is based on shape complementarity.
Inbar et al. (CombDock)	• Rigid body. • Asymmetric. • Different proteins. • Search Space: $m^2R^6 + m^{m-2}$	• Each pair is docked using feature correspondence based alignment. • A graph is built with edges for each interface between a pair of proteins. Multi-protein complex is found by finding spanning trees of the graph using a greedy algorithm.	• Scoring for docking is based on shape complementarity.

Table 2.2: Comparison of the search space, sampling and search methods and scoring terms of current multi-protein docking algorithms (Continued)

possible spanning trees and if CombDock's principle is applied, then the correct one gets the lowest score. Similar scenarios are even more probable in non-symmetric assemblies.

mRosetta can dock viral capsid proteins if they follow Caspar-Klug theory of quasi-equivalence. But there are many icosahedral viruses whose structure does

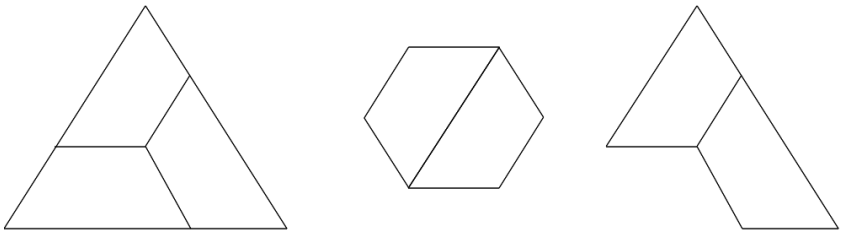


Figure 2.5: A simple example where greedy algorithms fail. We assume that the correct configuration of the three proteins is a trimer (left). Each protein looks like a trapezoid. Pairwise docking suggests two modes of binding. Either along the long edge (center), or one of the non-parallel edges (right).

not conform to the Caspar-Klug model (see Section 1.4 for details). The viral tiling theory (VTT) offers a more comprehensive set of rules to enumerate all possible ways or decorating an icosahedral shell. In this thesis, we define our search space and strategy based on the VTT. Also, mRosetta is mainly designed as a secondary stage docking method produce optimal solutions at the cost of slower speed. Our multi-protein docking is based on faster search based on FFT.

2.3.3 Viral Capsid Assembly

Researcher have long been trying to understand the structure and life-cycle of viruses in an attempt to design anti-viral drugs. The first unifying and generic model was proposed in 1962 by Caspar and Klug who theorized that most spherical viral capsids have quasi-equivalent icosahedral symmetry and developed a simple mathematical formalism to predict a series of viral capsid structures [55] (see Section 1.4 for details). This formulation succeeded in explaining the structures of most of the known capsids. Based on similar ideas Berger et al. [38, 39] proposed the local rules model. In local rules model, it is recognized that the proteins on a

capsid can be grouped into a few classes such that the members in each class have the same protein-protein interfaces. These rules are inferred by analyzing the structure of a given capsid and hence lacks predictive capacity. However, the model is quite suitable to drive simple molecular dynamics simulations where the proteins are allowed to move in solvent, and whenever two proteins come into contact such that the interface matches the local rules, then the proteins are considered to bind [229]. This process is continued until the whole capsid is formed. Such molecular dynamics simulations aim to understand the stages of the assembly from an energetic and statistical perspective. Various researchers including Rappaport et al. [203], Schwartz et al. [229], Hagan and Chandler [118] have used molecular dynamics. Schwartz's model is based on the local rule theory and the dynamics is driven using Brownian motions and spring like bonds between the proteins. Instead of proteins, Rappaport's model uses a coarser model with large triangles or the pentamers as building blocks with Brownian motions and binding is predicted by shape complementarity only. Hagan uses a a scoring term like the Lennard Jones' Potential (atomic attraction-repulsion) and uses Newtonian motion based on derivatives of the potential to move the proteins. Using Monte Carlo simulations, Zandi et al. [282] showed that icosahedral shapes are indeed energetically most favorable. Zlotnick [290–292] applied statistical thermodynamics based equation on the equilibrium concentration and binding affinity to analyze the rate of assembly. However, it is assumed that the shape of the capsomers as well as the final capsid are known and intermediate states are formed by adding one more capsomer at each step. Recently Sitharam et al. proposed a geometric constraint based model to predict the

assembly stages [240]. However, this model does not take physico-chemical properties of the proteins into consideration. Finally, Twarock et al. [143, 193, 256–258] proposed the viral tiling theory and applied it to explain the structure on some capsids which cannot be explained using the Caspar-Klug theory. They also apply the theorem to perform equilibrium analysis based on assembly of tiles [89, 144]. Keef et al. extended the tiling theory to explain the 3D structure of the virus at different radial levels [145].

2.3.3.1 Limitations

From the above discussion, it is clear that the methods which try to analyze the assembly pathway require that the structure of the capsid and the local interaction pattern among the proteins are known. On the other hand, the Caspar-Klug theory and viral tiling theory has the power to predict an infinite series of possible capsids, but cannot predict whether a given protein can form a specific capsid. Hence, using all the current methods, we still cannot predict whether a given protein can form a capsid, or how many different types of capsids it can form, or what kind of local interactions the protein supports, and whether a drug attached to the same protein can prevent such interactions and capsid formation. We provide a multi-protein docking algorithm tailored to detect local interactions based on icosahedral symmetry and use these interaction patterns to predict the possible tilings of the capsid. As a by product we predict the intermediate states as well as compute their binding affinity and perform equilibrium analysis.

Chapter 3

Dynamic Data Structure for Maintaining Collection of Weighted Points with Applications in Molecular Assembly Scoring

In this chapter, we present the “Dynamic Packing Grid" (DPG), a neighborhood data structure with applications in maintaining and manipulating flexible molecules and assemblies, efficient computation of binding affinities in drug design or in molecular dynamics calculations. DPG can efficiently maintain sets of points in 3D using only linear space and supports quasi-constant time insertion, deletion and movement (i.e., updates) of weighted points or groups of points. DPG also supports constant time neighborhood queries from arbitrary points. Applications of DPG in maintenance of molecular surface and polarization energy computations using exhibit marked improvement in time and space requirements over other spatial data structures.[1]

[1]The contents of this chapter appeared in the article-
C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. *Bioinformatics*, 27(1):55–62, 2011.
MR, RAC and CB developed the algorithms, MR and RAC implemented the code, MR performed the experiments and collected data, MR, RAC, and CB wrote the paper.

3.1 Introduction

Consider the case when we have a set M of n weighted points (or balls) such that each ball B is represented as (c, r) with center c and radius r. Such a set can represent a set of particles in a colloid, planets and stars in the galaxy, points on a mesh, atoms of a molecule etc. Now, let us assume we want to compute functions that depend on the location and weight of the points. Consider in particular the two functions we mention below-

Integral with Decaying Kernel

$$\mathcal{F}_{vdW} = \sum_{p \in M_1} \sum_{q \in M_2} \left(\frac{a_{p,q}}{dist(p,q)^12} - \frac{b_{p,q}}{dist(p,q)^6} \right) \tag{3.1.1}$$

where $a_{p,q}$ and $b_{p,q}$ are constants that depend on the nature of points p and q, and $dist(p, q)$ is the Euclidean distance between the points. $\mathcal{F}_{vdW}$ is a well known function that models the long range attraction and short range repulsion of atoms. Similar functions are also used in collision detection in computer graphics and simulations.

Functions with Distance Cutoffs

$$\mathcal{F}_{RC} = \sum_{p \in M_1} \sum_{q \in M_2} w_{p,q} \nu(p, q) \tag{3.1.2}$$

where $w_{p,q}$ is a constant that depend on the nature of points p and q, and $\nu(p,q) = 1$ is 1 if $dist(p,q) \leq \delta$, otherwise $\nu(p,q) = 0$. Such functions appear when we are interested in measuring properties of the overlap region or contact region of two sets of points (atom-atom contacts between molecules, regions of two meshes that come close to each other etc.). See Figure 3.1(b) for an example.

Clearly, a naive approach would require $O(n^2)$ time to compute both of the integrals mentioned above. However, in many cases, the number of pairs of points for which $\nu(p,q) = 1$ is going to be much smaller than n^2. In such cases, we would like to evaluate the functions much faster. It can be achived if we already knew which pairs are close to each other, or given a single point $p \in M_1$, identify its neighbors in M_2 faster than $O(n)$.

It is not hard to see that if all the points are stored in a uniform grid-like data structure, then given any point p, and a distance cutoff δ, one can quickly identify the grid-cells which are within δ from the point p and then focus only on the points inside those grid-cells (cf. Figure 3.1(c)). Though it leads to efficient neighborhood searches, and can support updates (add/remove points) in constant time [92, 93], such a data structure may require $O(n^3)$ space in the worst case.

Here, we present the *Dynamic Packing Grid* (DPG) – a space and time efficient neighborhood data structure that maintains a collection of balls (atoms) in 3-space allowing a range of spherical range queries and updates for rapid scoring of flexible protein-protein interactions.

The efficiency of the data structure results from the assumption that the cen-

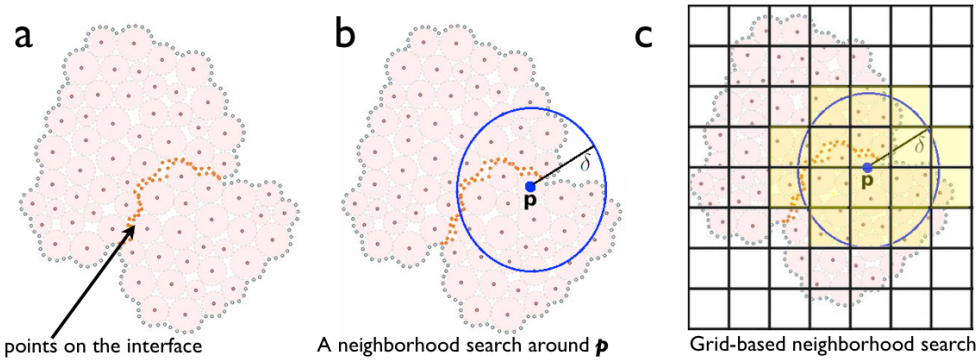


Figure 3.1: **Neighborhood search in 2D**. In (a), we show a collection of points representing the centers of some atoms part of two separate molecules (in 2D), and another set of points that represent the boundary of the molecules. The boundary points which are close to the boundary of the other, are colored differently. A function like $\mathcal{F}_{RC}$ only depend on these points. In (b), we illustrate a typical neighborhood search where we want to find all points within a δ distance from p. Finally, in (c), a grid-based construction is shown where the shaded grid-cells ar the ones that are within δ distance from p, and only points within those need to be checked rigorously.

ters of two different balls in the collection cannot come arbitrarily close to each other, which is a natural property of molecules. A consequence of this assumption is that any ball in the collection can intersect at most a constant number of other balls. On a RAM with w-bit words, our (DPG) data structure can report all balls intersecting a given ball or within $\mathcal{O}(r_{max})$ distance from a given point in $\mathcal{O}(\log \log w)$ time with high probability (w.h.p.), where r_{max} is the radius of the largest ball in the collection. It can also answer whether a given ball is exposed (i.e., lies on the union boundary) or buried within the same time bound. At any time the entire union boundary can be extracted from the data structure in $\mathcal{O}(m)$ time in the worst-case, where m is the number of atoms on the boundary. Updates (i.e., inser-

tion/deletion/movement of a ball) are supported in $\mathcal{O}(\log w)$ time (w.h.p.). The data structure uses linear space.

We also report how DPG can be used to maintain different surface representations of a molecule within the performance bounds mentioned above. An augmented hierarchical version of (DPG) is also reported which fast approximations of functions involving decaying kernels (as opposed to strict distance cutoffs), and show that an application in molecular energetics calculation achieves significant speedup compared to naive implementation, without small approximation error.

The rest of the chapter is organized as follows. We describe and analyze the packing grid data structure in Section 3.2. We give some preliminaries in Section 3.2.1, describe the layout of the data structure in Section 3.2.2, and describe and analyze the supported queries and updates in Section 3.2.3. In Section 3.3.1, we describe how DPG supports efficient maintenance of molecular surfaces and in Section 3.3.3 we describe applications of hierarchical packing grids for solvation energy computations. Some experimental results evaluating the performare of the queries and updates are presented in Section 3.4.2, comparative performance analysis for molecular surface maintenance are included in Section 3.4.3 and performance in solvation energy calculations are presented in Section 3.4.4.

3.2 The Dynamic Packing Grid Data Structure

We describe the *packing grid data structure* for maintaining a set M of balls in 3-space efficiently under the following set of queries and updates. By $B = (c, r)$ we denote a ball withcenter c and radius r.

Queries.

1. INTERSECT($c,\ r$): Return all balls in M that intersect the given ball $B = (c, r)$. The given ball may or may not belong to the set M.

2. RANGE($p,\ \delta$): Return all balls in M with centers within distance δ of point p. We assume that δ is at most a constant multiple of the radius of the largest ball in M.

3. EXPOSED($c,\ r$): Returns *true* if the ball $B = (c, r)$ contributes to the outer boundary of the union of the balls in M. The given ball must belong to M.

4. SURFACE( ): Returns the outer boundary of the union of the balls in M. If there are multiple disjoint outer boundary surfaces defined by M, the routine returns any one of them.

Updates.

1. ADD($c,\ r$): Add a new ball $B = (c, r)$ to the set M.

2. REMOVE($c,\ r$): Remove the ball $B = (c, r)$ from M.

3. MOVE($c_1,\ c_2,\ r$): Move the ball with center c_1 and radius r to a new center c_2.

We assume that at all times during the lifetime of the data structure the following holds.

	TIME COMPLEXITY	
OPERATIONS	ASSUMING $t_q = \mathcal{O}(\log\log w)$, $t_u = \mathcal{O}(\log w)$	ASSUMING $t_q = \mathcal{O}(\log\log n)$, $t_u = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$
RANGE($p,\ \delta$) | INTERSECT($c,\ r$) | EXPOSED($c,\ r$) ($\delta = \mathcal{O}(r_{max})$)	$\mathcal{O}(\log\log w)$ (w.h.p.)	$\mathcal{O}(\log\log n)$ (w.h.p.)
SURFACE( )	$\mathcal{O}$ (#balls on surface) (worst-case)	
ADD($c,\ r$) | REMOVE($c,\ r$) | MOVE($c_1,\ c_2,\ r$)	$\mathcal{O}(\log w)$ (w.h.p.)	$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ (w.h.p.)
ASSUMPTIONS: (i) RAM with w-bit Words, (ii) Collection of n Balls, and (iii) $r_{max} = \mathcal{O}$ (minimum distance between two balls)		

Table 3.1: Time complexities of the operations supported by the packing grid data structure.

Assumption 3.2.1. *If r_{max} is the radius of the largest ball in M, and d_{min} is the minimum Euclidean distance between the centers of any two balls in M, then $r_{max} = \mathcal{O}(d_{min})$.*

In general, a ball in a collection of n balls in 3-space can intersect $\Theta(n)$ other balls in the worst case, and it has been shown in [67] that the boundary defined by the union of these balls has a worst-case combinatorial complexity of $\Theta(n^2)$. However, if M is a "union of balls" representation of the atoms in a molecule, then assumption 3.2.1 holds naturally [119, 261], and as proved in [119], in that case, both complexities improve by a factor of n. The following theorem states the consequences of the assumption.

Theorem 3.2.1. *(Theorem 2.1 in [119], slightly modified) Let $M = \{B_1, \ldots, B_n\}$ be a collection of n balls in 3-space with radii $r_1, \ldots, r_n$ and centers at $c_1, \ldots, c_n$.*

Let $r_{max} = \max_i \{r_i\}$ *and let* $d_{min} = \min_{i,j} \{d(c_i, c_j)\}$*, where* $d(c_i, c_j)$ *is the Euclidean distance between* c_i *and* c_j*. Also let* $\delta M = \{\delta B_1, \ldots, \delta B_n\}$ *be the collection of spheres such that* δB_i *is the boundary surface of* B_i*. If* $r_{max} = \mathcal{O}(d_{min})$ *(i.e., Assumption 3.2.1 holds), then:*

(i) *Each* $B_i \in M$ *intersects at most* $216 \cdot (r_{max}/d_{min})^3 = \mathcal{O}(1)$ *other balls in* M*.*

(ii) *The maximum combinatorial complexity of the boundary of the union of the balls in* M *is* $\mathcal{O}\left((r_{max}/d_{min})^3 \cdot n\right)$
$= \mathcal{O}(n)$*.*

Proof. Similar to the proof of Theorem 2.1 in [119]. ■

Therefore, as Theorem 3.2.1 suggests, for intersection queries and boundary construction, one should be able to handle M more efficiently if assumption 3.2.1 holds. The efficiency of our data structure, too, partly depends on this assumption.

3.2.1 Preliminaries

Before we describe our data structure we present several definitions in order to simplify the exposition.

Definition 3.2.1 (r-grid and grid-cell)**.** *An* r-grid *is an axis-parallel infinite grid structure in 3-space consisting of cells of size* $r \times r \times r$ *(*$r \in \mathbb{R}$*) with the* root *(i.e., the corner with the smallest* x*,* y*,* z *coordinates) of one of the cells coinciding with*

origin of the (Cartesian) coordinate axes. The grid cell that has its root at Cartesian coordinates (ar, br, cr) *(where* $a, b, c \in \mathbb{Z}$*) is referred to as the* (a, b, c, r)*-cell or simply as the* (a, b, c)*-cell when* r *is clear from the context.*

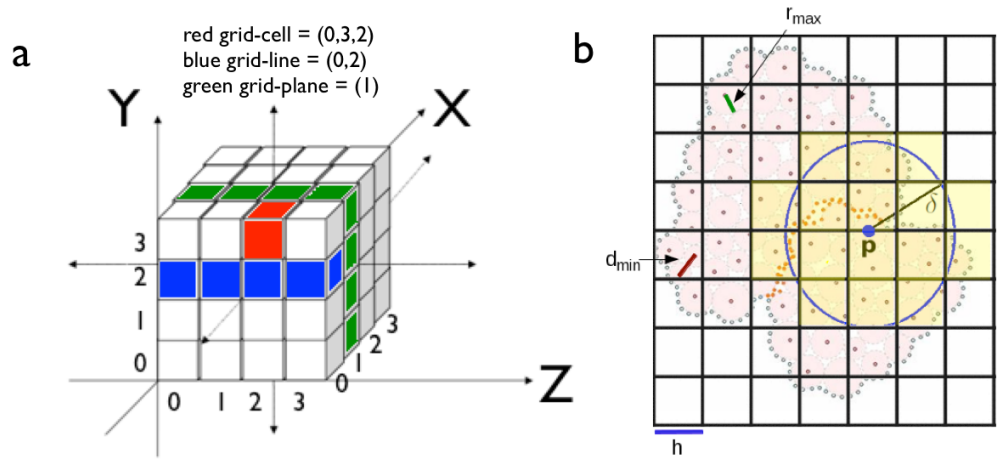


Figure 3.2: **The grid data structure and its parameters**. In (a), we show an uniform grid whose cells, lines and planes are indexed by integer coordinates. Note that DPG does not maintain the entire grid, but only the non-empty components (see text and Figure 3.3 for details). In (b), we provide and example 2D grid with points in it, to demonstrate the intuitive meanings of d_{min}, r_{max} and the cell width h (for instance, in Lemma 3.2.1, $h = 2r_{max}$.

Definition 3.2.2 (grid-line)**.** *The* (b, c, r)-line *(where* $b, c \in \mathbb{Z}$*) in an* r*-grid consists of all* (x, y, z, r)*-cells with* y *and* z *fixed to* b *and* c*, respectively. When* r *is clear from the context the* (b, c, r)*-line will simply be called the* (b, c)*-line.*

Observe that each cell on the (b, c, r)-line can be identified with a unique integer, e.g., the cell at index $a \in \mathbb{Z}$ on the given line corresponds to the (a, b, c, r)-cell in the r-grid. See Figure 3.2(a).

Definition 3.2.3 (grid-plane)**.** *The* (c, r)-plane *(where* $c \in \mathbb{Z}$*) in an* r*-grid consists of all* (x, y, z, r)*-cells with* z *fixed to* c*. The* (c, r)*-plane will be referred to as the* c*-plane when* r *is clear from the context.*

The (c, r)*-plane* can be decomposed into an infinite number of lines each identifiable with a unique integer. For example, index $b \in \mathbb{Z}$ uniquely identifies the (b, c, r)-line on the given plane. Also each grid-plane in the r-grid can be identified with a unique integer, e.g., the (c, r)-plane is identified by c. The proof of the following lemma is straight-forward.

Lemma 3.2.1. *Let* $M = \{B_1, \ldots, B_n\}$ *be a collection of* n *balls in 3-space with radii* $r_1, \ldots, r_n$ *and centers at* $c_1, \ldots, c_n$*. Let* $r_{max} = \max_i \{r_i\}$ *and let* $d_{min} = \min_{i,j} \{d(c_i, c_j)\}$*, where* $d(c_i, c_j)$ *is the Euclidean distance between* c_i *and* c_j*. Suppose* M *is stored in the* $2r_{max}$*-grid* G *(cf. Figure 3.2(b)). Then*

- (i) *If* $r_{max} = \mathcal{O}(d_{min})$ *(i.e., Assumption 3.2.1 holds) then each grid-cell in* G *contains the centers of at most* $64 \cdot (r_{max}/d_{min})^3 = \mathcal{O}(1)$ *balls in* M*.*
- (ii) *Each ball in* M *intersects at most 8 grid-cells in* G*.*
- (iii) *For a given ball* $B \in M$ *with center in grid-cell* C*, the center of each ball intersecting* B *lies either in* C *or in one of the 26 grid-cells adjacent to* C*.*
- (iv) *The number of* non-empty *(i.e., containing the center of at least one ball in* M*) grid-cells in* G *is at most* n*, and the same bound holds for grid-lines and grid-planes.*

At the heart of our data structure is a fully dynamic one dimensional integer range reporting data structure for word RAM described in [184]. The data structure in [184] maintains a set S of integers under updates (i.e., insertions and deletions), and answers queries of the form: report any or all points in S in a given interval. The following theorem summarizes the performance bounds of the data structure which are of interest to us.

Theorem 3.2.2. *(proved in [184]) On a RAM with w-bit words the fully dynamic one dimensional integer range reporting problem can be solved in linear space, and w.h.p. bounds of $\mathcal{O}(t_u)$ and $\mathcal{O}(t_q + k)$ on update time and query time, respectively, where k is the number of items reported, and*

(i) $t_u = \mathcal{O}(\log w)$ *and* $t_q = \mathcal{O}(\log \log w)$ *using the data structure in [184]; and*

(ii) $t_u = \mathcal{O}(\log n / \log \log n)$ *and* $t_q = \mathcal{O}(\log \log n)$ *using the data structure in [184] for small w and a fusion tree [98] for large w.*

The data structure can be augmented to store satellite information of size $\mathcal{O}(1)$ with each integer without degrading its asymptotic performance bounds. Therefore, it supports the following three functions:

1. INSERT($i,\ s$): Insert an integer i with satellite information s.
2. DELETE(i): Delete integer i from the data structure.
3. QUERY($l,\ h$): Return the set of all $\langle\ i,\ s\ \rangle$ tuples with $i \in [l, h]$ stored in the data structure.

3.2.2 Description (Layout) of the Packing Grid Data Structure

We are now in a position to present our data structure. Let DPG be the data structure. We represent the entire 3-space as a $2r_{max}$-grid (see Definition 3.2.1), and maintain the non-empty grid-planes (see Definition 3.2.3), grid-lines (see Definition 3.2.2) and grid-cells (see Definition 3.2.1) in DPG. A grid component (i.e., cell, line or plane) is non-empty if it contains the center of at least one ball in M. The data structure can be described hierarchically. It has a tree structure with 5 levels: 4 internal levels (levels 3, 2, 1 and 0) and an external level of leaves (see Figure 3.3). The description of each level follows.

The Leaf Level "Ball" Data Structure (DPG_{-1}). The data structure stores the center $c = (c_x, c_y, c_z)$ and the radius r of the given ball B. It also includes a Boolean flag $exposed$ which is set to *true* if B contributes to the outer boundary of the union of the balls in M, and *false* otherwise. If another ball B' intersects B, it does so on a circle which divides the boundary δB of B into two parts: one part is buried inside B' and hence cannot contribute to the union boundary, and the other part is exposed w.r.t. B' and hence might appear on the union boundary. The circular intersections of all balls intersecting B define a 2D arrangement A on δB which according to Theorem 3.2.1 has $\mathcal{O}(1)$ combinatorial complexity. A face of A is exposed, i.e., contributes to the union boundary, provided it is not buried inside any other ball. Observe that if at least one other ball intersects B, and A has an exposed face f, then each edge of f separates f from another exposed face f' which belongs to the arrangement A' of a ball intersecting B. We store all exposed faces (if any) of A in a set F of size $\mathcal{O}(1)$, and with each face f we store pointers to

the data structures of $\mathcal{O}(1)$ other balls that share edges with f and also the identifier of the corresponding face on each ball. Observe that if B does not intersect any other balls then F will contain only a single face and no pointers to any other balls.

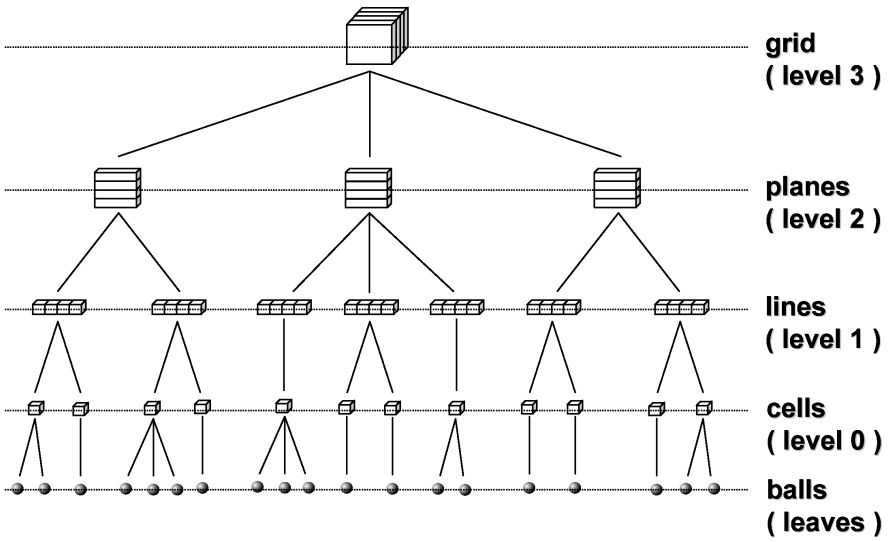


Figure 3.3: Hierarchical structure of DPG.

The Level 0 "Grid-Cell" Data Structure (DPG_0). The "grid-cell" data structure stores the root (see Definition 3.2.1) (a, b, c) of the grid-cell it corresponds to. A grid-cell can contain the centers of at most $\mathcal{O}(1)$ balls in M (see Lemma 3.2.1). Pointers to data structures of all such balls are stored in a set S of size $\mathcal{O}(1)$. Since we create "grid-cell" data structures only for non-empty grid-cells, there will be at most n (and possibly $\ll n$) such data structures, where n is the current number of balls in M.

The Level 1 "Grid-Line" Data Structure (DPG_1). We create a "grid-line" data

structure for a (b, c)-line provided it contains at least one non-empty grid-cell. The data structure stores the values of b and c. Each (a, b, c)-cell lying on this line is identified with the unique integer a, and the identifier of each such non-empty grid-cell is stored in an integer range search data structure RR as described in Section 3.2.1 (see Theorem 3.2.2). We augment RR to store the pointer to the corresponding "grid-cell" data structure with each identifier it stores. The total number of "grid-line" data structure created is upper bounded by n and possibly much less than n.

The Level 2 "Grid-Plane" Data Structure (DPG_2). A "grid-plane" data structure is created for a c-plane provided it contains at least one non-empty grid-line. Similar to the "grid-line" data structure it identifies each non-empty (b, c)-line lying on the c-plane with the unique integer b, and stores the identifiers in a range reporting data structure RR described in Section 3.2.1. A pointer to the corresponding "grid-line" data structure is also stored with each identifier. The data structure also stores c. The total number of "grid-plane" data structures created cannot exceed n, and will possibly be much less than n.

The Level 3 "Grid" Data Structure (DPG_3). This data structure maintains the non-empty grid-planes of the $2r_{max}$-grid in an integer range reporting data structure RR (see Section 3.2.1). Each c-plane is identified by the unique integer c, and each such integer stored in RR is also accompanied by a pointer to the corresponding "grid-plane" data structure. The "grid" data structure also stores a *surface-root* pointer which points to the "Ball" data structure of an arbitrary exposed ball in M.

We have the following lemma on the space usage of the data structure.

Lemma 3.2.2. *Let M be a collection of n balls as defined in Theorem 3.2.1, and let Assumption 3.2.1 holds. Then the packing grid data structure storing M uses $\mathcal{O}(n)$ space.*

Proof. The space usage of the data structure is dominated by the space used by the range reporting data structures, the grid-cells and the "ball" data structures. Since the range reporting data structures use linear space (see Theorem 3.2.2) and total number of non-empty grid components (i.e., planes, lines and cells) is $\mathcal{O}(n)$ (see Lemma 3.2.1), total space used by all such data structures is $\mathcal{O}(n)$. The grid cells contain pointers to "ball" data structures, and since no two grid-cells point to the same "ball" data structure, total space used by all grid-cells is also $\mathcal{O}(n)$. Each "ball" data structure contains the arrangement A and the face decomposition F of the exposed (if any) faces of the ball. The total space needed to store all such arrangements and decompositions is $\mathcal{O}\left(\left(r_{max}/d_{min}\right)^3 \cdot n\right)$ (see Theorem 3.2.1) which reduces to $\mathcal{O}(n)$ under Assumption 3.2.1. Thus the total space used by the data structure is $\mathcal{O}(n)$. ■

3.2.3 Queries and Updates

The queries and updates supported by the data structure are implemented as follows.

3.2.3.1 Queries.

(1) RANGE($p,\ \delta$)**:** Let $p = (p_x, p_y, p_z)$. We perform the following steps.

i. **Level 3 Range Query:** We invoke the function QUERY($l,\ h$) of the range reporting data structure RR under DPG_3 (i.e., the level 3 "grid" data structure) with $l = \lfloor (p_z - \delta)/(2r_{max}) \rfloor$ and $h = \lfloor (p_z + \delta)/(2r_{max}) \rfloor$. This query returns a set S_2 of tuples, where each tuple $\langle\ c,\ P_c\ \rangle \in S_2$ refers to a non-empty c-plane with a pointer P_c to its level 2 "grid-plane" data structure.

ii. **Level 2 Range Query:** For each $\langle\ c,\ P_c\ \rangle \in S_2$, we call the range query function under the corresponding level 2 data structure with $l = \lfloor (p_y - \delta')/(2r_{max}) \rfloor$ and $h = \lfloor (p_y + \delta')/(2r_{max}) \rfloor$, where $(\delta')^2 = \delta^2 - (c - p_z)^2$ if $c - p_z < \delta$, and $\delta' = r_{max}$ otherwise. This query returns a set $S_{1,c}$ of triples, where each triple $\langle\ b,\ c,\ P_{b,c}\ \rangle \in S_{1,c}$ refers to a non-empty $(\ b,\ c\ )$-line with a pointer $P_{b,c}$ to its level 1 "grid-line" data structure. We obtain the set S_1 by merging all $S_{1,c}$ sets.

iii. **Level 1 Range Query:** For each $\langle\ b,\ c,\ P_{b,c}\ \rangle \in S_1$, we call the integer range query function under the corresponding level 1 "grid-line" data structure with $l = \lfloor (p_x - \delta'')/(2r_{max}) \rfloor$ and $h = \lfloor (p_x + \delta'')/(2r_{max}) \rfloor$, where $(\delta'')^2 = \delta^2 - (b - p_y)^2 - (c - p_z)^2$ if $\delta^2 > (b - p_y)^2 + (c - p_z)^2$, and $\delta'' = r_{max}$ otherwise. This query returns a set $S_{0,b,c}$ of quadruples, where

each quadruples $\langle\ a,\ b,\ c,\ P_{a,b,c}\ \rangle \in S_{0,b,c}$ refers to a non-empty $(\ a,\ b,\ c\ )$-cell with a pointer $P_{a,b,c}$ to its level 0 "grid-cell" data structure. We obtain the set S_0 by merging all $S_{0,b,c}$ sets.

***iv*. <u>Ball Collection</u>:** For each $\langle\ a,\ b,\ c,\ P_{a,b,c}\ \rangle \in S_0$, we collect from the level 0 data structure of the corresponding $(\ a,\ b,\ c\ )$-cell each ball whose center lies within distance δ from p. We collect the pointer to the leaf level "ball" data structure of each such ball in a set S, and return this set.

The correctness of the function follows trivially since it queries a region in 3-space which includes the region covered by a ball of radius δ centered at p. It is straightforward to see that the function makes at most $\mathcal{O}\left(\pi \cdot (\lceil \delta / r_{max} \rceil + 1)^2\right)$ calls to a range reporting data structure, and collects balls from at most $\mathcal{O}\left(\frac{4}{3}\pi \cdot (\lceil \delta / r_{max} \rceil + 1)^3\right)$ grid-cells. Using Lemma 3.2.1 and Theorem 3.2.2, we conclude that w.h.p. the function terminates in $\mathcal{O}\left((\delta / r_{max})^2 \cdot t_q + ((\delta + r_{max})/d_{min})^3\right)$ time. Assuming $r_{max} = \mathcal{O}(d_{min})$ (i.e., Assumption 3.2.1) and $\delta = \mathcal{O}(r_{max})$, the complexity reduces to $\mathcal{O}(t_q)$ (w.h.p.).

(2) INTERSECT$(\ c,\ r\ )$: Let $B = (c, r)$ be the given ball. We perform the following two steps.

***i*. <u>Ball Collection</u>:** We call RANGE$(\ c,\ r + r_{max}\ )$ and collect the output in set S which contains pointers to the data structure of each ball in M with its center within distance $r + r_{max}$ from c.

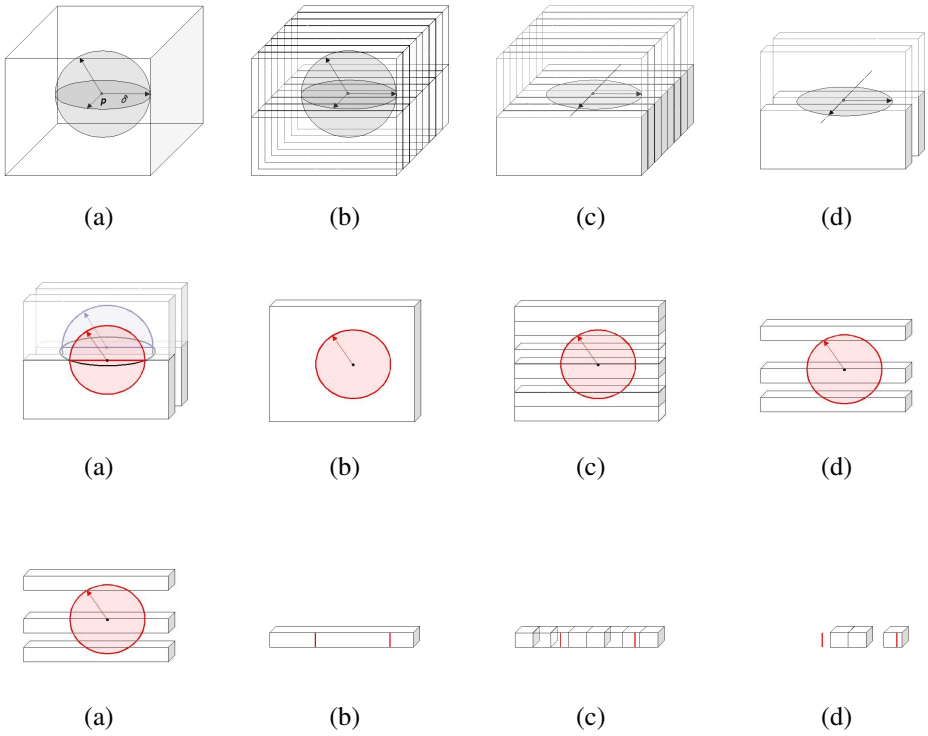


Figure 3.4: **Top row:** Level 3 Range Query. (a) Query region defined by the sphere of radius δ at point p inside the level 3 grid, (b) The level 3 grid as a collection of level 2 grid-planes, (c) and (d) Range reporting query returns the set of non-empty grid-planes within the query region. **Middle row:** Level 2 Range Query. (a) On each grid-plane, query region is defined by a circular slice of the sphere of radius δ at point p, (b), (c) and (d) Range reporting query on such a grid-plane returns the set of non-empty grid-lines within the query region. **Bottom row:** Level 1 Range Query. (a) and (b) Query region in each grid-line is defined as an interval, (c) and (d) For each grid-line, range reporting query returns the set of non-empty grid-cells

ii. **Identifying Intersecting Balls:** From S we remove the data structure of each ball that does not intersect B, and return the resulting (possibly reduced) set.

We know from elementary geometry that two balls of radii r_1 and r_2 cannot intersect unless their centers lie within distance $r_1 + r_2$ of each other. Therefore,

step (i) correctly identifies all balls that can possibly intersect B, and step (ii) completes the identification. Step (i) takes
$\mathcal{O}\left(t_q + (r_{max}/d_{min})^3\right)$ time w.h.p., and step (ii) terminates in $\mathcal{O}\left((r_{max}/d_{min})^3\right)$ time in the worst case. Therefore, under Assumption 3.2.1 w.h.p. this function runs in $\mathcal{O}(t_q)$ time.

(3) EXPOSED$(\ c,\ r\ )$**:** Let $B = (c, r)$ be the given ball. We locate B's data structure by calling RANGE$(\ c,\ 0\ )$, and return the value stored in its *exposed* field. Clearly, the function takes $\mathcal{O}\left(t_q + (r_{max}/d_{min})^3\right)$ time (w.h.p.) which reduces to $\mathcal{O}(t_q)$ (w.h.p.) under Assumption 3.2.1.

(4) SURFACE$(\ )$**:** The *surface-root* pointer under the level 3 "grid" data structure points to the "ball" data structure of a ball B on the union boundary of M. We scan the set F of exposed faces of B, and using the pointers to other exposed balls stored in F we perform a depth-first traversal of all exposed balls in M and return the exposed faces on each such ball. Let m be the number of balls contributing to the union boundary of M. Then according to Theorem 3.2.1 the depth-first search takes $\mathcal{O}\left((r_{max}/d_{min})^3 \cdot m\right)$ time in the worst case which reduces to $\mathcal{O}(m)$ under Assumption 3.2.1.

3.2.3.2 Updates.

(1) ADD$(\ c,\ r\ )$**:** Let $c = (c_x, c_y, c_z)$ and let $c'_u = \left\lfloor \frac{c_u}{2r_{max}} \right\rfloor$, where $u \in \{x, y, z\}$. We perform the following steps.

i. If $M \neq \emptyset$, let G be the grid data structure, otherwise create and initialize G. Add input ball to M.

ii. Query the range reporting data structure $G.RR$ to locate the data structure P for the c'_z-plane. If P does not exist create and initialize P, and insert c'_z along with a pointer to P into $G.RR$.

iii. Query $P.RR$ and locate the data structure L for the (c'_y, c'_z)-line. If L does not exist then create and initialize L, and insert c'_y along with a pointer to L into $P.RR$.

iv. Locate the data structure C for the (c'_x, c'_y, c'_z)-cell by querying $L.RR$. Create and initialize C if it does not already exist, and insert c'_x and a pointer to C into $L.RR$.

v. Create and initialize a data structure B for the input ball and add it to the set $C.S$.

vi. Call INTERSECT($c,\ r$) and find the set I of the "ball" data structures of all balls that intersect the input ball. Create the arrangement $B.A$ using the balls in I. The new ball may partly or fully bury some of the balls it intersects, and hence we need to update the arrangement $B'.A$, the set $B'.F$ and the flag $B'.exposed$ of each $B' \in I$. The set $B.F$ is created and $B.exposed$ is initialized using the information in the updated data structures in I. If the *surface-root* pointer was pointing to a ball in I that got completely buried by the new ball, we update it to point to B instead.

Observe that the introduction of a new ball may affect the surface exposure of only the balls it intersects (i.e., bury some/all of them partly or completely), and no other balls. Hence, the updates performed in step (vi) (in addition to those in earlier steps) are sufficient to maintain the correctness of the entire data structure. Steps (i) and (v) take $\mathcal{O}(1)$ time in the worst case, and w.h.p. each of steps (ii), (iii) and (iv) takes $\mathcal{O}(t_q + t_u)$ time. Finding the intersecting balls in step (vi) takes

$\mathcal{O}\left(t_q + (r_{max}/d_{min})^3\right)$ time w.h.p., according to Theorem 3.2.1 creating and updating the arrangements and faces will take $\mathcal{O}\left((r_{max}/d_{min})^3 \times (r_{max}/d_{min})^3\right) = \mathcal{O}\left((r_{max}/d_{min})^6\right)$ time (w.h.p.). Thus the ADD function terminates in

$\mathcal{O}\left(t_q + t_u + (r_{max}/d_{min})^6\right)$ time w.h.p., which reduces to $\mathcal{O}(t_u)$ (w.h.p.) assuming $r_{max} = \mathcal{O}(d_{min})$ (i.e., Assumption 3.2.1).

(2) REMOVE($c,\ r$)**:** This function is symmetric to the ADD function, and has exactly the same asymptotic time complexity. Hence, we do not describe it here.

(3) MOVE($c_1,\ c_2,\ r$)**:** This function is implemented in the obvious way by calling **REMOVE**($c_1,\ r$) followed by **ADD**($c_2,\ r$). It has the same asymptotic complexity as the two functions above.

Therefore, we have the following theorem.

Theorem 3.2.3. *Let M be a collection of n balls in 3-space as defined in Theorem 3.2.1, and let Assumption 3.2.1 holds. Let t_q and t_u be as defined in Theorem 3.2.2. Then the packing grid data structure storing M on a word RAM:*

(i) *uses $\mathcal{O}(n)$ space;*

(ii) *supports updates (i.e., insertion/deletion/movement of a ball) in $\mathcal{O}(t_u)$ time w.h.p.;*

(iii) *reports all balls intersecting a given ball or within $\mathcal{O}(r_{max})$ distance from a given point in $\mathcal{O}(t_q)$ time w.h.p., where r_{max} is the radius of the largest ball in M; and*

(iv) *reports whether a given ball is exposed or buried in $\mathcal{O}(t_q)$ time w.h.p., and returns the entire outer union boundary of M in $\mathcal{O}(m)$ worst-case time, where m is the number of balls on the boundary.*

In Table 3.1 we list the time complexities of the operations supported by our data structure.

3.3 Applications of DPG in Molecular Modeling

3.3.1 Molecular Surface Maintenace Using DPG

In this section, we briefly describe applications of the packing grid data structure for efficient maintenance of molecular surfaces.

3.3.1.1 Maintaining van der Waals Surface of Molecules

For dynamic maintenance of the van der Waals surface of a molecule we can use the packing grid data structure directly. Each atom is treated as a ball with a radius equal to the van der Waals radius of the atom (see [30] for a list of van der Waals radius of different atoms).

3.3.1.2 Maintaining Lee-Richards (SCS/SES) Surface

We can use the packing grid data structure for the efficient maintenance of the Lee-Richards surface of a molecule under insertion/deletion/movement of atoms. The performance bounds given in Table 3.1 remain unchanged. We maintain two packing grid data structures: DPG and DPG'. The DPG data structure keeps track of the patches on the Lee-Richards surface, and DPG' is used for detecting intersections among concave patches.

Before adding an atom to DPG, we increase its radius r_s, where r_s is the radius of the rolling solvent atom. The DPG data structure keeps track of all solvent exposed atoms, i.e., all atoms that contribute to the outer boundary of the union of these enlarged atoms. Theorem 3.2.1 implies that each atom in DPG contributes $\mathcal{O}(1)$ patches to the Lee-Richards surface, and the insertion/deletion/movement of an atom results in local changes of only $\mathcal{O}(1)$ patches. We can modify DPG to always keep track of where two or three of the solvent exposed atoms intersect, and once we know the atoms contributing to a patch we can easily compute the patch in $\mathcal{O}(1)$ time [16].

The Lee-Richards surface can self-intersect in two ways: (i) a toroidal patch

can intersect itself, and (ii) two different concave patches may intersect [16]. The self-intersections of toroidal patches can be easily detected from DPG. In order to detect the intersections among concave patches, we maintain the centers of all current concave patches in DPG', and use the INTERSECT query to find the concave patch (if any) that intersects a given concave patch.

3.3.2 Hierarchical Packing Grids for Mixed Resolution Surfaces

We construct a k-level hierarchical packing grid data structure $\mathsf{HPG}(k)$ as follows. For $i \in [0, k-1]$, level i contains a packing grid data structure $\mathsf{DPG}^{(i)}$ with parameters $\langle r_{max}^{(i)}, d_{min}^{(i)} \rangle$ for which Assumption 3.2.1 holds. We also assume that for $i \in [0, k-2]$, $r_{max}^{(i+1)} = \Theta\left(r_{max}^{(i)}\right)$ and $d_{min}^{(i+1)} = \Theta\left(d_{min}^{(i)}\right)$. The level 0 data structure $\mathsf{DPG}^{(0)}$ contains the atomic level union of balls representation of the given molecule M. For $i \in [1, k-1]$, $\mathsf{DPG}^{(i)}$ contains a coarser representation of the molecule represented in $\mathsf{DPG}^{(i-1)}$. Each ball in $\mathsf{DPG}^{(i)}$ represents a grouping several neighboring balls in $\mathsf{DPG}^{(i-1)}$. A single doubly linked list links the parent ball in $\mathsf{DPG}^{(i)}$ to all its child balls in $\mathsf{DPG}^{(i-1)}$. Additionally, each child ball maintains a direct pointer to its parent ball. Thus given the center of any ball in $\mathsf{DPG}^{(i)}$, the set of all its children in $\mathsf{DPG}^{(i-1)}$ can be found in $\mathcal{O}\left(t_q + l\right)$ time w.h.p., where l is the number of children of the given ball, and t_q is as defined in Theorem 3.2.2. We assume that each ball in $\mathsf{DPG}^{(i-1)}$ has at most one parent in $\mathsf{DPG}^{(i)}$, and thus the balls in $\mathsf{HPG}(k)$ form a forest. Now in order to create a mixed resolution surface of the given molecule M, we start at coarse resolution, say at some level $j > 0$, and copy $\mathsf{DPG}^{(i)}$ to an initially empty packing grid DPG with the same parameters.

Now we selectively replace balls in DPG with finer resolution balls from the appropriate level in HPG(k), and we keep replacing until we get the required mixed resolution representation of M in DPG.

3.3.3 Energetics Computation using DPG

Free energy computations is an integral component in molecular dynamics applications, where it is required to estimate the force fields and hence infer the motion paths. The difference of free energy between a candidate complex and the individual molecules is often used to filter false positives in molcular docking applications. But computing the energy efficiently with acceptable degree of accuracy has been a bottleneck in such applications. But leveraging hierarchical DPG, energetics computation can be performed efficiently.

Generally, the solvation energy G_{sol} of a molecule is decomposed into three components, namely, G_{cav} - the energy to form cavity in the solvent, G_{vdw} - the solute-solvent van der Waals interaction energy, and G_{pol} - the polarization energy or the electrostatic potential energy change due to the solvation.

$$G_{\text{sol}} = G_{\text{cav}} + G_{\text{vdw}} + G_{\text{pol}} \tag{3.3.3}$$

The first two terms G_{cav} and G_{vdw} in the sum above are linearly related to the solvent accessible surface area Ω_{SAS} of the molecule.

$$G_{\text{cav}} + G_{\text{vdw}} = \gamma \cdot \Omega_{\text{SAS}} \tag{3.3.4}$$

The last term, G_{pol}, can be approximated using the *Generalized Born* (GB) theory as follows [245].

$$G_{pol} = -\frac{\tau}{2} \sum_{i,j} \frac{q_i q_j}{\sqrt{r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}}}, \tag{3.3.5}$$

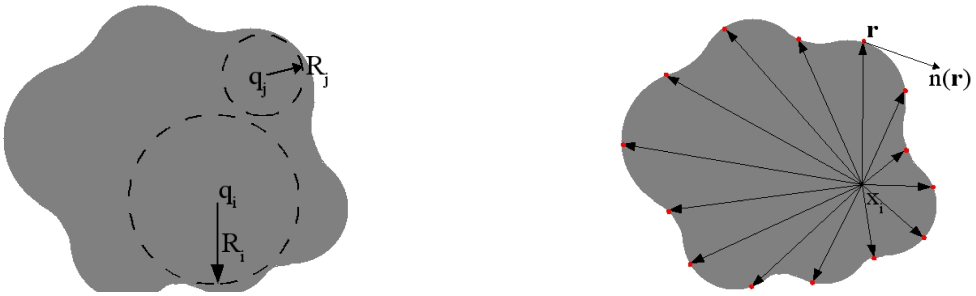
where $\tau = 1 - \frac{1}{\epsilon}$, and R_i is the effective Born radius of atom i (see Figure 3.5(a)). The R_i's can be approximated as follows.

$$R_i^{-1} = \frac{1}{4\pi} \int_{\Gamma} \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} dS, \tag{3.3.6}$$

where Γ is the boundary of the molecule, $\mathbf{n}(\mathbf{r})$ is the normal of the molecular surface at $\mathbf{r}$ pointing out of the molecule, and $\mathbf{x}_i$ is the center of atom i. A discrete approximation of R_i^{-1} based on equation 3.3.6 is as follows [22].

$$R_i^{-1} = \frac{1}{4\pi} \sum_{k=1}^{N} w_k \frac{(\mathbf{r}_k - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r}_k)}{|\mathbf{r}_k - \mathbf{x}_i|^4}, \tag{3.3.7}$$

where the $\mathbf{r}_k$'s are N carefully chosen integration points on the boundary of the molecule, and w_k is a weight assigned to $\mathbf{r}_k$ in order to achieve higher order of accuracy for small N (see Figure 3.5(b)).



(a) (b)

Figure 3.5: (a) G_{pol} is computed based on Born radii and charges of each atom pair, (b) Born radii of an atom can be approximated based on integration points, shown as red dots, sampled on the surface

The non-polar terms G_{cav} and G_{vdw} can be computed directly from the solvent accessible surface (SAS) area Ω_{SAS} of the molecule (see equation 3.3.4). The SAS of the molecule can be extracted in $\mathcal{O}\left(\widetilde{m}\log w\right)$ (w.h.p.) time and $\mathcal{O}\left(\widetilde{m}\right)$ space using a DPG data structure, where $\widetilde{m}$ is the number of atoms in the molecule. The DPG data structure outputs the SAS as a set of spherical (convex and concave) and toroidal patches, and we add up the area of each patch in order to calculate Ω_{SAS}.

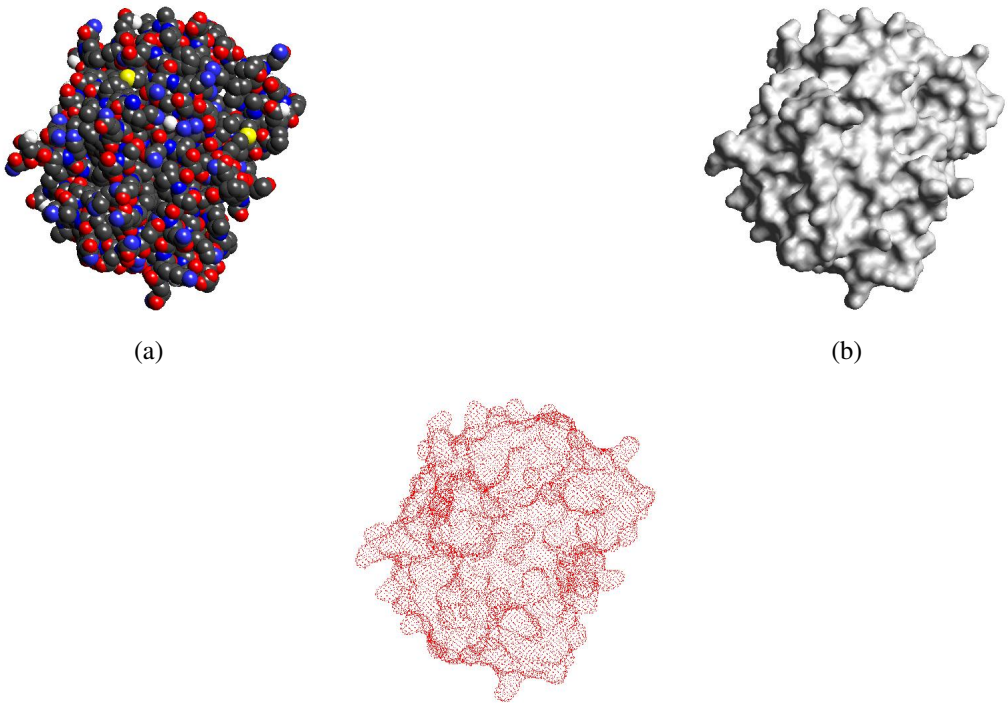
In order to approximate the polar term G_{pol} first we need to approximate the Born radius R_i of each atom i, the details of which is presented in Section 3.3.3.1. Assuming that $\widetilde{m}_{\widetilde{\delta}}$ is an upper bound on the number of atoms within distance $\widetilde{\delta}$ from any given point in space, the time spent for computing all R_i's is $\mathcal{O}\left(N\log\log w + N\widetilde{m}_{\widetilde{\delta}}\right)$ which reduces to $\mathcal{O}\left(N\log\log w\right)$ (w.h.p.) since $\widetilde{m}_{\widetilde{\delta}}$ is a constant (though could be quite large) for constant $\widetilde{\delta}$. Once all R_i's are computed G_{pol} can be computed using equation 3.3.5 in $\mathcal{O}\left(\widetilde{m}^2\right)$ time in the worst case. The space usage is $\mathcal{O}\left(\widetilde{m} + N\widetilde{m}_{\widetilde{\delta}}\right)$ which is $\mathcal{O}\left(\widetilde{m} + N\right)$ for constant $\widetilde{\delta}$.

3.3.3.1 Discrete Approximation of Born Radii

We use the discrete approximation equation 3.3.7 for computing R_i. Given the solvent excluded surface (SES) of the molecule, it has been shown in [21] how to choose N integration points $\mathbf{r}_k$ and weights w_k optimally in order to reduce the error in approximation. Figure 3.6 shows the distribution of integration points on the surface of 1MAG.PDB. We compute the SES using the A-spline based method introduced in [288], produce a quality improved meshing of the surface using the method of [286] and then sample integration points and their weights following [21].

Observe that direct computation of R_i^{-1} using Equation 3.3.7 requires $O(n^2)$ time, where n is the number of atoms and assuming that the number of sampled integration points is also $O(n)$. However, since the terms in the summation diminish very fast with the increase of distance, distance cutoffs can be used to approximate it. Given, the set of atoms $\mathcal{A}$, the set of integration points Ω sampled on the surface, and two user defined parameters $\alpha, \delta > 0$, for every integration point $q \in \Omega$, we place each atom $a \in \mathcal{A}$ in one of the following three categories based on the distance d between q and the center of a: (1) *near* ($d \leq \delta$), (2) *mid-way* ($\delta < d \leq \alpha\delta$), and (3) *far* ($\alpha\delta < d$). Figure 3.7 shows an example in 2D. For the near categories, the computation is performed exactly. For the mid-way category, clusters of atoms and integration points are viewed as pseudo atoms and pseudo-integration points and hence a coarse computation is performed. And, for the far category a single average distance is used for all pair of clusters.

To cluster the atoms and integration points, first all atoms and integration

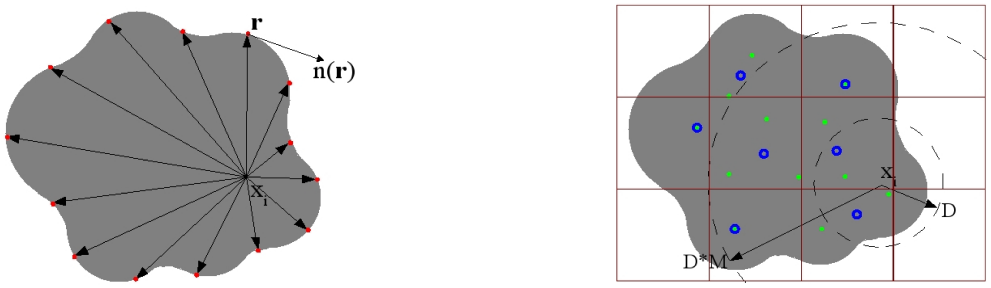


(a) (b)

(c)

Figure 3.6: Gaussian integration points (c) on the surface of nuclear transport factor 2 (1A2K) computed after generating a smooth surface (b) from the collection of balls model (a)

points are inserted into two separate (DPG) data structures. Let us denote them as $\mathcal{P}_a$ and $\mathcal{P}_i$ respectively. The size of the cells for both are set to δ. Now, each cell of the two (DPG)s contains a set of atoms (or integration points). Hence, it is natural to consider each cell as a cluster. So, we use a constant-time cell-membership query for each cell to get the list of atoms (or integration points), and compute their average center. For integration points we additionally need to compute the approximate normals. The approximate normal of the cluster is defined as a weighted sum of the normals of each integration point in the cluster. Then, each cluster is viewed



(a) (b)

Figure 3.7: (a) A simple 2D example depicting definition of near, medium and far atoms (centers shown as green dots) from a particular integration point x_i. In the example, 2 atoms are near, 7 are medium and 3 are far. (b) After cluster using hierarchical (DPG), each cell contains a pseudoatom (centers shown as blue circles). Now 2 atoms are near, 3 clusters are medium and 2 clusters are far

as a pseudoatom (or pseudo-integration point) and inserted into separate (DPG)s, denoted $\mathcal{P}'_a$ and $\mathcal{P}'_i$. RANGE queries are used to identify the near, mid-way and far atoms/pseudoatoms and their partial contribution to the sum are updated based on their distance category. Finally, R_i^{-1} is performed by summing up the partial contributions of each atom and pseudoatom in $\mathcal{P}_a$ and $\mathcal{P}'_a$.

3.3.4 Maintenance of Flexible Molecules

Suppose we are given a flexible molecule decomposed into several (mostly) rigid domains which interact either through connected chain segments or large interfaces. We refer to these chain segments and interfaces as connectors. Domains may move with respect to each other through motions applied to the connectors. Two domains connected by at least one connector may undergo bending motion applied to some hinge point around some hinge axis. If they are connected by only one connector, a twisting motion can also be applied to the connector by updating tor-

sion angles along its backbone. If two domains share a large interface area they may undergo a shearing motion with respect to each other. However, though domains are mostly rigid they may have flexible loops and side-chains on their surfaces.

We maintain a separate packing grid data structure $\mathcal{P}_i$ for each domain $\mathcal{D}_i$. If two domains $\mathcal{D}_i$ and $\mathcal{D}_j$ are connected and $i < j$, the set S_{ij} of all connectors between these two domains are included in $\mathcal{P}_i$, and a transformation matrix M_{ij} is kept with $\mathcal{P}_i$ that describes the exact location and orientation of the grid structure of $\mathcal{P}_j$ with respect to that of $\mathcal{P}_i$. Whenever some motion is applied to the connectors in S_{ij}, we update $\mathcal{P}_i$ in order to reflect the changes in the locations of the atoms in these connectors, and also update M_{ij} in order to reflect the new relative position and orientation of $\mathcal{P}_j$ with respect to $\mathcal{P}_i$. Hence such an update requires $\mathcal{O}\left(1 + m_{ij} \log w\right)$ time (w.h.p.), where m_{ij} is the number of atoms in the connectors in S_{ij}. We defer the tests to check whether any two domains intersect due to these movements until we need to construct the surface of the entire molecule in response to a surface query. At that point we extract the surface atoms from each $\mathcal{P}_i$ and insert them into an initially empty packing grid data structure $\mathcal{P}$ after applying necessary transformations. Thus generating the surface of the entire molecule requires $\mathcal{O}\left(\widehat{m} \log w\right)$ time (w.h.p.), where $\widehat{m}$ is the sum of the number of atoms on the surface of each domain. If we need to update the conformation of a flexible loop or a side-chain on the surface of some domain $\mathcal{D}_i$, we directly update the locations of the atoms affected by this change in $\mathcal{P}_i$. Such an update requires $\mathcal{O}\left(\widetilde{m} \log w\right)$ time (w.h.p.), where $\widetilde{m}$ is the number of atoms affected. Therefore, we have the following lemma.

Lemma 3.3.1. *The surface of a flexible molecule decomposed into (mostly) rigid*

domains can be maintained using packing grid data structures so that

(i) *updating for a bending/shearing/twisting motion applied between two domains takes* $\mathcal{O}\left(1+\overline{m}\log w\right)$ *time (w.h.p.), where* $\overline{m}$ *is the number of atoms in the connectors between the two domains;*

(ii) *updating the conformation of a flexible loop or a side-chain on the surface of a domain takes* $\mathcal{O}\left(\widetilde{m}\log w\right)$ *time (w.h.p.), where* $\widetilde{m}$ *is the number of atoms affected by this change; and*

(iii) *generating the surface of the entire molecule requires* $\mathcal{O}\left(\widehat{m}\log w\right)$ *time (w.h.p.), where* $\widehat{m}$ *is the sum of the number of atoms on the surface of each domain.*

3.4 Results

This section presents and analyzes the performance of the (DPG) data structure. After discussing the implementation details and the testing platform in Section 3.4.1, the performance of the basic functionalities of (DPG) are reported in Section subsec:performance. Sections 3.4.3 and 3.4.4 respectively analyzes performance of (DPG) in molecular surface maintenance and energetics calculation.

3.4.1 Implementation Details

In our current implementation, instead of the 1D integer range-reporting data structure presented in [184], we have implemented a much simpler data structure that supports both updates and distance queries in expected $\mathcal{O}\left(\log w\right)$ time and

uses linear space [75]. Since w is usually not more than 64, for most practical purposes a $\mathcal{O}(\log w)$ query time should be almost as good as $\mathcal{O}(\log \log w)$ time. This data structure builds on binary search trees, dynamic perfect hashing, and y-fast trees [272]. However, instead of dynamic perfect hashing we used "cuckoo hashing" [209] since it is much simpler, and still supports lookups in $\mathcal{O}(1)$ worst-case time, and updates in expected $\mathcal{O}(1)$ time.

In the following section, we report the results on the performance of our implementation of the packing grid data structure for the basic queries and updates. All experiments are performed on a 3 GHz $2\times$dual-core (only one core was used) AMD Opteron 2222 processor with 4 GB RAM. And, in Section 3.3.3.1, results of using hierarchical DPG to approximate born radii for solvation energy computation is reported.

3.4.2 Performance Analysis of Updates and Queries

To measures the performance of the update and query functions of DPG, we use more than 180k quadrature points, generated for energetics computations by sampling uniformly at random on the surface of PSTI (a variant of human pancreatic trypsin inhibitor: 1HPT.pdb) after protonation using PDB2PQR [3]. These points were randomly partitioned into four equal groups. Group 1 was first inserted into DPG and range queries were performed from each atom center of the molecule to report all quadrature points lying within a given distance from the center. Average running time was measured after executing each query multiple times. The same experiment was carried out with query distances 2, 4, 8 and 16 . After running

QUADR. POINTS	QUERY DISTANCE (Å)	AVG. TIME (MS) / QUERY	AVG. # POINTS RETURNED / QUERY	AVG. # POINTS RETURNED / MS
45,654	2 | 4 | 8 | 16	0.311 | 0.566 | 1.420 | 3.379	118 | 775 | 4,466 | 22,839	379 | 1,367 | 3,144 | 6,758
91,309	2 | 4 | 8 | 16	0.588 | 1.139 | 2.801 | 6.158	225 | 1,623 | 9,284 | 44,518	382 | 1,425 | 3,314 | 7,229
136,963	2 | 4 | 8 | 16	0.973 | 1.845 | 4.436 | 9.572	329 | 2,435 | 14,496 | 70,016	338 | 1,320 | 3,268 | 7,314
182,618	2 | 4 | 8 | 16	1.304 | 3,219 | 5.855 | 12.661	439 | 3,401 | 19,307 | 93,443	377 | 1,314 | 3,297 | 7,381

Table 3.2: Performance of the QUERY function of packing grid. We take a molecule (1HPT: a variant of human pancreatic trypsin inhibitor) consisting of about 850 atoms after protonation using PDB2PQR [3], and sample approximately 184,000 quadrature points uniformly at random on its surface. We randomly assign each point to one of four groups and thus obtain four approximately equal-sized groups. We then run queries from the 800 atom centers (100 queries per atom) on group 1; merge groups 1 and 2, and run queries on this merged group; merge groups 1, 2 and 3, and run queries again; and finally run queries on the entire set.

experiments with group 1, group 2 was also inserted into the data structure and the same set of experiments were performed again. In the same manner groups 3 and 4 were also added subsequently so that the results gives a clear measure of the scalability of the data structure. Table 3.2 shows the results of this experiment. The time required is $\mathcal{O}\left(\log w + K\right)$ where K is the size of the output or in this case, the number of points returned. The fifth column of the table shows that, as the point set becomes denser, the efficiency of the data structure remains almost the same.

Table 3.3 reports the performance of update functions of DPG's range reporting data structure. Four different macromoleculeswere used, and for each of them all atoms were first randomly inserted into the data structure followed by the random deletion of all atoms. The reported insertion and deletion times are averages of four such independent runs. The average time for a single insertion/deletion was never more than 5 μs.

MOLECULE (PDB FILE)	NUMBER OF ATOMS	INSERT		DELETE	
		Total Time (ms)	Average Time (μs)	Total Time (ms)	Average Time (μs)
GroEL (1GRL)	29,274	97	3.3	118	4.0
RDV P8 (1UF2: Chain P)	193,620	746	3.9	846	4.4
RDV P3 (1UF2: Chain A)	459,180	1,813	3.9	2,094	4.6
Dengue (1K4R)	545,040	2,176	4.0	2,432	4.5

Table 3.3: Insertion and deletion times of our current packing grid implementation. The results are averages of 4 runs. In each run, all atom centers are randomly inserted into the data structure followed by random deletion of all atom centers.

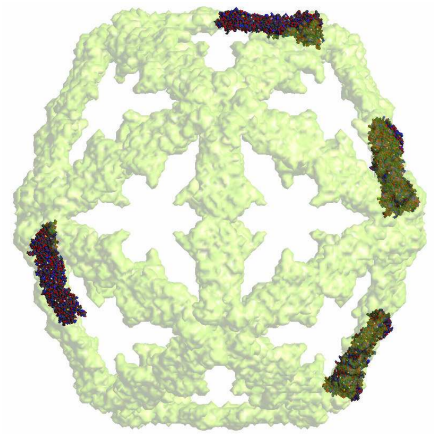


Figure 3.8: RDV capsid protein P3 chain A. The entire structure generated by applying all sixty transformations is rendered in transparent green. The chains generated by the first four transformations are rendered opaque and in atom-based coloring.

3.4.3 Performance of Molecular Surface Maintenance

We compared the performance of DPG with the 3D hashing used in [92, 93] in producing and maintaining molecular surfaces. As our experimental setup we used the same implementation of 3D arrangement and surface generation [93], but switched between the two different range query data structures. We measured the space and time requirements for generating the surface of various molecules and macromolecules. In addition to the molecules used in the experiments of [92,

93], we ran our experiments on some viruses and ribosomes we are interested in. To verify scalability, multiple chains of the same protein were inserted. For virus capsids as multiple chains are inserted, not only the number of atoms increases but also the overall structure becomes sparser. For example, Figure 3.8 shows that though a single chain is dense, if four chains are considered together then their bounding volume becomes sparse. The results of this experiment are reported in Table 3.4. From the table, one can verify that the space requirement of the DPG range query data structure is linear in the number of atoms. Also, its running times are comparable with that of 3D hash while using much less memory. The difference in space requirement becomes more pronounced for larger and sparser structures. Though 3D hash performs insertions and queries in optimal constant time, using too much memory can adversely affect its running time when the set of atoms is sparse as in virus capsids. For example, in the case of RDV P3 with 4 chains, 3D hash operations run slower than DPG range reporting operations. We believe that this slowdown is due to page faults caused by excessive space requirement of 3D hash.

3.4.4 Performance of Born Radii and Polarization Energy Calculation

The approximation scheme described in the Section 3.3.3.1 was applied to compute the Born Radii for 60 complexes of the ZDock Benchmark 2.0. Then these approximate Born radii were used to compute the polarization energy G_{Pol}. For each complex, the approximation performance was tested by comparing the results and the speed with an exact implementation of the discrete formulation of

MOLECULE (PDB FILE)	NO. OF CHAINS	NO. OF ATOMS	NUMBER OF CELLS		TIME (SEC)	
			DPG	3D hash	DPG	3D hash
Trypsin Inhibitor (4PTI)	1	454	196	1,089	0.58	0.54
Carbonic Anhydrase I (1BZM)	1	2,034	856	3,360	2.73	2.58
Fasciculin2 with Acetylcholinesterase (1MAH)	1	4,116	1,726	8,568	6.20	5.70
Anthrax Lethal Factor with MAPKK2 (1JKY)	1	5,614	2,389	16,456	8.52	8.13
RNA Polymerase II (1I3Q)	1	11,114	4,682	45,177	17.36	16.23
Glutamine Synthetase (2GLS)	1	3,636	1,444	9,177	5.43	5.06
	5	18,180	7,275	41,400	37.10	34.80
Nicotinic Acetylcholine Receptor (2BG9)	1	2,991	1,199	10,752	4.44	4.29
	5	14,955	6,027	31,200	24.31	22.95
Rice Dwarf Virus (RDV) P8 (1UF2: Chain P)	1	3,227	1,348	9,261	4.47	4.23
	2	6,454	2,739	1,124,040	9.23	8.56
	3	9,681	4,115	2,506,480	15.17	14.31
	4	12,908	5,467	4,426,110	19.36	18.14
	5	16,135	6,848	4,426,110	30.79	30.20
	6	19,362	8,224	6,052,800	35.65	34.42
	7	22,589	9,605	6,052,800	40.28	38.86
	8	25,816	10,981	6,332,160	45.22	44.44
Rice Dwarf Virus (RDV) P3 (1UF2: Chain A)	1	7,653	3,229	38,760	10.99	10.23
	2	15,306	6,458	927,442	22.73	21.44
	3	22,959	9,739	1,992,747	40.48	39.62
	4	30,612	12,985	2,591,700	119.28	128.37
Dengue Virus (1K4R: Chains A & B)	2	6,056	2,622	20,706	8.46	7.71
	4	12,112	5,237	138,600	17.56	16.52
	6	18,168	7,846	333,060	33.73	32.62

Table 3.4: Comparison of the performance of the 3D range reporting data structure used by DPG, and the 3D hash table used in [93]. The same 3D arrangement code was used in both cases [93]. Table shows the comparative running times and the space requirement (in terms of the number of cells used) for surface generation of different molecules. To verify scalability, molecules of varying sizes and in some cases, multiple chains were used. To generate multiple copies of the molecule we used the transformation matrices given in the corresponding PDBs (e.g., to generate k copies we used the top k matrices).

Born Radii. Three different approximations were performed by varying the D and M parameters, to understand the accuracy/speed tradeoff. In our results, we shall

refer to them as $coarser$, $medium$ and $finer$ approximations.

Figure 3.9(a) shows that the average error is between $5-10\%$ for most of the complexes and it is higher than 20% for only 3 complexes. But interestingly, Figure 3.9(b) shows that even with Born Radii containing 10% or more error, the polarization energy can be computed very accurately. Only 7 complexes have more than 2% error, and even the maximum error is less than 9%. So, the approximation is quite suitable and accurate enough to be used in practice. Another interesting aspect to notice in both Figure 3.9(a) and 3.9(b) is that, the errors consistently decrease for finer approximations.

Figure 3.9(c) shows the speedup gained with respect to using the exact computation. The approximation method consistently achieves 10 times or more speedup. Also, it is clear that finer approximation result is less speedup. So Figure 3.9(d) plots the speedup/error to compare the trade-offs and identify which level of approximation is most suitable. Note that the values on the y axis are not normalized, so the plot is only useful in comparing the different level of approximation.

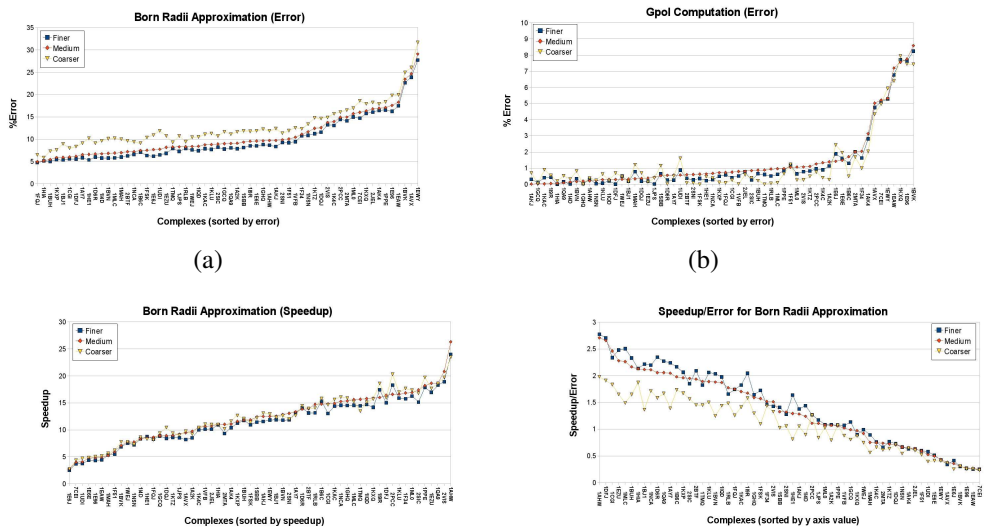


Figure 3.9: (a) Comparison of the approximation errors for Born Radii computation at various levels of approximation. Clearly, finer approximation scheme has consistently better accuracy than others and has an average error of 9.83 over the entire set of complexes. (b) Comparison of the approximation errors for G_{pol} computation at various levels of approximation. The average error over all complexes is only 1.43. Also, finer approximation scheme has consistently better accuracy than others. (c) Comparison of the speedup (with respect to the exact implementation) for Born Radii computation at various levels of approximation. On an average, DPG based approximation is more than 10 times faster. Also, notice that coarser approximation is consistently faster than finer ones. (d) Comparison of the speedup/error tradeoff (higher ratio is favorable) for Born Radii computation at various levels of approximation. Better trade-off is achieved for coarser models. But the actual choice should depend on the application.

Chapter 4

Dynamic Construction of Multi-resolution Smooth Surface Representation of Molecules

We report the Dynamic Adaptive Grid data structure [207] which adaptively subdivides space to provide higher resolution sampling near the boundary (surface) of a molecule with n atoms in $O(n \log n)$ time. It can support any surface approximation as long as it is expressed as a level set of a volumetric function. Our implementation includes the van der Waals surface, the solvent accessible surface (SAS), and the solvent excluded surface (SES) all of which requires the evaluation of an anlytical signed distance function at each gridpoint, and a faster Gaussian integration based surface approximation. This grid-based approach prodives simultaneous maintenance of molecules in atomic, smooth surface and volumetric representations, making it applicable for a wide range of applications. Also, experiments on a large set of proteins showed that our algorithm runs faster and requires less memory than regular uniform grid-based approaches. Finally, when an atom is moved (added or removed), the surface can be dynamically and locally updated by our algorithm in $O(\log n)$ time. Experiments showed that the constant hidden in the $O(\log n)$ update is not negligible, and dynamic updates are profitable if fewer than 10% of the atoms are moved in a step, otherwise reconstruction of the entire grid is faster. This indicates that, in many applications, for example in docking and struc-

ture refinement, where only a fraction of the atoms need to be updated frequently, the dynamic algorithm we report here is beneficial.

4.1 Introduction

Probably the simplest model for molecules is as a collection of atoms, each atom represented as hard spheres, with radii equal to their van der Waals radii. The boundary of this arrangement of spheres is a possible surface representation and is usually referred to as the van der Waals Surface (vdW). Lee and Richards [157] proposed the Solvent Accessible Surface (SAS) as the boundary of the region which is accessible for solvent molecules. It is modeled as the locus of the center of a water molecule, considered as a sphere with radius 1.4Å, as it rolled along the protein surface. While both of these models are sufficient for visualization purposes, they are not suitable for simulations and energetics computations that require numerical integrations over the surface, since it contains singularities at the sphere-sphere intersections. We need smooth surfaces for such applications.

Richards [213] proposed a model commonly known as the Solvent Contact Surface (SCS), or Solvent Excluded Surface (SES), as the boundary of the volume not penetrated by a solvent molecule. This surface is composed of convex patches where the probe touches the atom surfaces, concave spherical patches when the probe touches more than 2 atoms simultaneously and toroidal patches when the probe rolls between two atoms. An even smoother representation is to represent each atom using a Gaussian kernel whose mean is at the center of the atom [46, 50, 78, 110, 111]. A volumetric function is defined over the entire space as a summation

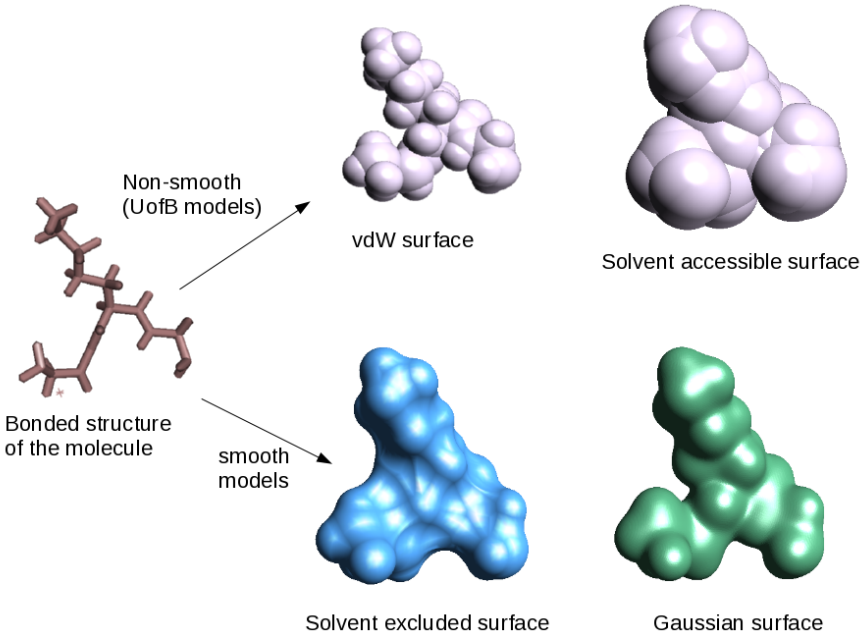


Figure 4.1: Different surface representations for the same collection of atoms. The vdW and SAS surfaces are non-smooth and represents each atom as a hard sphere with radius r and $r + r_p$ respectively, where r is the van der Waals radius of the atom, and r_p is the radius of the solvent (1.4Åin the case of water). The SES is a smooth surface, where the singuralities of the vdW surface are removed by using toroidal patches to cover pairwise intersections, and concave patches to cover 3-way or higher intersections of the vdW spheres. Finally the Gaussian surface is defined as a level set of a volumetric function defined as a sum of Gaussian kernels placed at the centers of the atoms.

of these Gaussian kernels, and a level set of this volumetric function is considered the molecular surface, usually called the Gaussian Molecular Surface (GS). Some variations on the Gaussian formulation were discussed in [85, 103, 280]. Figure 4.1 shows examples for each of the four representations we discussed here.

Since Richards introduced the SES definition, a number of techniques have been devised to compute the surface, for both static and dynamic cases, and using

both implicit and explicit representations. Connolly introduced two algorithms to compute the surface. First, a dot based numerical surface construction and second, an enumeration of the patches that make up the analytical surface (See [71]). In [264], the authors describe a distance function defined over a grid for computing surfaces of varying probe radii. Our data structure contains approaches similar to their idea. A number of algorithms were presented using the intersection information given by voronoi diagrams and the alpha shapes introduced by Edelsbrunner [87], including parallel algorithms in [261] and a triangulation scheme in [4]. SES based on arrangment of spheres for both static and dynamic cases were reported in [92, 93]. Another computation of SES is described in [221], using Reduced sets, which contains points where the probe is in contact with three atoms, and faces and edges connecting such points. Non Uniform Rational BSplines (NURBs) descriptions for the patches of the molecular surfaces are given in [18], [17] and [19]. You and Bashford in [279] defined a grid based algorithm to compute a set of volume elements which make up the Solvent Accessible Region. Another grid-based algorithm using 2D arrangment of circles was reported in [24].

Various modifications of the Gaussian kernels and their effects were analyzed in [85, 103, 280]. Recently, Zhang et al. considered representing the molecular surface as a level set of a high order polynomial and showed that the additional degrees of freedom available from such a model can be used to satisfy user-defined constrains and quality metrics for the molecular surface [20]. In this paper, we also report that the prevalent choice of Gaussian parameter blobbiness and isovalue to be 2.3 and 1.0 respectively does not always lead to surfaces close to the SES model.

We found that the choice of the parameters depend on the property of the molcule one is interested in. For example, to get a better estimate of the surface area, small blobbiness (1.0-1.2) with large isovalue (1.8-2.2) is required.

In many applications such as docking and pairwise alignment of molecules [23, 61, 65, 151], a volumetric/grid representation is more amenable for applying FFT-based fast computations. So, in this paper we first show that each of the four surface types can be expressed as level sets of volumetric functions and develop a data structure and algorithms to support and maintain atomic, smooth surface and volumetric representation of a molecule simultaneously. Additionally, our octree based scheme allows one to sample/construct the surface at multiple resolutions with provable approximation errors. Note that other existing fast approximation algorithms like [269–271] do not provide theoretical error bounds.

4.2 Molecular Surfaces as Level Sets

Given a set of atoms $M = \{a_i, \ldots, a_n\}$, where the center and radius of each atom is $\mathbf{c}_i$ and r_i, we define a volumetric function $\Phi^M(\mathbf{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$ and use its iso-contours $\Gamma(\Phi^M, v) = \{\mathbf{x} | \Phi^M(\mathbf{x}) = v\}$, where v is a scalar, to provide a family of molecular surfaces. In the following, we analyze three specific cases.

4.2.1 Level Set of Sum of Gaussian Kernels

Define $\Phi^M_{GS}(\mathbf{x})$ as follows-

$$\Phi^M_{GS}(\mathbf{x}) = \sum_i e^{-\beta(1-\frac{\|\mathbf{c}_i-\mathbf{x}\|^2}{r_i^2})} \tag{4.2.1}$$

where β is called the blobbiness parameter, which controls the width (or the decay rate) of the Gaussian kernel. Different choices of blobbiness and isovalues v would lead to different surfaces, not only in terms of the area/volume, but also in terms of topology. Traditionally, one chooses a β and v such that the resulting isosurface approaches the vdW or SES surface.

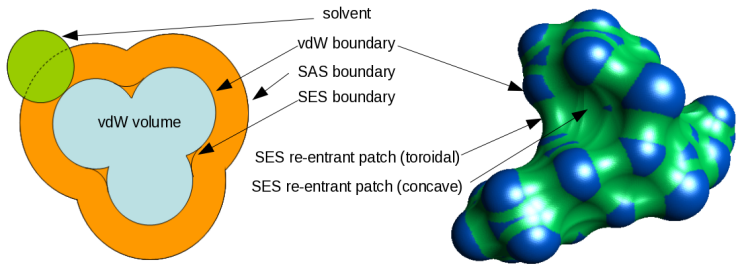


Figure 4.2: The different molecular surfaces and regions are shown for a 3 atom model in 2D. The SAS surface is the locus of the center of the rolling probe sphere. The VDW surface is the exposed union of spheres representing atoms with their van der Waals radii and contains the VDW volume. The lower side of the rolling probe defines the smooth SES which contains parts of the VDW surface and reentrant patches. We also define the SAS volume as the region between the SAS and SES. The region between the SAS and VDW volumes is later refered to as the SES volume.

4.2.2 vdW Surface as a Level Set

Define $\Phi^M_{vdW}(\mathbf{x})$ as follows-

$$\Phi^M_{vdW}(\mathbf{x}) = s^M_{vdW}(\mathbf{x})\delta^M_{vdW}(\mathbf{x}) \tag{4.2.2}$$

where $s^M_{vdW}(\mathbf{x})$ is a sign function indicating whether the point $\mathbf{x}$ is inside or outside the vdW sphere of any atom. In other words, $s^M_{vdW}(\mathbf{x}) = 1$ if $\exists_i \|\mathbf{c}_i - \mathbf{x}\|^2 \leq r_i^2$, and -1 otherwise. And $\delta^M_{vdW}(\mathbf{x})$ is the shortest distance from the point $\mathbf{x}$ to the vdW sphere of any atom, in other words, $\delta^M_{vdW}(\mathbf{x}) = min_i |\|\mathbf{c}_i - \mathbf{x}\| - r_i|$.

Functions of this form are called signed distance functions. Note that the $\Gamma(\Phi^M_{vdW}, 0)$ defined the vdW surface.

4.2.3 SAS and SES Surfaces as Level Sets

Since, the definitions of both SAS and SES depends on rolling a solvent molecule, perhaps it comes as no surprise that both can be defined as level sets of the same volumetric function defined as follows-

$$\Phi^M_{SAS}(\mathbf{x}) = s^M_{SAS}(\mathbf{x})\delta^M_{SAS}(\mathbf{x}) \tag{4.2.3}$$

where $s^M_{SAS}(\mathbf{x})$ is a sign function indicating whether the point $\mathbf{x}$ is inside or outside the solvent enlarged sphere of any atom. In other words, $s^M_{SAS}(\mathbf{x}) = 1$ if $\exists_i \|\mathbf{c}_i - \mathbf{x}\| \leq (r_i + r_p)$, and -1 otherwise. And $\delta^M_{SAS}(\mathbf{x})$ is the shortest distance from the point $\mathbf{x}$ to the SAS surface. Computing $\delta^M_{SAS}(\mathbf{x})$ is not trivial and is described in the next subsection.

- $\Gamma(\Phi^M_{SAS}, 0)$ defines the SAS surface.
- $\Gamma(\Phi^M_{SAS}, r_p)$ defines the SES surface.

See Figure 4.2 for 2D and 3D examples of the vdW, SAS and SES surfaces for an intuitive understanding of these particular signed distance functions and the chosen iso-values.

4.2.3.1 Computing Signed Distance Function $\Phi^M_{SAS}(\mathbf{x})$ from the SAS

Note that, we do not really have an analytical representation of the SAS, and hence cannot compute the distance $\delta^M_{SAS}(\mathbf{x})$ directly. We do however, have analytical representation of the SAS for individual atoms. Here we want to compute $\delta^M_{SAS}(\mathbf{x})$ using only local per atom definitions of the SAS.

Figure 4.3 provides an example that shows the complexity of computing $\delta^M_{SAS}(\mathbf{x})$ using local information only. In that 2D example, we see that the SAS boundary of individual atoms gets buried when two or more atoms intersect and hence do not contribute to the SAS boundary of the entire molecule. As such, if the closest point on the molecular SAS, from any point $\mathbf{x}$ may not be the closest point on the atomic SAS from the same point. We need to consider several cases. First we define some regions of the intersections of the spheres.

- **Cone of intersection** If the SAS boundaries of two atoms a_i and a_j intersect, then let C_{ij} define the circle of intersection of the spheres. Then we define two infinite cones with apexes at $\mathbf{c}_i$ and $\mathbf{c}_j$ and going through C_{ij}. The intersection of these cones are defined as CI_{ij}. A 2D anlog of this can be seen in 4.3 as the shaded quadrilateral. Note that for any point $\mathbf{x}$ inside CI_{ij}, the closest boundary point, not buried by a_i or a_j, must lie on C_{ij}.

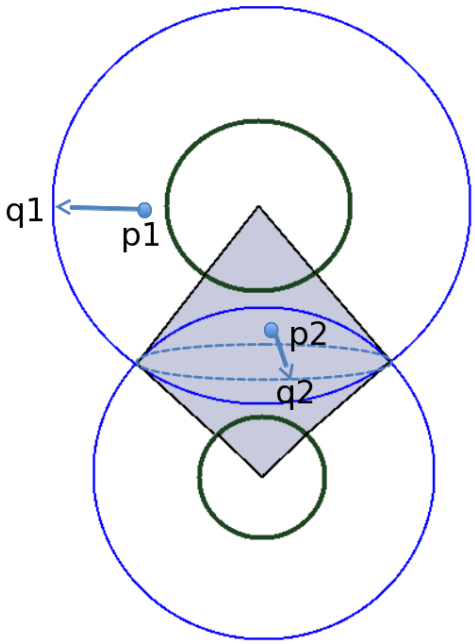


Figure 4.3: A 2D example with only 2 atoms showing that multiple cases should be considered when computing distance to the solvent accessible surface (SAS). The point p_1 lies inside the SAS of only one atom, and the its closest point on the SAS is simply the closest point on the atom. But in case when the point lies within the intersection region of 2 atoms (eg. p_2, the closest point on the SAS is no longer the closest point on either atom (since that closest point is not part of the SAS, but is actually buried inside the SAS). In this case, the closest point must be chosen from the intersection of the two atoms.

- **Tetrahedron of intersection** When 3 spheres a_i, a_j and a_k have a common intersection, then common region is like a trigon, three spherical patches with converging onto exactly 2 points of intersections $\mathbf{t}_{ijk1}$, and $\mathbf{t}_{ijk2}$. Then we define two tetrahedrons $T_{ijk1} = \mathbf{c}_i\mathbf{c}_j\mathbf{c}_k\mathbf{t}_{ijk1}$ and $T_{ijk2} = \mathbf{c}_i\mathbf{c}_j\mathbf{c}_k\mathbf{t}_{ijk2}$ as the tetrahedrons of intersection. For any point inside T_{ijk1}, the closest boundary point, not buried by a_i, a_j or a_k, is exactly $\mathbf{t}_{ijk1}$; and similarly for T_{ijk2}.

Let $exposed^M(\mathbf{x})$ is an indicator function which is true if $\mathbf{x}$ is not inside the SAS of any atom of M. Let $closest(\mathbb{P}, \mathbf{x})$ define the closest point $\mathbf{y}$ belonging to the set of the point $\mathbb{P}$ from the given point $\mathbf{x}$; and let $d(\mathbb{P}, \mathbf{x})$ be the Euclidean distance between them if $\mathbf{y}$ is exposed, otherwise it is Rr_p, where $R > 1$ is some constant.

Now, for each point $\mathbf{x}$, let $N^A(\mathbf{x})$ be a set of atoms such that $\mathbf{x}$ is inside the SAS of the atoms; $N^{CI}(\mathbf{x})$ be a set of cone intersections that contain $\mathbf{x}$; and $N^T(\mathbf{x})$ be a set of tetrahedral intersections that contain $\mathbf{x}$. Let n^A, n^{CI} and n^T be the cardinalities of these sets. Finally, we define d^A, d^{CI} and d^T as follows-

- $d^A = min_{a_i \in N^A} d(a_i, \mathbf{x})$
- $d^{CI} = min_{CI_{ij} \in N^{CI}} d(C_{ij}, \mathbf{x})$
- $d^T = min_{T_{ijkl} \in N^T} d(\mathbf{t}_{ijkl}, \mathbf{x})$

We are now ready to define the distance function.

- If $\mathbf{x}$ is exposed, then $\delta^M_{SAS}(\mathbf{x}) = min_i | \|\mathbf{c}_i - \mathbf{x}\| - r_i - r_p|$
- If $\mathbf{x}$ is not exposed, then $\delta^M_{SAS}(\mathbf{x}) = min(d^A, d^{CI}, d^T)$

Note that efficient computation of this function requires efficient computation of $exposed^M(\mathbf{x})$, $N^A(\mathbf{x})$, $N^{CI}(\mathbf{x})$ and $N^T(\mathbf{x})$. We use the Dynamic Packing Grid [14, 24] data structure to compute these in quasi-constant time. Details of the data structure is given in Chapter 4.4. In Section 4.4, we shall discuss its augmentation for the purposes of efficient computation of SES.

4.3 Dynamic adaptive grids for molecular surface computation

Given a molecule M, our algorithm constructs an adaptive octree T. The octree is subdivided at a higher resolution near the boundary of the molecule and is coarsely subdivided elsewhere. The dual of the highest resolution cells of the octree is contoured based on a user-defined isovalue to generate a smooth surface approximation of M. We define the resolution of the mesh as the resolution of the dual grid, which is equal to the resolution of the smallest cells (d_{min}) of the octree. The construction runs in $\mathcal{O}(n \log n)$ time provided that d_{min} is linearly related to the maximum radius of the atoms. Refer to Sections 4.3.1 and 4.3.2 for details about the model and the construction.

We define an update as adding, removing or moving an atom a_i. In Section 4.3.4 we discuss a $\mathcal{O}(\log n)$ time algorithm for locally updating the octree while maintaining the adaptive subvision property.

4.3.1 The octree and its dual

We define an octree node u as a 3-dimensional cube with a specific length, a center and a function value. The length $\text{LENGTH}(u)$ is the Euclidean length of an edge of the cube, $\text{BOX}(u)$, representing the node. The center $\text{CENTER}(u)$ is the geometric center of the cube. The function value $\text{VAL}(u)$ is defined as a volumetric function $\Phi^M(\text{CENTER}(u)))$, where M is the set of atoms. We define the sign of a node $\text{SIGN}(u) = 1$ if $\text{VAL}(u) - isoValue$ is positive and -1 otherwise. Refer to Section 4.2 for possible choices of $\mathcal{F}$ and corresponding iso-values.

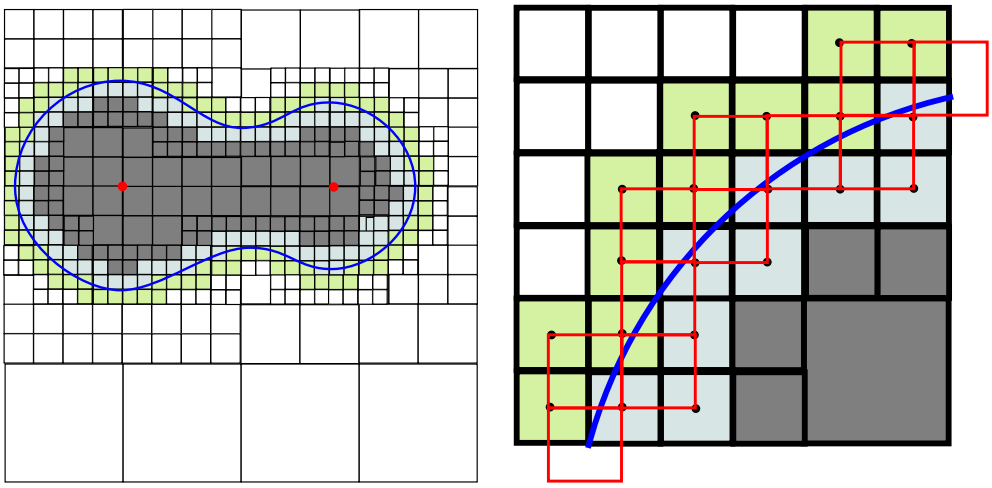
By $\text{CHILD}(u)$ we denote the set of non-empty octree nodes obtained by sub-

dividing node u, and PARENT(u) points to the parent of u in the octree. CHILD(u) = NIL if the node is a leaf of the octree and PARENT(u) = NIL if u is the root. For convenience, we set LEAF(u) to TRUE if u is a leaf, and FALSE otherwise. $\mathcal{N}(u)$ denotes the neighbor cells u. DEPTH(u) denotes the depth of u. Each leaf u of the octree stores a list of atoms ATOM(u) which intersect its bounding box.

The octree T contains a reference to the root node of the tree, a list of atoms M, a positive number d_{min} specifying the minimum acceptable length of any node of the octree, and a number $isoValue$ corresponding to the level-set of $\mathcal{F}$ representing the molecular surface.

Given M, d_{min} and $isoValue$, the octree is constructed by iteratively refining (splitting) nodes which contain the contour corresponding to the specified $isoValue$. Other parts of the octree remain coarse. The refinement continues until $d_{min} \leq$ LENGTH$(v) \leq 2 * d_{min}$ for every node v containing the contour. Let BOUNDARY(T) define the set of all such nodes. We provide a simple 2D example in Figure 4.4(a).

We construct a dual grid G based on the nodes in BOUNDARY(T). The length of each cell in the dual grid is equal to LENGTH(v) where $v \in$ BOUNDARY(T). The vertices/grid-points of the dual grid are made congruent to the center of the nodes in BOUNDARY(T). See Figure 4.4(b) for an example dual grid superimposed on an octree. Each grid-point gp of G is assinged a value equal to VAL(w), where $gp =$ CENTER(w). Finally, we use fast marching cubes to produce a mesh for the isocontour in each of the gridcells of G.



(a) (b)

Figure 4.4: **(a)** A 2D example of adaptive decomposition. Cells are adaptively subdivided depending on their distance from the boundary. In the figure, we color the a cell u- (i)light green if $u \in \text{BOUNDARY}(T)$ and $\text{SIGN}(u) = 1$, (ii) light blue if $u \in \text{BOUNDARY}(T)$ and $\text{SIGN}(u) = -1$, (iii) dark gray if $u \notin \text{BOUNDARY}(T)$ and $\text{SIGN}(u) = 1$, and (iv) white if $u \notin \text{BOUNDARY}(T)$ and $\text{SIGN}(u) = -1$. Atom centers are shown in red and the isocontour representing the molecular surface is shown in blue. (b) A zoomed in view of a part of the octree with the dual grid superimposed on it. Note that, the dual grid is very sparse, and yet sufficient to compute the contour.

Theorem 4.3.1. *The dual of a dynamic octree containing a molecule with n atoms has $\theta(k)$ cells provided that $d_{min} = \Theta(r_{max})$ where r_{max} is the radius of the largest atom in M and $k < n$ is the number of atoms on the boundary of the molecule.*

Proof. Since $d_{min} = \Theta(r_{max})$, each atom intersects a constant number of octree leaves at the lowest level (see Theorem 2.1 and Lemma 2.1 in [14]). So, if k atoms contribute to the molecular surface, the number of such leaves can be at most Ck, where C is a constant. Hence, the number of dual cells is $\theta(k)$. ■

4.3.2 Construction

Given a molecule M, a positive number d_{min} specifying the expected resolution of the dual grid and the isovalue for the surface, we initialize an Octree T with a single node u (root of T), such that $\text{BOX}(u)$ is large enough to contain all of the atoms. All atoms are inserted into u and $\text{VAL}(u)$ is computed.

The octree is refined by iteratively subdividing cells which are on the boundary of the molecule. Since we define the boundary as a isocontour $\Gamma(\Phi^M, isoValue)$, the boundary intersects an edge (v_i, v_j) of dual cell gc if and only if $\text{SIGN}(v_i)$ and $\text{SIGN}(v_j)$ are different, where v_i and v_j are neighboring octree cells. We consider all such (v_i, v_j) pairs as being on the boundary and mark them for subdividing.

To split a node u, we create its children $u_1, \ldots, u_8$. Then, we update the list of atoms in the children as $\text{ATOMS}(v_i) = \{a | a \in \text{ATOMS}(u) \& (\text{BOX}(v_i) \cap a) \neq \phi$. The intersection detection is performed using the Dynamic Packing Grids data structure in constant time per atom. The number of times an atom is checked for intersection is bounded by the maximum depth of the octree. After assigning the atoms, we compute $\text{VAL}(v_i)$.

Signs of each of the newly created nodes are compared with their neighbors to identify pairs of nodes on the boundary and marked for splitting. The process continues until for each leaf cell v, $d_{min} \geq \text{LENGTH}(v)$.

Special note on initialization Let u and v be two neighboring cells of the octree such that $\text{SIGN}(u) \neq \text{SIGN}(v)$, then the isocontour crosses the line connecting

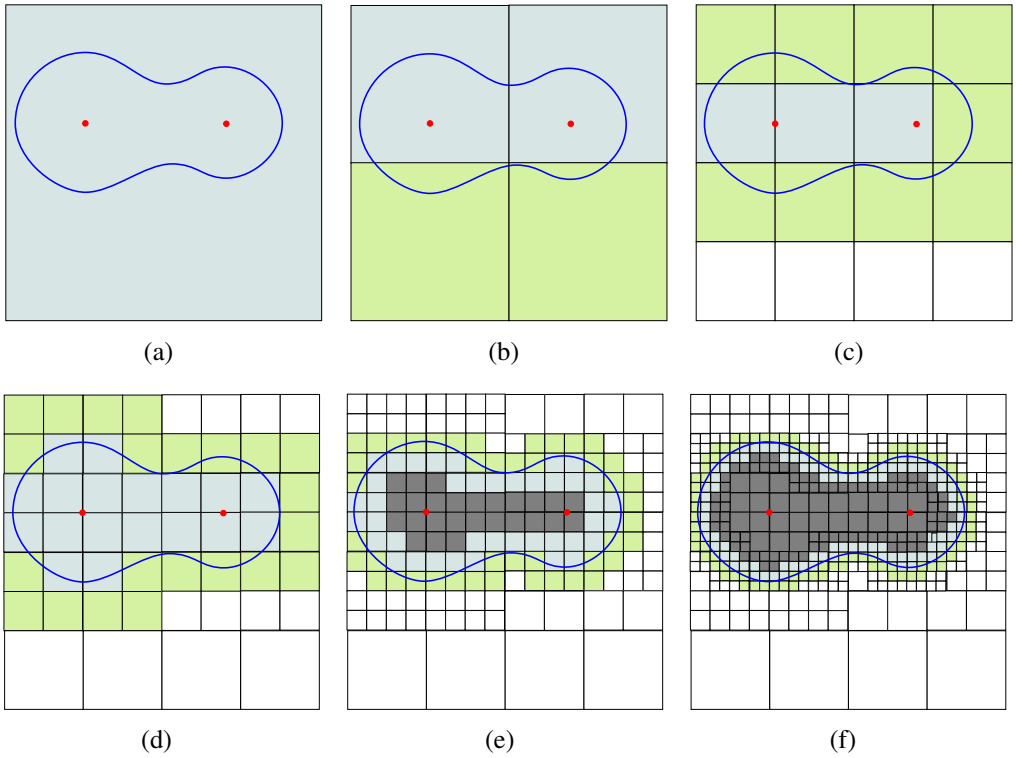


Figure 4.5: Figure (a) shows the initial octree, which is iteratively refined and the results of each successive iterations are shown in Figues (b)-(f). We follow the same coloring convention used in Figure 4.4. Note that only the cells belonging to BOUNDARY(T) are split in the next iteration.

CENTER(u), CENTER(v) an odd number of times (and at least once). Otherwise, the isocontour crosses the line an even number of times (or never). So, if we define the minimum topological feature size $\epsilon_{\Phi^M,isoValue}$ (eg. width of a pocket/tunnel) of the molecule M under the given function and isoValue, then no grid-based algorithm whose finest level grid-size $d_{min} > \epsilon_{\Phi^M,isoValue}$ can produce a topologically accurate surface. So, we assume that $epsilon_{\Phi^M,isoValue}$ is known or the user specifies a specific value ϵ such that s/he is willing to tolerate errors below that threshold.

We incorporate this into the algorithm by ensuring that the octree is uniformly subdivided until every leaf has length at most ϵ. Let the minimum depth required to achieve this is l_{min}. Note that after this point, we continue to subdivide adaptively near the boundary until the lowest level leaves $d_{min} \geq \text{LENGTH}(v)$.

Contouring Once the octree is constructed, the surface is computed by dual contouring (see previous section), based on the cells corresponding to any level (greater than l_{min}) of the octree, with progressively finer (deeper) level nodes providing progressively better approximation of the surface.

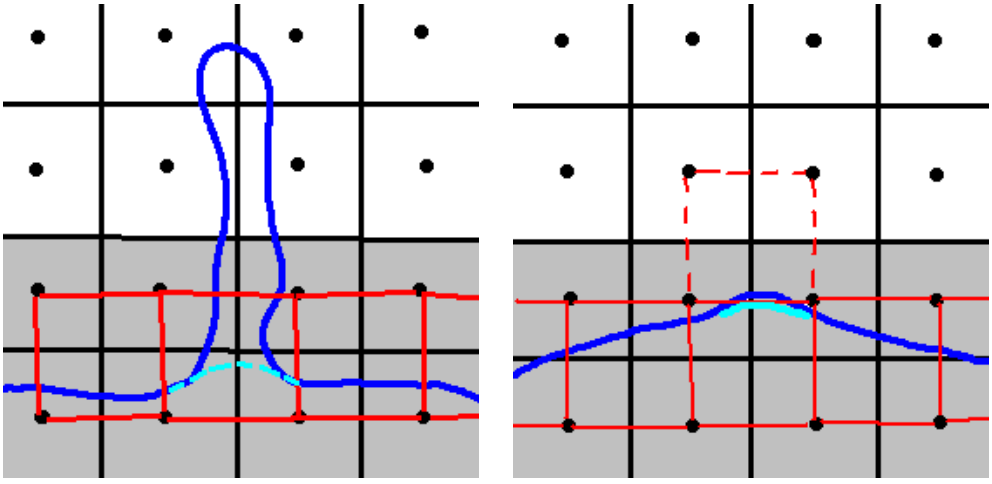
Figure 4.5 provides a detailed example of the octree construction algorithm.

4.3.3 Analysis

Our grid-based algorithm is a discrete approximation of a continuous domain (the isocontour). Hence it does not generate a perfect representation as analytical methods would produce. However, we can provide theoretical bounds on the topological and geometric errors.

Theorem 4.3.2. *Dual contouring is performed using gridcells at any level $l > l_{min}$ of the octree produces a isocontour with bounded geometric error if $\epsilon_{\Phi^M, isoValue} > d_{min}$.*

Proof. There can be three types of error. The severest error is like the one shown in Figure 4.6(a), where very small features lie on in between grid centers and hence discrete evaluation of Φ^M at the grid-centers fail to detect such features. In molec-



(a) (b)

Figure 4.6: **(a)** Topological error. If the minimum dimension of such a topological feature (like the tunnel in the figure) is less than ϵ, then it is lost during the contouring. **(a)** Geometric error. The approximate isocontour (light blue) is only slightly off the actual isocontour (dark blue). The error is bounded by the cell size.

ular applications, however, typically the feature sizes are bounded by the smallest radius. There can sometimes be small tunnels and pockets whose width is less than the smallest radii, but for most purposes such tunnels are inaccessible to other molecules including water, and hence loss of such features will not affect the molecular properties calculation too much. The second type of error can that happen is when the isocontour goes through a dual cell which is not identified, but its neighbor is identified (see Figure 4.6(b)). In this case, the maximum distance between the actual and the approximate contour is at most d_{min}. Since, we assumed that the smallest topological feature size $\epsilon_{\Phi^M,isoValue} > d_{min}$, this error is only a geometric error. Finally, the last type of error is simply due to the limitation of contouring where the correct dual cells are contoured, but the approximated contour inside a

dual cell is slightly off the real isocontour within the same cell. Here, the maximum error, in terms of Hausdorff distance, can be at most $d_{min}/2$. The proof follows. ■

Theorem 4.3.3. *A dynamic octree containing a molecule with n atoms is constructed in $\mathcal{O}(n \lg n)$ time if $d_{min} = \Theta(r_{max})$.*

Proof. Since, $d_{min} = \Theta(r_{max})$, following Theorem 2.1 and Lemma 2.1 in [14], the number of cells in the lowest level (with length d_{min}) is $\mathcal{O}(n)$. By the same logic, the number of nodes at any level above that is also bounded by $\mathcal{O}(n)$. Finally, since the maximum dimension of the bounding box is at most nr_{max} assuming that the molecule M is compact, the depth of the octree is $\log \frac{nr_{max}}{d_{min}} = \log n$.

At each node, the decisions to split/merge takes constant time, computing the function value requires summing over the atoms intersecting the cell and may require $BigOh(n^3)$ time per node at worst. This is addressed in section 4.4 to make sure that only a constant number of atom is required to compute the value at a cell. Finally, during split, we must go over the list of atoms and copy them to new nodes, indicating a worst case $\mathcal{O}(n)$ time per node. However, note that evergy atom will be copied/moved at most $\log n$ time during the entire construction, hence the cost is amotized to $\mathcal{O}(1)$ per node.

Hence the overall amostized cost of construction is $\mathcal{O}(n \lg n)$. ■

4.3.4 Update

When an atom is updated from a to a' by either changing the position or the radius, we must update the function values $\text{VAL}(v)|v \in T\&(\text{BOX}(v) \cap (a \cup a')) \neq \phi$. For each updated cell v, we verify whether it should be split or merged. An internal node v is marked for merging if all of its children have the same sign and none of them are candidates for splitting. Once a cell is marked for merging, it cannot be split again for the same update operation. The decision to split is made based on the same criteria described in Section 4.3.2. We iteratively split cells and identify new cells for splitting/mergning until no more cells are marked for splitting. Then we merge the cells marked for such.

Note that, if the update of the atom does not change the boundary of the molecule, then our algorithm would only update the function values, but would not split/merge any cells.

Theorem 4.3.4. *After completion of* UPDATEOCTREE(a, a')*, if dual contouring is performed using gridcells at any level $l > l_{min}$ of the octree produces a isocontour with bounded geometric error if $\epsilon_{\Phi^M, isoValue} > d_{min}$.*

Proof. The accuracy of the merge operation (and decision) follows from the accuracy of split operation (and decision) discussed in Theorem 4.3.2, since the merger is performed if only if the merged cell would not have been marked for splitting if we had started constructing the octree with the moved atom in the new position.

■

Theorem 4.3.5. UPDATEOCTREE(a, a') *completes in* $\mathcal{O}(\lg n)$ *time for a dynamic octree containing a molecule with* n *atoms.*

Proof. A single atom intersects at most a constant number of leaf cells. Hence, the total number of nodes affected by the move at any level of the octree is bounded by the depth $\mathcal{O}(\log n)$. The proof follows, since the algorithm spends $\mathcal{O}(1)$ time per node (see proof of Theorem 4.3.3). ■

4.4 Augmented Dynamic Packing Grids for SES Molecular Surface Computation

We describe a data structure and algorithm which supports $O(1)$ time updates (add/remove/move an atoms) such that the molecular surface after the update is consistent. As a by-product, the algorithm also updates the list of exposed atoms and boundary cells (SAS).

The algorithm is an augmentation of the octree based surface construction algorithm for the purpose of computing the Solvent Excluded Surface. As we have dicussed in Section 4.2, SES computation requires evaluation of the signed distance function and the computation is slightly more involved that than the sum of Gaussian kernels. The octree always uses all atoms inside a cell to define the function values in the cell, but this becomes extremely costly when $\theta(n)$ atoms are in a cell, during the initial stages of the construction and leads to $O(n^3)$ computations for the double and triple intersections. Here, we propose to use a mixed model, where a simple uniform grid G with a constant grid spacing of $r_p/2$ is created initially,

but the SDF is not computed on the grid directly. Rather, we also create separate Dynamic Packing Grids (DPG) with a constant grid spacing of $2r_{max}$ and use the DPGs to label gridpoints and gridcells of G. We show that the labeling is sufficient to identify all the gridcells where the SES boundary lies, and also the atoms which contribute to the SES boundary. Once this classification is achieved, the regular octree based refinement can be performed only at the boundary cells and the number of atoms that contributes to the boudary and intersects a given cell is also reduced to a constant, and hence the overall construction time reduces from $O(n^3)$ to $O(n \log n)$, without any loss of accuracy. Furthermore, we show that the labeling can be updated under dynamic motions of the atoms in constant time per atom.

4.4.1 Notations

Let the molecule M is represented as a collection of atoms a_i and each atom is represented using a center $\vec{c}_i$ and radius r_i. Let the radius of the probe be r_p. Let the dynamic adaptive grid be defined as a set of gridpoints and a set of gridcells. We also assume that each grid cell contains exactly 8 gridpoints.

A grid point gp is marked as V_{VDW} if it is inside the vdW surface of at least one atom, is marked V_{SAS} if it is not inside the vdW surface of any atoms, but inside SAS surface of at least one atom, and is marked V_{OUT} if it is not inside the SAS of any atoms. We mark a gridcell gc as $C_{BURRIED}$ iff $\forall gp_j \in gc, (gp_j \in V_{VDW})$, as C_{SAS} iff $\exists gp_j \in gc, (gp_j \in V_{VDW}) \wedge \exists gp_k \in gc, (gp_k \in V_{SAS})$, as C_{BAND} iff $\forall gp_j \in gc, (gp_j \in (V_{SAS} - V_{VDW}))$, C_{VDW} iff $\exists gp_j \in gc, (gp_j \in (V_{SAS} - V_{VDW}))$ $\wedge \exists gp_k \in gc, (gp_k \notin V_{SAS})$, and C_{OUT} iff $\forall gp_j \in gc, (gp_j \notin V_{SAS})$. See Figure 4.7

for a 2D example.

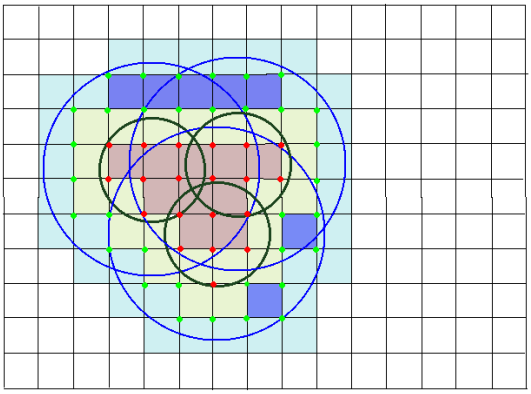


Figure 4.7: A 2D example grid with uniform grid spacing showing how the grid-points and the gridcells are labeled based on thier positions with respect to the vdW and SAS surfaces. SAS boundary of the atoms are shown as blue lines, vdW boundary of atoms are shown as dark green lines. Gridpoints labeled vdW, SAS and OUT are marked red, green and unmarked in the figure. Gridcells labeled $C_{BURRIED}$, C_{VDW}, C_{BAND}, C_{SAS} and C_{OUT} are shaded brown, light green, dark blue, light blue and white.

If the SAS boundary of an atom a_i contains at least one gridpoint marked as V_{SAS}, then a_i is marked as an exposed atom (see Lemma 4.4.1 for details).

4.4.2 Algorithm Sketch

During an update, the algorithm uses DPG to identify and re-classify the affected grid-points (as V_{VDW}/V_{SAS}/V_{OUT}) and then uses these classification to mark atoms and gridcells. The algorithm maintains 4 separate packing grids, namely, (i) for all atoms, (ii) for exposed atoms, (iii) for grid-points marked as V_{SAS}, and (iv) for gridcells marked as C_{SAS}.

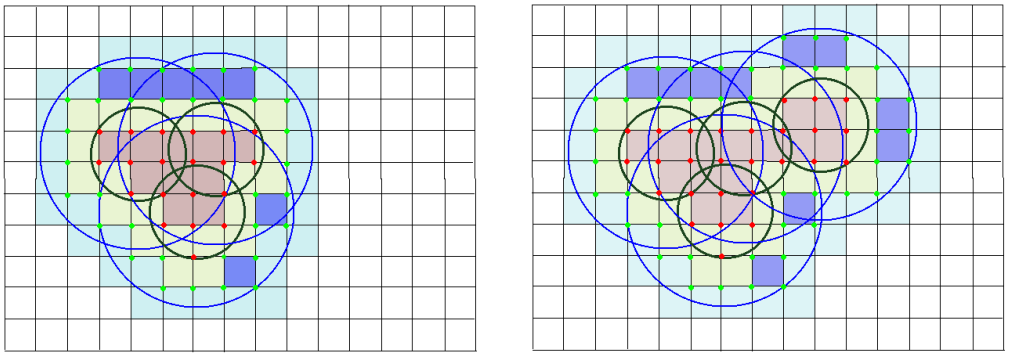


Figure 4.8: Updating the labeling of the grid points and gridcells when an atom is added. (left) before adding, (right) after adding.

Add When an atom is added (cf. Figure 4.8), it can have the following effects-

- Some grid-points might be marked as V_{SAS}. This is reflected in steps 2-3 of the pseudocode for the Add method.
- Some grid-points might be marked as V_{VDW}. This is reflected in steps 4-5 of the pseudocode for the Add method.
- The new atom will be marked either buried/exposed. The new atom is exposed if it marks a gridpoint as V_{SAS} (i.e. it contributes to the surface). This is handled by step $3(a)iii$ of the pseudocode.
- Some exposed atoms might get buried. Only exposed atoms intersecting the current atom are affected. If any of those exposed atoms no longer contributes to the surface (does not contain any gridpoint marked V_{SAS} within its solvent enlarged volume), then it is marked as buried (see step 6 of the algorithm for Add).

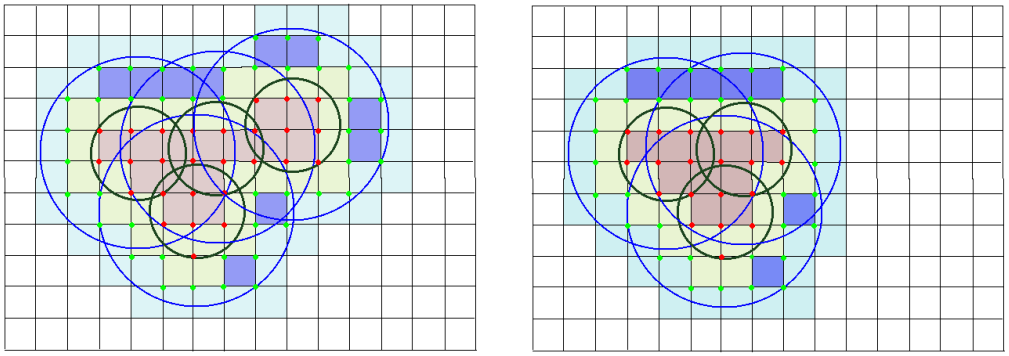


Figure 4.9: Updating the labeling of the grid points and gridcells when an atom is added. (left) before removing, (right) after removing.

Remove When an atom a_i is removed (cf. Figure 4.9), it can have the following effects-

- Some gridpoints marked as V_{SAS} can become V_{OUT}. (see step 4a of $Remove$).
- Some gridpoints marked as V_{VDW} can become V_{SAS}. And a previously buried atom which contains the gridcell within its solvent inflated volume would become exposed. Step 4b of the algorithm handles this case.

Updating the surface Any change in the classification of grid-points immediately affects the classification of the eight cells neighboring that point. Once a cell's classification is updated the contour/mesh inside the cell need to be recomputed. We use three auxiliary lists C_{rem}, C_{add} and C_{mod} to facilitate the management of the update. C_{rem} is the list of cells whose classification changed from C_{SAS} to something else. C_{add} are the cells whose classification changed from something else to C_{SAS}. C_{mod} contains cells whose classification remain C_{SAS}, but the surface inside the

cell needs to be modified (re-contour) to account for the recent movement of atoms.

4.4.3 Analysis

Correctness

Lemma 4.4.1. *In a grid with a grid spacing of $r_p/2$, an atom a_i is buried if $S_{SAS}(a_i)$ does not contain any grid-point marked as V_{SAS}.*

Proof. First of all, if the grid spacing is $r_p/2$, then it is easy to show that if one corner of a grid-cell is marked as V_{VDW}, then no part of the cell is outside the S_{SAS}.

$S_{SAS}(a_i)$ can intersect a cell in two different ways. Either the intersection contains at least one corner of the cell or it does not contain any corners.

If the intersection contains a corner, then that corner must be marked as V_{VDW}. And since all of these cells are completely inside the S_{SAS} (before adding a_i), a_i cannot have exposed surface inside these cells.

On the other hand if $S_{SAS}(a_i)$ intersects only a face f of a cell G_k and does not contain any corner, then clearly it has to contain at least one corner g of the neighboring cell G_l on the opposite side. According to the assumption of the lemma, g is marked V_{VDW} and whichever atom a_j has g inside its VDW surface, has the entire face f buried inside its SAS and hence the portion of a_i inside G_k is also completely buried ubsude the SAS of inside a_j.

Hence, a_i is not exposed inside any cell it intersects. In other words, it is buried. ■

Lemma 4.4.2. ***Add(***a_i, c_i, r_i***) and Remove(***a_i, c_i, r_i***) correctly updates the list of solvent exposed atoms.***

Proof. An atom a_j is marked buried only if $S_{SAS}(a_j)$ contains no gridpoints marked as V_{SAS} (see step 5(c) of Add method). And an atom a_j is marked as exposed as soon as $S_{SAS}(a_j)$ has at least one gridpoint marked as V_{SAS} (see step 2(a)i of Add and step 3(b)i of $Remove$). The correctness follows from Lemma 4.4.1. ■

Lemma 4.4.3. ***UpdateCells(***g***) correctly updates the lists*** C_{rem}***,*** C_{add} ***and*** C_{mod}***.***

Proof. Trivially follows from the definitions and the algorithm. ■

Time Complexity

Lemma 4.4.4. ***UpdateCells(***g***) operate in constant time.***

Proof. There are exactly 8 cells neighboring g and to classify each cell, the classification of exactly 8 gridpoints need to be checked. So, the total number of operations is constant. ■

Lemma 4.4.5. ***Add(***a_i, c_i, r_i***) and Remove(***a_i, c_i, r_i***) operate in constant time.***

Proof. Each range query of DPG runs in expected time $O(lglgh + k)$ where h is the word size in the memory, and k is the number of results returned (see Theorem 2.3 in [14]). h is clearly constant. The value of k depend on the type of the query. We discuss each of them separately.

A query in $DPG_{expatom}$ or DPG_{atom} with a distance cutoff of $O(r_{max})$ always returns $O(1)$ results (see Theorem 2.1 in [14]).

A query in DPG_{grid} or $DPG_{sasgrid}$ would return $O(\frac{r_max+r_p}{gridspacing})^3$. Provided that the grid spacing is $O(r_p)$ and $r_p \approx r_{max}$, the number of results returned becomes $O(1)$.

Hence all the queries runs in constant time and also returns constant number of results. Which means all internal loops run in constant time as well and hence the total complexity remains constant.

The rest follows from Lemma 4.4.4. ■

4.5 Results

4.5.1 Construction Accuracy

We have previously provided theoretical proofs that only the cells marked as boundary cells need to be contoured to get an airtight surface. In Figure 4.10, we provide practical examples showing that the SES is indeed correctly recovered.

The generated smooth solvent excluded surfaces using MSMS [222] for the molecules in Zlab's non-redundant protein benchmark. For the same dataset, we also applied our dynamic octree algorithm to compute approximate SES. Since MSMS computes an analytical representation of the SES, it is considered the ground truth in this context, and we report the error in terms of surface area, volume and Hausdorff distance between surfaces produced by MSMS and our algorithm (see Figure 4.11). We found that the average errors for area and volumes are 2.7% and

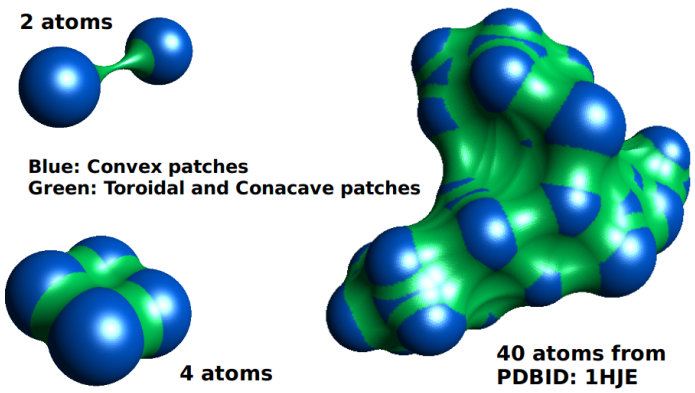


Figure 4.10: 3 separate examples with 2, 4 and 40 atoms showing the results of SES computation using the DAG. Notice that our level set of signed distnace function formulation correctly produced the re-entrant patches (green) of the SES.

1.53% respectively. The average distance between the surfaces are 0.255Å.

4.5.2 Construction Time

We compared the time required to compute volume representation and iso-value based surface construction using regular uniform grids, and octrees such that the grid-size of the lowest level of the octree matched the grid-size of the regular grid. Note that, the accuracy/quality of the surfaces produced from both would be similar, but the runtime is expected to be different. The regular grid would have to compute the function values at more positions, on the other hand an irregular data structure like an octree incurs some maintenance overhead. In our experiment (see Figures 4.12 and 4.13) on the dataset mentioned above, we found that the dynamic octree requires less time and less memory than the regular grids.

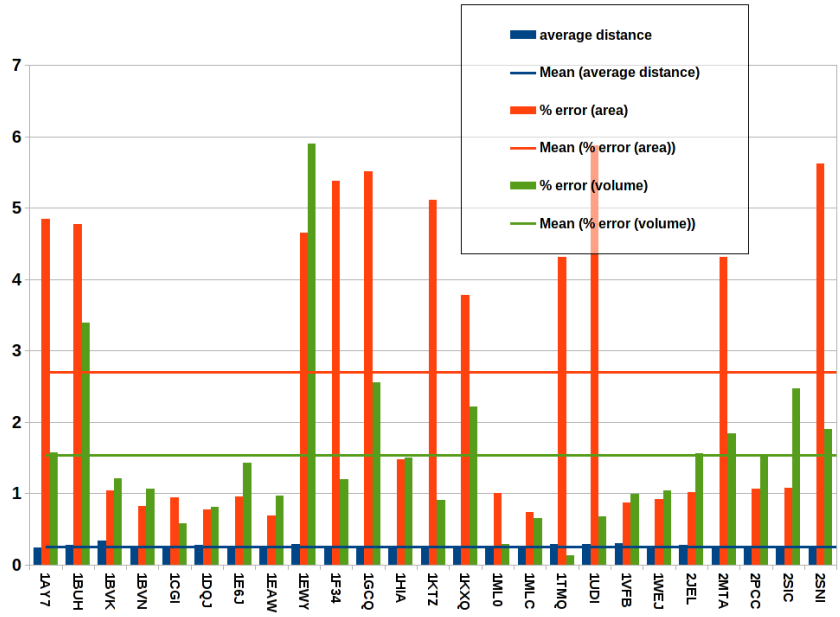


Figure 4.11: The plot compares the surface area, volume and average distance of the solvent excluded surfaces (SES) for various proteins generated by our DAG data structure to those generated by analytically modeling the SES as a collection of spherical and toroidal patches.

4.5.3 Updates

Theoretically, each update operation runs in $O(\log n)$ time and the overall construction from the scratch runs in $O(n \log n)$ time (see Theorems 4.3.3 and 4.3.5). This would indicate that it is always preferable or at least equivalent to construct the octree using a sequence of updates. However, the runtime for update has a larger constant which makes it slightly slower. For a large set of proteins, we moved different number of atoms and applied the update operation to recompute/update the surface to identify a threshold where reconstruction from scratch becomes more attractive than locally updating the surface. We found that even when upto 6% of the

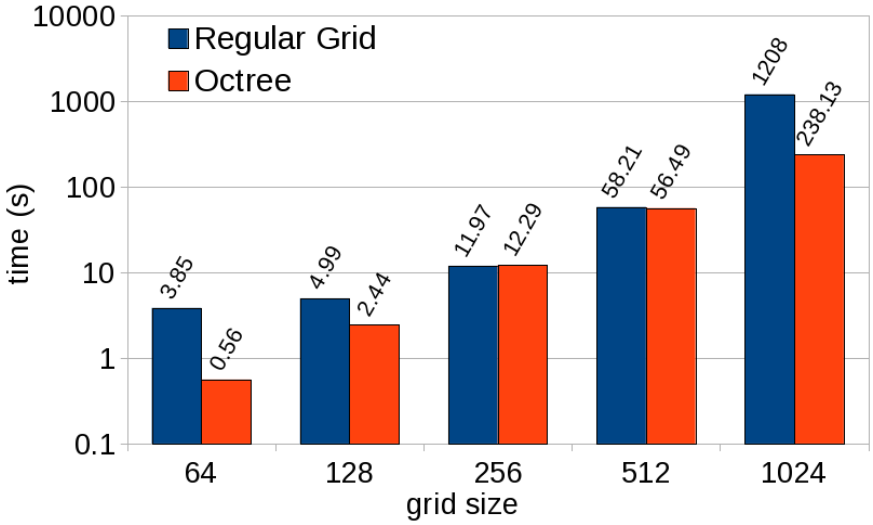


Figure 4.12: Comparison of the construction time using regular grids and adaptive grids. The numbers on the x-axis represent the resolution of the grid in terms of the number of gridpoints on each direction. The octree based construction clearly outperforms regular grid based construction. Note that this plot shows the runtimes for Gaussian surfaces.

atoms are moved, dynamic update provides 4 times faster performance than reconstruction. In fact, reconstruction only becomes a better choice if more than 10% of the atoms are moved (see Figure 4.14.

4.5.4 Choice of blobbiness and isovalue for Gaussian surfaces

Gaussian representation of the molecular surface is attractive for several reasons including the higher degree of continuity, simple computation (as opposed to computing arrangment of spheres for SES), and an intuitive relationship of the Gaussian blurring model to electron scattering of molecules, making it a better choice for energetics and force calculations. However, for shaphe based analysis

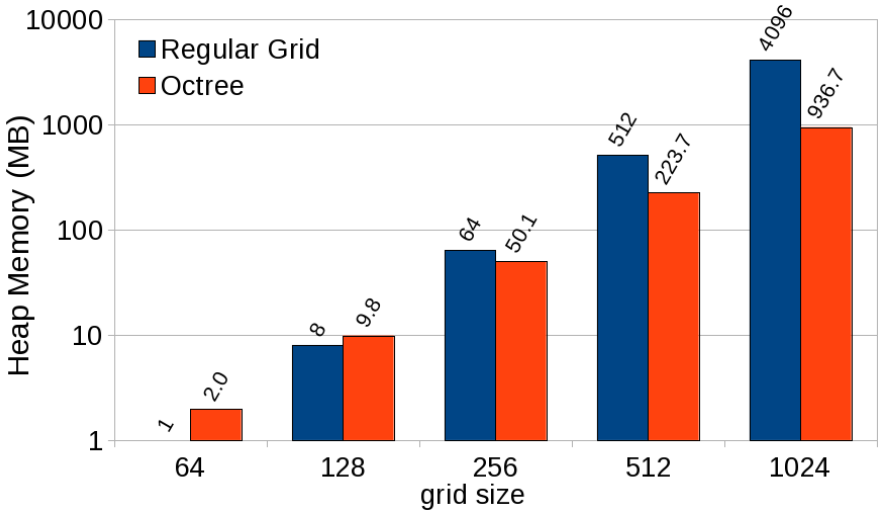


Figure 4.13: Comparison of the memory requirement of regular grids and adaptive grids.

or comparison of molecules, an SES approximation seems more favorable. We postulate that for such applications, it is possible to choose appropriate values for the Gaussian paramters to make the isosurface as close to the SES as possible. Similar ideas and analysis were also reported in [284] and [20]. In [284], in particular it was reported that a blobbiness (β) of 0.9 was a good choice for shape complementarity analysis. In this paper, we compared the Gaussian surfaces at different blobbiness and isovalues to their SES counterparts, in terms of surface area, volume, average distance and Hausdorff distance. Note that, given two surfaces A and B represented as a collection of points, we define the average distance as $\frac{1}{|A|}\sum_{p\in A} min_{q\in B} dist(p,q)$ and the Hausdorff distance as $max_{p\in A} min_{q\in B} dist(p,q)$.

We found that low blobbiness is better for area, and Hausdorff distances. Average distance is also low for low blobbiness, if a large enough isovalue is se-

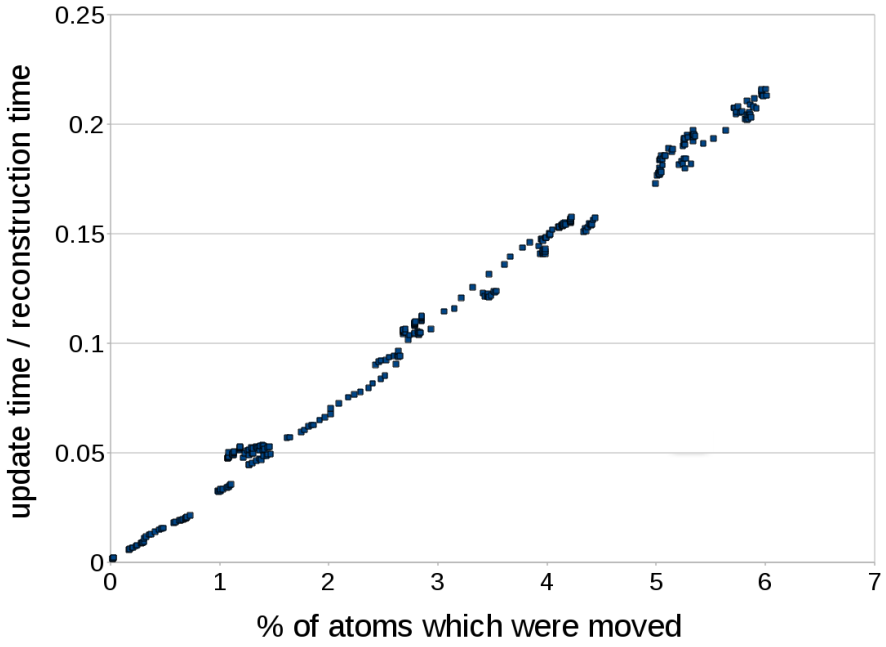


Figure 4.14: The plot compares the time required to dynamically update the surface when a fraction of the total atoms are moved, vs, the time required to reconstruct the entire surface.

lected. For instance, a blobbiness of 1 and isovalue of 2.0 provides low errors in all three terms. However, to minimize volume error, an even larger isovalue might be chosen. Interestingly, there are other choices which provide similar error bounds for volume and average distace. But we believe the area and Hausdorff distance provides a more reliable estimate of the closeness of the surfaces.

Algorithm Details

Algorithms for Constructing Octree

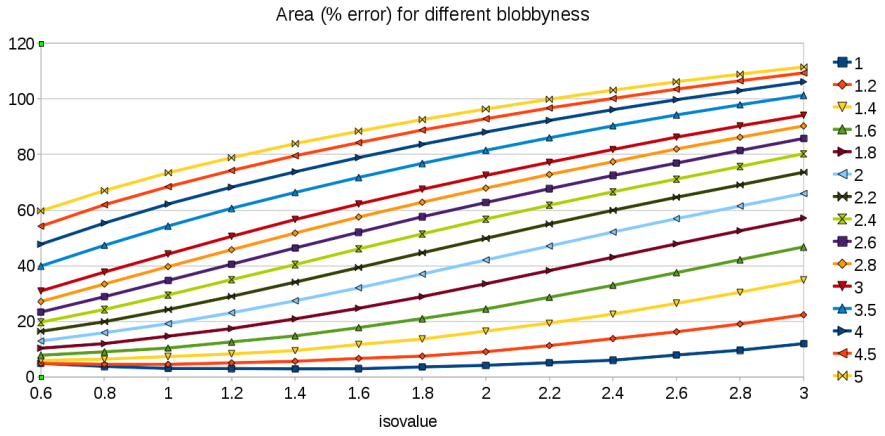


Figure 4.15: Difference, in terms of area, between Gaussian and SES surfaces for different choices of blobbiness and isovalues. The different lines represent different choices of blobbiness, then for each blobbiness the x-axis represents different isovalues and the y-axis represents the percentage error.

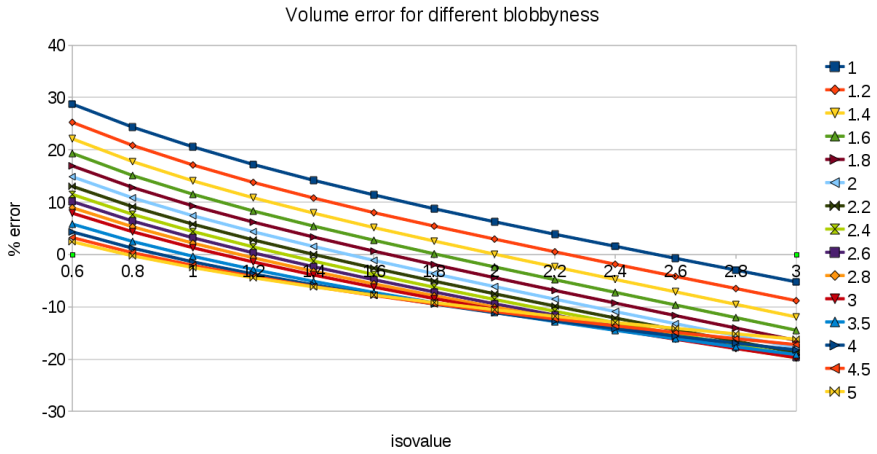


Figure 4.16: Difference, in terms of volume, between Gaussian and SES surfaces for different choices of blobbiness and isovalues. The different lines represent different choices of blobbiness, then for each blobbiness the x-axis represents different isovalues and the y-axis represents the percentage error.

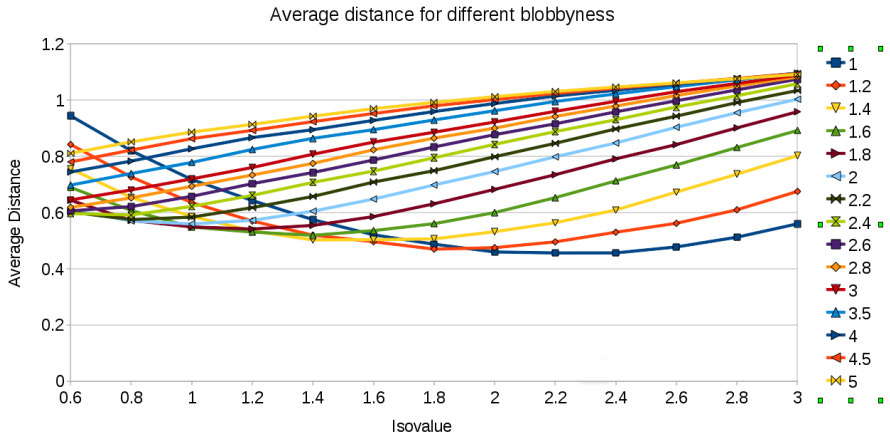


Figure 4.17: Difference, in terms of average distance between surfaces, between Gaussian and SES surfaces for different choices of blobbiness and isovalues. The different lines represent different choices of blobbiness, then for each blobbiness the x-axis represents different isovalues and the y-axis represents the percentage error.

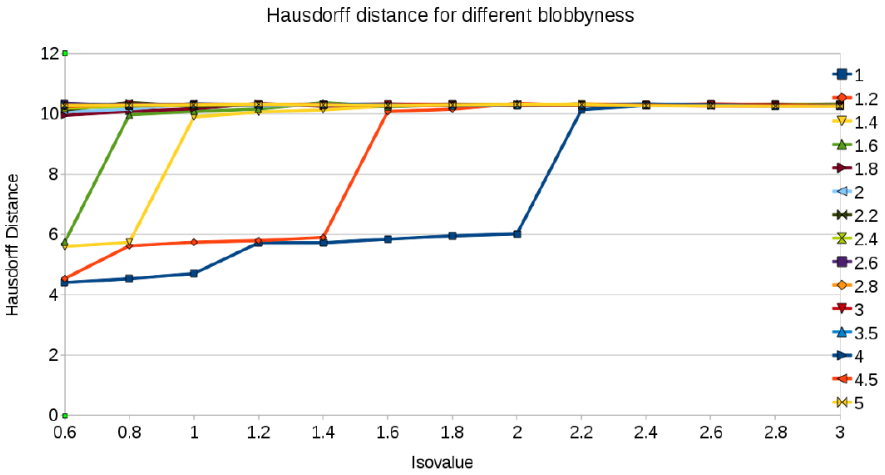


Figure 4.18: Difference, in terms of Hausdorff distance, between Gaussian and SES surfaces for different choices of blobbiness and isovalues. The different lines represent different choices of blobbiness, then for each blobbiness the x-axis represents different isovalues and the y-axis represents the percentage error.

CONSTRUCTOCTREE($Molecule, d_{min}, \epsilon, isoValue$) {*Constructs a octree for the atoms in set* $Molecule$}

Inputs: $Molecule$ is a set atoms in 3-dimensions. d_{min} is a positive number specifying the minimum length of a cell of the octree. ϵ is the expected minimum feature size. $isoValue$ is a real number used to define the surface.
Output: This routine constructs a dynamic octree from the atoms in $Molecule$. The octree is adaptively refined based on function values in cells and the given $isoValue$, while maintaining the constraint that LENGTH(u) $\geq$ d_{min} for all cells in the octree. Where, BOX(u) denotes the cube corresponding to any octree node (cell) u and LENGTH(u) denotes the length of a side of BOX(u).

1. create an octree $\mathcal{T}$ with a single node u as the root
2. make BOX(u) large enough to contain all atoms in $Molecule$
3. $minDepth \leftarrow \log_2(\text{LENGTH}(u))$
4. DEPTH(u) $\leftarrow 0$
5. intialize set $newLeaves \leftarrow \{u\}$
6. intialize set $markedNodes \leftarrow \phi$
7. ***while*** MARKFORREFINEMENT($newLeaves$, $minDepth, isoValue, markedNodes$) {*Finds nodes for splitting*}
8. clear $newLeaves$
9. ***for*** each node $v \in markedNodes$
10. SPLIT($v, d_{min}, newLeaves$) {*refines the octree and populates* $newLeaves$}
11. ***return*** $\mathcal{T}$

Figure 4.19: Algorithm for the construction of a dynamic octree from a given set M of atoms in 3D

Algorithms for Updating Octree

4.5.5 Details for efficient cell classification using augmented DPG

- **Initialization:**

  1. Consider a grid GP with uniform grid-size of $r_p/2$
  2. Mark all cells of GP as C_{OUT}. Each cell contains a list of pointers to the mesh inside the cell. Initially the list is empty.
  3. Mark all grid-points of GP as V_{OUT}

MARKFORREFINEMENT ($newLeaves, minDepth$, $isoValue, markedNodes$) {*Identifies all nodes which should be split*}

Input: A set of newly created octree leaves $newLeaves$, the $isoValue$ and a boolean $contourDetected$ denoting whether the the current octree contains a contour. If DEPTH of a node is less than $minDepth$, then it is automatically marked for splitting. Otherwise, $isoValue$ is used to mark nodes for splitting. **Output:** Populates the $markedNodes$ set of nodes which should be split. Returns true if such nodes are found.

1. $refineReq \leftarrow$ FALSE
2. ***for*** each node $u \in newLeaves$
3. ***if*** DEPTH$(u) < minDepth$ or SPLITREQUIRED$(u, isoValue)$
4. ***for*** each node $w \in \mathcal{N}(u)$
5. ***if*** $(\text{VAL}(u) - isovalue) * (\text{VAL}(w) - isovalue) < 0$ ***then*** {*contour is between the centers of* u *and* w}
6. ***if*** $w \notin markedNodes$ ***then*** add w to $markedNodes$
7. ***if*** $u \notin markedNodes$ ***then*** add u to $markedNodes$
8. $refineReq \leftarrow$ TRUE
9. return $refineReq$

Figure 4.20: Algorithm for identifying candidates for splitting

4. Insert all grid-points into a DPG. Let's call it DPG_{GP}
5. Create an empty DPG, named DPG_{atom} for storing all the atoms
6. Create an empty DPG, named $DPG_{expatom}$ for storing all the exposed atoms
7. Create an empty DPG, named $DPG_{sasgrid}$ for storing all grid points marked V_{SAS}
8. Define empty lists C_{rem}, C_{add} and C_{mod}.

- **Add(a_i, c_i, r_i):**

  Given an atom a_i with center c_i and radius r_i, (i) updates classifications of atoms, gridpoints and gridcells, (ii) updates the DPGs DPG_{atom}, $DPG_{expatom}$

SPLIT$(u, d_{min}, newLeaves)$ {*Splits the node* u}

Input: An octree node u and a positive number d_{min}. $newLeaves$ is a set where reference of the children are added. **Output:** This routine splits node u, by creating its children and distributing the atoms intersecting u to the children. And finally, it computes $\mathcal{F}(M, \text{CENTER}(u))$ for each $v \in \text{CHILD}(u)$.

1. ***if*** LENGTH$(u)/2 < d_{min}$ ***then*** return
2. ***else***
3. create children nodes of u
4. ***for*** each $v \in$ CHILD(u) ***do***
5. copy atoms from u which intersect BOX(v) to v
6. VAL$(v) \leftarrow \mathcal{F}(M, \text{CENTER}(v))$
7. LEAF$(v) \leftarrow$ TRUE
8. DEPTH$(v) \leftarrow$ DEPTH$(u) + 1$
9. add v to $newLeaves$
10. clear the list of atoms of u
11. LEAF$(u) \leftarrow$ FALSE

Figure 4.21: Algorithm for splitting an octree node u.

SPLITREQUIRED $(u, isoValue)$ {*Decides whether to split* u}

Input: An octree node u and the $isoValue$. **Output:** Based on VAL(u) and the values of its neighbors, decides whether the iso-contour potentially intersects u, in which case u should be split and SPLITREQUIRED returns TRUE.

1. ***for*** each node $w \in \mathcal{N}(u)$
2. ***if*** (VAL$(u) - isovalue) * ($VAL$(w) - isovalue) < 0$ ***then***
3. return TRUE
4. return FALSE

Figure 4.22: Algorithm for deciding whether a specific node u should be split

and $DPG_{sasgrid}$, (iii) updates the lists C_{rem}, C_{add} and C_{mod}, and finally (iv) updates the surface mesh.

1. Reset C_{rem}, C_{add} and C_{mod}
2. Let, $G1_i =$ all gridpoints inside $S_{SAS}(a_i)$.

MERGEREQUIRED $(u, isoValue)$ {*Decides whether to merge* u}

Input: An octree node u and the $isoValue$. **Output:** Returns true if u can be coarsened, i.e. u does not contain the isocontour.

```
1. merge ← TRUE
2. for each node v ∈ CHILD(u)
3.     if CHILD (v) ≠ NIL then
4.         merge ← merge&MERGEREQUIRED(v, isoValue)
5.     else
6.         merge ← merge&!SPLITREQUIRED(v, isoValue)
7. return merge
```

Figure 4.23: Algorithm for deciding whether children of a specific node u should be merged

MERGE(u) {*Merges the subtree under* u}

Input: An octree node u and a positive number d_{min}. $newLeaves$ is a set where reference of the children are added. **Output:** This routine merges the subtree under u. A node is merged by removing its children and copying the atoms in the children back to the parent. Finally, u is marked as a leaf.

```
1. if CHILD(u) = NIL then return                    {It is already a leaf}
2. else
3.     for each v ∈ CHILD(u) do
4.         if CHILD(u) ≠ NIL
5.             MERGE(v)
6.         copy atoms from v to u
7.         delete v
8.     LEAF(u) ← TRUE
```

Figure 4.24: Algorithm for merging the children of an octree node u.

3. For each grid-point $g \in G1_i$

(a) If g is marked V_{OUT}

i. Mark g as V_{SAS}

ii. Insert g into $DPG_{sasgrid}$

iii. Mark a_i as exposed.

iv. Insert a_i into $DPG_{expatom}$

v. UpdateCells(g)

4. Let, $G2_i = DPG_{GP} -> Range(c_i, r_i)$

5. For each grid-point $g \in G2_i$

(a) If g is marked as V_{SAS}

i. Mark g as V_{VDW}

ii. Remove g from $DPG_{sasgrid}$

iii. UpdateCells(g)

6. If a_i is exposed

(a) Let $A_i = DPG_{expatom} -> Range(a_i, r_i + r_{max})$

(b) For each $a_j \in A_i$

i. If $DPG_{sasgrid} -> Range(a_j, r_j + r_p)$ is empty then remove a_j from $DPG_{expatom}$

ii. Mark a_j as buried.

7. Insert a_i into DPG_{atom}

8. UpdateContour()

- **Remove**(a_i, c_i, r_i)**:**

  Given an atom a_i with center c_i and radius r_i, (i) updates classifications of atoms, gridpoints and gridcells, (ii) updates the DPGs DPG_{atom}, $DPG_{expatom}$

and $DPG_{sasgrid}$, (iii) updates the lists C_{rem}, C_{add} and C_{mod}, and finally (iv) updates the surface mesh.

1. Reset C_{rem}, C_{add} and C_{mod}

2. Remove a_i from DPG_{atom}

3. Let, $G1_i = DPG_{GP} -> Range(C_i, r_i + r_p)$

4. For each grid-point $g \in G1_i$

(a) If g is marked as V_{SAS}

i. Remove a_i from $DPG_{expatom}$

ii. If $DPG_{expatom} -> Range(g, r_max + r_p)$ is an empty set

A. Mark g as V_{OUT}

B. Remove g from $DPG_{sasgrid}$

C. UpdateCells(g)

(b) Else if g is marked as V_{VDW}

i. If $A1_g = DPG_{atom} -> Range(g, r_max)$ is an empty set and $A2_g = DPG_{atom} -> Range(g, r_max + r_p)$ is not empty

A. Mark g as V_{SAS}

B. Mark each $a_j \in A2_g$ as exposed and insert it into $DPG_{expatom}$

C. UpdateCells(g)

5. UpdateContour()

- **UpdateCells(g):**

1. For each gridcell C which is a neighbor of g
   (a) Update the classification of C based on the definition of the classes.
   (b) If applicable, add C to the appropriate list among C_{rem}, C_{add} and C_{mod}.

- **UpdateContour():**

1. For each gridcell $C \in C_{rem} \cup C_{mod}$
   (a) Remove the mesh inside C.
2. For each gridcell $C \in C_{add} \cup C_{mod}$
   (a) Compute a function F inside C: either a sum of Gaussian of atoms in the neighborhood of C, or a sign distance function for C
   (b) Contour a level set F
   (c) Add the mesh to C

- **Move(a_i, c_{i1}, c_{i2}, r_i):**

1. Remove (a_i, C_{i1}, r_i)
2. Add (a_i, C_{i2}, r_i)

- **getExposedAtoms():** Return all atoms in $DPG_{expatom}$

UPDATEOCTREE$(a, a', isoValue)$ {*Updates the octree when atom a moves to a new position* a'}

Input: The previous and new positions of an atom, and the $isoValue$. **Output:** This routine updates the octree by modifying the function values of affcted cells. The change in function values might produce a shift in the isocontour. So, the octree is refined/coarsened locally to reflect the change.

1. $affectedNodes \leftarrow \{u | (\text{BOX}(u) \bigcap (a \cup a')) \neq \varnothing\}$ {*identified by traversing the octree*}

2. intialize set $markedForMerge \leftarrow \phi$

3. intialize set $markedForSplit \leftarrow \phi$

4. ***while*** $affectedNodes \neq \phi$ ***do***

5. ***for*** each $u \in affectedNodes$ ***do***

6. ***if*** MERGEREQUIRED(u, isoValue) ***then***

7. add u to $markedForMerge$

8. mark u

9. remove u from $markedForMerge$

10. ***for*** each $u \in affectedNodes$ ***do***

11. remove u from $affectedNodes$

12. ***for*** each node $w \in \mathcal{N}(u)$

13. ***if*** $(\text{VAL}(u) - isovalue) * (\text{VAL}(w) - isovalue) < 0$ ***then***

14. ***if*** w is not marked ***then***

15. add w to $markedForSplit$

16. mark w

17. ***if*** u is not marked ***then***

18. add u to $markedForSplit$

19. mark u

20. ***for*** each node $v \in markedForSplit$

21. SPLIT$(v, d_{min}, affectedNodes)$

22. unmark v

23. remove v from $markedForSplit$

24. ***for*** each $u \in markedForMerge$ ***do***

25. MERGE(u)

26. unmark u

Figure 4.25: Algorithm for updating the octree due to atom a moving to a'

Chapter 5

Design and Calibration of Scoring Functions for Predicting of Protein-Protein Assemblies

In this chapter[1], we report the F^2Dock 2.0 + GB-rerank protocol which improves the state of the art in initial stage rigid body exhaustive docking search, scoring and ranking by introducing improvements in the shape-complementarity and electrostatics affinity functions, a new knowledge-based interface propensity term with FFT formulation, a set of novel knowledge-based filters and finally a solvation energy (GBSA) based reranking technique. Our algorithms are based on highly efficient data structures including the dynamic packing grids and octrees which significantly speed up the computations and also provide guaranteed bounds on approximation error. The improved affinity functions shows superior performance compared to their traditional counterparts in finding correct docking poses at higher ranks. We found that the new filters and the GBSA based reranking individually and in combination significantly improve the accuracy of docking predic-

[1]Part of contents of this chapter appeared in the article-
R. A. Chowdhury, M. Rasheed, D. Keidel, M. Moussalem, A. Olson, M. Sanner, and C. Bajaj. Protein-protein docking with f2dock 2.0 and gb-rerank. *PLoS ONE*, 8(3):e51307, 2013.
RAC (major contributor) and MR developed the scoring functions, and wrote the code ; MR developed and applied the machine learning approaches (Section 5.3); RAC, MR, DK, MM processed and curated dataset; RAC, MR, DK, MM performed experiments and summarized results; MR, RAC, DK, AO, MS and CB analyzed the data and wrote the paper.

tions with only minor increase in computation time. We compared F^2Dock 2.0 with ZDock 3.0.2 and found improvements over it, specifically among 176 complexes in ZLab Benchmark 4.0, F^2Dock 2.0 finds a near-native solution as the top prediction for 22 complexes; where ZDock 3.0.2 does so for 13 complexes. F^2Dock 2.0 finds a near-native solution within the top 1000 predictions for 106 complexes as opposed to 104 complexes for ZDock 3.0.2. However, there are 17 and 15 complexes where F^2Dock 2.0 finds a solution but ZDock 3.0.2 does not and vice versa; which indicates that the two docking protocols can also complement each other.

5.1 Introduction

The study of protein-protein interactions plays an important role in understanding the processes of life [100]. Though advancements in X-ray crystallography and other imaging techniques have led to the extraction of near-atomic resolution information for numerous individual proteins; the creation, crystallization and imaging of macromolecular complexes, as extensively required for drug design, still remains a difficult task. Among the atomic structures of proteins deposited in the *Protein Data Bank* [40], only a very small percentage are complexes. Hence, the need for fast and robust computational approaches to reliably predict the structures of protein-protein complexes is growing. An important step towards understanding protein-protein interactions is *protein-protein docking* which can be defined as computationally finding the relative transformation and conformation of two proteins that results in a stable (energetically favorable) complex if one exists.

Given two rigid proteins and some characteristic (e.g., electron density)

function(s) of the molecules, one can construct an appropriate representation of them and also define a correlation function based on cumulative overlap of the characteristic functions. Then it is possible to conduct a combinatorial search in a 6D parameter space of all possible relative translations and orientations of the two proteins to find the optimal. Hence in computational perspective, docking is a search over the space of possible orientation of two proteins to find the (set of) optimum(s) of a scoring function designed to mimic physico-chemical interaction of proteins.

The combinatorics of the search can be reduced by using coarse grids and rotational angles [142], and by using a-priori knowledge of suitable binding sites [102]. For docking without prior knowledge about possible binding sites, exhaustive sampling is required to improve the probability of finding the global minimum energy configuration. In such cases, Fast Fourier Transforms has been used to speed up the cumulative scoring function computations [102, 142, 215]. Spherical Fourier correlation based approaches were presented in multiple studies [86, 150, 175, 214, 215]. However, if binding sites are known, or inferred based on some initial stage docking, then a finer resolution search involving local flexibility can be applied to improve the accuracy of the fit [73, 80, 128].

Accuracy of docking predictions is dependent on the scoring model's ability to distinguish between native and non-native poses. Docking based on structural (shape) complementarity alone has shown to be adequate for a range of proteins [61, 102, 114]. To represent shape complementarity, a grid based double skin layer approach became the base of many variations and software, e.g., DOT [170], ZDOCK [60], PIPER [151], MolFit [32, 33] and RDOCK [159]. However, en-

ergy and bioinformatics based scoring terms have been shown to improve the accuracy of predictions and a combination of multiple scoring terms have become the norm in current docking software. For example, DOT 2.0 [170] is based on van der Waals energy and Poisson-Boltzmann electrostatics, ZDock 3.0.2 [179] uses pairwise shape complementarity, electrostatics, and pairwise potentials known as Interface Atomic Contact Energies (IFACE), PIPER [151] is based on shape complementarity and electrostatics using a Generalized Born (GB) type formulation, and uses a new class of structure-based potentials referred to as DARS (Decoys As the Reference State) where the potentials are derived from a large set of docking conformations as decoys. FRODOCK [104] is a recent spherical harmonics based docking tool that uses van der Waals, electrostatics and desolvation potential terms in its correlation function. Some docking or reranking techniques solely use coarse-grained potentials trained on large benchmark of decoys [162, 210]. We leave the reader to consult the reviews [49, 112, 120, 197, 216] for further information.

In [56] we described a non-equispaced Fast Fourier Transform (NFFT) based rigid-body protein-protein docking algorithm for efficiently performing the initial docking search (based on shape and electrostatics complementarity). Compared to traditional grid based Fourier docking algorithms, the algorithm was shown to have lower computational complexity and memory requirement. The algorithm was extended in [26] to F^{2}Dock (F^2 = Fast Fourier), which included an adaptive search phase (both translational and rotational) for faster execution.

Here, we describe a new version (F^{2}Dock 2.0) which includes improved shape-complementarity and electrostatics functions as well as a new on-the-fly

affinity function based on interface propensity and hydrophobicity. The current version uses uniform FFT, but exploits the sparsity of FFT grids for faster execution and restricts its search within a narrow band around the larger molecule. A clustering phase penalizes docking poses that are structurally similar to poses with better scores and a set of efficient on-the-fly filters penalize potential false positives based on Lennard-Jones potential, steric clashes, interface propensity, interface area, residue-residue contact preferences, antibody active sites, and glycine richness at the interface for enzymes. The filters are implemented using fast multipole type recursive spatial decomposition techniques [25, 66]. A solvation energy based reranking program GB-rerank [28, 66] has also been implemented using an approximation scheme which can be tuned for speed-accuracy trade-off. Both F^{2}Dock 2.0 and GB-rerank have been implemented as multithreaded programs for faster execution on multicore machines. Our molecular visualization software *TeχMol* serves as a front-end to F^{2}Dock 2.0 in a client-server mode of execution [15]. *F^{2}Dock* has been calibrated based on an extensive experimental study of the rigid-body complexes from Zlab benchmark 2.0 [180] and tested on Zlab benchmark 4.0 [129] (which includes the complexes from benchmark 2.0).

5.2 Scoring and Search Protocol

Let A and B be two proteins with M_A and M_B atoms respectively. Without loss of generality, we assume that $M_A \geq M_B$, i.e., A is the larger of the two proteins. We refer to A and B as "receptor" and "ligand", respectively.

Figure 5.1 gives a high level overview of the algorithm. The rest of this sec-

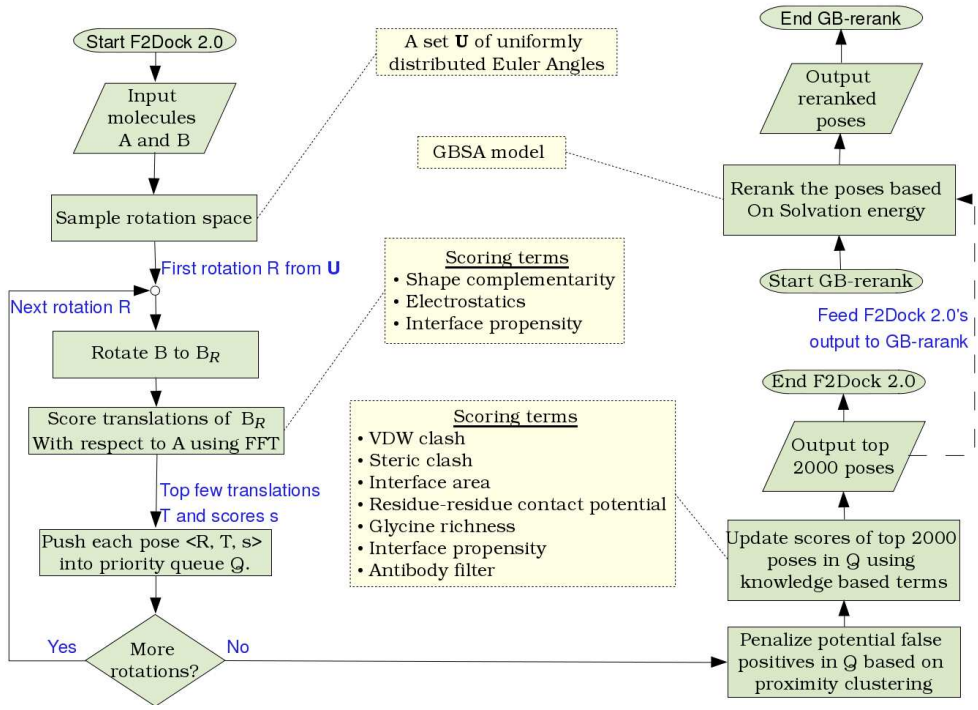


Figure 5.1: **High-level overview of rigid-body protein-protein docking using F^{2}Dock 2.0 and GB-rerank.** F^{2}Dock 2.0 performs exhaustive 6D search in discretized rotational and translational space where it computes a score for each sampled orientation of the ligand with respect to a stationary receptor. The scoring function is a weighted combination of shape complementarity, electrostatics and interface propensity based affinity terms. The top few orientations (poses) of the ligand are kept in a priority queue. Then top several thousand poses from the queue are clustered based on the distance between the geometric centers of different poses of B. All but the best scoring pose of a cluster is penalized by reducing the score. The resulting reordered list is then passed through several soft filters in order to further penalize potential false positives. Finally, as a separate post-processing step, the ranked docking poses are re-scored and reranked based on the change in solvation energy caused by each pose.

tion details the various aspects of the algorithm. Even greater details are available in [65].

5.2.1 Overall Strategy

First, F^{2}Dock 2.0 performs exhaustive search in 6D space of relative configuration of B with respect to A. We use a discrete and uniform sampling of 3D rotational space and then use FFT to score a discrete 3D translational space. Given N_R rotational samples and N^3 translational grid, F^{2}Dock 2.0 computes $N_R N^3$ scores. However, only a constant multiple of N_R scores and their corresponding poses are retained for the next step. Let us denote this set as Q. A particular pose is expressed as a tuple $< \mathbf{t}, \mathbf{r}, s >$ where $\mathbf{t}$ is the translation, $\mathbf{r}$ is the rotation and s is the corresponding score.

We apply a very simple clustering scheme based on proximity of the poses in Q to reshuffle the order such that the top few poses are dissimilar to each other. Though this step does not affect the overall ratio of true and false positives, it increases the chance of finding at least one near-native solution at the top of the order. It is important because the next stage of filtering is only performed on the top few (2000 by default) poses. Let this reduced list be called Q'.

The filters are designed based on knowledge-based scoring potentials to update the scores of the poses of Q', reorder them and output them as final predictions from F^{2}Dock 2.0. Some filters are defined only for specific types of proteins like Antibodies or Enzymes.

The results from F^{2}Dock 2.0, or a subset of it, can optionally be reranked using a solvation energy (generalized Boltzman model) based reranker called GB-rerank which generally improves ranks of near native solutions.

5.2.2 Phase I (Exhaustive 6D Scoring and Search with FFT):

F^2*Dock* exhaustively searches over a discretized $SO^3 \times R^3$ space. First, it samples the rotational space SO^3 uniformly [181] and applies the rotation on the ligand after placing it's mass center at the origin. Then it scores each relative positions of the rotated ligand w.r.t. the stationary receptor over the set of translations in R^3. FFT (Fast Fourier Transform) based 3D convolution is used to compute scores for all points on the uniform 3D translational grid. The current version uses uniform FFT, but exploits the sparsity of FFT grids for faster execution, and also restricts its search within a narrow band around the larger molecule. The top several thousand poses from this FFT-based scoring phase are inserted in a priority queue sorted by the scores.

The scoring function is a weighted combination of shape complementarity, electrostatics and interface propensity (or hydrophobicity) based affinity terms. We briefly introduce the terms here (refer to [26, 65] for further details).

5.2.2.1 Shape Complementarity

Shape complementarity was originally introduced to model the lock-and-key matching idea of docking, i.e the proteins have complementary shapes at the binding interface. Energetically, this models the van der Waals interaction to some extent.

F^{2}Dockuses an improved double-skin layer approach to shape complementarity. Atoms on the ligand which are exposed to the solvent are considered $skin$ atoms, and the remaining atoms are considered $core$. All atoms of the receptor are

$core$ and a layer of atoms are added outside the solvent excluded surface (SES) of the receptor and these extra atoms are considered the $skin$ for the receptor. The shape complementarity function is designed to maximize skin-skin overlaps and minimize skin-core and especially core-core overlaps, since it indicates that the ligand is coming close to the receptor without penetrating it. This can be achieved numerically by assigning positive real affinity values on the $skin$ atoms and positive imaginary affinity values on the $core$ atoms. Hence, a convolution would generate positive real contributions S_{SS} from skin-skin overlaps, negative real contributions S_{CC} from core-core overlaps and positive imaginary contributions S_{SC} from skin-core overlaps. The complete shape complementarity score is defined as a weighted combination of them - $S_{shape} = W_{SS}S_{SS} + W_{CC}S_{CC} + W_{SC}S_{SC}$, where W_{SS}, W_{CC} and W_{SC} control the relative importance of the different terms. For example, a high W_{CC} would heavily penalize any penetrations.

F^{2}Dock improves on the above double-skin idea by further refining the affinity values assigned to the atoms. First, F^{2}Dock uses a depth based scheme for assigning affinity values to the $core$ atoms which penalizes deeper penetrations more than shallow ones. Secondly, F^{2}Dock assigns the affinity values on $skin$ atoms based on curvature to promote binding near pockets and mouths. And finally, the receptor's $skin$ atoms do not touch its SES, but are placed a small distance away to define a 'floating' skin. This tries to mimic the fact that in most complexes the SES of the ligand and receptor does not actually touch each other, but rather stay at a very close distance.

5.2.2.2 Electrostatics

F[2]Dock models long distance electrostatic interactions using the simplified model for Coloumbic electrostatics proposed by Gabb et. al. [101] which allows efficient FFT-based computation of the term during docking search. Two affinity functions f_A^E and f_B^E are defined for molecule A and B, respectively.

$$f_A^E(\mathbf{x}) = \sum_{k \in A} \frac{q_k}{E(\mathbf{x} - \mathbf{c}_k)(\mathbf{x} - \mathbf{c}_k)} \cdot g_{A,k}^E(\mathbf{x})$$

$$\text{and } \; f_B^E(\mathbf{x}) = \sum_{k \in B} q_k \delta(\mathbf{x} - \mathbf{c}_k) \cdot g_{B,k}^E(\mathbf{x}),$$

where, q_k is the Coloumbic charge[2] on atom k, $\delta(\mathbf{x})$ is the Kronecker delta function with value 1 at $||\mathbf{x}|| = 0$, and 0 everywhere else, $g_{A,k}^E(\mathbf{x}) = g_{B,k}^E(\mathbf{x}) = 1$ and $E(\mathbf{x})$ is the distance dependent dielectric constant [101]. The convolution of the two affinity functions produce the electrostatic score S_{elec}

5.2.2.3 Interface Propensity or Hydrophobicity

Unlike shape complementarity (vdW) and electrostatics (Coloumbic) terms which are based on molecular free energies, the Interface propensity or Hydrophobicity term is based completely on statistical and empirical observations. It has been observed that Hydrophobic residues tend to be found near the core of the molecules and Hydrophilic residues are found on the surface. However, if there are some Hydrophobic residues on the surface of a protein, then they tend to act as binding sites so that they can get buried by forming a complex with another protein. Based on

[2]charge assignments are made using PDB2PQR [3]

this idea, we reward docking poses where the binding interface contains Hydrophobic residues. We use per residue Hydrophobicity values from [44] to define weights on the surface atoms and then compute the convolution using FFT. Refer to [65] for details about the formulation and FFT adaptation.

However, there are other factors which also promotes the possibility of a residue to be on the binding interface. Jones and Thornton [133, 134] studied the interface of 63 protein-protein interfacs and computed a interface propensity value IP for each residue type. The interface propensity is defined as the log normalized probability of a residue being on the binding interface given that it is present in the molecule. The IP values for the 20 amino acid residues lie between -0.38 (for ASP) and 0.83 (for TRP). A residue with a higher IP value is likely to occur more frequently in a protein-protein interface than one with a lower IP value. After assigning the IP values to atoms (based on their residues), we follow the same technique used for Hydrophobicity to compute a score S_{IP} for a given pose.

In F^{2}Dock, we model interface propensity and hydrophobicity as alternate ways of modeling the same phenomenon and the user has a option to select either one.

5.2.2.4 Overall score of Phase I

The overall score of phase I is-

$$S_{phaseI} = S_{shape}W_{shape} + S_{elec}W_{elec} + S_{IP}W_{IP}$$

where, W_{shape}, W_{elec} and W_{IP} controls the relative importance of the dif-

ferent scoring terms. We have observed that the contribution of the terms vary significantly for different class of complexes (Enzymes, Antibodies, Others etc.), and hence the weights must be learned separately for each class.

5.2.3 Phase II (Bioinformatic scoring terms computed by fast multipole methods)

In this phase, the top several thousand poses, inserted into the priority queue in phase I based on their S_{phaseI}, are evaluated using several statistical scoring terms to prune away false positives. Each of the terms reward or penalize a pose by updating its score. When all the terms have been applied, the final updated scores are used to re-rank the poses. The terms are computed using fast multipole type recursive spatial decomposition techniques [66].

We briefly introduce these terms below. Refer to [65, 66] for details.

5.2.3.1 Proximity Clustering

The poses in the priority queue are examined for structural similarity based on how close the geometric centers of B are to each other (note that A is kept static). If a structurally similar docking pose with a better score exists, then the docking pose with lower score is further penalized by reducing its score.

5.2.3.2 Lennard-Jones

Penalize if the Lennard-Jones potential of a docking pose is above a threshold. We approximate the Lennard-Jones (LJ) potential between molecules A and

$B_{\mathbf{t},\mathbf{r}}$ given by the following expression.

$$LJ(A, B_{\mathbf{t},\mathbf{r}}) = \sum_{i \in A, j \in B_{\mathbf{t},\mathbf{r}}} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} \right),$$

where r_{ij} is the distance between atoms $i \in A$ and $j \in B_{\mathbf{t},\mathbf{r}}$, constants a_{ij} and b_{ij} depend on the type (e.g., C, H, O, etc.) of the two atoms involved. For any fixed pair of atom types a_{ij} and b_{ij} are fixed, and are calculated from the Amber force field.

5.2.3.3 Steric Clash

Penalizes all docking poses with the number of steric (atom-atom) collisions above a threshold. Two atoms $a \in A$ and $b \in B$ with van der Waals radii r_a and r_b, respectively, are said to be in a *clash* provided the distance between their centers is smaller than $\alpha(r_a + r_b)$, where α is a user-defined positive constant.

5.2.3.4 Interface Area (Dispersion)

Penalizes a docking pose if the interface area is outside acceptable range. Interface area is computed by defining a smooth surface representation (triangulated mesh) of the proteins. Then Gaussian quadrature points are sampled on the triangles such that the weight of a quadrature point corresponds to the area of the triangle supporting it. Hence, the interface area is the sum of the weights of the quadrature points which are on the interface (have a neighboring quadrature point on the other surface within a distance threshold).

5.2.3.5 Interface Propensity or Hydrophobicity

Using statistical information from [134] it computes a score for each pose which from a high level can be viewed as the ratio of the interface area of the pose corresponding to residues that typically appear in high frequencies in protein interfaces to the interface area corresponding to residues that appear in low frequencies. A docking pose is penalized if this ratio is below a threshold. Alternatively, Hydrophobicity values from [44] can be used.

5.2.3.6 Residue-Residue Contact

It was observed in [106] that large hydrophobic residue pairs typically have high contact preferences while the smallest contact preferences were observed between pairs of residues that are small in size. Interfaces do not seem to favor contacts between hydrophobic and polar residues, and between charged residues that do not have charge complementarity. F^2Dock uses the pairwise contact preference values listed in either Table III (without volume normalization) or Table IV (normalized w.r.t. residue volumes) of [106]. This term penalizes a pose if the sum of residue-residue contact values of the given pose fall below a threshold. Two residues are considered to be in contact if the distance between their C_β atoms (C_α for Gly) is less than 6 Å.

5.2.3.7 Antibody-Antigen Contact

This term uses statistical information on antibody-antigen contact preferences derived in [1, 169]. It is based on the observation that in each antibody each

of the following three regions will make at least one antigen contact: (1) either CDR-L1 or CDR-H1, (2) CDR-L3, and (3) CDR-H3.

5.2.3.8 Glycine Richness

This term exploits the observation that enzyme active sites are rich in glycines, particularly *G-X-Y* and *Y-X-G* oligopeptides, where X and Y are polar and non-polar residues, respectively, and G is glycine [277].

5.2.4 Phase III (Solvation Energy Based Reranking)

The ranked docking poses obtained from phase I are re-scored and reranked based on the change in solvation energy caused by each pose. The polar part of the solvation energy is approximated using the surface-based formulation of Generalized Born (GB) energy [28], and implemented using a fast octree-based approximation scheme which we describe in detail in [66]. Among the non-polar parts the dispersion energy is also approximated using octrees while the cavity forming energy is approximated by computing an approximate interface area of the two molecules using our fast linear-space *Dynamic Packing Grid* (DPG) data structure described in [25].

5.3 Learning the Best Combination and Weighting of Scoring Terms

Several machine learning based approaches have been proposed recently. Ravikant and Elber in [211] used quadratic programing [187] to learn 441 inter pa-

rameter for residue contact potentials in which their objective function maximizes a distance metric between correct solution and decoys. Similarly, Andersson et al. in [6] used a multi-variance approach to optimize linear parameter sets. Hetenyi et al. in [124] used linear regression models to learn the linear relationship between multiple score terms and empirical binding free energy. Other than linear models, Teramoto et al. in [248] used random forest classifier as supervised scoring method and Gozalbes et al. in [109] learned threshold parameters from statistical results of empirical data. Adaptive parameter swiping is another type of learning approach in which the initial parameter values are randomly sampled and then updated in an iterative manner. Pham et al in [196] and Seifert et al. in [232] used gradient descend and line search for high dimensional searching and Antes et al. in [9] used a neural network to parameterize the relationship between parameter vector and objective function (RMSD for top 5 results). Genetic programming [242] has also been applied in parameter space search where evolution and mutation are exploited to update the generation of parameter vectors. Another score driven approach is applied by Yang et al. in [278] in which multiple criteria (Z-scores etc.) are used to instruct the iterative steps. Notice that the learning approaches presented in all the previous work assume homogeneity of the parameters they are trying to learn, i.e. they are applicable for either linear sums of weighted terms, or for interval thresholds etc. Here, we identify the different types of parameter spaces and the need for tailoring the learning methods for each. We design a multi-stage mixed learning model, a combination of quadratic programming and random forest classifiers, for optimizing the parameters of the scoring functions described above.

5.3.1 Quadratic Programming

The quadratic programming approach targets at maximizing the separating distance between correct and incorrect solutions. For transformation τ, recall the scoring function

$$E(\tau) = w^T \cdot P_\tau$$

The parameter vector we need to train is

$$w = [w_{ss}, w_{sc}, w_{cc}, w_{elec}, w_{hbond}, w_{ip}]$$

and P_τ is the corresponding feature vector for a given transformation τ. The training data (input) of the algorithm is a set of n receptor-ligand pairs $X = \{X_1, X_2, ..., X_n\}$ where $X_i = \{X_{i1}, X_{i2}\}$ and their corresponding correct transformation $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$. The algorithm input also includes a constant C, tolerated error v and size of region in output space ϵ.

At the beginning of the algorithm, we sample set of clusters $\Gamma_i^0 = \{G_1, G_2, ...\}$ where $G_j = \{\tau_1, \tau_2, ...\}$ of incorrect transformations for each receptor-ligand pair X_{i1}, X_{i2}. In this definition, Γ_k is a set of clusters, G_k is a cluster and τ_k is a transformation. Then we compute the set constraints for each set of clusters such that

$$S_i \leftarrow \{\forall G_k \in \Gamma_i^0 \forall \tau_i^j \in G_k : w^T(P_{\tau_i} - P_{\tau_i^j}) \geq 1 - \frac{\delta_{ik}}{\Delta(\tau_i, \tau_i^j)}\}$$

In this step, we compute a slack variable δ_{ik} for each cluster G_k so that the error between all transformations τ_i^j in G_k and the correct transformation τ_i cannot violate the minimum threshold $1 - \frac{\delta_{ik}}{\Delta(\tau_i, \tau_i^j)}$ where $\Delta(\tau_i, \tau_i^j)$ is the rmsd value between

τ_i and τ_i^j. We solve the QP

$$(w, \delta) = \arg\min_{w,\delta} \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i,k}\delta_{ik}$$

to get the minimum value of each slack variable δ_{ik} which maximize the separating distance between scores of correct and incorrect transformations. Until now we have assigned the initial values for the variables which will update during iterations:

- Γ_i^0: Initial set of clusters
- w: Initial values for parameter vectors
- δ: Initial values for slack variables

Then we begin the iteration. In the α^{th} iteration, we find the set of top violated transformations T_i^α in each set of clusters $\Gamma_i^{\alpha-1}$ and re-cluster them. For each cluster G_k in T_i^α, we will use it to generate the k^{th} cluster Γ_{ik}^α for the new set of clusters Γ_i^α. The generating process is described as: For each transformation τ_i^j in G_k, if it is possible for it to violate the constraint for the k^{th} cluster such that

$$\Delta(\tau_i, \tau_i^j)(1 - w^T(P_{\tau_i} - P_{\tau_i^j})) > 0$$

We'll determine whether to add this transformation into the new cluster Γ_{ik}^α by the following two criteria:

- If the k^{th} cluster exists in set $\Gamma_i^{\alpha-1}$ but transformation τ_i^j violates the slack variable δ_{ik} that

  $$w^T(P_{\tau_i} - P_{\tau_i^j}) < 1 - \frac{\delta_{ik} + v}{\Delta(\tau_i, \tau_i^j)}$$

  add τ_i^j into Γ_{ik}^α.

- If the k^{th} cluster doesn't exist in set $\Gamma_i^{\alpha-1}$, add τ_i^j into Γ_{ik}^{α}.

If either criteria is met, we will add the transformation into Γ_{ik}^{α}. After all the new clusters are generated, we have the new set of clusters Γ_i^{α}. Similar as the beginning, we set constraint δ_{ik} for each cluster G_k in Γ_i^{α} such that

$$S_i \leftarrow \{\forall G_k \in \Gamma_i^{\alpha} \forall \tau_i^j \in G_k : w^T(P_{\tau_i} - P_{\tau_i^j}) \geq 1 - \frac{\delta_{ik}}{\Delta(\tau_i, \tau_i^j)}\}$$

and solve the QP

$$(w, \delta) = \arg\min_{w,\delta} \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i,k}\delta_{ik}$$

again to update w and δ_{ik}.

Until now the three variables Γ, w and δ are all updated and then we use them to begin the iteration $\alpha + 1$. The iteration will terminate if no new constraint is generated during the iteration and the w at that time is our final output.

5.3.2 Random Forest Classifier

The random forest classifier [51, **?**] consists of multiple decision tree classifiers each of which satisfies the conditions below:

- Nodes are constructed from a subset of data. Root node contains all data. Each data item is a vector.
- At each node, search through all variables to find best split into two children nodes.
- Split all the way down and then prune tree up to get minimal test error

The generation process of the random forest is described as:

- Root node contains a bootstrap sample of data of the same size as original data. Each tree has different bootstrap sample
- For each node, a random subset of training samples are selected and the best split is found by searching through these samples. The search algorithm is called classification and regression tree (CART) search with Gini criterion. The Gini criterion is a statistical measure of the statistical dispersion of a set of data points. In the CART algorithm, each split only depends on one predictor value i.e. one dimension of the training sample. For a training set with k training samples, there are k possible splits for each dimension.
- Find each dimension's best split: Sort the value of this dimension from the smallest to the largest. For the sorted values, go through each value from top to examine each candidate split point (call it v, if x *CORRECT* v, the case goes to the left child node, otherwise, it goes to the right) to determine the best. The best split point is the one that maximize the splitting criterion the most when the node is split according to it.
- the Gini criterion for node t is defined as:

$$\Delta i(s,t) = i(t) - P_L i(t_L) - P_R i(t_R)$$

where

$$t(t) = 1 = \sum_j p^2(j|t)$$

Gini criterion describes the purity of the node and the split the maximize $\Delta i(s,t)$ will be found. Note that for n k-dimensional examples there are nk possible splits.

- Normally a large number of trees (> 100) are generated and the final prediction result is the average of the votes of all trees.

For a docking exercise, we represent each relative pose using a vector containing the score for each individual term of the multi-term scoring model. These score vectors are used as data items for the random forest. During the training step, we label score vectors corresponding to transformations with rmsd smaller than 5 as positive training sample and all transformations with rmsd greated than 20 are labeled as negative. We have used 1342 positive training samples and 214,532 negative training samples in total drawn from all complexes. The number of decision trees contained in the forest is set to be 500. Given a test pose, we simply compute the rf-score as the average of the votes of all trees with the corresponding score vector used as input. Finally, for each complex, we take the originally ranked results (top $10,000$) and use their rf-scores for reranking.

5.3.3 Dataset Preparation

F2Dock takes two PDB files as inputs. First the PDB files are processed by PDB2PQR [79] where missing atoms such as Hydrogens are added, the protein is optimized for hydrogen bonding, and charge and radius parameters are assigned using the AMBER force-field available in PDB2PQR. If the given PDB has missing

residues or too many residues with missing backbone atoms, then our curation process fails and F2Dock cannot be used without manually curating the PDB or using other curation software.

Then pseudo-atoms are added above the surface of the receptor (i.e., stationary molecule), and surface atoms of the ligand (i.e., moving molecule) are detected. These atoms are marked as skin atoms, and the rest as core atoms.

5.3.4 Parameter Selection Based on Complex Type

F^2Dock 2.0 has several free parameters in its pipeline. We can broadly classify the parameters into several groups. For parameters like the charge and radii of atoms, or the hydrophobicity and interface propensity of residues etc., we either use well-established parameters (for example, from the AMBER [54] force field) or derive from previously published results (for example, interface propensity values from [134]). Some parameters are internal to a scoring function for example the distance dependent dielectric for electrostatics, or the thickness of the skin used in shape complementarity. These parameters are trained using manual parameter sweeps based on a small number (4-5 per complex type) of complexes. However, we produced multiple configurations for each complex and chose the set of parameters which maximizes the corresponding individual scoring term for the near native poses. A similar strategy was used for selecting the thresholds used to penalize poses during filtering. Finally, there are the parameters that govern the weights assigned to different scoring terms when they are combined as well as the weights (or percentages) by which poses are penalized. These parameters are the most difficult

to train as the scoring terms are not independent and the relative influence of a term might vary for different complexes. These parameters were trained based on the 60 complexes from Zlab's protein-protein benchmark 2.0 [180] as follows.

The complexes in the benchmark are categorized into four main types: Antibody-Antigen (A) and Antibody-bound Antigen (AB), Enzyme-Inhibitor/Enzyme-Substrate (E), and other (O) types. We identify that the classification is not only functional, but it also has significant effect on scoring function design since different scoring terms bear different level of significance for different categories of complexes. For example, it is known that binding interfaces of Enzymes are rich in Glycines, which lead us to design a filter based on Glycine richness and it is applied only for Enzyme type of complexes. For each class of complexes (9 Antibody-Antigen, 9 Antibody-bound Antigen, 21 Enzyme-Inhibitor/Enzyme-Substrate and 21 Others), we train the weight parameters separately. The objective for the training is to improve the ranks of near-native solutions for as many complexes as possible. We performed parameter sweeps for each of the weights that combines the FFT based scores based on the above objective for each of the categories. Then we examined the effect of applying each of the filter, one at a time, and controlled its penalty to improve the results.

We do realize that our manual scheme has its drawbacks, specially since it does not sufficiently cover the entire space of possible values for the parameters. We are actively trying to use machine learning schemes to train the parameters in a more robust way. However, so far our attempt of using quadratic programming and random forest learning based on thousands of negative and positive examples based

on this benchmark have failed to produce a set of parameters which outperform the manually calibrated set of parameters.

Default values of all the parameters for different types of complexes can be found in the user manual for F^{2}Dock 2.0 downloadable from our website (link given in the abstract).

5.3.4.1 Automated Detection of Complex Types

Since F^{2}Dock 2.0's parameters are optimized separately for antibody-antigens and enzyme-inhibitors/ enzyme-substrates, and a general set of parameters are used for all other types of complexes, the user only needs to specify the complex type to ensure the set of optimized parameters are applied. If the type is unknown, F^{2}Dock 2.0 tries to determine which set of parameters to use as follows. If F^{2}Dock 2.0 locates the six CDR loops (L1, L2, L3, H1, H2 and H3) in the protein sequence using the algorithm in [174], it identifies it as an antibody and uses the corresponding parameter set. Otherwise, if neither molecule is identified as an antibody and at least one of the molecules has at least 200 residues and at least 8% of its surface residues are Glycines then F^{2}Dock 2.0 uses the enzyme complex parameter set. Finally, if both tests fail, a set of parameters for the general case is used. Among the complexes in the Zlab benchmark 2.0, F^{2}Dock 2.0 fails to identify only one antibody (1KXQ) and three enzymes (1AY7, 1UDI and 2MTA). See supplement for details.

5.4 Results

We present the results of our experiments to explore the contribution of the new scoring terms and filters available in F^2Dock 2.0 as well as the solvation energy based re-ranker GB-rerank on prediction accuracy. These experiments are carried out on the set of complexes in Zlab's benchmark 2.0 [180] which contains 60 complexes. Then we run F^2Dock 2.0 with the best set of parameters on the complexes in the Zlab benchmark 4.0 [129], and compare the performance with ZDock 3.0.2 [179]. The complexes in both the benchmarks are categorized into rigid-body (easy), medium and difficult (flexible) based on the RMSD between the bound and unbound states of the proteins. They are also categorized into four main types: Antibody-Antigen (A) and Antibody-bound Antigen (AB), Enzyme-Inhibitor/Enzyme-Substrate (E), and other (O) types. As mentioned before, F^2Dock 2.0 uses different set of parameters for the different categories and we have also compared our results for each category separately.

5.4.1 Evaluation Criteria

F^2Dock 2.0's search leaves the receptor stationary and searches over the orientations of the ligand. Hence, to evaluate the accuracy of a predicted pose, we compute the deviation between the predicted position of the ligand and its correct position as the root mean squared distance (RMSD) of the interface atoms. Note that correct position of the ligand for unbound test cases can be approximated by aligning the unbound components to their bound counterparts. The unbound ligands in the ZLab benchmarks are provided after alignment with bound counterparts

and hence can be used as the approximate truth without further manipulations. We assume that an atom is on the interface if the distance between its center and the center of any atom on the other molecule is less than 10Å. We define L_I as the set of all backbone atoms of the ligand which are on the interface when the ligand is placed in its native pose w.r.t the receptor (to find the native pose for an unbound case, we simply align the unbound receptor and unbound ligand to their bound counterparts). If the position of ligand atom a_i is x_i^* in the native pose and x_i^P in a predicted pose P, then the interface RMSD is computed as $IRMSD = \sqrt{\frac{1}{|L_I|}\sum_{a_i \in L_I}(|x_i^* - x_i^P|^2)}$. A predicted solution is considered a *hit* provided its IRMSD value is at most 5Å.

In the remaining text and supplement, we refer to the hit with the lowest RMSD as the 'best' hit and the hit with the highest rank as the 'top' hit. In most of our results, we compare protocols based on the rank of the 'top' hit. Given a set of complexes C, and a protocol S, we define $C_S(x)$ as the set of complexes such that for each complex $c \in C_S(x)$, the top hit lies within the range $[1, x]$. Clearly, for a given x a higher $C_S(x)$ is better. Hence, to compare the accuracy of two protocols S_1 and S_2, we can simply compare $C_{S_1}(x)$ and $C_{S_2}(x)$ for different x. In general we use a few specific values for x ([1,1], [1,5], [1,10], [1,50], [1,100], [1,500] and [1,1000]). We are specially interested in the first few ranges which shows off the accuracy of the scoring model, and the last range which shows off the applicability of the model over a broad range of complexes.

Two residues $R_i \in A$ and $R_j \in B$ are considered to be in contact if the distance between the centers of any atom $a_{ii} \in R_i$ and any atom $a_{jj} \in R_j$ is less than a threshold. The set of residue-residue contacts for the native pose of the

receptor and ligand are defined as the native contacts $\mathbf{N}$. For a given predicted pose, we compute the set of residue-residue contacts for that pose as $\mathbf{C}$. The set of native contacts for that pose is hence defined as $\mathbf{N}' = \mathbf{N} \cap \mathbf{C}$. Now, we define another metric based on native contacts as $F_{nat} = |\mathbf{N}'|/|\mathbf{N}|$. We follow the well known CAPRI criteria that uses a combination of F_{nat} and $IRMSD$ to classify predictions as high, medium, acceptable and incorrect.

5.4.2 Analyzing the Improvements due to New Affinity Functions and Filters

5.4.2.1 Effectiveness of the New Skin-Core Definition

We have compared the new improved double skin approach to the traditional approach (used in F^{2}Dock [26]) in terms of their prediction accuracy on the rigid-body complexes of the Zlab Benchmark 4.0. In these tests only the shape complementarity term was used, and hence the results are not as accurate as the default combination of scoring and filtering terms can produce.

In Figure 5.2(a), we clearly notice the improvement offered by the floating skin approach over the traditional which validates our idea that a softer definition of skin is better for unbound docking. However, the traditional skin approach performs slightly better for the bound-bound (re-docking) test cases (Figure 5.2(b)). Figure 5.2(c) shows that as a result of the improved skin definition, F^{2}Dock 2.0's shape complementarity function outperforms DOT and ZDock on the rigid complexes from Zlab benchmark 2.0 (bound-bound).

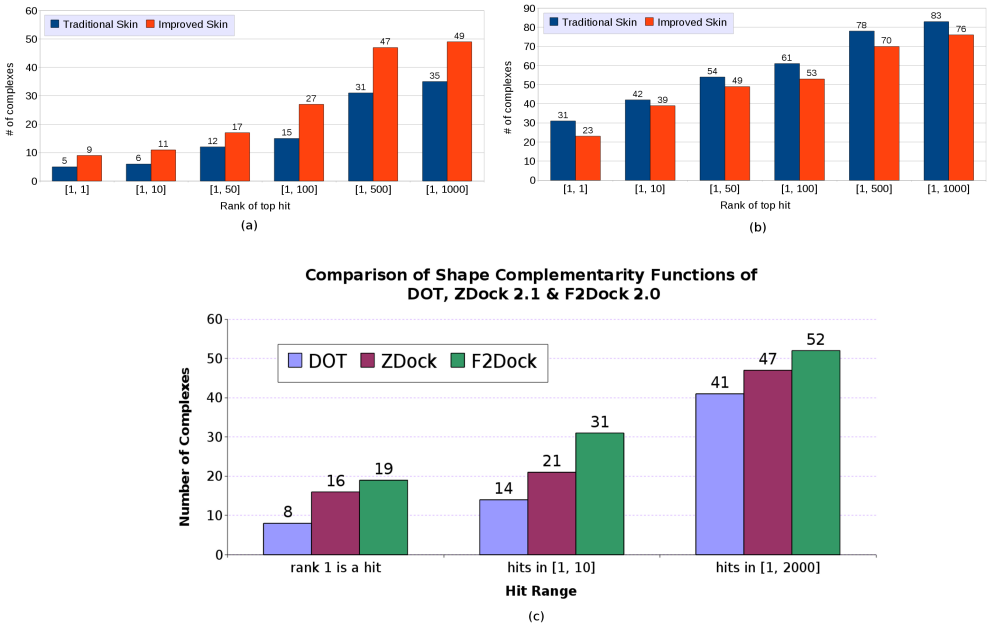


Figure 5.2: **Effectiveness of the New Skin-Core Definition**. (a-b) Comparison of the performance of F^{2}Dock 2.0's shape complementarity function with traditional skin and the new floating skin approach, in terms of the number of complexes for which the top hit is within the ranges mentioned in the X-axis. (a) On the rigid-body unbound-(un)bound complexes from Zlab Benchmark 4.0. (b) On the rigid-body bound-bound complexes from Zlab Benchmark 4.0. (c) Comparison of the shape complementarity functions of DOT, ZDock 2.1 and F^{2}Dock 2.0 on the rigid-body bound-bound complexes from Zlab benchmark 2.0.

5.4.2.2 Effects of Various Filters on Quality of Solutions

Figure 5.3(top) shows how the number of test cases (rigid-body test cases from Zlab benchmark 2.0 [180]) with at least one hit in top 1, top 10, top 50, top 100, top 500 and top 1000 changes as various affinity functions and filters in F^{2}Dock 2.0 are applied. The filters are applied to the top 2000 predictions after using the FFT based affinity terms and clustering. In this experiment, we have specified the

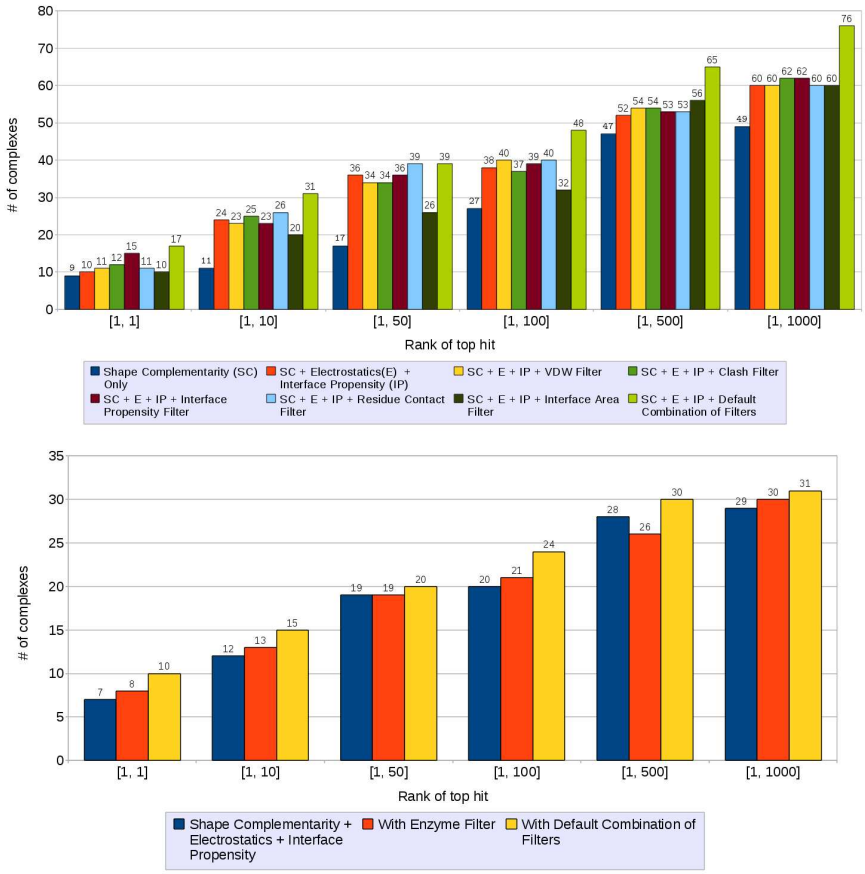


Figure 5.3: **Analysis of the efficacy of the different filters and affinity terms used in F^{2}Dock 2.0. (top)** Improvements in the rank of the top hit (of rigid-body test cases from Zlab benchmark 4.0) as various affinity functions and filters in F^{2}Dock 2.0 are activated one after another. **(bottom)** Improvements in the rank of the top hit for the Enzyme type of complexes from Zlab benchmark 4.0.

complex type (A/AB, E and O) for each test case. Clearly, each of the filters (except interface area filter) individually improves the ranks of the top solution, and the best outcome is generated when the default combination of filters are used. For example, after the FFT based scoring, we get a hit at rank 1 for 10 complexes, but

after filtering it improves to 17. Since the antibody and enzyme filters do not apply to all types of complexes, we compare their effect only on the particular type of complexes. For example, Figure 5.3(bottom) shows the effectiveness of the enzyme filter.

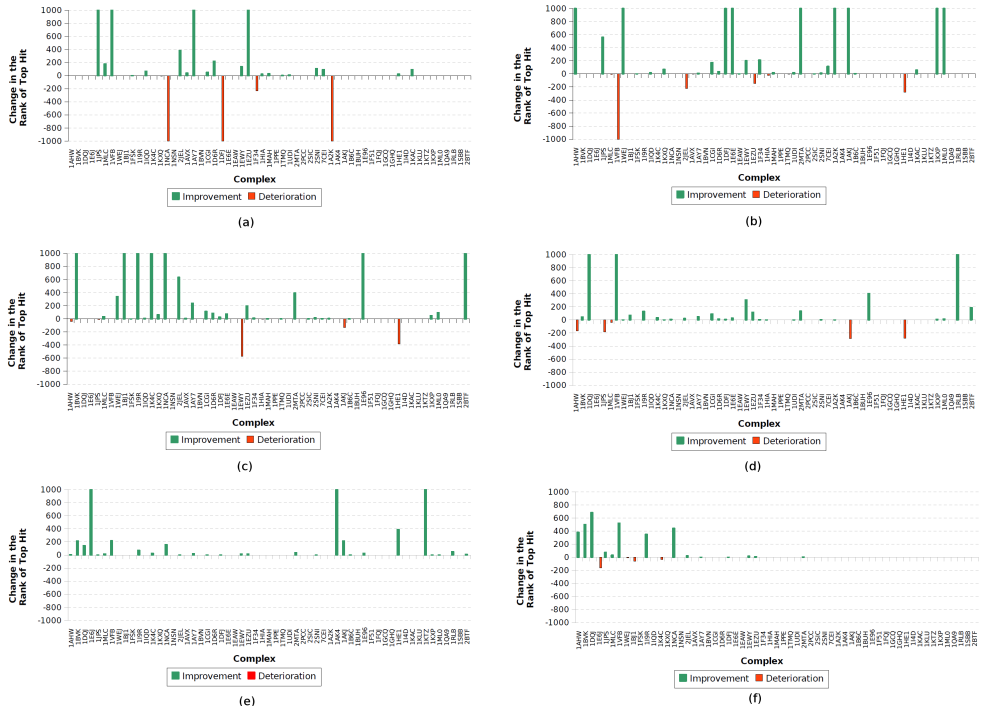


Figure 5.4: **Changes in the rank of top hit as various options in F^{2}Dock 2.0 are activated one after another (on the rigid-body test cases from Zlab benchmark 2.0 [180]).** **(a)** Lennard-Jones Filter (LJ), clash filter (CL) and proximity clustering (PC) are activated after shape complementarity (SC), **(b)** electrostatics & charge complementarity (EL) after SC+LJ+CL+PC, **(c)** interface propensity (IP) after SC+LJ+CL+PC+EL. **(d)** interface propensity filter (PF) after SC+LJ+CL+PC+EL+IP, **(e)** residue-residue contact filter (RC) after SC+LJ+CL+PC+EL+IP+PF, and **(f)** antibody contact filter (AF) or glycine filter (GF) after SC+LJ+CL+PC+EL+IP+PF+RC.

The series of plots in Figure 5.4 shows a detailed breakdown of the effect of different scores/filters for each complex separately. On the X-axis, we list the complexes and the Y-axis shows the change of the rank of the top hit. In the figures, an improvement is defined as producing the top hit at a better rank. We use the results of using just shape complementarity as the base case and analyze the relative improvements as more and more terms are added.

When we activate Lennard-Jones filter, clash filter and proximity clustering after shape complementarity we get hits for 4 new test cases, and the rank of the top hit improves for 15 more (see Figure 5.4(a)). However, we also lose hits in top 1000 for 3 test cases, and the rank of the top hit degrades for one test case. Overall, the application of these filters and clustering seem largely beneficial. The best results are obtained for enzyme-inhibitor/enzyme-substrate complexes, as for more than 50% of these complexes rank of the top hit improves.

When electrostatics is turned on we get hits in top 1000 for 9 test cases for which we did not have a single hit before, and for 14 other cases rank of the top hit improve (see Figure 5.4(b)). However, we lose hits 1 test case, and for 4 others rank of the top hit degrades.

The FFT-based interface propensity scoring is activated next which improves the rank of the top hit for 30 test cases (i.e., for around 50% of all cases) among which 7 cases did not have a single hit before (see Figure 5.4(c)). Among these 7 cases with new first hits 5 are antibody-antigen or antigen-bound antibody complexes, and none are enzyme-inhibitor or enzyme-substrate.

The interface propensity filter is turned on next. It improves the rank of the top hit for 25 complexes, and degrades for 5 (see Figure 5.4(d)). For 3 test cases we did not have a single hit in top 1000 before among which 2 are antibody-antigens.

The residue-residue contact filter which is activated next improves the rank of the top hit for 27 test cases, and degrades for none (see Figure 5.4(e)). The enzyme-inhibitor and enzyme-substrate complexes seem to have benefited the least from this filter.

Next we apply the antibody contact filer and the Glycine filter. The antibody contact filter improves the rank of the top hit for 9 antibody-antigen and antigen-bound antibody test cases, and degrades for 3, while the Glycine filter slightly improves the same for 4 enzyme-inhibitor/enzyme-substrate complexes (see Figure 5.4(f)).

More comparisons with respect to the RMSD of the best hit, the total number of hits, and the lowest RMSD are provided in the supplement.

5.4.2.3 Effects of Post-processing with GB-rerank

Figure 5.5 shows the impact of applying GB-rerank (after the initial docking phase) on the rigid-body test cases from Zlab benchmark 2.0 [180]. GB-rerank improves the ranks of the top hit for 9 antibody-antigen and antigen-bound antibody complexes, and 10 complexes of type "other" (see Figure 5.5).

The post-processor is least effective on enzyme-inhibitor/enzyme-substrate complexes since the enzyme filter has already improved the ranks quite well. On

Figure 5.5: **Effect of performing GBSA based reranking.** The plot shows the change of the rank for the first hit. A positive change indicates that the reranker improves the result. For most complexes, specially for complexes where a knowledge-based based filter (Antibody or Enzyme) could not be applied, GB-rerank improves the rank of top hit compared to the results produced by F^{2}Dock 2.0 (for the rigid-body test cases from Zlab benchmark 2.0 [180]).

the other hand, for the 'other' complexes, GB-rerank produces the most significant improvements, since specific filters cannot be applied in these cases. Hence if the complex is known to be Enzyme, then GB-rerank should not be applied.

5.4.2.4 Performance of F^{2}Dock 2.0 with and without User-specified Complex Type

Figure 5.6 compares the performance of F^{2}Dock 2.0 with and without user-specified complex types on Zlab's protein-protein docking benchmark 2.0. When no complex type is specified F^{2}Dock 2.0 tries to identify antibody-antigen complexes by locating the CDR loop regions of the antibody. Among the 17 such

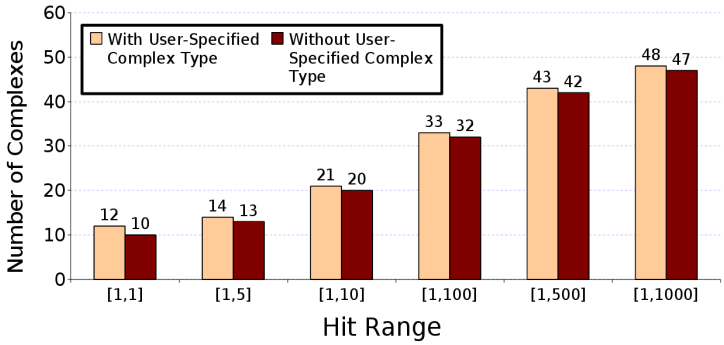


Figure 5.6: **Performance of F^{2}Dock 2.0 with and without user-specified complex type**. When complex type is not specified in the input, F^{2}Dock 2.0's performance does not change significantly. In most cases, it can automatically detect the complex-type and apply the correct set of parameters. Tests are based on rigid body cases from Zlab's Protein-protein docking Benchmark 2.0.

complexes in our experiments 16 are correctly identified by F^{2}Dock 2.0. It fails to identify 1KXQ which is an antibody-antigen complex from a Camelid (camels, llamas, etc.) [77]. Camelids produce functional antibodies that do not have light chains and CH1 domains, and so F^{2}Dock 2.0's antibody detection system fails to identify such antibodies. Hence for 1KXQ the set of parameter values optimized for complexes of "other" type is applied, and the result is only slightly worse than what is obtained with the parameter set optimized for antibody-antigen complexes. F^{2}Dock 2.0 fails to select the correct parameter set for the following three enzyme-inhibitor/enzyme-substrate complexes among the 21 included in the experiments: 1AY7, 1UDI and 2MTA. While for 1UDI and 1AY7 F^{2}Dock 2.0 is still able to get a hit in the top 100 and top 500, respectively, it fails to get any hit in the top 1000 for 2MTA. For all other complexes the results remain the same except for 1WEJ for

which we get slightly different results in the two set of experiments due to the non-determinism (arising from multiple concurrent threads) that exists in the proximity clustering phase.

5.4.3 Comparison with ZDock

In this section we compare the performance of F^{2}Dock 2.0 and ZDock 3.0.2 [179, 199] on the complexes from Zlab benchmark 4.0 [129]. We acquired the executable for ZDock 3.0.2 from their website and ran it following the steps specified in the accompanying instructions and used the PDB files downloaded from ZLab's website without any modification. F^{2}Dock 2.0 used the same set of PDBs after performing the preprocessing we mentioned in Section 5.3.3. Note that ZDock 3.0.2 also applies their own preprocessing which is part of the mark_sur script provided with the executable. Both programs used 15° rotational sampling. F^{2}Dock 2.0 used user-specified complex types.

In Figure 5.7, we show a summary of the performances in terms of the number of complexes where each protocol found at least one hit in different ranges (see the X-axis). Note that having a higher Y-axis value for any instance shows that the corresponding protocol is successful on complexes than the other. In Figure 5.7(a) we compare the performances over the entire Zlab benchmark 4.0 containing 176 complexes. We find that for each of the ranges except one, F^{2}Dock 2.0 performs better than ZDock 3.0.2. F^{2}Dock 2.0 is specially impressive since it gets a hit at rank 1 for 22 of the complexes (which is 1/8th of the dataset) as opposed to 13 found by ZDock 3.0.2. Overall both ZDock 3.0.2 and F^{2}Dock 2.0 finds at least one solution

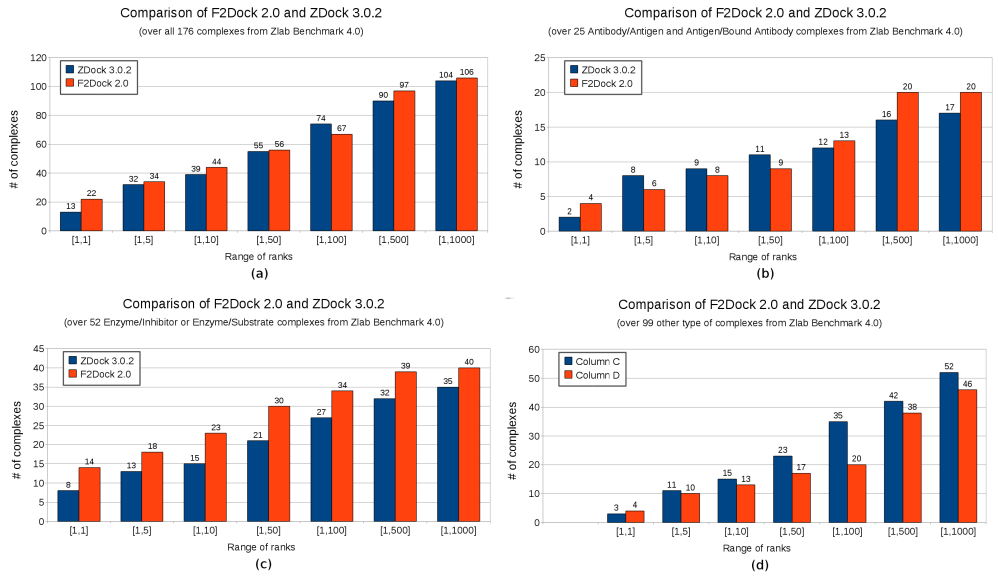


Figure 5.7: **Comparison of ZDock 3.0.2 [179] and F^{2}Dock 2.0.** (a) On all 176 complexes from Zlab Benchmark 4.0 [129], (b) On 25 antibody-antigen and antigen-bound antibody complexes, (c) On 52 enzyme-inhibitor and enzyme-substrate complexes, and (d) on the 99 other type of complexes.

for about the same number of complexes, 104 and 106 respectively.

Figures 5.7(b)-(d) compares F^{2}Dock 2.0 and ZDock 3.0.2 using the same metrics but considers each type of complex separately. For antibodies there is not much to choose between the two protocols. For other types F^{2}Dock 2.0 is successful for a lower number of complexes, and is comparable only at relatively high ranks. However, for Enzymes, F^{2}Dock 2.0 completely outperforms ZDock 3.0.2 across the board.

Based on these results, we can clearly see that F^{2}Dock 2.0 produces much more reliable predictions for Enzymes, but there is not much difference for antibodies and other type of complexes. But Tables 5.1 to 5.4 shows that even for anti-

bodies and other types F^{2}Dock 2.0 provides significant contributions since the two protocols are often successful for different complexes and hence compliment each other. For example, among the antibodies, F^{2}Dock 2.0 finds a solution for 1QFW and 1I9R for which ZDock 3.0.2 does not find any solutions, on the other hand ZDock 3.0.2 finds a solution for 1NSN where F^{2}Dock 2.0 fails. Similarly among the other complexes, only F^{2}Dock 2.0 is successful for 1J2J, 2A5T, 2A9K, 2HQS, 3BP8, 1K5D, 1R6Q, 2Z0E, 3CPH and 1ATN. Hence, it is advisable to use both of these protocols specially for other type of complexes to increase the possibility of finding a correct solution.

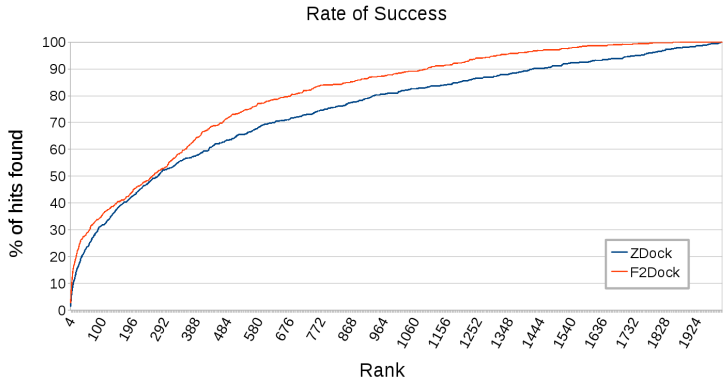


Figure 5.8: **Comparison of the rate of success of F^{2}Dock 2.0 and ZDock 3.0.2**. On the 176 complexes from ZLab's benchmark 4.0. Rate of success is defined as the percentage of the hits found within the top x ranks, where x is the corresponding value of the X-axis. Clearly F^{2}Dock 2.0 has a better ratio.

Next we compare the rate of success of the two protocols. Let us assume that the total number of hits (counting multiple hits found for a complex) found within a range $[0, x]$ across all the complexes be $H(x)$. Now we define the rate of success as $y(x) = H(x) * 100/H(1000)$ which measures how quickly a protocol

finds its hits. A protocol with a higher ratio has higher true positive rate near the top of the list. If we plot this function, we expect to see a curve which rises sharply and then gradually flattens and converges to $y(x) = 1.0$. In Figure 5.8, we see that F^{2}Dock 2.0 has consistently better success rate than ZDock 3.0.2.

A closer look at Tables 5.1 to 5.4 shows that the RMSDs of the predictions by F^{2}Dock 2.0 is poorer than ZDock 3.0.2 in more occasions than it is better. This is due to our softer skin approach which rewards docking poses which have slightly larger gap between them, and our stringent clash and VDW filters which discard ligand poses which comes too close. This is beneficial for unbound complexes with larger conformational change, but prevents ligands of rigid (easy cases in the benchmark) from getting as close as they could be placed. The result clearly shows that ZDock 3.0.2 gets better RMSDs for rigid cases, and F^{2}Dock 2.0 is better for non-rigid cases. At this point, it should be mentioned that F^{2}Dock 2.0 is designed solely as a initial stage docking tool, which can quickly perform exhaustive search and return good leads at high ranks. Hence the poses it finds are generally acceptable or medium quality as defined in the criteria used in the CAPRI [131] challenge (tables summarizing F^{2}Dock 2.0's performance using the CAPRI criteria can be found in the supplement). Local refinements (rigid body or flexible) can then be performed on a small number of top solutions to further improve their RMSDs and minimize the energies. There are a host of such tools available including ROSETTA [73], Amber [54], FireDock [8] etc.

We conclude this section with the observation that F^{2}Dock 2.0 shows better overall performance, with significant improvement for Enzymes. For other type of

complexes the performance is comparable and sometimes complementary.

5.4.4 Running Times

To evaluate the average running times and the relative consumption by each scoring term/filter we performed a set of experiments run on a 3 GHz $2\times$dual-core (i.e., 4 cores) AMD Opteron 2222 processor with 4 GB RAM. On average, the FFT phase took around 23 minutes or 35% of the total running time, the interface propensity filter took 20%, GB-rerank accounted for around 42%, and the remaining 3% is spent on the other filters. GB-rerank and interface propensity filter take longer to compute than other filters, since the computation is based on surface quadrature points, whose number is a constant multiple of the number of atoms. Figure 5.9 shows how the different components of F^{2}Dock 2.0 and GB-rerank contribute to the total running time of the docking and reranking process on the rigid-body test cases from Zlab benchmark 2.0 [180]. Overall, about 30% time is taken up by the FFT based affinity functions, 30% is taken up the the filters (mostly the interface propensity filter), and around 40% by the GB-rerank.

F^{2}Dock 2.0 leverages from the embarrassingly parallel nature of the computation using multithreaded computations on multi-core machines. Note that each of the N_R FFT computations are independent of each other and can be run in parallel. Scores for each of filter terms for each of the poses in Q can also be computed in parallel. Specifically, given q cores and T tasks the simplest strategy is to distribute T/q tasks to each cores. But this approach often leads to unbalanced exploitation of the cores if the tasks given to different cores take different amount of time to

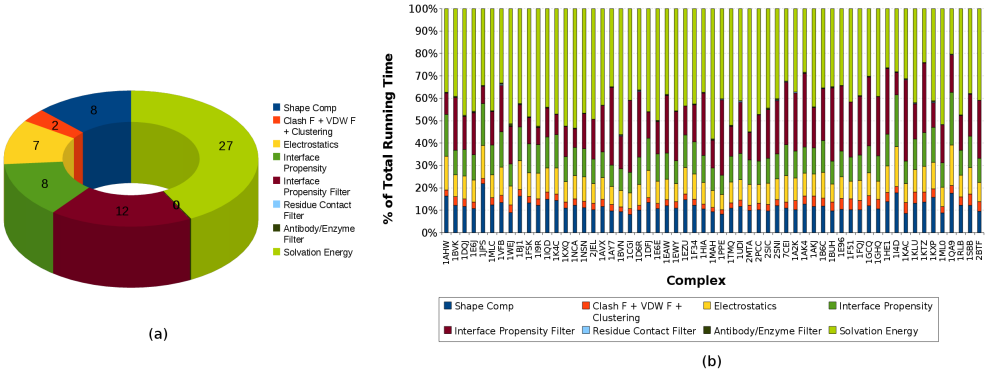


Figure 5.9: **Running time of F^{2}Dock 2.0 and its components. (a)** Average running time of each affinity function and filter of F^{2}Dock 2.0. GB-rerank consumes a major portion of the time (42%), the FFT phase takes about 30% time and the rest is taken by the filters and clustering. The labels in the figure are actual time in minutes. **(b)** Running times of F^{2}Dock 2.0 on the rigid-body test cases from Zlab benchmark 2.0 [180] showing percentage of running time due to each affinity function and filter of F^{2}Dock 2.0 for each complex.

complete. For example, the running times of the filters are proportional to the size of the interface which varies between different poses. So our technique initially sends only one task to each core and maintains a queue of remaining tasks, and then whenever a core is done with its task, it gets another one from the queue. This scheduling ensures that every core is exploited equally and hence the overall completion time is quicker.

5.5 Conclusion

We have developed an enhanced version (F^{2}Dock 2.0) of our protein-protein docking program F^{2}Dock 2.0 with improved scoring functions, complete with dynamic clustering and filtering and generalized Born based solvation energetic rerank-

ing. The on-the-fly FFT-based scoring function is a weighted combination of shape-complementarity, Coulombic electrostatics complementarity, and interface propensity terms. The on-the-fly docking also includes filters based on Lennard-Jones potential, steric clashes, residue-residue contact statistics and an extremely fast approximation of solvation energy using a newly developed fast multipole type implementation with octree data structures. Our implementation results and numerous tests show that each of these terms and filters significantly improves the accuracy of docking predictions. Our use of highly efficient data structures including the dynamic packing grids for near constant time neighborhood search and near-far distance clustering using octrees, significantly speed up the computations for each of the 'on-the-fly' scoring and filtering terms. GB-rerank's solvation energy based post-processing suite is also optimized using these efficient data structures with the best tradeoffs of docking accuracy vs. speed. The entire software is highly parallel and can be run efficiently on multicores and clusters of multicores (e.g., many modern supercomputers). We have also developed a GUI based interface (TexMol) for easily preparing and running a docking process and interactively visualize, compare different solutions along with several relevant statistics including interface area, residue contacts, binding energy etc.

Difficulty	Complex	Rank of First Hit		RMSD of First Hit		Rank of Lowest RMSD		Lowest RMSD	
		ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock
Easy	1AHW	354	**8**	4.5	**4.4**	1242	**457**	**0.9**	1.8
	1BJ1	**1**	63	**1.9**	2.3	**1**	63	**1.9**	2.3
	1BVK	**184**	205	**3.6**	4.9	358	**264**	**1.9**	4.1
	1DQJ	374	**74**	**4.0**	4.9	1787	**74**	**3.3**	3.6
	1E6J	**3**	126	**4.1**	5	181	**126**	**2.7**	5
	1FSK	1	1	**2.9**	3.2	**2**	3	1.8	**1.5**
	1I9R	-	**9**	-	**3.9**	-	**9**	-	**3.9**
	1IQD	18	**4**	4.3	**3**	68	**4**	**1.7**	3
	1JPS	1266	**186**	**2.1**	2.7	1266	**186**	**2.1**	2.7
	1K4C	583	**105**	**2.9**	4.4	583	**165**	2.9	**2.2**
	1KXQ	2	**1**	**1.2**	1.6	2	**1**	**1.2**	1.6
	1MLC	57	**11**	**2.0**	3.8	**57**	114	2.0	**1.3**
	1NCA	**11**	168	**1.7**	3.7	**11**	168	**1.7**	3.7
	1NSN	**1267**	-	**1.6**	-	**1267**	-	**1.6**	-
	1QFW	-	**80**	-	**1.9**	-	**80**	-	**1.9**
	1VFB	250	**191**	**3.1**	4.8	560	**434**	**2.9**	3.4
	1WEJ	9	**5**	**1.5**	3.2	9	**5**	**1.5**	3.2
	2FD6	**3**	62	5.0	**4.4**	282	**62**	**3.3**	4.4
	2I25	**2**	122	**3.0**	3.9	**40**	242	**1.7**	2.6
	2JEL	4	**1**	3.5	**3.3**	753	**1**	**2.6**	3.3
	2VIS	-	-	-	-	-	-	-	-
	9QFW	2	**1**	4.0	**3.9**	48	**3**	**1.9**	2.9
Medium	1BGX	-	-	-	-	-	-	-	-
Hard	1E4K	-	-	-	-	-	-	-	-
	2HMI	-	-	-	-	-	-	-	-

Table 5.1: Comparison of the performance of F^2Dock 2.0 and ZDock 3.0.2 for each of the 25 antibody-antigen and antigen-bound antibody complexes from ZLab's benchmark 4.0 in terms of the rank and RMSD of the top hit and the best hit. Bold-faced entries indicate better performance on the particular metric for the complex. Empty entries indicate that no hits were found for that complex by the corresponding protocol.

Difficulty	Complex	Rank of First Hit		RMSD of First Hit		Rank of Lowest RMSD		Lowest RMSD	
		ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock
Easy	1AVX	25	**1**	**3.5**	4.5	194	**4**	**1.5**	2
	1AY7	577	**2**	**2.5**	4	577	**6**	**2.5**	2.5
	1BVN	3	**1**	**1.2**	3.2	3	**2**	**1.2**	3
	1CGI	**10**	76	4.0	**3.4**	**173**	199	**2.6**	3.3
	1CLV	3	**1**	**2.3**	2.5	**21**	350	2.3	**2.1**
	1D6R	-	**59**	-	**4.6**	-	**249**	-	**4.3**
	1DFJ	**1**	9	**4.1**	4.4	**2**	9	**3.2**	4.4
	1E6E	**5**	20	**3.2**	4.7	**10**	20	**1.5**	3.9
	1EAW	68	**1**	3.4	**1**	579	**1**	1.7	**1**
	1EWY	53	**14**	4.2	**3.2**	231	**14**	3.6	**3.2**
	1EZU	841	**170**	4.9	**4.5**	**841**	1554	4.9	**3.8**
	1F34	62	**2**	**3.4**	4.3	925	**3**	**2.4**	3.7
	1FLE	31	**3**	5.0	**3.7**	1102	**192**	3.4	**3**
	1GL1	**73**	326	**2.6**	3.8	**73**	881	2.6	**2.3**
	1GXD	**1173**	-	**4.9**	-	**1173**	-	**4.9**	-
	1HIA	-	**18**	-	**3.4**	-	**258**	-	**2.2**
	1JTG	**1**	7	**2.6**	4.6	1	**1173**	**2.6**	3.4
	1MAH	1	1	3.1	**2.7**	4	4	**1.4**	1.9
	1N8O	**7**	11	**3.4**	4.8	**20**	1330	**0.6**	4
	1OC0	**1590**	-	4.8	-	**1590**	-	**4.8**	-
	1OPH	**1694**	-	**3.9**	-	**1694**	-	**3.9**	-
	1OYV	15	**7**	4.9	**3.6**	153	**105**	3.3	**2.9**
	1PPE	1	1	2.9	**2.3**	3	3	**1.1**	1.3
	1R0R	138	**39**	**2.2**	4.3	1298	**1164**	2.0	**1.4**
	1TMQ	16	**1**	**3.6**	4.8	885	**515**	3.0	**2.4**
	1UDI	24	**1**	3.5	**3.1**	**24**	229	3.5	**2.5**
	1YVB	**1**	-	**2.4**	-	**18**	-	**2.2**	-
	2ABZ	-	**5**	-	**2.8**	-	**5**	-	**2.7**
	2B42	3	**1**	4.2	**3.9**	**6**	12	**0.6**	2.2
	2J0T	-	**19**	-	**2.8**	-	**21**	-	**2.6**
	2MTA	**76**	90	4.4	**4.3**	716	**100**	**0.7**	3.7
	2O8V	**29**	654	5.0	**3.7**	852	**654**	4.0	**3.7**
	2OUL	1	1	**1.7**	4.9	**1**	329	**1.7**	3
	2PCC	496	**10**	**2.6**	4.3	496	**10**	**2.6**	4.3
	2SIC	5	**1**	1.1	**1.1**	5	**1**	1.1	**1.1**
	2SNI	177	**1**	**3.8**	4.7	**299**	403	2.8	**1.3**
	2UUY	693	**7**	4.4	**4.1**	1946	**44**	3.1	**3**
	3SGQ	428	**110**	4.0	**2.6**	**576**	624	**1.0**	2
	4CPA	1	1	**4.4**	4.8	465	**202**	2.5	**2.4**
	7CEI	1	1	4.4	**4.1**	88	**2**	**0.8**	1.4
	BOYV	-	**220**	-	**3.6**	-	**220**	-	**3.6**
Medium	1ACB	126	**22**	4.4	**3.2**	393	**49**	**2.6**	2.6
	1IJK	**81**	88	**3.0**	4.8	1317	**142**	**2.0**	3.4
	1JIW	-	-	-	-	-	-	-	-
	1KKL	-	-	-	-	-	-	-	-
	1M10	-	-	-	-	-	-	-	-
	1NW9	-	**321**	-	**4.9**	-	**321**	-	**2.3**
Hard	1F6M	-	-	-	-	-	-	-	-
	1FQ1	-	-	-	-	-	-	-	-
	1PXV	-	-	-	-	-	-	-	-
	1ZLI	-	-	-	-	-	-	-	-
	2O3B	-	-	-	-	-	-	-	-

Table 5.2: Comparison of the performance of F^2Dock 2.0 and ZDock 3.0.2 for each of the 52 enzyme-inhibitor and enzyme-substrate complexes from ZLab's benchmark 4.0 in terms of the rank and RMSD of the top hit and the best hit.

Difficulty	Complex	Rank of First Hit		RMSD of First Hit		Rank of Lowest RMSD		Lowest RMSD	
		ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock
Easy	1A2K	1348	**44**	4.3	**2.4**	1894	**44**	3.2	**2.4**
	1AK4	1090	**964**	**3.5**	4.3	1090	**964**	**3.5**	4.3
	1AKJ	546	**39**	**2.9**	3.4	1632	**39**	**1.7**	3.4
	1AZS	**42**	-	**2.9**	-	**61**	-	**2.0**	-
	1B6C	**1**	3	**2.9**	4	**1**	3	**2.9**	4
	1BUH	**30**	431	**3.6**	4.8	1961	**431**	**3.0**	4.8
	1E96	1171	**278**	**3.8**	5	1171	**314**	**3.8**	4.2
	1EFN	-	-	-	-	-	-	-	-
	1F51	**589**	-	**4.6**	-	**589**	-	**4.6**	-
	1FC2	-	**1190**	-	**5**	-	**1570**	-	**4.1**
	1FCC	-	-	-	-	-	-	-	-
	1FFW	**73**	325	**4.5**	4.7	1349	**1291**	4.0	**3**
	1FQJ	-	-	-	-	-	-	-	-
	1GCQ	**1105**	-	**1.4**	-	**1105**	-	**1.4**	-
	1GHQ	-	-	-	-	-	-	-	-
	1GLA	**1708**	-	**3.9**	-	**1708**	-	**3.9**	-
	1GPW	3	**1**	**3.6**	3.7	134	**5**	**2.1**	2.6
	1H9D	**1006**	-	**4.5**	-	**1006**	-	**4.5**	-
	1HCF	**175**	1225	**4.0**	4.7	**225**	1225	**1.9**	4.7
	1HE1	1141	**574**	**4.7**	4.9	1141	**574**	**4.7**	4.9
	1I4D	**571**	-	**4.2**	-	**571**	-	**4.2**	-
	1J2J	-	**182**	-	**4.6**	-	**182**	-	**4.6**
	1JWH	**7**	-	**3.6**	-	**78**	-	**1.9**	-
	1K74	**2**	3	**1.2**	3.8	**2**	7	**1.2**	2.6
	1KAC	592	**8**	4.5	**4.4**	1527	**99**	**1.9**	4.1
	1KLU	**1957**	-	**3.4**	-	**1957**	-	**3.4**	-
	1KTZ	535	**98**	**2.8**	3.9	535	**166**	**2.8**	2.9
	1KXP	**1**	7	**1.6**	3.5	**1**	260	**1.6**	2.6
	1ML0	4	**2**	**3.1**	4.3	**8**	123	**3.1**	3.2
	1OFU	**84**	-	**4.5**	-	**347**	-	**3.1**	-
	1PVH	**748**	-	**4.5**	-	**1192**	-	**1.5**	-
	1QA9	-	-	-	-	-	-	-	-
	1RLB	**3**	555	**4.6**	5	**232**	555	**3.4**	5
	1RV6	2	2	**1.3**	4	**2**	694	**1.3**	2.2
	1S1Q	**756**	-	**1.9**	-	**1243**	-	**1.4**	-
	1SBB	-	-	-	-	-	-	-	-
	1T6B	**58**	525	**3.6**	4	1510	**752**	2.8	**2.7**
	1US7	**74**	-	**1.1**	-	**74**	-	**1.1**	-
	1WDW	2	**1**	**1.2**	2.5	2	**1**	**1.2**	2.5
	1XD3	8	**1**	**4.0**	4.2	**86**	1298	**2.6**	3.9
	1XU1	**912**	-	**5.0**	-	**912**	-	**5.0**	-
	1Z0K	**8**	307	**3.3**	3.3	**8**	307	**3.3**	3.3
	1Z5Y	**20**	-	**3.4**	-	423	-	2.5	-
	1ZHH	-	-	-	-	-	-	-	-
	1ZHI	**65**	202	4.4	**4**	324	**202**	**2.1**	4
	2A5T	-	**268**	-	**3.6**	-	**618**	-	**2.9**
	2A9K	-	**558**	-	**3.4**	-	**558**	-	**3.4**
	2AJF	**475**	-	**3.6**	-	**475**	-	**3.6**	-
	2AYO	**37**	1108	3.3	**2**	**138**	1108	2.5	**2**
	2B4J	-	-	-	-	-	-	-	-
	2BTF	**53**	95	4.7	**4.5**	**148**	377	3.8	**3.4**
	2FJU	261	**228**	**3.2**	4.2	**261**	333	**3.2**	3.5

Table 5.3: Comparison of the performance of F^{2}Dock 2.0 and ZDock 3.0.2 for each of the 99 other type of complexes from ZLab's benchmark 4.0 in terms of the rank and RMSD of the top hit and the best hit. Continued as Table 5.4.

Difficulty	Complex	Rank of First Hit		RMSD of First Hit		Rank of Lowest RMSD		Lowest RMSD	
		ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock	ZDock	F2Dock
Easy	2G77	15	**8**	**1.5**	3.7	**15**	917	1.5	**1.3**
Cont	2HLE	31	**4**	4.1	**3.8**	31	**4**	4.1	**3.8**
	2HQS	-	**27**	-	**4.1**	-	**125**	-	**2.7**
	2OOB	-	-	-	-	-	-	-	-
	2OOR	766	**16**	4.4	**4**	766	**63**	4.4	**2**
	2VDB	**5**	-	**1.2**	-	**5**	-	**1.2**	-
	3BP8	-	**474**	-	**5**	-	**699**	-	**3.3**
	3D5S	71	**1**	**3.1**	3.2	609	**4**	**2.5**	2.7
Medium	1GP2	**61**	193	4.4	**4**	**107**	193	**2.8**	4
	1GRN	1299	**401**	**4.3**	4.8	1299	**401**	**4.3**	4.8
	1HE8	-	-	-	-	-	-	-	-
	1I2M	**267**	545	**2.2**	2.6	**267**	545	**2.2**	2.6
	1IB1	-	-	-	-	-	-	-	-
	1K5D	-	**521**	-	**4.3**	-	**521**	-	**4.3**
	1LFD	**85**	990	4.6	**4.6**	**466**	1235	4.5	**4.1**
	1MQ8	**1455**	-	**3.2**	-	**1455**	-	**3.2**	-
	1N2C	-	-	-	-	-	-	-	-
	1R6Q	-	**180**	-	**3.7**	-	3**11**	-	**3.5**
	1SYX	211	**2**	4.8	**4.7**	211	**11**	4.8	**3**
	1WQ1	**81**	-	**4.0**	-	**81**	-	**4.0**	-
	1XQS	**19**	61	**3.8**	4.2	**45**	833	**2.6**	3.7
	1ZM4	**6**	-	**4.1**	-	**631**	-	**2.7**	-
	2CFH	**1**	119	3.8	**2.6**	**2**	119	**1.7**	2.6
	2H7V	**1112**	-	**4.6**	-	**1112**	-	**4.6**	-
	2HRK	**3**	-	**3.7**	-	**3**	-	**3.7**	-
	2J7P	-	-	-	-	-	-	-	-
	2NZ8	**64**	-	**4.5**	-	**64**	-	**4.5**	-
	2OZA	-	-	-	-	-	-	-	-
	2Z0E	-	**169**	-	**3.9**	-	**169**	-	**3.9**
	3CPH	-	**250**	-	**4.3**	-	**250**	-	**4.3**
Hard	1ATN	-	**1307**	-	**2.7**	-	**1307**	-	**2.7**
	1BKD	-	-	-	-	-	-	-	-
	1DE4	**84**	-	**4.7**	-	**84**	-	**4.7**	-
	1EER	-	-	-	-	-	-	-	-
	1FAK	-	-	-	-	-	-	-	-
	1H1V	-	-	-	-	-	-	-	-
	1IBR	-	-	-	-	-	-	-	-
	1IRA	-	-	-	-	-	-	-	-
	1JK9	510	**422**	4.2	**2.5**	790	**422**	4.1	**2.5**
	1JMO	-	-	-	-	-	-	-	-
	1JZD	**44**	144	4.6	**4**	**44**	144	4.6	**4**
	1R8S	-	-	-	-	-	-	-	-
	1Y64	-	-	-	-	-	-	-	-
	2C0L	-	-	-	-	-	-	-	-
	2I9B	-	-	-	-	-	-	-	-
	2IDO	**130**	156	**3.6**	4.5	**154**	156	**3.5**	4.5
	2OT3	**121**	-	**4.6**	-	**327**	-	**4.5**	-

Table 5.4: Comparison of the performance of F^2Dock 2.0 and ZDock 3.0.2 for each of the 99 other type of complexes from ZLab's benchmark 4.0 in terms of the rank and RMSD of the top hit and the best hit (cont.).

Chapter 6

Characterization, Enumeration and Construction of *Almost-regular* Polyhedra

The symmetries and properties of the 5 known regular polyhedra are well studied. These polyhedra have the highest order of 3D symmetries, making them exceptionally attractive templates for (self)-assembly using minimal types of building blocks, from nanocages and virus capsids to large scale constructions like glass domes. However, the 5 polyhedra only represent a small number of possible spherical layouts which can serve as templates for symmetric assembly. In this paper, we formalize the notion of symmetric assembly, specifically for the case when only one type of building block is used, and characterize the properties of the corresponding layouts. We show that such layouts can be generated by extending the 5 regular polyhedra in a symmetry preserving way. The resulting family remains isotoxal and isohedral, but not isogonal; hence creating a new class outside of the well-studied regular, semi-regular and quasi-regular classes and their duals Catalan solids and Johnson solids. We also show that this new family, dubbed *almost-regular polyhedra*, can be parameterized using only two variables and constructed efficiently. In Chapter 7, we discuss some applications on modeling, predicting and analyzing the structures of virus capsids of various sizes.

6.1 Introduction

Regular polyhedra are combinatorial marvels. They are simultaneously isogonal (vertex-transitive), isotoxal (edge-transitive), and isohedral (face-transitive). In this context, transitivity means that every vertex (or edge or face) are congruent to each other, and for any pair of vertices (or edges or faces), there exists a symmetry preserving transformation of the entire polyhedron which isometrically maps one to the other. Note that two vertices are congruent if they have the same number of edges incident on them with the same angles between the edges; congruency for edges and faces are defined in the general way.

Transitivity plays a vital role is assembly, specially self-assembly. For instance, if we consider each face as a building block, then face transitivity indicates that a single type of block is sufficient to form a shell-like structure; and edge-transitivity indicates that there is exactly one way to put any two blocks together. Viruses, natures smallest organisms, utilize this simplicity by having only encoding the blueprint (RNA/DNA) for a single type of protein, multiple copies of which can follow a simple assembly rule to form a shell, called a capsid, within which the same RNA/DNA (and more) can be safely stored. There are only 5 regular polyhedra- the tetrahedraon, the cube, the octahedron, the dodecahedron and the icosahedron with respectively 4, 6, 8, 12, and 20 faces. But, more than half of the viruses, though they seem to have the polyhedral symmetry, have capsids which are formed by more than the number of faces of the corresponding polyhedron. In other words, the building blocks seem to be arranged in a different (denser) layout where the faces of the original polyhedra is subdivided into smaller facets. In this paper,

we refer to these families as *almost-regular* polyhedra. These polyhedra preserves the *global* polyhedral symmetry, and also introduce *local* symmetries so that only one type of building block still suffices.

Caspar and Klug [55] first addressed this layout of virus capsids, using triangular tiles, and called it 'quasi-symmetry'. A similar class of assembly is seen in fullerene like particles, with 12 pentagonal and many hexagonal faces, which was first characterized by Goldberg [108]. Recently, several researchers have developed efficient constructions and parameterizations of Goldberg-like particles [96, 125, 231]. Recently, Deng et al. [76] studied extensions of Goldberg's construction to other platonic solids, but their study is not exhaustive in characterization and enumeration of all the possible cases. In another recent work, Schein and Gayed [225] developed a numerical optimization scheme that takes a Goldberg-like polyhedra, which by construction does not have planar faces and is not always convex, and produces strictly convex polyhedra while preserving the edge-lengths. However, the resulting polyhedra no longer have any face-transitivity, and even though the edges have the same length, they are not strictly congruent as their neighboring faces are different- hence making such polyhedra unsuitable for using as layouts for assembly. In this paper, we study the class of spherical tilings which preserve global symmetry, keeps edge-lengths equal and makes the faces as congruent as possible, but relaxes the convexity condition.

We believe that our results will greatly impact research in several areas including the field of nano-materials, where the primary focus is to understand, analyze and design components at the atomic level, which can self-assemble to cre-

ate nano-structures with desirable properties. For example, gold nanorods have been used in cancer imaging and therapy [58, 275], virus capsids and protein-cages have been used for targeted drug delivery [238]. One of the emerging topics of research in nanotechnology is the engineering and design of nano-particles for specific applications. For example, designing a protein-cage or modifying a virus cage [234, 241, 244] which can contain/hold desired quantity of a drug and only dissociates when it reaches a specific organ/tissue to release the drug, and have specific tensile strength, deformability and mass. Advanced scientific computation techniques, to explore and automatically predict possible nano-structures that can be formed symmetrically by one type of building block (eg. an engineered protein) and to automatically analyze mechanical and biophysical properties of entire nano-shells involving millions of atoms etc. would surely accelerate the development of new nano-shell structures. Our theoretical groundwork would greatly support such extensive computational techniques, for instance we show that even though assembly prediction is an NP-hard problem, but using our symmetry characterization, a symmetric assembly of n particles can be predicted using a algorithm whose running time is only polynomial in n.

6.2 Characterization of *Almost-Regular* Polyhedra

First, we briefly introduce the concepts of symmetry groups, symmetry axes, order etc. Symmetry groups consists of a set of symmetry operations, i.e. transformations which maps an object to itself, such that the set is closed under the composition (one transformation followed by another) operation. The actions of the

polyhedral symmetry groups can be expressed as pure rotations around different axis through the center of the polyhedra, which we assume to be at the origin without loss of generality. We generally refer to the points of intersection of these axes with the polyhedra as the locations of symmetry, and refer to the axes as the symmetry axes. For instance, the octahedron have 6 axes of 4-fold rotational symmetry[1] going through the four vertices, 8 axes of 3-fold rotational symmetry going through the centers of the faces, and 12 axes of 2-fold rotational symmetry going thorugh the centers of the edges.

Interestingly, duals of regular polyhedra are also regular- tetrahedron is self-dual, octahedron and cube are duals of each other, and icosahedron and dodecahedron are duals of each other.

6.2.1 The *Almost-Regular* Polyhedra and Their Duals

The *almost-regular* polyhedra have exactly the same number of rotational symmetry axes with the same symmetry orders as any specific regular polyhedron, such that there exists a rigid body transformation that perfectly aligns these axes to those of the regular polyhedron. We refer to these as the *global* symmetry axes and locations. We shall refer to these global symmetry axes as gv-symmetry axes, ge-symmetry axes and gf-symmetry axes respectively for axes of symmetry going through, respectively, the vertices, edge-centers and face-centers of the regular polyhedron. Additionally, all vertices, faces and edges of an *almost-regular* poly-

[1] n-fold rotational symmetry, also referred to as the symmetry order n, means that a rotation by $2\pi/n$ maps the polyhedron to itself

hedron must have locations of *local* symmetry. Local symmetry operations map the vertices, edges, faces immediately neighboring the location of local symmetry to themselves, but may or may not map the remaining parts of the polyhedron to itself. These axes will be referred to as lv-, le-, and lf-symmetry axes.

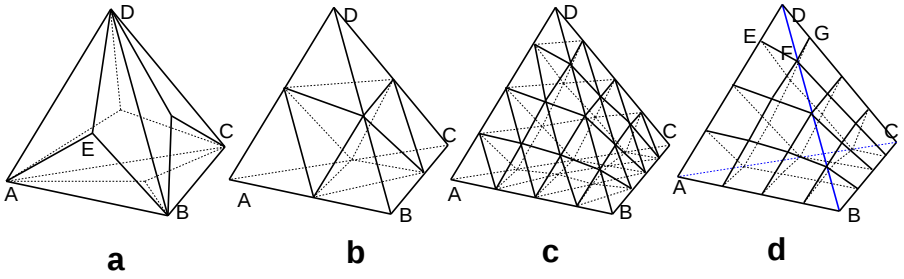


Figure 6.1: **Illustration of the criterion for almost-regular polyhedra**. The polyhedra shown in **(a)** and **(d)** are not almost-regular. In **(a)**, the global symmetries are preserved, but the local symmetries are not (for example, it is not locally 2-fold symmetric around the center of DE). In **(d)**, local symmetries are intact (note that DEFG and other creased faces are considered a single face), but the global 3-fold symmetries are not (for example, around the center of ABD).

It is clear that the almost-regular polyhedron must be isotoxal, isohedral and have only cyclic symmetric faces. However, it is not neccessarily isogonal. This class is similar to the Catalan solids, but an important distinction is that the Catalan solids allow non-symmetric faces (hence, are not isotoxal), as long as all the faces are congruent (for example, the solid in Figure 6.1(a) is a Catalan solid, but is not almost-regular.

Note that, the transivity properties that make regular polyhedra suitable for assembly, is preserved in almost-regular polyhedra, but now the class is richer and more importantly can model structures with more than 20 building blocks.

The duals of almost-regular polyhedron would have regular faces, be isotoxal and isogonal, may or may not be isohedral, and may or may not be convex. The closest known family is the semi-regular polyhedra (duals of Catalan solids) which are also isotoxal and isogonal. But the semi-regular polyhedra, which includes the 13 Archimedean solids and the family of prisms with regular faces, are always convex.

6.2.2 Related Prior Work

Our construction scheme closely follows the one proposed by Goldberg [108]. The family of polyhedra generated by the Goldberg construction rule [108] are fullerene like structures. Fullerene like structures have icosahedral symmetry (symmetry group of the icosahedron), and consists of many hexagonal faces and exactly 12 pentagonal faces. The soccer ball is the smallest example of such structure. See Figure 6.2 for an illustrative description of the construction.

Caspar and Klug [55] were studying virus capsids and their symmetric organization and inspired by Goldberg's construction, they proposed a similar approach, but using a triangular lattice, instead of a hexagonal one, and required that the corners of an edge of the unfolded icosahedron falls on the vertices of the lattice (Figure 6.3). Since, the triangular lattice is simply the dual of the hexagonal lattice, the mapping is essentially the same. But the refolded polyhedron now has only regular triangular faces. It has 12 vertices where 5 such faces are incident, and many vertices where 6 faces are incident- the first set are exactly the original vertices of the icosahedron. Notice that this polyhedron is exactly the dual of the one

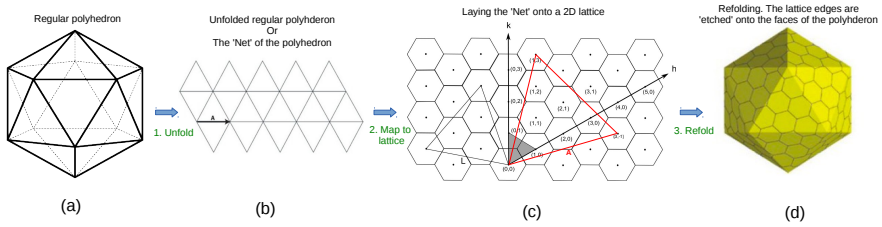
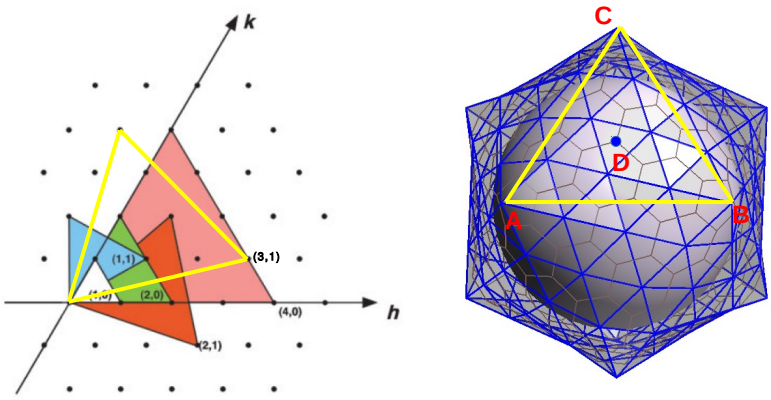


Figure 6.2: **Illustration of the Goldberg construction**. The Goldberg construction involves unfolding an icosahedron (see **(a)** and **(b)**), and then mapping the unfolded icosahedron onto a 2D hexagonal lattice scaled and oriented such that all corners of the unfolded icosahedron (its original vertices) falls on the centers of some hexagon of the grid (some example scale and orientations (for one triangle only) are shown **(c)**). Finally, the icosahedron is folded back, along with the hexagonal grid etched onto its faces. For example, for the scaling and orientation of the red triangle in **(c)**, would result in the tiled icosahedron shown in **(d)**. Notice that the new polyhedron has exactly 12 regular pentagonal faces where the icosahedral vertices originally were, and many regular hexagonal faces.

constructed using Goldberg's method (shown in Figure 6.2(d), and also overlayed in Figure 6.3(b)).

Polyhedra produced by Caspar and Klug's construction method are *almost-regular*, and the ones produced by Goldberg's are duals of *almost-regular*. But notice that both Goldberg, and Caspar and Klug focused only on the icosahedral case and considered a specific unfolding onto specific 2D lattices, and hence only covers a fraction of the possible almost-regular polyhedra and their duals. Separately, Pawley [194] studied other ways of wrapping different polyhedra using different lattices from the wallpaper group. However, Pawley did not provide any theoretical characterization of the factors related to the possibility or impossibility of such wrappings. We address this issue in the following section.



(a) (b)

Figure 6.3: **Illustration of the Caspar-Klug construction**. The Caspar-Klug construction involves unfolding an icosahedron onto a triangular lattice scaled and oriented such that all corners of the unfolded icosahedron (its original vertices) falls on the vertices of the grid (some example scale and orientations (for one triangle only) are shown **(a)**). Then, the icosahedron is folded back, along with the grid etched onto its faces. For example, for the scaling and orientation of the yellow triangle in **(a)**, would result in the tiled icosahedron shown in **(b)**.

6.2.3 Characterizing All Possible *Almost-Regular* Polyhedra

Both Goldberg and Caspar-Klug constructions can be expressed as unfolding a regular polyhedron onto a 2D lattice and then refolding it with the lattice etched onto its faces. Pawley's wrapping idea is equivalent. We call these procedures *unfold-etch-refold* method. Here, we prove the conditions that must be satisfied to produce almost-regular polyhedra using the *unfold-etch-refold* idea for any regular solid, unfolded in any way, onto any 2D lattice.

Shepherd's conjecture [237] states that all convex polyhedra have a non self-overlapping planar unfolding with only edge-cuts. This conjecture is not proved

or disproved yet for all possible convex polyhedra, however for the set of special classes we are interested in, it is true. Hence, in principle it is possible to unfold one such polyhedra and lay it down on a 2D grid, use the grid to draw tiles of the unfolded polyhedron, and then fold it back up to get a tiled polyhdron. However, every polyhedron actually have many unfolding. For example, isosahedron have 43380 unique unfoldings. Caspar and Klug's construction produced almost-regular polyhedra using 1 such unfolding, but it is not clear whether other unfoldings would also produce similar *almost-regular* polyhedra, or different types of *almost-regular* polyhedra, or not be *almost-regular*. To address this question, we characterize the relationship of the local and global symmetries of the almost-regular polyhedra, and the etched polyhedra (henceforth called *tiling*) produced using unfold-etch-refold construction.

First of all, we prove that the lattice onto which the polyhedron is unfolded must be regular.

Lemma 6.2.1. *The polyhedra generated by an unfold-etch-refold using any regular polyhedra and unfolded in any way, cannot be* almost-regular *if a non-regular grid/lattice is used.*

Proof. The unfold-etch-refold essentially wraps the lattice/grid over a regular polyhedron, thereby vertices, edges and faces of the regular polyhedron is suppressed, and a new set of vertices, edges and faces appear, all of which belong to the lattice. Now, the *local* symmetry condition of the almost-regular polyhedron requires that every face be symmetric around its center and be congruent to each other. This

cannot be satisfied if the lattice itself was not regular nor symmetric around some points. ■

There are exactly 3 regular lattices in 2D- the square lattice, the triangular lattice and the hexagonal lattice. The square lattice have 4-fold rotational symmetries at each vertex and face-center, the triangular lattice have 6-fold and 3-fold symemtries at each vertex and face-center; and the hexagonal lattice have 3-fold and 6-fold symemtries at each vertex and face-center. All of them have 2-fold symmetry on the center of each edge.

Among regular polyhedra, the tetrahedron, the octahedron and the icosahedron have 3-fold symmetries at face-centers and respectively 3, 4 and 5-fold symmetries at vertices. The cube has 4-fold symmetries at vertices and face-centers. The dodecahedron has 3-fold and 5-fold symmetries at vertices and face-centers. Now we prove another lemma relating the symmetries of the regular polyhedra and the lattice wrapped onto it. We address satisfying global symmetry conditions at gf-, ge-, gv-symmetry axes.

Lemma 6.2.2. *To satisfy gf-symmetry conditions, all gf-axes must go through a point of the lattice that have cn-fold rotational symmetry where n is the order of rotational symmetry around the gf-symmetry axes of the regular polyhedron and c is a positive integer.*

Proof. Recall that a polyhedra is almost-regular iff it has exactly the same number of rotational symmetry axes with the same symmetry orders as any specific regular polyhedron, such that there exists a rigid body transformation that perfectly

aligns these axes to those of the regular polyhedron. Since, the unfold-etch-refold is a wrapping, the expected locations and axes of global symmetry of the new tiled/etched polyhedra, and the underlying regular polyhedra are already aligned. For example, in Figure 6.3(b), A, B and C are locations of 5-fold global symmetry and D is a location of 3-fold local symmetry. Let, D be the location of one axis of symmetry goind through the face of the regular polyhedron and n be its symmetry order. Depending on the chosen unfolding and mapping, a particular gf-symmetry axis can have the following three cases-

- If gf-symmetry axis goes through a vertex of the tiled polyhedra, then clearly the tiled polyhedra can only be n-fold symmetric around that axis, if the lattice have cn-fold rotational symmetry around its vertices.
- If gf-symmetry axis goes through the center of a face of the tiled polyhedra, then the tiled polyhedra can only be n-fold symmetric around that axis, if the lattice face is cn-regular.
- If gf-symmetry does not go through a vertex or a face-center, then clearly the tiled polyhedra can not n-fold rotational symmetric around the axis, irrespective of the symmetry of the regular grid.

Hence, the lemma is proved for any unfolding/mapping. ■

A regular 2D grid/lattice can be parameterized using two vectors e_1, e_2, and a fixed origin O as follows. Let O is a vertex of the lattice. O has at least 2 neighbors

U and V such that the vectors $O \rightarrow U$ and $O \rightarrow V$ are not colinear. Then, defining $e_1 = O \rightarrow U$ and $e_2 = O \rightarrow V$, every point of the lattice can be expressed as linear combinations $he_1 + ke_2$ where h and k are integers. This defines a coordinate system $\mathcal{L}$ with O as the origin and e_1, e_2 as the primary axes and a co-ordinate (h, k) representing points on the 2D plane, such that if both h and k are integers then, the point lies on the lattice (is a lattice vertex).

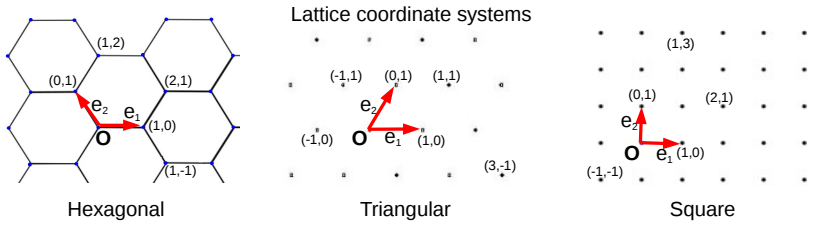


Figure 6.4: **Coordinate systems for the 2D regular lattices**. In the figures, O is the origin and e_1 and e_2 are two coordinate axes. Any point (x, y) can be reached by vector $xe_1 + y_e 2$ from the origin. Coordinates of some of the lattice points are shown.

Lemma 6.2.3. *A lattice described in the $\mathcal{L}$ coordinate system is 2-fold rotationally symmetric around a point (x, y) for the following cases-*

- *Triangular lattice: if and only if both $2x$ and $2y$ are integers*
- *Square lattice: if and only if both $2x$ and $2y$ are integers*
- *Hexagonal lattice: if and only if both $2x$ and $2y$ are odd integers, or $x = 2(y + 1) - 3k$ where x, y and d are integers.*

Proof. Triangular lattices have 2-fold (or multiples of 2-fold) symmetries around their vertices and edge-centers, both of which satisfy the condition that $2x$ and $2y$

are integers. Also, no other point satisfy these conditions. Square lattices have 2-fold (or multiples of 2-fold) symmetries around their vertices, face-centers and edge-centers, all of which satisfy the condition that $2x$ and $2y$ are integers. Also, no other point satisfy these conditions. Hexagonal lattices have 2-fold (or multiples of 2-fold) symmetries around their face-centers and edge-centers. The edge-centers satisfy the condition that both $2x$ and $2y$ are odd integers. The face centers are integer solutions of x, y for the family of lines $x = 2(y + 1) - 3d$ where d is an integer. No other point satisfy either of these conditions. ■

Lemma 6.2.4. *If a face $\mathfrak{T}$ of a regular polyhedron is mapped to a 2D regular lattice in the following ways, then the part of lattice inside $\mathfrak{T}$ is n-fold rotational symmetric around the center of $\mathfrak{T}$, where n is the rotational symmetry around the gf-axes of the regular polyhedron, and the set of faces intersected by each edge of $\mathfrak{T}$ is 2-fold rotationally symmetric around the center of the edge (location of the ge-axes of the regular polyhedron).*

1. *If $\mathfrak{T}$ belongs to either a tetrahedron, a octahedron or a icosahedron and is mapped to triangular lattice, such that all corners have integer coordinates.*

2. *If $\mathfrak{T}$ belongs to either a tetrahedron, a octahedron or a icosahedron and is mapped to a hexagonal lattice such that all corners fall on face centers of the lattice (have integer coordinates (x, y) such that $x = 2(y + 1) - 3d$ where x, y and d are integers).*

3. *If $\mathfrak{T}$ belongs to a cube and is mapped to a square lattice, such that either all*

the corners fall on vertices of the lattice, or all the corners fall on face-centers of the lattice.

Proof. First, we show that such mapping is possible.

1. $\mathcal{T}$ is an equilaterial triangular face being placed on a triangular lattice. Without loss of generality, we assume that one corner is placed at the origin and another at (h, k) where h and k are integers. Then, it is trivial to show that the third point can be at $(h + k, -h)$.

2. $\mathcal{T}$ is an equilaterial triangular face being placed on a hexagonal lattice. Note that the hexagonal lattice is simply the dual of the triangular lattice, hence a mapping that puts corners of $\mathcal{T}$ on vertices of the triangular lattice would put corners of $\mathcal{T}$ on face-centers of the hexagonal lattice.

3. $\mathcal{T}$ is a square being placed on a square lattice. Without loss of generality, we assume that one corner is placed at the origin and another at (h, k) where h and k are integers. Then, it is trivial to show that the other points can be placed at $(h - k, h + k)$ and $(-k, h)$.

When an equilaterial triangular face $\mathcal{T} = ABC$ is placed on a triangular lattice such that the corners are at $(0, 0)$, (h, k) and $(h + k, -h)$, the the center O of $\mathcal{T}$ is at $(\frac{2h+k}{3}, \frac{-h+k}{3})$ which is at a face center, or at a vertex (if $h = k$, or $k = -2h$). Hence O falls on a location of 3-fold or 6-fold symmetry. Hence, gf-symmetry is satisfied. The center (x, y) of any edge would satisfy the condition that $2x$ and $2y$

are integers, and hence falls on a location that have 2-fold (or 6-fold) symmetry and satisfies the ge-condition.

Similar arithmetic can be applied to prove the theorem for the remaining two cases. ■

We shall refer to the mapping described in Lemma 6.2.4 a *compatible mapping*.

Lemma 6.2.5. *For any unfolding of a regular polyhedron onto a regular lattice, if any face $\mathcal{T}$ is compatibly mapped, then all faces are also compatibly mapped. And, the etching inside each face are congruent.*

Proof. For any unfolding, there must be at least another face adjacent to $\mathcal{T}$ and shares an edge with it. Let that edge be AB. According to Lemma 6.2.3, for each other point P belonging to $\mathcal{T}$, there exists a point P' produced by rotating P around the center of AB by 180 degrees. Also, if P was on a vertex (or a face-center), then P' will also be the same. Since only a rigid body motion is applied, the new face $\mathcal{T}' = ABP' \ldots$ will also be regular and be congruent to $\mathcal{T}$ (i.e. it is exactly the unfolded face which was neighboring $\mathcal{T}$ along AB). Also, for any point X in $\mathcal{T}$, the same transformation would map it to a point X' inside $\mathcal{T}'$ such that the location of X with respect to the corners of $\mathcal{T}$ is the same as the location of X' with respect to the corners of $\mathcal{T}'$. Hence the etching inside $\mathcal{T}$ and $\mathcal{T}'$ are congruent. By propagation of the same argument, all unfolded faces have corners at integer coordinates and are congruent, irrespective of the unfolding. ■

Lemma 6.2.6. *For any unfolding of a regular polyhedron onto a regular lattice, if any face $\mathcal{T}$ is compatibly mapped, then the resulting polyhedron will be* almost-regular *or a dual.*

Proof. The polyhedron generated by unfold-etch-refold method have local symmetry, due to the lattice being regular.

Lemma 6.2.4 showed that within the face $\mathcal{T}$, it satisfies global symmetry around the gf-axes. Even after the faces are folded back into the complete polyhedron, the congruency of the all the face (Lemma 6.2.5) guarantees that the remaining faces would also map to each other (including the etching inside them).

Lemma 6.2.5 showed that all faces (and edges) are congruent, and also that the etching is 2-fold symmetric around the edges of the face $\mathcal{T}$. Hence when folded back, the etchings from a neighboring face $\mathcal{T}'$ will match up perfectly (intersect the shared edge of $\mathcal{T}$ and $\mathcal{T}'$ at exactly the same points), and in case there are fractions of a lattice-face inside $\mathcal{T}$, exactly its complement will show up on the other side (inside $\mathcal{T}'$) of the shared edge, thereby all etched-faces are complete. Topologically, the etched-faces that cross the face boundaries and the ones that do not, are identical.

After folding back, the corners of the faces $\mathcal{T}, \mathcal{T}', \ldots$ meet at a point. Note that since all faces are congruent (Lemma 6.2.5) and all corners of each face are also symmetric to each other (Lemma 6.2.4), gv-symmetry is satisfied. Moreover, when $\mathcal{T}$ was mapped such that the corners fell on vertices of the lattice, then the gv-axis is surrounded by exactly n congruent and regular etched-faces, where n is the rotational symmetry of the gv-axis, and not the rotational symmetry around the

lattice-vertices. For instance, mapping a tetrahedron onto a triangular lattice will produce 4 vertices where 3 edges are incident, and many more (depending on the scale of the mapping) where 6 edges are incident. Hence, two types of vertices appear on the polyhedron, the ones coinciding with the gv-axis and the ones that do not. On the other hand, if $\mathcal{T}$ was mapped such that the corners fell on face-centers of the lattice, then the gv-axis is surrounded by exactly a regular polyhedron with nc-fold symmetry, where n is the rotational symmetry of the gv-axis and c is an integer. Other etched faces will simply depend on the regularity of the lattice. For example, if an icosahedron is mapped to a hexagonal grid, then there would be 12 pentagonal faces and many hexagonal faces.

Hence, the polyhedron is *almost-regular* if the corners of the faces fell on lattice vertices, or dual if the corners fell on face-centers. Tetrahedron mapped to triangular grid's face-centers is an exception, where the dual construction also produce *almost-regular* polyhedron. ■

Figure 6.5 shows a few examples of constructing *almost-regular* polyhedron and their duals by mapping a face of regular polyhedron on regular lattices in a compatible way. Note that the etched-faces that cross an edge of $\mathcal{T}$ are geometrially not identical to the ones that do not. The ones crossing the boundary have a crease inside them, or if they are flattened, they are no longer regular. This is addressed in the Section 6.3.4.

Now we show that any deviation from a compatible mapping results in a violation of some global symmetry condition.

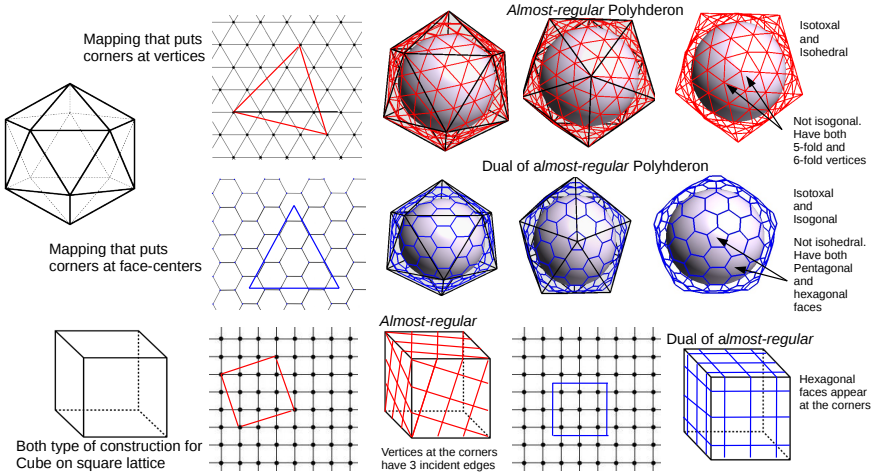


Figure 6.5: **Illustration of the constructing *almost-regular* poleyhdron and their duals**. Top row shows how placing corners of a polyhedral face on vertices of a compatible lattice produces an *almost-regular* polyhedron. The black lines show the original polyhedron, and the red lines show the etching/tiling induced by the lattice. The second row shows and example of placing the corners at face centers and producing duals of *almost-regular* polyhedron. Finally, the bottom row shows examples of both primal and dual construction using square lattice.

Lemma 6.2.7. *If a regular polyhedron is mapped to a regular lattice without fulfilling the compatible mapping conditions, then the resulting polyhedron cannot be* almost-regular.

Proof. If some corners of a face $\mathcal{T}$ fall on lattice vertices (or face-centers) and other don't, then the etching inside the face cannot be symmetric around the center of the face. Hence the polyhedron is not *almost-regular*.

If none of the corners of $\mathcal{T}$ fall on lattice vertices (or face-centers), and the center of the face is not on a lattice vertex or center of a lattice face, then again the polyhedron is not *almost-regular* (by Lemma 6.2.2).

Finally, we consider the case where none of the corners of $\mathcal{T}$ fall on lattice vertices (or face-centers), but the center O of $\mathcal{T}$ is on a lattice vertex or a face-center. This would not violate the symmetry around the gf-symmetry axes. But we shall show that it would violate the symmetry around the ge-symmetry. Since, we are mapping a regular polyhedron onto a compatible lattice, there exists a mapping that would place a face $\mathcal{T}'$ such that the center falls at O, and the corners lie at integer coordinates. We can generate $\mathcal{T}$ by sclaing and/or rotating $\mathcal{T}'$ around O. Now, we shall prove that any scaling or rotation of $\mathcal{T}$ that moves the vertices off lattice-vertices violate ge-symmetry. The proof depends on the geometry of coordinate system and here we shall only show if for the cube mapped to the square lattice case. Other cases follow the same pattern of proof.

Without loss of generality, we assume that the center O is the origin, and we focus on one edge AB of the face $\mathcal{T}$ with integer coordinates (x_1, y_1) and (x_2, y_2). Hence, the midpoint $C(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ falls on a face-center, vertex, or edge-center. After rotating around O by θ degrees, the new location of the center will be $C'(\frac{(x_1+x_2)cos\theta-(y_1+y_2)sin\theta}{2}, \frac{(x_1+x_2)sin\theta+(y_1+y_2)cos\theta}{2})$. Hence, C' can be on a 2-fold location if and only if the numerators are integers, which can only happen if θ is a multiple of $\pi/2$ and in that case, $\mathcal{T}'$ coincides with $\mathcal{T}$.

Now, if $\mathcal{T}$ was scaled by s then the new position of the edge-center C would be $C(\frac{(x_1+x_2)s}{2}, \frac{(y_1+y_2)s}{2})$, and again the numerators are integers iff s is an integer. And if s is an integer, then the corners of $\mathcal{T}'$ would also be on lattice vertices or face-centers.

Hence, scaling or rotating $\mathcal{T}$ around its center O will not keep the C on a

face-center, vertex, or edge-center unless the corners of the transformed face $\mathcal{T}'$ fall on lattice-vertices. Finally, the fact that a regular lattice is not 2-fold symmetric around any point other than the face-centers, vertices, or edge-centers completes the proof. ■

Finally, we conclude our characterization with the following theorem.

Theorem 6.2.1. *The polyhedron generated by an unfold-etch-refold is* almost-regular *if and only if a compatible mapping of a regular polyhedron onto an unfold-etch-refold compatible lattice is performed.*

Proof. The proof follows from Lemmas 6.2.1, 6.2.2, 6.2.6 and 6.2.7. ■

6.3 Construction and Combinatorics of Families of *Almost-regular* Polyhedra

In the previous section we characterized the conditions that must be satisfied by a *unfold-etch-refold* protocol to produce an *almost-regular* polyhedron. The characterization immediately lends itself to efficient generation of families of such polyhedra whose topology can be parameterized using just two variables (discussed below). Furthermore, the symmetry at global and local levels lets us represent the geometry using a minimal set of points. Finally, we show how these properties lead to efficient optimization algorithms for constructing 3D shapes with spherical symmetries.

For the sake of simplicity of presentation, the discussion in this section, in some cases, is focused solely on mapping icosahedron onto triangular lattices. Other compatible mappings can be discussed in the same manner with almost no difference in the theorems/lemmas presented here except for minor changes in the counting parts. In many cases, the differences are mentioned in the text, but detailed/proved only for the icosahedral case. The choice to focus on the icosahedral case is primarily due to two reasons- first, it has the highest level of symmetry among the regular polyhedra which have a compatible mapping, and second, it has applications in modeling viruses, fullerenes etc.

6.3.1 Topological Considerations

Let $\mathcal{L}$ be a lattice with origin O and axes H and K. Any point in the lattice is expressed using coordinates (h, k) where both h and k are integers.

Lemma 6.3.1. *Assuming that one corner A of the face $\mathcal{T}$ of the polyhedron is mapped to the origin O of the lattice (or the nearest face-center for dual constructions). Then specifying the position of another compatibly placed point $B(h, k)$ is sufficient to parameterize the entire mapping.*

Proof. Since $\mathcal{T}$ is regular, there is exactly 2 possible ways to define the other points of $\mathcal{T}$. Lemma 6.2.4 showed that both of these choices will result in a compatible mapping as long as A and B are also compatibly mapped. Also by Lemma 6.2.5, these two are congruent. So, any one of them can be chosen arbitrarily. ■

We mention the following lemma whose proof is immediate.

Lemma 6.3.2. *Topology of any* almost-regular *polyhderon or its dual can be expressed using a tuple* $< \mathcal{P}, \mathcal{L}, h, k >$*, where* $\mathcal{P}$ *is a regular polyhderon,* $\mathcal{L}$ *is a lattice represented using two axes, and* h *and* k *are integers.*

6.3.2 Combinatorics and Symmetry

We consider the case where $\mathcal{P}$ is the icosahedron whose symemtry group will be denoted as I, and $\mathcal{L}$ is the triangular lattice which will be denoted as $\mathcal{L}^3$. Now, we discuss some properties of the lattice.

Definition 6.3.1. *We define each triangle of the lattice* $\mathcal{L}^3$ *as a small triangle and would use* t *to denote such a triangle. Let us define a triple* $< i, j, k >$ *where* i *and* j *are integers and* $k \in \{+, -\}$*. Now we define the triangle produced by the intersections of the lines* $h = i$*,* $k = j$ *and* $h + k = i + j + 1$ *and the vertices* $(i, j), (i + 1, j)$ *and* $(i, j + 1)$*, as* t_{ij+}*. Similarly, the triangle* t_{ij-} *have vertices* $(i, j), (i + 1, j - 1)$ *and* $(i + 1, j)$*, and is produced by the intersections of the lines* $h = i + 1$*,* $k = j$ *and* $h + k = i + j$*.*

The proof of the following lemma is immediate from this definition.

Lemma 6.3.3. $t_{i_1 j_1 k_1}$ *coincide with* $t_{i_2 j_2 k_2}$ *if and only if* $i_1 = i_2$*,* $j_1 = j_2$ *and* $k_1 = k_2$*. For any small triangle in* $\mathcal{L}^3$*, there exists a triple* $< i, j, k >$ *such that* t_{ijk} *represents that small triangle.*

Through etching, the triangular lattice $\mathcal{L}^3$ produces a tiling of a face $\mathcal{T}$ (which will be called a large triangle in this section) of $\mathcal{P}$ where each tile is a small triangle. Now we consider some properties of this tiling.

Assuming A is at $(0, 0)$, B is (h, k) such that h and k are integers, the tiling produced by $\mathfrak{L}^3$ on $\mathfrak{T}$ satisfies-

- The area of $\mathfrak{T}$ is $\frac{\sqrt{3}}{4}(h^2 + hk + k^2)$, which is equal to the area of $h^2 + hk + k^2$ small triangles.
- In addition to A, B and C, $\mathfrak{T}$ includes exactly $\frac{h^2+hk+k^2-1}{2}$ more vertices of $\mathfrak{L}^3$. Note that any vertex that lie on an edge of $\mathfrak{T}$ is counted as half a vertex.
- Each edge of $\mathfrak{T}$ is intersected by at most $2(h+k)-3$ lines of the form $h = c$, $k = d$ and $h + k = e$, where c, d and e are integers.
- The number of small triangles intersected by any edge of $\mathfrak{T}$ is at most $2(h + k - 1)$.

Now, some combinatorial properties of the overall tiled polyhedron-

- There are exactly $20(h^2 + hk + k^2)$ small triangles, and the same number of local 3-fold axes.
- The 12 gf-symmetry axes are surrounded by 5 small triangles.
- There are exactly $10(h^2 + hk + k^2 - 1)$ vertices (not lying on the gf-axes) with 6-fold local symmetry.

Similar properties can easily be derived for other mappings as well. The important point to note is that not only the topology, but also the symmetry and combinatorics are also parameterized by only h and k.

6.3.3 Construction and Geometric Considerations

Note that given a point P with coordinate (i, j) inside $\mathcal{T}$, there exists two other points Q and R such that P, Q and R are 3-fold symmetric around the center D of $\mathcal{T}$. The two points Q and R have coordinates $(h - i - j, k + i)$ and $(h + k + j, -h - i)$ respectively. This can be seen by noticing that stepping along the H and K axis by i and j units from $A(0, 0)$ is C^3 symmetric (around D) to stepping in $-H + K$ and $-H$ directions by the same units from $B(h, k)$, and stepping in $-K$ and $H - K$ directions by the same units from C. We can further extend it to triangles and deduce the following.

Lemma 6.3.4. *If $A(h_1, k_1)$, $B(h_2, k_2)$ and $C(h_3, k_3)$ are three points in the HK coordinate system such that $h_1, h_2, h_3, k_1, k_2, k_3$ are integers and ABC is an equilateral triangle whose centroid is O, then the small triangles $t_{h_1+i,k_1+j,\pm}$, $t_{h_2-i-j-1,k_2+i,\pm}$ and $t_{h_3+j,k_3-i-j-1,\pm}$ are C^3 symmetric around O.*

Now, we define the minimal set of points or non-redundant set of points $\mathbb{S}$ such that no two points $s_i, s_j \in S$ are C^3 symmetric to each other around D, and all points in $\mathbb{S}$ lie inside or on $\mathcal{T}$. Clearly, $|\mathbb{S}| = \lceil \frac{h^2+hk+k^2}{3} \rceil$. Note that applying C^3 operations on $\mathbb{S}$ produces all points inside and on $\mathcal{T}$.

Now we are ready to specify a concrete algorithm for computing *almost-regular* polyhedra (see Figure 6.6).

Theorem 6.3.1. *The algorithm* TILINGGEN *constructs a minimal geometric representation of the* almost-regular *polyhderon in terms a set of points $\mathbb{S}$ embedded onto the XY plane and a set of 3D transformations $\mathbb{T}_{all}$.*

TILINGGEN($\mathcal{P}$, $\mathcal{L}$, h, k)
Constructs an *almost-regular* polyhderon using compatible mapping of polyhderon $\mathcal{P}$ onto lattice $\mathcal{L}$, such that the scaling and combinatorics are specified by h,k

1. Assume that the lattice coordinate system is aligned with the Cartesian coordinate system such that the origings coincide and one of the axes is aligned to the X axis, and the other lies on the XY plane.
2. Place one point A at the origin $(0, 0)$ of the lattice, a second point at (h, k). Compute the other corners of the face $\mathcal{T}$. Note that we only need to know the number of vertices n of $\mathcal{T}$.
3. Let $\mathbb{T}_{\mathbb{C}}$ be the set of cyclic symmetry operations aruond D, such that $|\mathbb{T}_{\mathbb{C}}| = n$.
4. Compute the location of the center D of the face $\mathcal{T}$.
5. Initialize empty set $\mathbb{S}$
6. For each lattice point p inside or on $\mathcal{T}$ do
7. Add p to $\mathbb{S}$ if and only none of the trasnformations in $\mathbb{T}_{\mathbb{C}}$ applied to p produces a point which is already in $\mathbb{S}$.
8. Compute the transformation T_{map} which maps the face $\mathcal{T}$ to a face of the polyhedron $\mathcal{P}$. T_{map} is composed of $T_{map_T} T_{map_S} T_{map_A}$ such that T_{map_A} translates $\mathcal{T}$ along the lattice to take D to the origin O, T_{map_S} is a scaling that resizes $\mathcal{T}$ to the size of the faces in $\mathcal{P}$, then T_{map_T} is a translation along Z-axis by an amount equal to the distance from the center of $\mathcal{P}$ to a face-center.
9. Let $\mathbb{T}_{\mathbb{P}}$ be the set of global symmetry operations (from the symmetry group of $\mathcal{P}$.
10. Define a set of tranformations $\mathbb{T}_{all} = \{T_2 T_{map} T_1 | T_2 \in \mathbb{T}_{\mathbb{P}} \& T_1 \in \mathbb{T}_{\mathbb{S}}\}$.
11. All points on the almost-regular polyhedron is now generated by simply computing $\mathbb{T}_{all}(\mathbb{S})$.

Figure 6.6: TILINGGEN: Algorithm for constructing an *almost-regular* polyhderon using compatible mapping

Proof. Follows from the definition of $\mathbb{S}$ and Lemma 6.2.6. ■

Some polyhderon generated by applying TILINGGEN are shown in Figure 6.7.

Note that if the tiles that cross the boundaries of a face $\mathcal{T}$ of $\mathcal{P}$ are not regular, they would look like they have a crease along the edge of the $\mathcal{T}$ by definition of the unfold-etch-refold technique. Tiles generated by this algotithm will also have the same problem and such tiles will be non-regular, and in some cases even non-planar.

In the next section we address this issue.

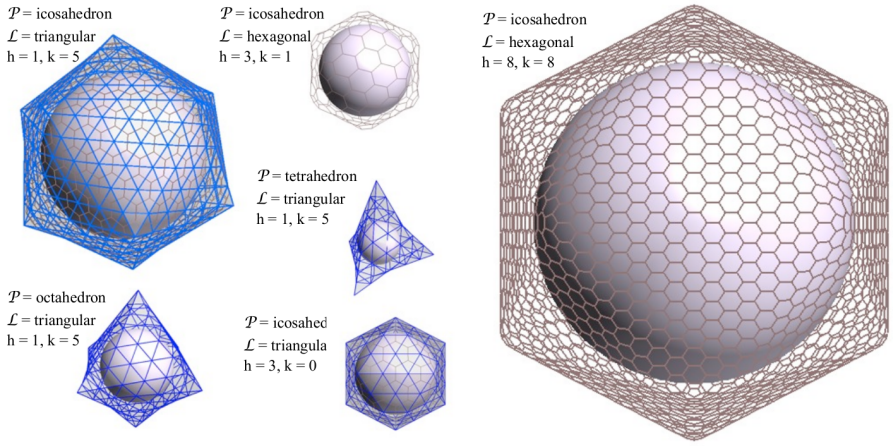


Figure 6.7: **Some polyhderon generated by applying** TILINGGEN.

6.3.4 Curation of Tiles

As mentioned before, sometimes the lattice faces, which corresponds to tiles/faces of the genrated *almost-regular* polyhderon, crosses the boundary of the polyhedral face $\mathcal{T}$ embedded on the lattice. During folding, these faces get warped. There can be exactly three types of scenarios-

1. For mapping to a square or triangular lattice, if one corner of $\mathcal{T}$ is at $(0, 0)$, and the other corner is at (h, k) such that either $h = 0$ or $k = 0$, then no lattice face crosses the edges of $\mathcal{T}$, and no curation is required. (see Figure 6.8 top row).

2. For mapping to a square or triangular lattice, if one corner of $\mathcal{T}$ is at $(0, 0)$, and the other corner is at (h, k) such that $h = k$ then some lattice faces are

exactly bisected by the edges of $\mathcal{T}$. The curation is quite trivial in this case. If, $h = k$, the center of $\mathcal{T}$ would lie on a lattice vertex. Let the center be D, and the face $\mathcal{T}$ be ABC. Then, folding along AD, BD and CD will not warp any lattice face. Additionally, connecting D to the centers of the neighbring polyhedral faces $\mathcal{T}$ would satisfy all global symmetry conditions as well. This folding will produce a base polytope which actually looks like the *almost-regular* polyhedron with $h = k = 1$. In fact, for any integer i, to polyhedron generated for $h = k = i$ is nothing but subdivisions of the faces of the $h = k = 1$ polyhedron. Interestingly, in some cases, the new folding produces a polyhedron with base geometry like some Catalan solids, but will unlike Catalan solids, these will have regular faces and may be non-convex. For example, the $h = k = 1$ polyhedron have the same topology as the pentakis dodecahedron (see Figure 6.8 middle row).

3. For mapping to a square or triangular lattice, if one corner of $\mathcal{T}$ is at $(0, 0)$, and the other corner is at (h, k) such that $h \neq k, \& h, k > 0$, and for all mappings on the hexagonal lattice, some lattice faces cross the edges of $\mathcal{T}$ in variable ways, and there are no folding lines which can avoid the crossing while maintaining global symmetry (as the lines will not meet at the center of the face $\mathcal{T}$). See Figure 6.8 bottom row. In this case, no exact solution exists, and we provide a numerical approximation which maximizes the regularity while ensuring that global symmetries are not violated (see below).

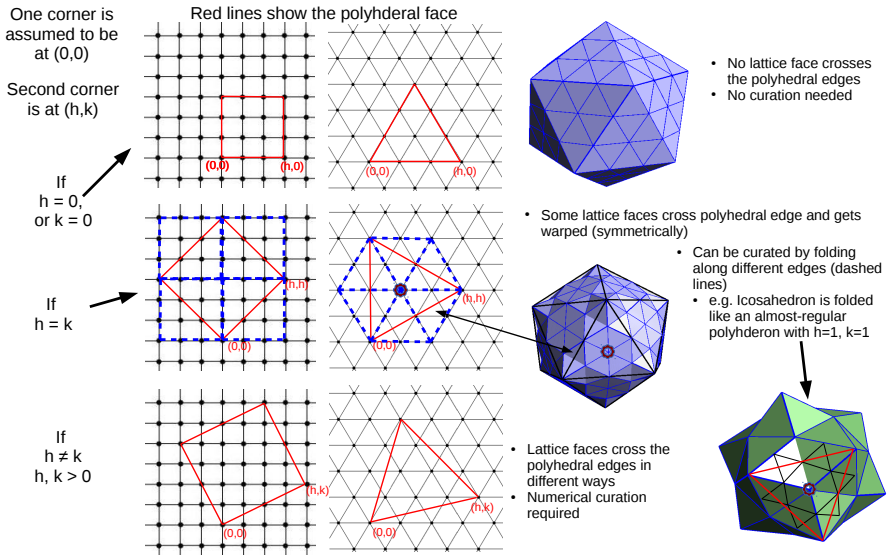


Figure 6.8: **Different cases of warping of tiles, and their curations**.

6.3.4.1 Curation as a Numerical Optimization Problem

The goal of curation would be to make all the faces as regular as possible without violating global symmetry. A similar problem for specifically the family of polyhedra generated by mapping an icosahedron onto a hexagonal lattice was addressed in [225]. In that work Schein and Gayed aimed to make all the hexagonal faces that cross the face boundary, and become creased/non-planar, into planar ones while keeping the edge lengths equal. They also showed that it was possible to provide an efficient numerical solution to the problem which also ensure that no two hexagonal/pentagonal tiles lie on the same plane and the overall polyhedron is convex. However, the shapes of the hexagons are allowed to get distorted such that the angles are no longer equal, and may vary a lot within the same hexagon. This is a very elegant mathematical solution, however, the faces are no longer congruent (or

even nearly congruent) to each other. This makes such a polyhedron non-amenable for modeling structures formed using a single type of building block, for instance viral capsids. In contrast, we want to maintain the congruence of the tiles as much as possible.

For the cases of mapping a polyhedron onto a triangular lattice, the generated polyhedra falls under the class called deltahedra, polyhedra whose faces are all equilateral triangles. Even though there are an infinite family of deltahedra [255] (our families are also infinite), It has been known since Freudenthal and van der Waerden's work [99], that there are exactly eight convex deltahedra, having 4, 6, 8, 10, 12, 14, 16 and 20 faces; among them only three are regular or have symmetries like the regular ones. So, our family of *almost-regular* polyhedra cannot be convex and regular at the same time. We prioritize regularity.

First we introduce some notations-

- Let the set of all points on the generated polyhedron be $\mathbb{S}_{all} = \mathbb{T}_{all}(\mathbb{S})$.
- Let $\mathbb{E}_1$ be the set of lattice/tile edges on the generated polyhedron
- Let $\mathbb{E}_2$ be the set of diagonals of all the tiles on the generated polyhedron
- Let, for any point $p \in \mathbb{S}_{all}$, the functions $s(p)$ and $t(p)$ returns respectively a point $q \in \mathbb{S}$ and a transformation $T \in \mathbb{T}_{all}$ such that $p = T(q)$.
- Let $dist(u, v)$ is the square of the Euclidean distance between two points.

In our calculations we shall only update the positions of the points in $\mathbb{S}$, and all other points $p \in \mathbb{S}_{all}$ on the polyhedron will be generated as $t(u)(s(u))$. This

ensures that the points are always moved in a symmetric way with respect to the global symmetry axes. Hence global symmetry is never violated.

Let $\mathbb{S}^0$ be the initial positions of the points in $\mathbb{S}$. As we update the locations of the points in $\mathbb{S}$ in our algorithm, the displacement of each point $p \in \mathbb{S}$ will be defined as $\delta(p) = (dist(p, p^0))^2$, where $p^0 \in \mathbb{S}^0$ is the initial position of p. ALso, the length of a line segment $e(u, v) \in \mathbb{E}_1 \cap \mathbb{E}_2$ will be computed as $dist(t(u)(s(u)), t(v)(s(v)))$ and be denoted $l(e)$.

Let us also define $\mu_1 = \frac{1}{|\mathbb{E}_1|} \sum_{e \in \mathbb{E}_1} (l(e))^2$ and $\mu_2 = \frac{1}{|\mathbb{E}_2|} \sum_{e \in \mathbb{E}_2} (l(e))^2$.

Finally, we define an energy function $\mathcal{F}(\mathbb{S})$ as follows-

$\mathcal{F}(\mathbb{S}) = \frac{1}{|\mathbb{E}_1|}(\sum_{e \in \mathbb{E}_1}(l(e) - \mu_1)) + \frac{1}{|\mathbb{E}_2|}(\sum_{e \in \mathbb{E}_2}(l(e) - \mu_2)) + \frac{1}{|\mathbb{S}|}(\sum_{p \in \mathbb{S}} \delta(p))$

Now, we minimize the function $\mathcal{F}(\mathbb{S})$ over the positions of the points in $\mathbb{S}$.

This is clearly a quadratic optimization problem over $h^2 + hk + k^2$ variables, and for most practical values of h and k it can be solved efficiently using any numerical solution techniques. We chose to use the limited memory variant of Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [52] due to its faster convergence rates. the BFGS algorithm requires the first and second derivatives (hessian) of the energy function. In our case, the energy function, if expanded is a simple polynomial of the free variables and can the first and second derivatives are straight-forward to compute, making the application of BFGS possible.

6.4 Conclusion

We have characterized a new family of polyhedra with regular faces such that it is isotoxal, isohedral, and have exactly 2 types of vertices; as well as a dual family which is isogonal, isotoxal and have exactly 2 types of regular faces. We have shown that both of polyhedrons of these families generated by unfolding a regular polyhedron onto a lattice in a compatible way, thereby allowing the lattice vertices, edges and faces to etch out a tiling on the unfolded polyhedron, and finally folding it back again. Further, the compatible ways are specified using only a couple of integer parameters. We also provided a deterministic and efficient algorithm for generating such polyhedra of any size (determined by the two parameters). We have proved that our construction covers all possible polyhedron which satisfies the stated properties. When considering the geometric aspects of the generated polyhedra, we characterized the cases where the faces may become non-regular, and provided solutions for each case.

Finally, we point out that our class of polyhedron is not similar to the known families like Catalan solids, Johnson solids, or Archimedean solids. Some Catalan solids like the tetrakis hexahedron, triakis octahedron, triakis icosahedron, rhombic dodecahedron, rhombic triacontahedron, or pentakis dodecahedron may seem like they satisfy the properties of *almost-regular* polyhedron, but actually all of them violate either the global of the local symmetry conditions. Also Archimedean solids like the truncated cube can be generated by placing the triangles of a tetrahedron on a hexagonal lattice such that the corners of each triangle fall on the centers of 3 faces surrounding a single face. Many other Archimedian solids are isogonal and

isotoxal, but are none of them (not even the truncated cube) are duals of of any *almost-regular* polyhedron.

Chapter 7

Predicting Symmetric Spherical Shell Assemblies

We use our characteriation of tiled, symmetric shell structures (described in Chapter 6 to address the computational problem of predicting the assembly of 3D spherical shell structures from primitive 3D building blocks (or primitive tiles). We provide an efficient polynomial time, shell assembly approximation solution (based on a combinatorial and motion-space search, and complementarity scoring) to an otherwise NP-hard geometric optimization problem. Our 3D shell assembly prediction, first uses a small set of tiles to generate either periodic or aperiodic surface tilings, parameterized by a small set of inter-tile matching rules. To form a 3D shell, we decorate each tile using appropriate number of copies of the given 3D blocks. The decorations must cover the tile, and those placed on a single tile must have a favorable and symmetric 'interface' with each other. Furthermore, the interfaces of the decorated tile with its neighboring tiles must follow the symmetry imposed by the inter-tile matching rules. The problem thereby reduces to a multi-dimensional search (and maximal scoring) of a consistent set of symmetric interfaces between each unique pair of neighboring decorations of the chosen tiling. We have successfully applied this procedure to the prediction of spherical protein shells of biological viruses of different sizes, all of which exhibit icosahedral symmetry. Our implemented technique has been successful in predicting the tiling/packing of known

and experimentally reconstructed protein shell structures of a fixed size, amongst several other computed assemblies of various sizes. The success of our prediction mechanism based on complementarity scoring is specially noteworthy in light of the complicated biophysical/biochemical interactions between protein structures in nature, and provides testimony to the stability of such tiled shell structures.

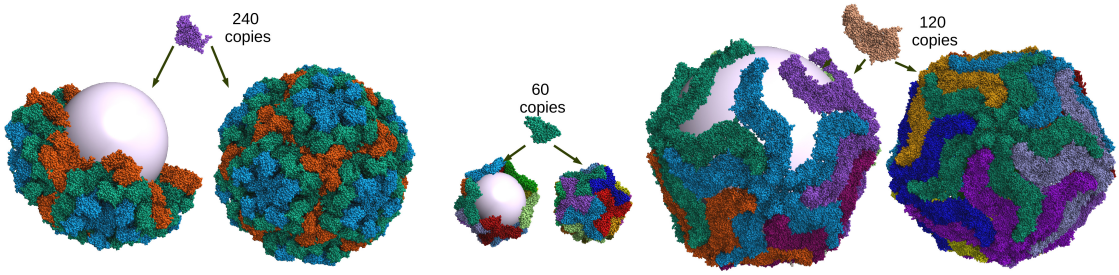


Figure 7.1: Automated prediction of compact symmetric shell assemblies from primitive building blocks. Figure shows three example shell structures of different sizes and thicknesses assembled using 240, 60 and 120 copies of the corresponding building blocks.

7.1 Introduction

In general, 3D assembly requires determining the correct arrangement (aka relative positions and orientations) of a set of building blocks, to yield the final correctly assembled structure. As the 3D analog of 2D jigsaw puzzles, the 3D building blocks (henceforth called primitive tiles or p-tiles) are the puzzle pieces, and these as well as the target assembled structure may have various free-form shapes. Assembly prediction with n blocks in 3-dimensions can be cast as a combinatorial geometric optimization problem with a search space of $6(n-1)$ dimensions. The optimization becomes computationally prohibitive for such high dimensional

search spaces, particularly since typically $n >= 100$. However, and as we show here, the optimization problem is made tractable by imposing restrictions on the shape and properties of the building blocks, and assemblies.

Prior work has addressed this problem from the perspective of reconstructing or reassembling historic 3D artifacts or statues from fractured 3D pieces recovered by archaeologists. For example, [127] and [191] both assume that the neighboring fractured pieces have good shape complementarity and use shape descriptor based matching to find possible interfaces between each pair of components. The remaining challenge is then to select a subset of those interfaces which are consistent, and optimize a global assembly. The approach is equivalent to computing the best simple spanning tree from a multi-graph where each graph node represent a p-tile, and k edges between two nodes represent k possible interface configurations. The number of simple spanning trees is exponential in the number of components, i.e. $O(n^{n-2}k^{n-1})$. Furthermore, its quite easy to reduce the known NP-hard 2D Monkey Puzzle problem to the more difficult 3D assembly problem. One can show that a polynomial time approximation scheme ($PTAS$) can be designed if the number of parallel edges between each pair of nodes is at most 2. However, for all practical cases, $PTAS$s do not exist. The paper [127] proposed a greedy forward search algorithm and [191] used genetic programming to solve this combinatorial problem. A simpler greedy algorithm was used by [130] for predicting the assembly of multiple proteins. The 2D counterpart is a well-studied problem, for both texture or image driven matching [62, 219] as well as based purely on the shape [107].

Here, we provide an efficient polynomial time solution to the 3D assem-

bly problem for the special case of symmetric shell structures. We show that the combinatorial search space has only 6 degrees of freedom, irrespective of the total number of components used in the final assembly. Moreover, we can predict shell assemblies with different local symmetric arrangements, using periodic and/or aperiodic tilings, as well as of varying size. These symmetric shell assemblies occurs often and in many shapes and sizes, i.e. in the shape of bucky balls, fullerene domes, carbon-nano tubes, quasicrystals and protein shells that house the genome of viruses [55, 83, 113, 224, 233].

The 3D symmetric shell assembly problem can be stated as a problem of predicting the local arrangement of multiple copies of a single 3D p-tile, to form a closed shell structure, enclosing an inner void space. See for example, figure 7.1. Local symmetry indicates that each p-tile has the same kind of interfaces with its neighbors, and each of the interfaces have d-fold cyclic symmetry for certain integer ds. In some cases, we allow the pieces to be quasi-symmetric where the number of interfaces remain the same for every p-tile, but the cyclic symmetric order of each interface may vary. Global symmetries are also achieved for the 3D assembled shell structures, and provides shell structure which are invariant under some transformations. For example, given a 3D triangular prism tile with extremely small thickness, a simple tetrahedron shell can be predicted and assembled using four copies of the 3D tile.

Our solution to the 3D shell assembly problem is a combinatorial and geometric search and optimization, over the space of all symmetric placements of p-tiles on a shell, such that the arrangement maximizes a local (and a global) scor-

ing function. The scoring functions are designed to reward complementarity at the interfaces of neighboring p-tiles, where an interface is defined as the portion of one p-tile which is within a distance from any point on the neighboring p-tile. We show that the symmetric search space is sufficiently characterized by the space of regular and semi-regular tilings of regular polytopes and their decorations. A tiling or tessellation is a covering of a given space using a set of non-overlapping tiles, and a decoration is the strategic placement of components on each tile such the same type of tiles, are decorated the same way.

To the best of our knowledge this 3D symmetric shell assembly problem has not been approached as a prediction problem before. Previous papers with some similarity include:[163] where they describe a method to decompose a given surface into polyominos and then transform them by adding notches and knobs to create a set of 3D tiles which has a unique reassembly solution; and [202] describe algorithms to compute optimal geodesic patterns on free-form shells. Other related works analyzing, enumerating or classifying symmetric shells include[55, 173, 193].

Our discussion is organized as follows. In Section 7.2 we provide a brief description of our algorithm. We discuss the mathematical principles of symmetric tilings in Section 7.3 and show that given a specific p-tile and a desired size for the final shell, only a small finite number of possible tilings need to be considered for decoration. We also describe how to pre-compute all possible tilings. In Section 7.4, we explain the decoration algorithm in greater detail. Finally in Section 7.5, we apply this algorithm to predict the shapes of biological virus protein shell

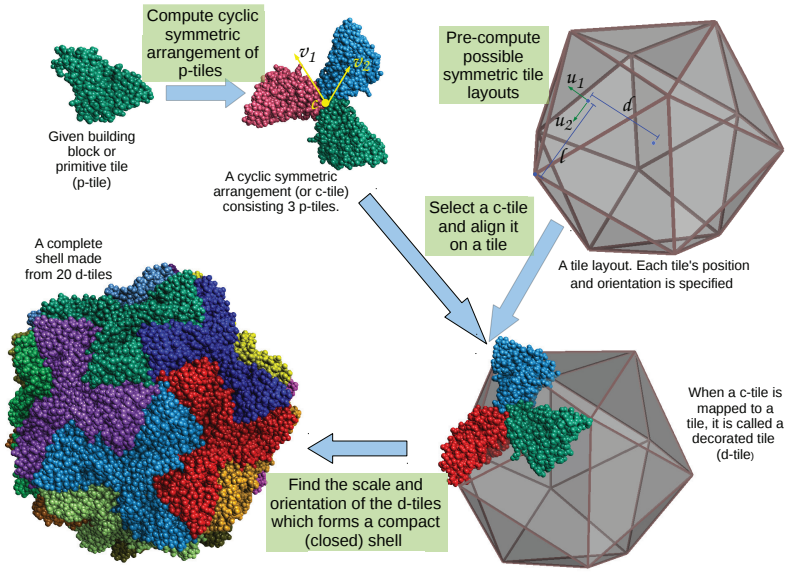


Figure 7.2: Overview of the algorithm. Given a 3D p-tile, we compute possible cyclic symmetric arrangement (henceforth c-tiles) that can be formed by them and then use many such c-tiles to decorate precomputed sets of spherical tilings. The c-tile computation requires a geometric search and optimization over 4 dimensional space. The complexity of selecting appropriate c-tiles for a specific spherical tiling is polynomial in the number of unique tiles that are used to cover the tiling. Finally, placing the decorations for optimal global arrangement requires only a 2-dimensional search.

structures exhibiting global icosahedral symmetry (and quasi-symmetry). Symmetric shells made of 3D molecules, and their corresponding inter-molecular interfaces are much more difficult to score since they do not have sharp features, or yield exact shape complementarity. Additionally, there our other bio-physical factors like electrostatics complementarity that often play a much bigger role in stabilizing the interface between neighboring molecular building blocks. These stabilizing forces cause for spontaneous self-assembly of the shells to occur, a phenomena that is still

not fully characterized however but truly one of the marvels of nature. So in this application of predicting molecular assemblies, we construct and optimize a combined multi-term biophysical and bioinformatics based scoring function. The scoring function include shape and electrostatic complementarity terms, polarization energy and terms based on knowledge-based potentials defined on the interface. We however surprisingly found that even though the molecular shell assembly is weakly scored, our algorithm in many cases produces the native (experimentally observed) assembly configurations as one of its predictions. Note, that for non-biological applications (architectural or manufacturing or 3D interlocking puzzles), using only shape complementarity scoring suffices in yielding 3D shell assemblies.

7.2 Sketch of the Algorithm

Below, we present the overall algorithm (Figure 7.2 provides a simple example flow). Separate sections are dedicated for the exposition of different aspects of the algorithm as well the mathematical foundation behind it.

1. We pre-compute the possible tilings of spherical shells. The layout of the tiling is uniquely described by specifying the details of each tile and unique inter-tile interface. A tile is described using five parameters $(\mathbf{u_1}, \mathbf{u_2}, d, l, f)$.

    - $\mathbf{u_1}$ is a unit vector pointing from the origin to the center of symmetry of the tile.
    - $\mathbf{u_2}$ is a unit vector pointing from the center of symmetry of the tile to one representative corner of the tile.

- d is the distance of the tile from the origin along $\mathbf{u_1}$.
- l is the distance from the center of symmetry of the tile to one corner of the tile along $\mathbf{u_2}$.
- f is the order of symmetry of the decoration to be placed inside the tile.

The interfaces are described by simply listing each unique type of interface and the number of such interfaces in the entire tiling. A pair of tiles have an interface if they are neighbors on the tiling. Note that all inter-tile interfaces essentially correspond to a cyclic symmetric configuration. For example, 5 tiles meeting at a vertex with 5-fold symmetry produces 5 interfaces of the same type. Section 7.3 describes the mathematical principles of enumerating and generating all possible spherical tilings.

2. Given a 3D p-tile we compute cyclic symmetric configurations (aka c-tiles) of different orders. The search for such configurations is carried out by optimizing a local score, based on the interface of a pair of c-tiles, over a 4D search space. Several configurations of each order are generated and ranked by their scores. Section 7.4.1 provides details on the search space, the sampling and search strategy and the principles of scoring. Each c-tile is represented using five parameters $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{c}, o, s)$.

- $\mathbf{v_1}$ is a unit vector representing the symmetry axis.
- $\mathbf{v_2}$ is a unit vector orthogonal to the symmetry axis pointing to the centroid of one copy from the center of symmetry.

- $\mathbf{c}$ is the center of symmetry.
- o is the order of symmetry.
- s is the score of the c-tile configuration.

3. We compute a graph $\mathcal{G}$ where each node represents a c-tile and we add edges between two nodes if the interface configurations are consistent. Further details about this step can be found in Section 7.4.2.

4. For each tiling from the precomputed set, we first identify the set of unique tile types and their corresponding order f. We search $\mathcal{G}$ to find cliques which include exactly one node corresponding to each required order. For each such clique we perform the following procedure to find a globally acceptable assembly.

- Align a c-tile of required order onto each tile, by moving $\mathbf{c}$ to the origin and aligning $\mathbf{v_2}$ and $\mathbf{v_1}$ with $\mathbf{u_2}$ and $\mathbf{u_1}$ and then translating it along $\mathbf{u_1}$ by d. This produces a assembly which has the required kinds of symmetry. However, since we have not adjusted the scale of the shell to match the scale of the p-tile, this layout is not a proper assembly yet. Also note that, though the alignment of $\mathbf{v_1}$ with $\mathbf{u_1}$ is sacred due to symmetry, the alignment of $\mathbf{v_2}$ with $\mathbf{u_2}$ is only necessary for ensuring that all c-tiles are decorated the same way. it does not ensure that this is indeed the correct way to orient the c-tile to cover the unique spherical tile. Hence we move to the next step.

- We search the space of scaling and rotations. Symmetry preserving scaling is performed by simply translating each c-tile along the respective $\mathbf{u_1}$ vectors by multiples of d. For each scaling search the space of rotations around the $\mathbf{u_1}$ axis. Apply the same rotation for each spherical tile of the same type to preserve global symmetry. The angles only need to be sampled within a band $[-\pi/f, \pi/f]$. The scaling is also performed within a band such that the scaled l is within $[t/2, 2t]$ where t is the farthest point from c if the entire c-tile is projected onto the plane perpendicular to $\mathbf{v_1}$. If l is outside this band then there is guaranteed to be no contact or penetrations respectively.
- Report the scale and rotation which produces the best global score. The global score is computed as weighted sum of the scores for each unique interface, where the weights are simply the number of interfaces of the corresponding type. See Section 7.4.3 for further details about the global optimization step.

7.3 Enumerating All Possible Symmetric Tilings of Spherical Shells

The symmetric search space can be adequately characterized using a mathematical framework dealing with tilings and decorations. In the next few subsections we briefly introduce this framework and how it characterizes the search space as well as motivates a search strategy. First we describe the concept of tilings and various kinds of symmetry related to the tiling. Then we describe a formal way of

generating and enumerating all tilings which have specific kinds of global symmetry using their root systems. Next, we show how an extension of the root system lets us generate denser tilings, which can be decorated using more puzzle pieces to generate larger shells, such that the global symmetry is preserved and local symmetry is created. Finally, after elaborating the concept of decorations we will show how the search space is restricted to only 6 DOFs.

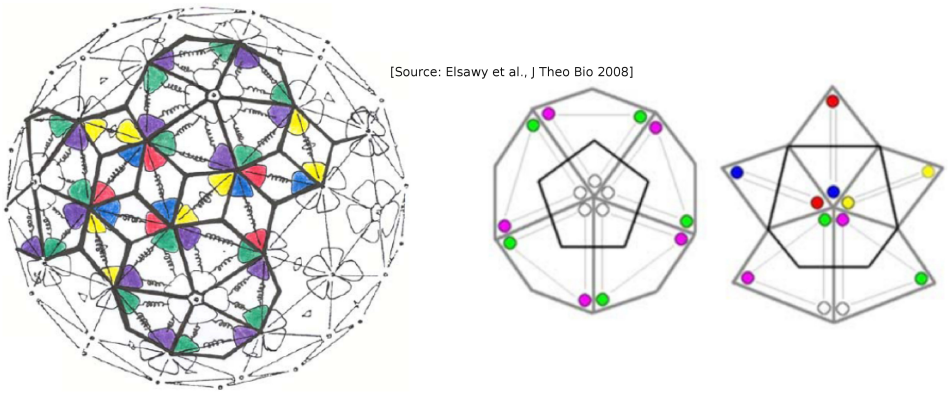


Figure 7.3: Left A semi-regular aperiodic tiling of a spherical surface using two types of tiles, a kite and a rhomb. The tiles are decorated with petal-like building blocks at the corners. The cyclic symmetry relationships between the decorations in each tile are shown by squiggly lines. Right The set of vertex stars of the tiling.

7.3.1 Tiling and Symmetry

A tiling or tessellation is a covering of a given space using a set of non-overlapping tiles. Formally, given a space S, a tiling of S is a set of tiles $t_1, \ldots, t_k$ such that $S = \bigcup_{i=1}^{k} t_k$ and $\forall_{i \neq j} t_i \cap t_j = \phi$. A 2D tiling is regular if all tiles are congruent regular polygons and have edge-to-edge interfaces, i.e. each interface spans an of the tile. The definition can easily be extended to higher dimensions. A

tiling is sometimes called semi-regular if the congruency restriction is lifted. Notice that regular and semi-regular tilings consists of only one and a few unique types of tiles respectively. In edge-to-edge tilings, the tiles meeting at a vertex are called a vertex star and the collection of all unique vertex stars is called the vertex atlas. See Figure 7.3 for an example (both images are from [89]). Again, regular and semi-regular tilings have very small vertex atlases.

7.3.1.1 Symmetries of Tiling

A regular tiling is often expressed using Schäfli notations. If a tiling consists of regular polygons with p edges and q of them meet at each vertex, then its Schäfli notation is $\{p, q\}$. It is easy to see that for 2D regular planar tiling, the angles of the polygons are $2\pi/q = (p-2)\pi/p$ or $(p-2)(q-2) = 4$, which has only three integral solutions producing the tilings by squares, by equilateral triangles and by hexagons. Regular and semi-regular tilings exhibit various kinds of symmetry. For example, a regular tiling of an infinite 2D plane using equilateral triangles is invariant under any translation in the direction of any of the edges by a magnitude equal to an integer-multiple of the edge-lengths. It is also invariant under rotations by multiples of 60 degrees around any vertex or reflections around any edge. All of these operations under which a tiling remains invariant forms the set of elements for symmetry group of the tiling. Any two of these operations can be composed (multiplications of transformation matrices) and the resulting operation is also member of the set. In other words, the elements of a symmetry group are closed under the application of the composition operation. The order of a group is the number of of its elements.

Interested readers should consult [72] for more details on symmetry groups, their representations, classifications and enumerations.

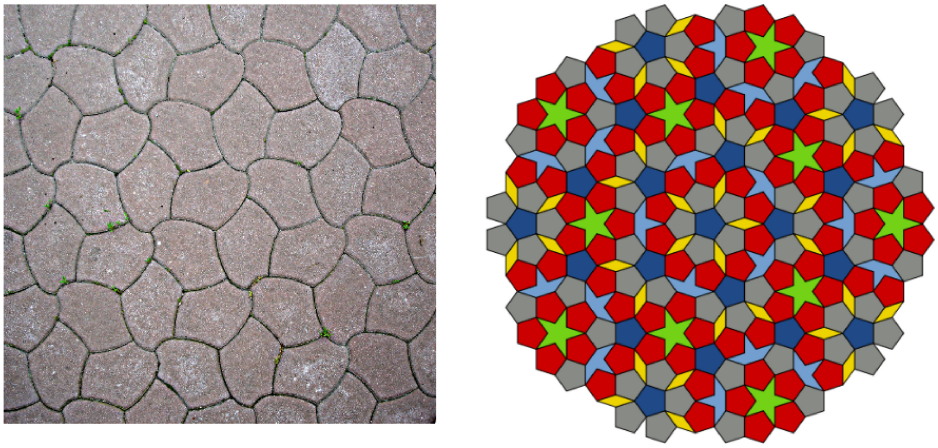


Figure 7.4: A periodic and an aperiodic tiling.

At this point we would like to point out that it is possible to produce semi-regular tilings which have rotational and reflection symmetries, but lacks translational symmetries. These tilings are classified as aperiodic tilings, and tiles which can only be used to produce aperiodic tilings are called aperiodic tiles. tilings which have translations symmetry as well are called periodic tilings. See Figure 7.4 for examples of 2D aperiodic tilings (the images were acquired from wikimedia and they are licensed for public use under the creative commons).

7.3.1.2 Tiling of Spherical Shells

We are specifically interested in regular tilings of a shell (tiling a sphere with 2D tiles) and their symmetries. A sphere can be tiled regularly iff integral solutions exists for $(p-2)(q-2) < 4$, since polygons mapped to the sphere have

larger angles. We can easily find that there are only 5 non-degenerate solutions. These solids, usually called the Platonic solids, are the tetrahedron $\{3, 3\}$, the cube $\{4, 3\}$, the octahedron $\{3, 4\}$, the icosahedron $\{3, 5\}$ and the dodecahedron $\{5, 3\}$. Note that the cube and icosahedron are duals of each other, the icosahedron and the dodecahedron are duals, and the tetrahedron is self-dual. The tetrahedral symmetry group T_d has order 24, the octahedral (and cubic) symmetry group O_h has order 48 and the icosahedral (and dodecahedral) symmetry group I_h has order 120.

Hence, given a puzzle piece we would like to make a shell which has any of the above 5 symmetry groups. However, to make larger shells we need to find a way to subdivide each facet of these polyhedra into regular tiles and then place puzzle pieces in each of the tiles. This subdivision also has to follow the global symmetry of the polyhedra. The next two subsections address this.

7.3.2 Generating all Symmetric Tiling

Here, we identify the minimal representation/parameterization which is sufficient for generating a tiling with specific global symmetry. We employ two primary techniques in this regard. The first, uses the unfold-etch-refold scheme described in Chapter 6 to generate families of *almost-regular* polyhedra. These polyehdra have congruent regular faces and define regular spherical tilings with the required global symmetries. The second approach uses minimal representations of the symmetry groups of the planotic solids and extends them using Kac-Moody algebra to generate denser 3D point groups. Tilings generated by this approach may be semi-regular. Finally, the tilings generated by both of these methods can be

locally subdivided without violating global symmetries.

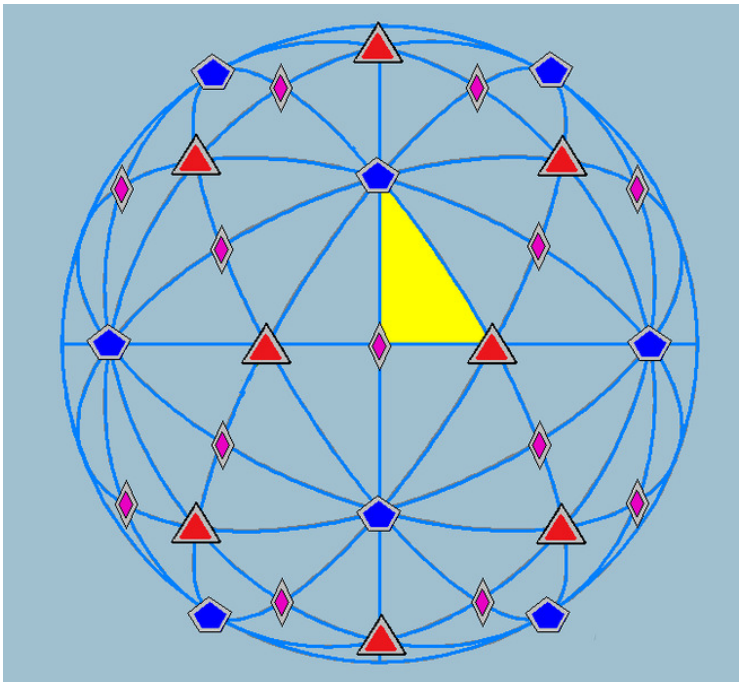


Figure 7.5: An example reflection group. The image shows a 2D projection of a sphere showing the fundamental region (yellow triangle) as well the resultant tiling of the sphere by repeated reflections of the fundamental regions. The three mirror planes (projected here as edges of the yellow triangle) have dihedral angles of 90, 60 and 36 degrees between them. This Kaleidoscope generates a tiling of the sphere with 12 locations of 5-fold, 20 locations of 3-fold and 60 locations of 2-fold symmetries.

7.3.2.1 Generating Tilings using Reflection Groups

Let us consider several distinct planes going through the origin. Each plane can be thought of as a one-way mirror such that the 'front' sides of the planes cut out a polygonal cone-like region out of a sphere centered at the origin. If finite number of repeated reflections by the mirrors of the polygonal region reproduces

the complete sphere then the planes define a finite reflection group. A reflection group is irreducible if they cannot be emulated using (compositions of) a smaller group. The planes and their reflections form circular intersections with the sphere. The circles also intersect each other. Interestingly the intersection of the circles are governed by the dihedral angle between the planes. If there are three planes and the angles are a_1, a_2 and a_3 such that $\pi/a_i = r_i$ is an integer, then the points where the circles intersect will have r_1, r_2 and r_3 fold rotational symmetries. For example, using the H_3 reflection group with angles equal to 90, 60 and 36 degrees we can generate the icosahedral symmetries. Furthermore, the number of copies of the fundamental region created by reflection is equal to the order of the corresponding symmetry group (see Figure 7.5). Coxeter [72] enumerated all finite irreducible reflection groups corresponding to the regular 3D solids as well as for other dimensions.

To generate a denser tiling of a platonic solid than that generated by the corresponding reflection group, we need to extend it. The extension is easier to model using the concept of root systems that we discuss now.

Generating Reflection Groups using Root Systems A root system is a set of vectors with one endpoint at the origin. It defines a basis for generating root lattices. For example, the unit length vectors along the Cartesian X and Y axis forms a root system which generates a 2D chessboard lattice. Each reflection group can be mapped to a unique root system (and vice versa) by defining vectors starting at the origin and perpendicular to the mirror planes. The root vectors pointing to the

Table 7.1: List of all finite non-decomposable crystallographic root systems relevant for 3D. We did not include G_2, F_4, E_6, E_7 and E_8 since they are only defined for specific dimensions and we are also not interested in their projections to 3D. In this table, n represents the dimension and e_i, e_j are the basis vectors of the coordinate systems of the corresponding dimensions.

Name	Order	Roots
A_n	$n(n+1)$	Project $e_i - e_j$ $(1 \leq i, j \leq n+1)$ to n-space
B_n	$2n^2$	$\{\pm e_i, \pm e_i \pm e_j\}$ $(1 \leq i, j \leq n)$
C_n	$2n^2$	$\{\pm 2e_i, \pm e_i \pm e_j\}$ $(1 \leq i, j \leq n)$
D_n	$2n(n-1)$	$\{\pm e_i \pm e_j\}$ $(1 \leq i, j \leq n)$

inside of the fundamental cone of the reflection group are called the fundamental roots. The entire root system can be generated by applying the reflections on the fundamental roots. For example, the root system generated by the H_3 root system generates points at the vertices of an icosahedron.

Root systems have been studied in detail by Crystallographers to describe lattices and quasi-lattices. There are only a finite number (for a specific dimension) of irreducible crystallographic root systems which we have listed in Table 7.1. However, the crystallographic root system allows only 2,3,4 and 6 fold symmetries. However, a non-crystallographic root system can be generated by projection from higher dimensional a crystallographic root system [233]. In particular, the icosahedral group can be generated by projecting the D_6 root system to 3 dimensions along a specific direction which maps the 6 roots of the D_6 system to 6 non-planar vertices of an icosahedron.

Extending the Root System to Generate Local Symmetries Reflection operations on the fundamental roots generate the vertices and tiling of the corresponding basic polyhedra. To generate a denser tiling, we extend the root system by adding another root. Given a root system with vectors $v_1, \ldots, v_k$, the additional root can be computed by first computing a Cartan matrix C such that each entry c_{ij} of the matrix defined as $2(v_i.v_j)/(v_j.v_j)$ where . indicates the dot product. Then this matrix is expanded by adding a new row and new column using the affine extension method of Kac-Moody algebra. The new row and column describes the additional root and its relationship to the others.

For example, for the icosahedral root system H_3, the extended root corresponds to a simple translation operation. Hence, when used in conjunction with the original roots and the corresponding reflection planes, it generates a series of new points at different radial levels. It can also be shown that the same set of points can also be generated by integer linear combinations of the original roots [193]. Note that the planes perpendicular to the extended set of points would intersect the sphere and generate new geodesic intersections which would correspond to local symmetries. The global symmetries remain unchanged since the original roots are still used, and the local symmetries are also distributed in a way that remain invariant under the global symmetry operations since they are generated by appropriate extension on the root system corresponding to the global symmetry.

Figure 7.6 summarizes the process of computing possible tilings.

Note that there are only a few (4) root systems relevant for regular or semi-regular tiling of a spherical shell. The root systems can be extended to generate

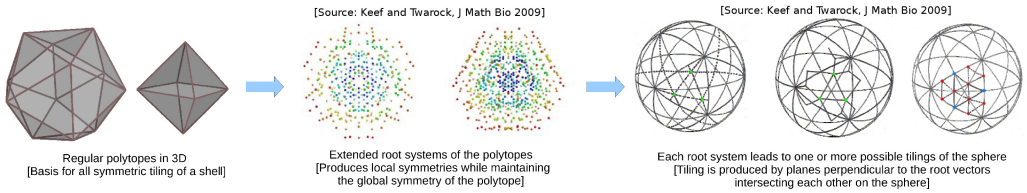


Figure 7.6: Pipeline of generating all possible regular and semi-regular tiling of spherical shells.

local symmetries, thereby increasing the number and density of tiles on the sphere. The number of unique extensions of each root system is also limited since many of the roots are colinear and hence does not affect the tiling. For example, [145] showed that there are exactly 342 possible unique extensions which can be made using the icosahedral root system, and among them there are only 26 unique outer shell configurations (the extended system produces vectors of many sizes, the outer shell refers to the distribution of the longest vectors).

7.3.3 Subdivisions

There is a fractal like process which divides a group of unique tiles into scaled copies of the same group (in other words, does not change the number of tile types or the vertex atlas). For example, in Figure 7.7 we show a set of tiles their decompositions.

7.4 Decorating the Tiles

Given a spherical tiling, it is easy to identify the set of unique tiles and unique tile-tile interfaces. Since the tiling must be edge-to-edge, neighboring tiles

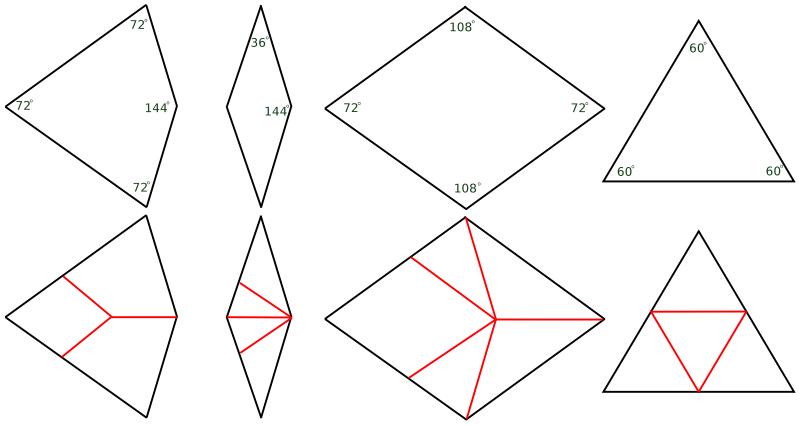


Figure 7.7: Subdivision of tiles while maintaining symmetry produces scaled versions of the same tile set

meet at their vertices. Each of these vertices correspond to cyclic symmetric locations. The decoration process deals with placing the 3D c-tiles onto each tile such that following hold.

1. The same type of tiles are decorated exactly the same way.

2. If more than one p-tile is needed to decorate a tile, then they must have local cyclic symmetry.

3. All c-tiles must have the same number and types of interfaces (with neighboring c-tiles) after placed as decorations.

4. The decoration scheme should minimize gaps between the c-tiles and generate as airtight a shell as possible, note that a perfect airtight shell may not be possible to make using a given building block (consider the case when a ball is given as a building block).

The first condition is trivial to fulfill. The second condition requires us to find favorable cyclic symmetric arrangements (c-tiles) of different orders using the 3D p-tiles. We accomplish this by optimizing a local score over a 4D search space (see Section 7.4.1). According to the third condition, if there are multiple types of tiles which need to be decorated using c-tiles with different order, then we also need to ensure that the selected c-tiles are consistent with each other. For example, if a p-tile is to be involved in both a 3-fold and a 5-fold c-tile, then it has to support both simultaneously. Hence, the footprint of the 3-fold interface and 5-fold interface must be disjoint. Finally, to fulfill the third and fourth condition, we define a global score which needs to be optimized by motions which leave the intra and inter-tile symmetries intact (see Section 7.4.3).

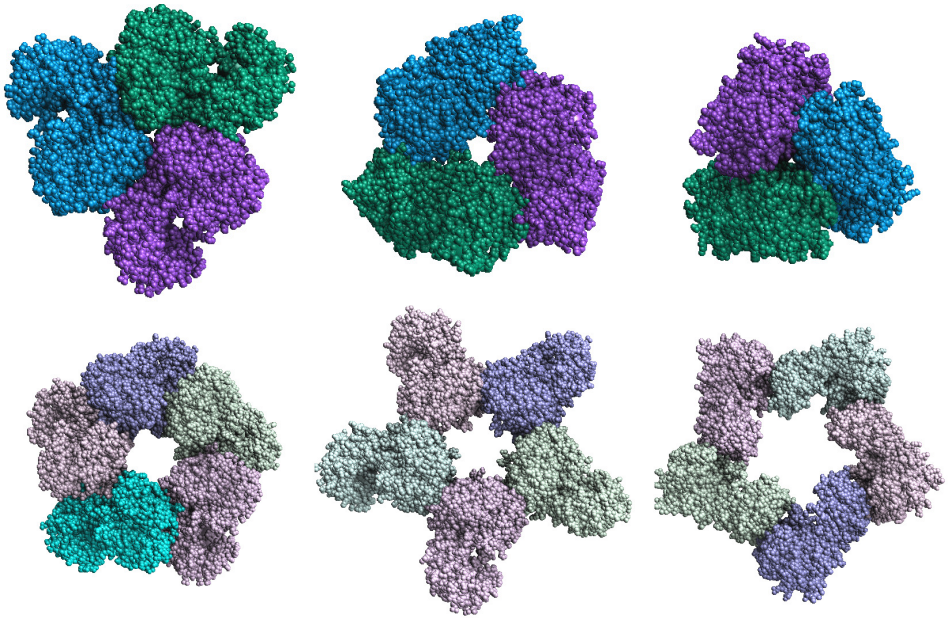


Figure 7.8: **Top row:** 3 best scoring trimers, and **bottom row:** 3 best scoring pentamers composed using copies of a single building block.

7.4.1 Computing Cyclic Symmetric c-tiles

Due to the symmetry, the number of unique tiles and unique tile-tile interfaces are very few for any given tiling. So, to identify a set of c-tile sufficient for the decoration, we need to search only a few types of cyclic symmetric configurations. In general, to find a favorable (under a specific metric/scoring model) arrangement between two 3D objects one needs to search over 6 degrees of freedom. For example, we can keep one stationary and rotate the other using 3 DOFs and then translate it using 3 DOFs. However, for the case of cyclic symmetric configurations involving q copies, 4 DOFs in total is sufficient since one translational DOF is lost because the centroids of the pieces would lie on a plane perpendicular to the symmetry axis and one rotational DOF is lost since two copies are related by a rotation of $2\pi/q$ around the symmetry axis.

We define a favorable symmetric arrangement as an arrangement which maximizes a local score.

Following the method of [198], we take one p-tile A and place its centroids at the origin and then sample the rotations Z and the X axis. For each rotational sample, we make a copy B and rotate it by $2\pi/q$ degrees around the Z axis and finally search for favorable relative positions of B w.r.t. A by sampling the space of translations on the XY plane. Hence, there are only 4 degrees of freedom. Once the relative position of the copy with respect to the other is found, the remaining ones can be computed by repeatedly applying the relative transformation. More specifically, if the relative transformation between the two is given by M, then we can generate the complete cycle as and form a macro-tile $O(A, M, q) = A \cup M(A) \cup$

$M^2(A) \cup ... \cup M^{q-1}(A)$. Since the scoring model is not expected to be perfect, we produce a list of k top scoring relative transformations for each order. For example, Figure 7.8 show the best three macro-tiles of order 3 and 5 respectively.

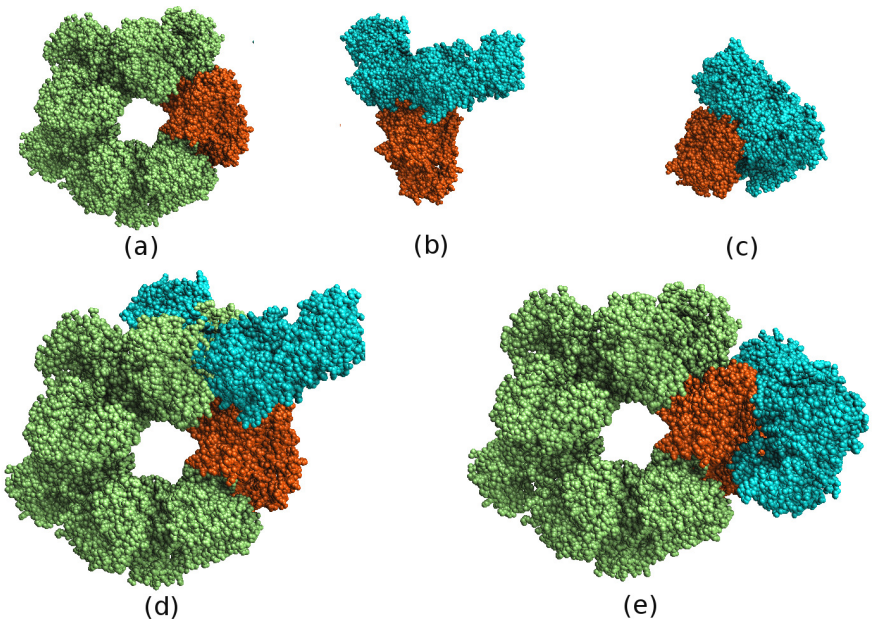


Figure 7.9: **(a)-(c)** A pentamer and two different trimer configurations using a given building block. **(d)-(e)** Testing for consistency between two c-tiles by aligning one component of the c-tiles and then verifying whether the remaining components from different c-tiles are clashing. There are clashes in (d). But in (e), we find a consistent pair.

7.4.2 Finding Consistent Pairs of Cyclic Symmetries

A simple technique to verify the consistency between a pair of c-tiles C_1 and C_2, is to verify if they induce disjoint interfaces on their constituent p-tiles, where an interface is defined as part of the surface of a p-tile which comes into contact with its neighboring p-tile while forming the c-tile. If the interfaces are not disjoint, then

clearly the same p-tile cannot be part of both C_1 and C_2 at the same time. However, parts of C_1 and C_2 may clash/penetrate even if the interfaces are disjoint. So, in this case, we further verify that the arrangement $O(A, M_1, q) \cup O(A, M_2, r)$ does not have severe penetrations/overlaps, wherre M_1 is the the relative transformations of order q which forms C_1, and M_2 is of order r and forms C_2 (see Figure 7.9 for examples). To compute the penetration scores, we first align both c-tiles such that the centroid of one p-tile of each lies at the origin and the principal components of the p-tile aligns with the coordinate axes. Then, we only need to verify that the remaining parts of the c-tiles are not clashing.

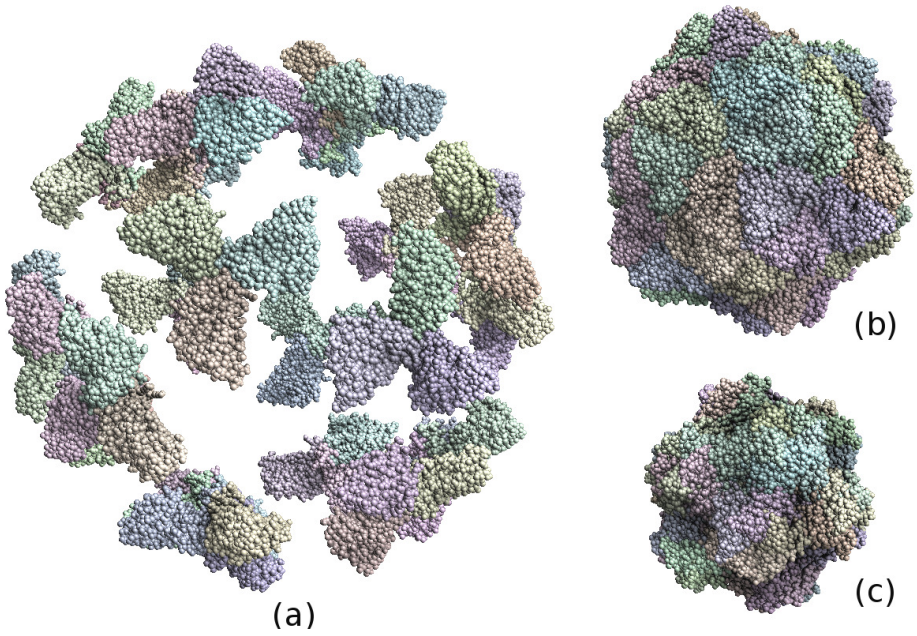


Figure 7.10: Searching for correct scaling of shell. Symmetric assemblies where the c-tiles are translated, (a) foo far away, (b) at the correct distance, and (c) too close. In (a) the decorations are not making any interface across tiles and in (c) decorations from different tiles are penetrating.

7.4.3 Global Optimization

As we have already discussed in the algorithm sketch, the decoration and global optimization phase starts with aligning a tile $(\mathbf{u_1}, \mathbf{u_2}, d, l, f)$ with a c-tile $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{c}, o, s)$. The shell produced by this alignment step fulfills the first three conditions stated above, but the fourth remains to be satisfied (see Figure 7.10 and Figure 7.11). It is addressed by sampling two degrees of freedom, a translation along $\mathbf{u_1}$ to scale the shell and a rotation around $\mathbf{u_1}$ to possibly improve interfaces across tile-boundaries.

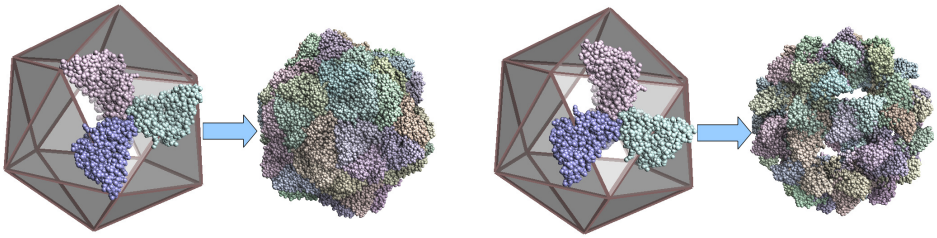


Figure 7.11: Searching for correct orientation of the decoration. Two assemblies are shown here, both are at the same scale. The orientation of the decoration in the left figure is perfect for creating complementary interface with the decorations on neighboring tiles. However, the orientations of the decorations on the right figure causes penetrations as well as leaves a lot of empty spaces.

We initially set broad limits for the translations such that the scaled l is within $[t/2, 2t]$ where t is the farthest point from c if the p-tile is projected onto the plane perpendicular to $\mathbf{v_1}$. We coarsely sample translation steps T between these limits. For each translation, we also coarsely sample rotations R within a band $[-\pi/f, \pi/f]$ and for each sample in (T, R) we compute two scores. One score is designed to penalize penetration the other is to award proximity. Both can be computed together using complex valued affinity functions defined on a double-skin

representation of the surfaces (similar approach have been used by others, for example [60, 170]). Note that, because of the symmetry it is not necessary to score all the interfaces between all the tiles. Rather, it is sufficient to compute the score for one copy of each unique type of inter-tile interfaces. Since this list is pre-computed, we simply compute the relative orientation of the corresponding c-tiles which decorate those tiles and compute the score. The tilings generated by the *almost-regular* polyhedron needs only a single type of tile and tilings based on extended root systems require only a limited number (2 for icosahedral). The total number of unique local interfaces are also bounded. For instance, if dihedral angles are ignored (assuming that the tiles are projected to a sphere), then there are exactly 2 types of neighborhoods for the regular case, and upto 16 types (usually much smaller) of vertex atlases for the semi-regular case. So, the number of unique interfaces to score is a small constant only.

The coarse sampling quickly recognizes all scales which are either too big or too small and retains a very narrow band. Then we perform a more refined sampling within that band.

7.4.3.1 Heuristics to Bound the Search Space

In most situations some information is known or some design goals are prespecified which can help restrict the range of allowed subdivisions and/or the scale parameter m. For example, we consider the following special cases-

- The expected diameter, or volume (either inner, outer or average) of the shell is known.

- The expected number (or range of numbers) of proteins on the capsid is known.
- The EM density map of the capsid is known, giving both the diameter as well as the thickness of the shell.

Definition 7.4.1 (Order of a tiling/layout)**.** We define the order $n^{\mathcal{L}}$ of a tiling $\mathcal{L}$ as $\sum_{t_i \in \mathcal{L}} f_i$, where f_i is the order of internal symmetry of the tile t_i.

Lemma 7.4.1. *If the expected number of proteins on the capsid is within* $[n_{min}, n_{max}]$*, then the space of possible tilings is restricted to all tilings such that* $n_{min} \leq n_{\mathcal{L}} \leq n_{max}$.

We are also able to bound the space of tilings and decoration based on the shape of the p-tiles (proteins). We define the shape using the principal components, in other words by replacing the p-tile with a ellipsoid. Let the three main diameters be $2a, 2b, 2c$ (in decreasing order of magnitude). The most number of proteins can be used to tile the shell if their smallest cross-section is orthogonal to the u_1 vectors. We estimate the footprint as $2bc$ (allowing penetrations of the ellipsoids). The thickness of the shell is estimated to be $2a$ in this case. On the other hand, the sparsest configuration is achieved when the thickness is $2c$. However, the lower bound would depend on the global and local symmetry requirements which dictates the number of interfaces a p-tile must have. For example, if we are using a triangular tiling, then the sparsest case is when a is equal to the in-radius of the small triangles, and hence each p-tile leaves a $\sqrt{3}a^2$ size footprint.

Definition 7.4.2 (Footprint of a p-tile)**.** If a p-tile is approximated as a ellipsoid of diameters $(2a, 2b, 2c)$, then for a triangular tiling the maximum and minimum footprint of the p-tile on the surface of a shell is respectively approximated as $F^p_{min} = 2bc$ and $F^p_{max} = \sqrt{3}a^2$.

Similar bounds can be derived for other types of symmetric tiles as well.

Definition 7.4.3 (Footprint ratio of tile)**.** We define the footprint ratio F^t of a tile as $a_t/(a_S \times f)$ where a_t is the surface area of the tile and a_S is the surface area of the entire shell.

Definition 7.4.4 (Footprint ratio of a tiling/layout)**.** We define the footprint ratio $F^{\mathcal{L}}$ of a layout $\mathcal{L}$ as the average footprint ratio of all its tiles.

Depending on the amount of prior information the search space can be restricted in the following ways-

- Given a p-tile and an expected surface area A_S of the shell (as either expected radius, area or volume), the space of possible tilings is restricted to all tilings whose footprint ratio satisfies $F^p_{min}/A_S \leq F \leq F^p_{max}/A_S$. Also, the scale of the shell has to be within the range defined as $R - 2a \leq m \leq R + 2a$, where R is the expected radius of the shell.

- Given a p-tile, the expected surface area A_S of the shell and a range $[n_{min}, n_{max}]$ for the number of proteins, the space of tilings is restricted to all tilings such that $n_{min} \leq n_{\mathcal{L}} \leq n_{max}$, and the possible orientations of the p-tiles on the

surface is also restricted such that the footprint of the p-tile $F^p \approx F^t$ the footprint ratio of the chosen tiling.

- If both the expected surface area A_S as well as thickness of the shell is known (e.g. from the EM density map), then the possible orientations of the p-tiles on the surface is also restricted such that the average thickness of the p-tile in a direction orthogonal to the surface is approximately equal to the expected thickness. Moreover, if F^{p*}_{min} and F^{p*}_{max} define the range of footprints for the allowed set of orientations of the p-tile, then space of possible tilings is restricted to all tilings whose footprint ratio satisfies $F^{p*}_{min}/A_S \leq F \leq F^{p*}_{max}/A_S$. Also, the scale of the shell has to be within the range defined as $R - 2a \leq m \leq R + 2a$, where R is the expected radius of the shell.

7.4.4 Scoring

The design objective of both the local and global scoring functions is to distinguish between acceptable and non-acceptable solutions as well to rank the acceptable solutions. Our search procedure can be used in conjunction with any scoring function as long as it conforms to this principle.

We speed up the scoring by using FFT based fast summation techniques which defines affinity functions on the two components such that the convolution of the affinities is the score and then computes the score for all translational samples (or rotational samples if spherical FFT is used [215]) by two forward and one inverse FFT computations. Further details on FFT based fast summation can be found in [142].

Some scoring terms are however not easy to decompose and express as convolutions. So, as a rule of thumb, we use such terms only as filters and for refinement steps. We have also developed fast multipole implementations for several scoring terms which only depends on pairwise summations over neighboring surface points at the interface of two components. The complexity of the algorithms are linear to the size of the interface.

Since scoring terms need to be custom-made based on the application and nature of the building blocks, we defer the description of different terms until Section 7.5 where we introduce them in the context of protein-protein interfaces.

7.5 Application in Molecular Biology

Viruses are an amazing example of symmetry and economy of design. Viruses usually have some generic material (DNA/RNA) packed inside at least one layer of a membrane or shell. This shell, usually called the capsid, protects the genetic material as well as acts as a delivery vehicle. The shell is formed by multiple copies of a protein assembled together. The shell disassembles when the virus reaches the host (victim) cell's nucleus to release the genetic material. The genetic material encodes a blueprint for making new copies of the protein which forms the shell. Many copies of the DNA/RNA as well as the proteins are manufactured by piggybacking the molecular machinery of the host cell and in the process depleting all its resources. Then the newly created proteins automatically assembles themselves into the same shell structure. For us, the most relevant aspect of this process is that the shells are generally icosahedral and the number of proteins in each shell

often exceed the global symmetry order, and hence produces local symmetries. So, their shells falls within the categories of the shell structures our algorithm can produce. Hence, we chose several viruses to verify whether the shells generated by our algorithm matches the experimentally observed shells and their symmetries.

In this section we first briefly discuss the current knowledge on the structures of viral capsids. Then we describe the properties of the proteins (puzzle pieces) and explain how the properties dictate the design of scoring functions. Finally we describe the results of applying our algorithm and contrast it with real data acquired by cryo-electron microscopy and x-ray crystallography.

7.5.1 An Overview of the Symmetry of Viral Capsids

Though there are some viruses whose capsids are shaped like tubes, filaments or sometimes without a fixed shape, the shape found most predominantly is the spherical one. It was observed long ago that all spherical viruses have icosahedral symmetry. We have already seen that the icosahedral or dodecahedral symmetry provides the highest symmetry order among all regular polytopes and hence allows the construction of larger shells using the same p-tile. Recent studies have also shown that icosahedral shells are also the most stable both in terms of biochemical interactions as well as against external pressure [282]. However, the presence of capsids of different sizes suggested that the number of copies of proteins used in each are different and it a general characterization to address this issue was first put forward by Caspar and Klug in their seminal work in [55].

The Caspar-Klug model introduced the concept of local symmetries and

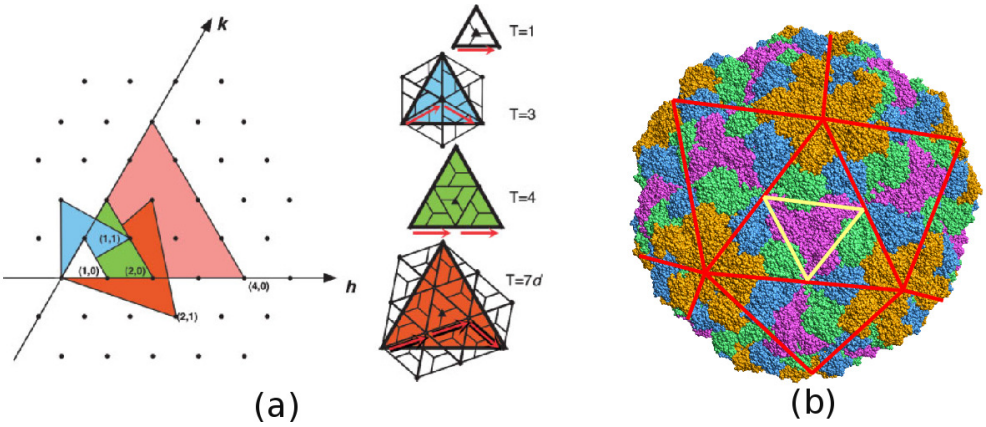


Figure 7.12: **(a)** The Caspar-Klug model lays out the icosahedral faces on a hexagonal grid. Larger triangles on the grid contain smaller triangles. With one corner of the large triangle fixed at $(0, 0)$ and another at (h, k), it would contain $T = h^2 + k^2 + hk$ small triangles. The model decorates each small triangle with 3 proteins. Icosahedrons made by larger triangles hence contains $60T$ proteins. **(b)** An example capsid (Nudeaurelia Carpensis Virus) with $T = 4$.

quasi-equivalence. They showed that the triangular faces of an icosahedron can be unraveled and placed on a regular hexagonal grid such that the corners of the icosahedra have integer coordinates. If we imagine that one corner lies at the origin, then the smallest such triangle we can lay out on this grid would have the other endpoints at $(1, 0)$ and $(0, 1)$. Caspar and Klug proposed that this small triangle is composed of three copies of the protein placed at each corner resulting in a capsid with 60 proteins. However, if the size of the triangle is increased such that the corners still have integer coordinates, then these large triangles would contain many smaller (unit sized) triangles. If a large triangle has two corners at $(0, 0)$ and (h, k) coordinates, then the number of small triangles in it is exactly $h^2 + k^2 + hk$. This number is called the T-number. According to the Caspar-Klug theory, a capsid has

exactly $60T$ triangles, 12 vertices with 5-fold symmetry corresponding to corners of large triangles, and $10(T-1)$ locations of 6-fold symmetry where small triangles meet each other (see Figure 7.12).

According to our formulation, the Caspar-Klug model produces tilings of the icosahedron with only a single type of tile, a triangle. Different T number changes the density of the tiling, but the types of symmetry requirement at the vertices remain the same.

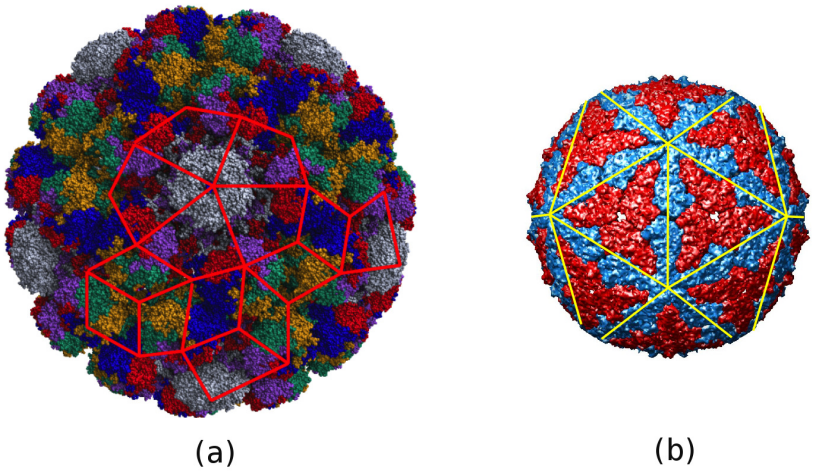


Figure 7.13: **(a)** Simian Virus, pseudo $T = 6$. **(b)** The inner shell of the Rice Dwarf Virus which is pseudo $T = 2$. Both of these only allows aperiodic tiling. We show one such tiling in (a). In (b) we show a failed attempt tile using Caspar-Klug like triangles.

However, Caspar-Klug theory fails to explain the capsid structure of some viruses in the Papilloma, Papova and Polyoma families. For example, the Simian Virus 40 (of Papova family) is predicted to have T=7 capsid, i.e 420 proteins with 12 locations with 5-fold symmetry and 60 locations with 6-fold symmetry, but in

reality it has only 360 proteins and 72 locations with 5-fold symmetry. Another example is the inner layer of the capsid of the Blue-tongue virus which has 120 proteins or T=2, a fact considered impossible under the Caspar-Klug model. See Figure 7.13.

These viruses must be tiled using aperiodic tiles. It has been shown that all such capsids can be tiled using either rhombus tiles, or a rhombus and kite tiles. The rhomb has two angles each of $\pi/5$ and $4\pi/5$. The kite has three angles of $2\pi/5$ and one angle of $4\pi/5$. Notice that the extended root system is able to produce such tilings as well. The decoration of these tiles is done by placing two proteins on the rhomb tiles and placing three proteins on the kite tile, leaving the corner with the largest angle empty.

We have studied several viruses with differing sizes and requiring different kinds of tilings and present the results after describing the scoring model.

7.5.2 Scoring Function Design based on Physico-chemical Properties of Protein-Protein Interfaces

Proteins are polypeptides, or a sequence of amino acids (peptides). There are 20 naturally produced amino acids for proteins. Each amino acid has different number and arrangements of atoms and hence different chemical properties. They have different polarities, different affinity towards the solvent (cellular environment) surrounding the protein. The interactions with the solvent, other amino acids on the same protein as well as other proteins makes the polypeptide chain to fold and coil and finally produce a more compact 3D shape. This final shape is of-

ten referred as the tertiary structure of a protein. For our purposes, we assume that the tertiary structures are given to us. When proteins form complexes, they undergo further changes and forms quaternary structures. Here, we consider the tertiary and quaternary structures to be the same (i.e. assume rigidity).

Proteins form complexes to minimize their free energy. If the net change in free energy due to the binding of the proteins A and B is $\delta E = E(A+B) - E(A) - E(B)$, then the stability of the complex is proportional to $e^{-\delta E}$. Hence, we would like to design our scoring function $\mathcal{F} \rightarrow \mathbf{R}$ such that it is positively correlated with δE and hence its minimization leads to favorable complex structures.

There are several ways to model the free energy. From the most comprehensive quantum mechanical models to purely empirical statistical potentials. We choose to use a a collection of terms including a semi-empirical model called the Gibbs free energy model which is based on statistical thermodynamics, and several empirical potentials. We have found that after training [208] based on large set of positive and negative examples, a combination of these terms can successfully distinguish between correct and incorrect configurations/interfaces [65]. See Chapter 5 for details.

7.5.2.1 Scoring Terms Based on Statistical Thermodynamics

Under this model, the free energy E of a molecule of a molecule (or a complex) is given by

$$E = E_{MM} + G_{sol} - TS \tag{7.5.1}$$

E_{MM} is called the molecular mechanical energy. It represents energy due to the atom-atom interactions among atoms of the molecule(s). G_{sol} is the solvation energy representing the interaction of the molecule(s) with the solvent. T is the temperature and S in the entropy. The change of TS is too time-consuming to compute accurately and in most cases, the change is negligible.

The molecular mechanical energy is decomposed into bonded and non-bonded interaction terms (see Equation 7.5.2). The bonded energy terms measure the energy required to deviate from an optimal bonded position (for example, changing the length of a bond). These terms are considered constant when it is assumed that the proteins are rigid.

$$E_{MM} = \underbrace{E_d + E_{theta} + E_{phi}}_{\text{bonded interactions}} + \underbrace{E_{vdw} + E_{coul}}_{\text{nonbonded interactions}} \tag{7.5.2}$$

$$E_{vdw} = \sum_i \sum_{j>i} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} \right) \text{ and } E_{coul} = \sum_i \sum_{j>i} \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}}, \tag{7.5.3}$$

However, the change of the non-bonded interactions (Equation 7.5.3) are very relevant to the scoring. The first term is called the VDW interaction which represents short range attraction (of electron of one atom to the proton of the other) and a very high repulsion if the distance r_{ij} gets too close (representing the positive nuclei of the atoms coming too close). The a_{ij} and b_{ij} are two weights which have been determined based on quantum mechanics for different types of atom-atom pairs. This energy encourages two molecules to come close to each other such that

their shapes complement each other, but does not allow penetration. This term can also be approximated using simple geometric complementarity. The second term of the non-bonded interaction is the long range electrostatic interaction between two molecules. This term is dependent on the charges q as well the distance dependent dielectric $\epsilon(r_{ij})$ of the solvent.

Several models for the solvation energy G_{sol} have been proposed. Here, we present an implicit solvation energy model called the GBSA model [105].

$$G_{sol} = \underbrace{G_{cav} + G_{vdw}}_{\text{nonploar}} + \underbrace{G_{pol}}_{\text{polar}} \tag{7.5.4}$$

G_{cav} depends on the volume of the protein and the exposed surface area and G_{vdw} is the Van der Waals interaction between exposed atoms and solvent atoms. The polar part E_{pol}, can be approximated using *Generalized Born* (GB) theory [245] as-

$$E_{\text{pol}} = -\frac{\tau}{2} \sum_{i,j} q_i q_j / \sqrt{r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}}, \tag{7.5.5}$$

where $\tau = 1 - \frac{1}{\epsilon}$, and R_i is the effective Born radius of atom i. There are other models for E_{pol} as well, e.g. [250].

7.5.2.2 Knowledge-based Scoring Terms

We have also designed and used several light-weight scoring models based on observed properties of the interfaces of protein-protein complexes. These models have the advantage of being computationally cheap and they can also be trained for specific types of interactions separately. Below we list the terms we have used.

- **Interface Propensity:** The relative probability of a residue (amino acid) appearing on the interface, given that it appears somewhere in the protein is defined as the interface propensity of the residue. Per-residue interface propensity values were computed in [134] which are based on the relative frequencies of different residues in the interfaces of a set of 63 protein-protein complexes. For each candidate assembly, we compute the residues which are on the interface and sum up their interface propensities.

- **Steric Clash:** Due to extremely high repulsion forces between the nuclei of two atoms, they cannot come too close to one another. We penalize all arrangements where the center of atom gets too close to the center of another atom.

- **Interface Area:** Larger interface area reduces the interactions with the solvent and also reduces the DOF of more atoms and reducing the entropy. So, we score assemblies with larger interface areas higher.

All of these terms can be computed as simple linear sums over the atoms which are on the interface of a complex. We have developed efficient data structures [14, 24] for computing such sums in time linear to the size of the interface.

7.5.3 Results

Below we present the results of applying our algorithm to some biological viruses with the aim of reproducing the capsid structures seen in nature as well to predict new compact shells of different sizes using the same p-tile. In the results

below we have also mentioned the global scores of the assemblies which is defined in terms of the three knowledge-based terms introduced above. If the interface propensity score is IP, the number of clashing atoms is NC and the interface are is IA, then the score is defined as $w_{IP} * IP + w_{NC} * n - NC + w_{IA} * IA$. NC scores can range from 0 to n^2 where n is the number of atoms, IP usually ranges between 0 and 25, and IA usually ranges between 0 and 0.3 (it is a ratio of the interface area and the total area).

7.5.4 Reproducing Known Shell Structures

Assembling a T=1 Shell We took one protein out of the 60 which makes a $T = 1$ shell for the Tobacco Necrosis Virus (TNV). The capsid structure, reconstructed from microscopic images by [135], is available as a atomic resolution in the protein data bank [2].

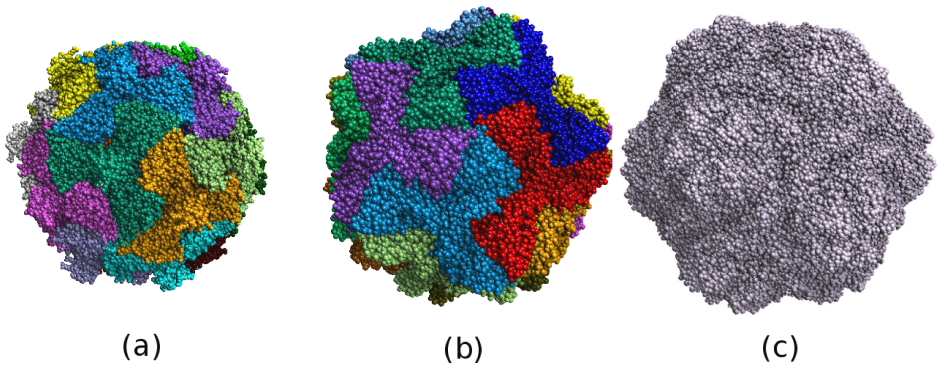


Figure 7.14: **(a)-(b):** Top 2 predicted models for the Tobacco Necrosis Virus. **(c):** The experimentally observed capsid. Our second prediction reproduces the original structure.

A $T = 1$ configuration is the simplest instance of icosahedral packing and

we can either use a layout based on dodecahedral point set or a layout based on icosahedral point set. In either case, only one type of cyclic symmetric oligomers are needed. Figure 7.14 shows the top two predictions produced by our algorithm with scores 447.5756 and 355.7908 respectively.

Assembling a T=4 Shell The capsid of the Nudaurelia Carpensis Virus (NCV) [123] is formed by 240 copies of a single protein. Hence, it is modeled as a $T = 4$ capsid. We use a tiling with only triangular tiles generated from the icosahedral system via decomposition. Figure 7.15 shows the top two predictions whose scores are 163.9997 and 91.2074 respectively.

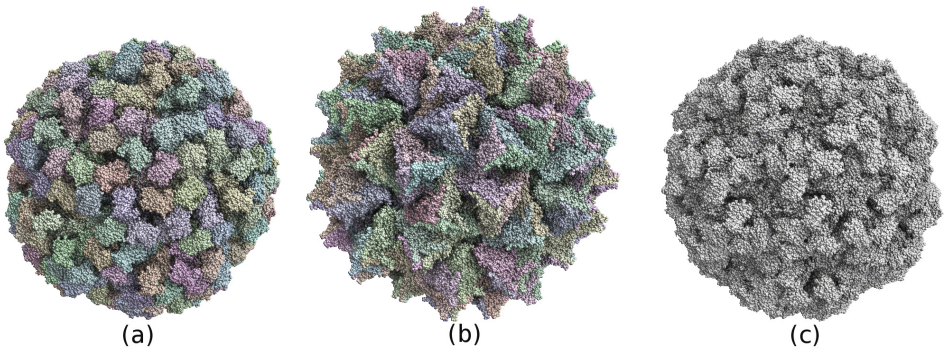


Figure 7.15: **(a)-(b):** Top 2 predicted models for the Nudeaurelia Carpensis Virus. **(c):** The experimentally observed capsid. Our first prediction reproduces the original structure.

Assembling a T=2 Shell The inner shell of the Rice Dwarf Virus (RDV) [185] is composed of 120 proteins. We mentioned before that this cannot be tiled using periodic tiles. The tiling which matches it best is an extension of the dodecahedral roots which produces a tiling with rhombs. There are 60 rhombs in total such that

five such rhomb meets a the vertex on the axis along one the dodecahedral root vectors.

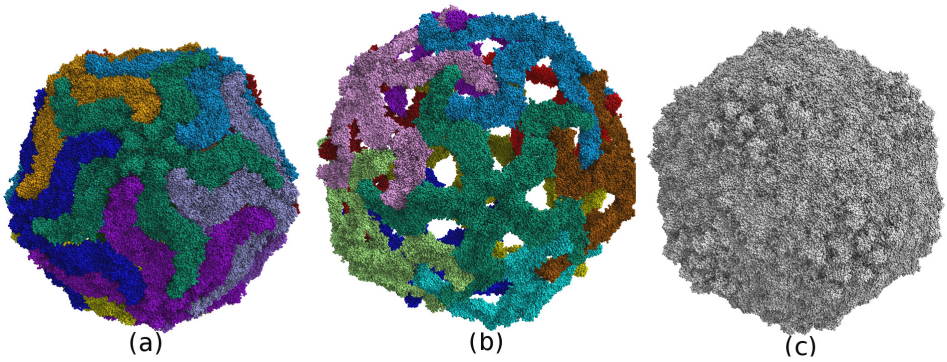


Figure 7.16: **(a)-(b):** 2 predicted models for the Rice Dwarf Virus (inner shell). **(c):** The experimentally observed capsid. Our first prediction reproduces the original structure. We show the second figure as an example case where the macro-tiles used for decoration cannot be transformed in anyway to produce tighter interfaces without causing penetrations.

Our algorithm handles this by composing consistent sets of 5 fold and 2 fold symmetric configurations and then decorating using these (Figure 7.16). The score of the first pose shown in Figure 7.16 is 591.6053 and the score is 71.2835 for the other one. The other result is included to highlight that even with consistent macro-tile, sometimes it is not possible to get a compact shell. However, the interface based affinity functions do have the ability to rank such cases lower.

Assembling a T=13 Shell The outer shell of the Rice Dwarf Virus (RDV) [185] is composed of 780 proteins. This corresponds to the T=13 tiling. T can be 13 for h=1, k 3 or h=3, k=1, giving rise to two different chirality of the tiling- one of them matches the correct chirality (Figure 7.17 (top-right)), and the other do not (Figure

7.17 (bottom-right))

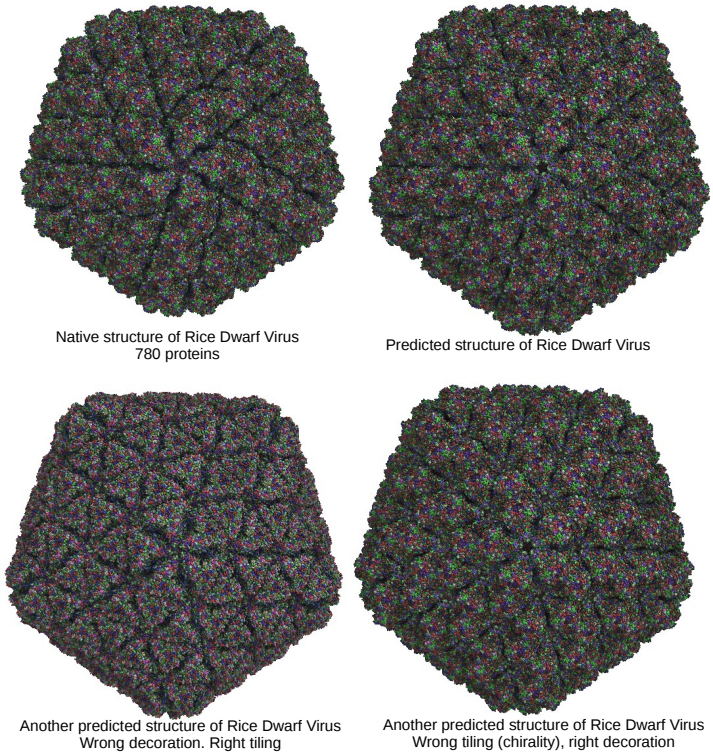


Figure 7.17: The top row shows the correct and a predicted structure for the rice dwarf virus outer shell. The bottom row shows two predictions which are incorrect due to wrong geometry (left), and wrong topology/tiling (right).

This is a particularly difficult problem for an automated search and ranking procedure to handle, since for geometrically and energetically there is very little difference, if at all, between the interfaces between the proteins in the two different models.

7.5.5 Modeling Virus Shells that Match EM-maps

This is one application where our technique really shows its practicability. In many cases, it is possible to get a coarse representation of the virs capsid using Electron-microscopy imaging, but the resolution is sufficient to identify individual atoms of even groups of atoms with certainty. The sequence of the protein is also easy to get. However, it is extremely difficult to crystallize the protein and get an atomic resolution model.

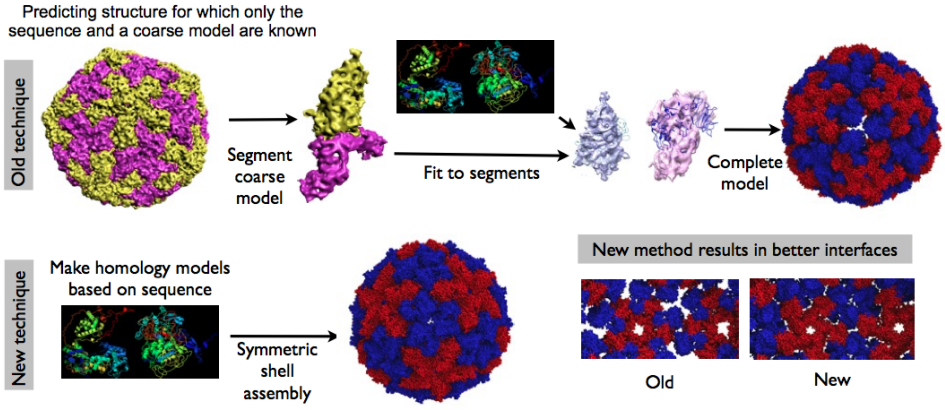


Figure 7.18: EM-based prediction of viral shells

In this type of scenarios, the current approaches like [91] segment the EM-map based on prior knowledge about the symmetry, generate homology models (computationally predicted template based models) for the protein using the sequence, and then fit copies of the protein into the segments to produce the whole shell (See Figure 7.18). However, this process is error-prone due to unreliability of the segmentation, and often result in poor interfaces between the assembled proteins. On the other hand, using our approach, we assemble the wntire capsid based

on the interface quality, and by decorating a tiling of the known symmetry and size while ensuring that the assembled shell fits to the density. This approach results in better interfaces.

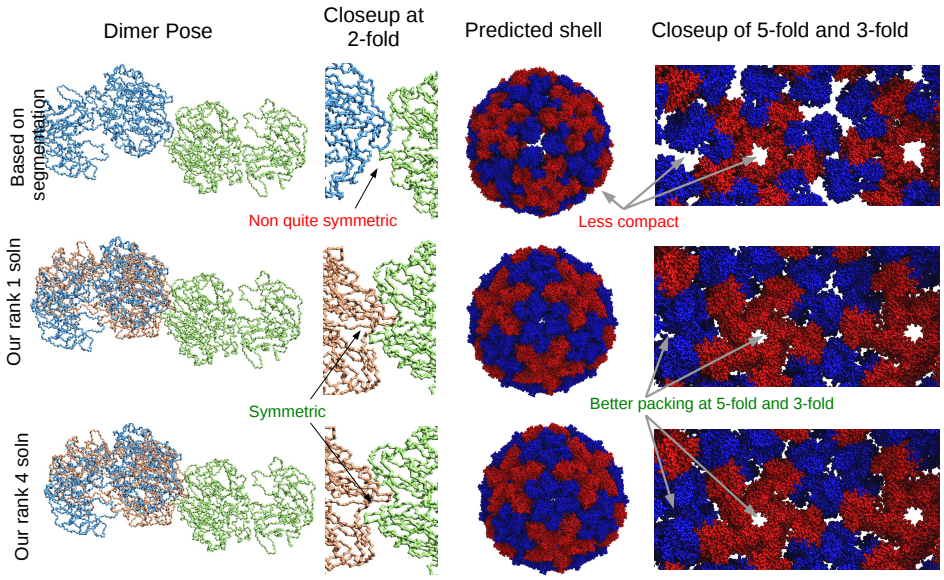


Figure 7.19: Comparison of segmentation-based methods with ours

For example, we applied both of the techiques to model the Helminthosporium victoriae virus 190S whose EM map and sequence were available. For the first method, VolumeRover [285] was used to segment the EM-map and then homology models produced by swiss-model were fit into the segments using PF2Fit [23, 41]. The results are shown in the top row of Figure 7.19. The middle and bottom row of the figure shows the results obtained by applying our docking based approach. Our approach not only results in more symmetric interfaces, it also produces more compact interfaces at 2-, 3-, and 5-fold locations.

7.5.5.1 Assembling Multiple Sized Shells Using the Same Protein

In the above sections, we focused our search based on prior knowledge of the target. In figure 7.20 we show that our algorithm can easily produce shells of different sizes using the p-tiles. Here we used the same protein, and the same c-tile but decorated tilings of different complexities and reported the highest scoring models for each size.

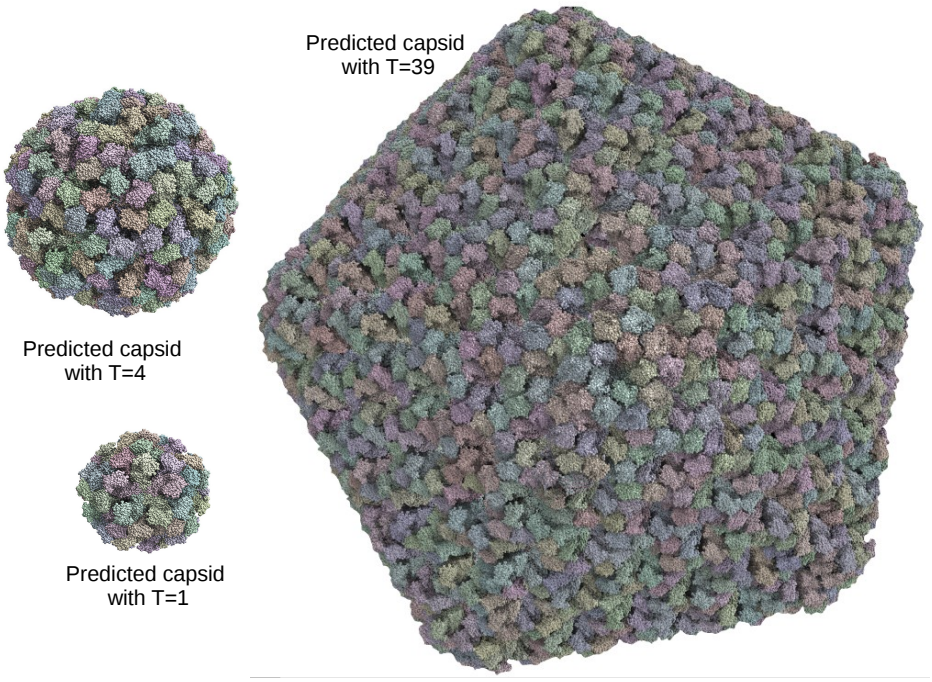


Figure 7.20: Shells of different sizes using the same protein

7.5.6 Running Times

Computing lists of possible c-tiles is the most time consuming part of the algorithm. Given a tiling and lists of c-tiles, decorating the tiles usually finishes

Table 7.2: Comparative running times for different sized datasets.

Dataset	Number of points	c-tile time (minutes)	Number of tiles	Decoration time (seconds)
TNV	4281	12.4	T1 (20)	12.55
			T4 (80)	11.93
NCV	12402	21.2	T1 (20)	37.78
			T4 (80)	38.27
RDV	76530	67.4	T2 (60)	65.33

within one minute even for really large p-tiles. As expected, the decoration time does not depend on the total number of tiles, rather on the number of unique interfaces and of course the size of the 3D tile. Table 7.2 presents the running times as well as provides a glimpse of the size of each dataset. The experiments were performed on a machine with 12GB memory.

7.6 Conclusion

Our polynomial time, 3D shell assembly prediction solution has several novel features. By characterizing our search space to symmetric decorations of 2D periodic and non-periodic tesselations, we are able to reduce the otherwise potentially huge combinatorial search space to only 6 motion parameter dimensions, and independent of the number of 3D tiles used to create the assembly. Furthermore, the use of finite symmetry groups of polyhedra, and their root systems and their extensions, yields a generalized procedure of subdividing such spherical tilings to have increased complexity while preserving local symmetries. This allows us to

predict 3D shell structures of varying sizes, typically relevant for tiled domed architectures. We have also successfully applied this procedure to the prediction of spherical protein shells of biological viruses of different sizes, all of which exhibit icosahedral symmetry. Our implemented technique has been successful in predicting the tiling/packing of known and experimentally reconstructed protein shell structures of a fixed size, and moreover predicts possibilities of different sized shells. In this sense our method provides emergent solutions for biological discovery. Our shell assembly predictive solution based on mixed biophysical and knowledge based models, is specially noteworthy in light of the complicated biophysical/biochemical interactions between protein structures in nature, and provides testimony to the stability of such tiled shell structures. Our 3D assembly method is also generalizable to flexible 3D assemblies, wherein the p-tiles are deformable and of parameterized flexibility. Our search space only increases from 6D with the additive deformation dimensions of a single p-tile.

Chapter 8

Multi-resolution and Data-driven Modeling and Validation of Molecular Assemblies

Elucidating functionally important structural details at finer resolutions of highly flexible proteins or proteins with large variable domains remains an elusive task. It is difficult to discern a complete and high resolution structure of such proteins in their native state. For such proteins, x-ray crystallography reports atomic resolution structure but often cannot resolve the variable regions, or has to remove those regions entirely before the protein becomes amenable to crystallization, a process which can also induce conformational changes to the retained part as well. On the other hand, electron-microscopy (EM) can often produce lower resolution model of the entire protein in its in-vitro state. Here, we report a computational protocol that not only model structures of protein complexes involving multiple chains while optimizing both inter-protein interaction potentials as well as internal structure of each protein, but also model the missing variable parts at atomic resolutions. The protocol uses known EM models are coarse constraints to restrict and validate the conformational search, uses known partial structures and prior knowledge on binding sites to train/learn high confidence scoring model tailored to the target complex. It also uses a consensus of multiple state of the art structure vaidation protocols for structural refinement.

This comprehensive multi-resolution protocol promises to greatly accelarate structure and functional study of molecular complexes, as evidenced by our application of the protocol in determining the first rigorously validated and complete atomic resolution model of HIV spike protein gp120 in complex with 17b and CD4, a complex which is the precursor to HIV infection of a host cell. gp120 have several variable regions which are known to play an important role in binding to CD4 and then CCR5 which acts as a precursor to infection. But the precise structure and configuration of these variable regions' interaction with CD4 and CCR5 have not been resolved yet by x-ray crystallography. We found that the method was extremely successful in predicting near-native configurations for a large set of protein-protein complexes involving the HIV spike protein gp120 and various antibodies. In 18 out of 32 cases, our method picked the lowest RMSD solution as the top solution. Further, we showed that our validated scoring model can be combined with off-the-shelf threading and energy minimization algorithms to produce reliable models of the variable regions.

8.1 Introduction

A key component of the study of molecular interactions is to resolve the structure and biophysical properties for the entire complex as well as for the interfaces (regions of contact) between proteins. Resolving the structure of a molecule at the atomic resolution (2Å) is not a trivial problem. Several experimental protocols have been developed, most prominently X-ray Crystallography [146], Nuclear Magnetic Resonance (NMR) [31, 274], and Cryo electron microscopy (EM) [97].

According to the protein data bank repository 85372 atomic resolution structural models have been resolved using x-ray crystallography, 10512 models by nuclear magnetic resonance techniques as of the end of December 2013. X-ray crystallography, first applied for proteins in 1958 to resolve the structure of the Myoglobin molecule [146], works by crystallizing a purified sample of the protein-complex and then analyzing the scattering pattern of x-ray shot at the specimen. However, many proteins, for example proteins with disordered regions are difficult to crystallize, and most insoluble membrane proteins, x-ray crystallography cannot be applied [165], leaving a large portion of the protein landscape beyond its reach. Another drawback of x-ray crystallography is the possibility that the crystallization process induces conformational changes in the protein and the resolved structure is not the natural state of the protein. Nuclear magnetic resonance (NMR) [31, 274] is applicable to an even smaller set of proteins since the magnetic response from the nuclei of the atoms, which is used to resolve their position and type, tends to decay too quickly for large molecules. The largest structure resolved by NMR [94] is still much smaller than macromolecules like viral capsids and ribosomes. Cryo electron microscopy (EM) is an alternate technique which while does not resolve atomic positions, it is however able to provide a volume occupancy model for a protein-complex in its natural state. Experimental protocols like isothermal titration calorimetry (ITC) are used to analyze the thermodynamic properties of protein-protein or protein-small molecule interactions [200].

Our comprehensive computational protocol (Figure 8.1) completes partial atomic resolution x-ray structures by integrating available data from other x-ray

structures, coarse resolution EM models, as well as stoichiometry and binding site information. It first generates an ensemble of feasible structural models for each missing fragment, and then clusters, ranks and assembles them into complete models while optimizing a multi-term scoring function which takes into account the agreement of the complete structure with the EM model, the feasibility of the interfaces between the fragments and other ligands, and stereochemistry. Our protocol bridges a gap between ab-initio loop/fragment modeling and threading/homology modeling. Ab-initio loop modelers can accurately predict or model loops which are fairly short, and fails for longer loops. For example, among the most popular loop modelers ModLoop [95] and FREAD [63] support loops up to 20 residues long, FALC-Loop [158] supports loops between 4-12 residues, and YASARA, based on Canutescu and Dunbrack's algorithm [53], supports loops of up to 18 residues. This makes it impossible to directly use such tools to model the large variable regions (for example, the V1V2 loop of gp120 is 66 residues long). Threading and homology modeling, on the other hand, have been successful [91, 230, 273] in modeling small to medium sized proteins (about 100 residues). This range is sufficient for modeling to missing portions of most complexes. However, the current tools does not take into account the interplay between multiple chains in a complex, and hence are not applicable for modeling complexes with more than one chain. There are some recently reported integrative modeling tools [121, 156, 263] which can handle multiple chains. They separately model components (protein chains or RNA) of a macro-molecular assembly by homology and/or threading, segment EM map of the complete macro-molecule, and then fit the individual homology models into

different segments of the EM model. While these methods improve the tertiary arrangements, they do not address modelling partial fragments of proteins, loop closure and satisfaction of local stereochemical constraints. Here, we address this limitation. Also our method does not require pre-segmentation of the EM model (which can become arbitrary and error-prone).

Our protocol was calibrated and rigorously validated based on a control set of high resolution structures of gp120 and achieved statistically significant correlation with ground truth. Finally, the validated protocol was used to generate a complete structural model of the envelope glycoprotein gp120 of HIV (including all its variable loops) in complex with CD4 with 17b.

gp160, the only solvent accessible protein of mature HIV-1, forms spikes protruding outside a bi-layer lipid membrane. It is cleaved by a fusion peptide at position 512 into a membrane extremal gp120 chain and a partially buried gp41 chain. gp120 itself has a relatively conserved core and several variable regions named V1, V2, V3, V4 and V5, spanning residues 131-156, 157-196, 297-330, 385-418 and 461-471 respectively. gp120 is believed to be instrumental in the initiation of the process of infecting a cell [190]. It first binds with the primary receptor CD4 present on the surface of T-cells and then binds with chemotaxis receptor CCR5. This induces conformational changes to gp120 and then to gp41, which starts to fuse with the membrane of the host cell and the RNA of the virus is released into the cell [140]. Understanding the structural basis of these interactions, and designing antibodies to disrupt them have been the mainstay of anti-HIV research for almost 30 years.

Elucidating the structure of gp120 in its native state have proved extremely challenging for several reasons. The variable domains V1/V2, V3, V4 and V5 are highly flexible and the surface is heavily glycosylated- both of which makes it hard to purify and crystallize gp120, and also introduce heterogeneity in EM imaging leading to low resolution reconstruction. All of the previously reported x-ray models of the complex of gp120 with CD4 and 17b, are missing the V1V2 and V3 loops. Out of around 480 residues of gp120, only 321 are present in x-ray models 1G9M, 1GC1 and 1RZJ [126, 152] deposited in the protein data bank (PDB). Other x-ray models in the PDB are missing even more residues (please see Discussion). At the low resolution end, there exists an EM model, EMD:5020 [161] at 20Å resolution, of the same complex (gp120+CD4+17b). Though the resolution is insufficient to correctly identify locations of secondary structural components or for manually grafting missing fragments, it does provide coarse spatial restraints. For instance, rigidly fitting 1GC1 into EMD5020 validates that the relative orientations of the three molecules in the x-ray model are close to their in-vitro state and also reveals large unoccupied portions of the EM map, where the variable missing loops are expected to be (Figure 8.4D). This provides the starting point of our protocol.

In the final model we report here, the variable loops occupy the vacant regions of the EM map. Moreover it improves the interface of gp120 with CD4 and 17b. The tertiary architecture of the variable loops in our model mostly agrees with recently submitted complete models of gp120 complexed with PGV04 and PGT122 [136, 167], however the ternary arrangement of the loops with respect to each other and specially 17b is vastly different, reinforcing previous observations

(e.g. [243]) that the loops undergo large conformational changes when bound to different partners. Our model also shows some new residue-residue contacts of CD4 and 17b with gp120, which upon experimental validation (e.g. isothermal titration calorimetry), can lead to new insights for neutralizaing antibody design.

Our results exhibit it is possible to generate improved-resolution structural models with high confidence. Our computational protocol and our reported model of gp120, can additionally be applied to generate complete improved resolution structural models of gp120 in complex with only other partners based on several recently reported EM-maps with resolutions between 16-25 in [29, 147, 148, 220, 254] for instance. It could of course also be applied to the structural refinement of many more lower resolution EM models of other macromolecules as well.

8.2 Materials and Methods

Given sequences of one or more chains and a low resolution EM map of a complex, we first identify the best representative atomic structure of the complex involving the chains in the protein data bank, and fit the structure into the density map (Figure 8.1A). Then we identify fragments of the sequence for which atomic structure is missing, identify corresponding locations in the fitted density map, and then generate and fit multiple models for each fragment (Figure 8.1B-C). In the second stage, a diverse subset of the fragments are chosen (Figure 8.1D) based on a scoring function which rewards favorable fitting with a EM density map of the protein complex, favorable interactions of the fragments with the partial chains derived from the protein data bank, lower energy (under the GBSA model), and better stere-

ochemistry (measured as a consensus of a large set of protein structure quality assessment tools). See supplement for details. Different combinations of the selected fragments are then assembled in all possible combinations to generate an ensemble of complete models (Figure 8.1E). In the third stage, the complete models are ranked by their scores after structural optimization and energy minimization [64], and a small subset of these complete models are selected. Then Local refinement-based docking [65] and fitting [23, 42], together with energy minimization is used to improve the binding interactions of chains in the selected models, and a single one out of these co-optimized models is chosen (Figure 8.1F-G). If multiple good candidates are generated, we perform binding site analysis to rank them.

Our search and scoring protocol was tested on 20 existing crystal structures of gp120 complexed with CD4 and/or 17b, and it successfully predicted a near-native pose as the top-ranked solution in 13/16 cases for gp120-17b interactions and 11/16 cases for gp120-CD4 interactions (with 3 more cases having a correct solution within top 10). In 18 out of the overall 32 cases, our method picked the lowest RMSD solution as the top solution. Hence, it is strongly validated that the scoring method clearly distinguishes between native and non-native poses, and the algorithm successfully samples the conformational space to find the native conformation.

8.2.1 Protocol Details

Stage 1: Fragment Modeling For each fragment of gp120 (core, v1v2, v3, and v4), we use two state of the art homology/threading platforms, namely Swiss-Model

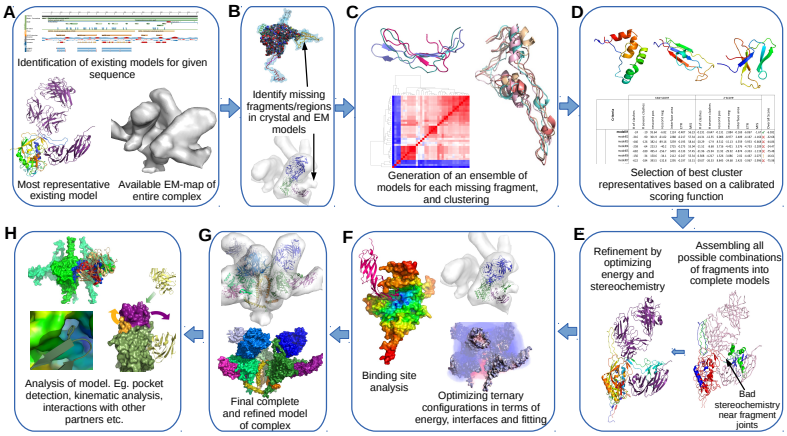


Figure 8.1: **Integrative refinement and validation protocol for modeling proteins with variable domains**. **(A)-(B)** Given a sequence we identify candidate partial crystal structures and EM-maps for the protein and identify the fragments which are missing in the crystal model and corresponding regions in the EM. **(C)** Threading and homology modeling to generate ensemble of candidate models for each fragment. **(D)** Existing partial crystal structures, known binding interfaces, prior knowledge about residue contacts and stereochemical properties of proteins are used to calibrate a multi-term scoring function which is used to rank the clusters and select small number of models for each fragment. The same scoring model is used in ranking and selecting models in the remaining steps as well. **(E)** Fragments are assembled in all possible combinations to generate a large set of complete models, which are then refined iteratively in terms of both the energy and stereochemistry. **(F)** A small set of refined models are now co-optimized with other chains in the complex to improve the ternary interfaces and fitting to the EM-map. A single model **(G)** is chosen based on the scoring function and binding site analysis. **(H)** Further analytics are performed to validate the model, compare with previous models and also to infer new kinematic, energetic, and binding information.

[230] and I-TASSER [217], to generate multiple initial models, based on different templates and having different folds. Each model was then flexibly fit using PF3Fit [23, 42] into EMD5020 [161]. The fitted models were next clustered based on their fold similarity (measured using TM-score [276]), and the best scoring model from each cluster was picked (Figures 8.1A-B).

Stage 2: Chain Modeling The selected fragments were assembled in all possible combinations to form a large set of models. These assembled structures are not stereochemically sound as the bond lengths/angles at the joint are too far from ideal (Figure 8.1C for example). To remove the gaps in the chain and improve the stereochemistry, we used the threading pipeline of Swiss-model [230] where the assembled model was specified as template to generate a new model with better local stereochemistry but whose 3D folds exactly match the assembled model (Figure 8.1D). Then energy minimization was performed on the new model using KoBaMin [64]. The two steps were repeated until no significant improvement was observed. A few of the improved models were selected (based on their score).

Stage 3: Co-optimization and Trimer Modeling In the previous steps, CD4 and 17b were kept fixed at their gp120 bound configuration in 1GC1 [152], and each of the generated models were aligned with the gp120 core of 1GC1 to evaluate the quality of the interface with CD4 and 17b. Now, we applied a multi-scale docking protocol, F2Dock [65], to refine the configuration of CD4 and 17b for each of the selected models. For some of the models, the refined configurations removed all clashes and also improved residue contacts. The complete model of the complex (for each selected model) was energy-minimized using KoBaMin [64] (Figure 8.1E). Finally, one model was chosen and the trimeric complex was formed by fitting. If multiple good candidates are generated, we perform binding site analysis to rank them.

Stage 4: Binding Site Analysis The binding site analysis applies our exhaustive docking protocol [65] and verify that known binding sites remain the most favorable binding sites even after the missing regions have been added. We consider the top 1000 docking results and for each residue R_i on the receptor (gp120), we count the number C_i of poses of the ligand (CD4 or 17b) for which the residue is on the contact region (Figure 8.1F). Then we define the probability of the residue R_i being part of the binding site as $P_i = C_i/1000$. These statistical inference of binding site is then compared to known binding site data available from known bound structures (in which variable regions are missing). Let, $\mathbb{BS}$ be the set of residues that belong to the binding site of the known structure, and $\mathbb{NBS}$ be the set of residues which do not. Then the quality of a complete model is defined as $\frac{1}{|\mathbb{BS}|}\sum_{R_i \in \mathbb{BS}} P_i - \frac{1}{|\mathbb{NBS}|}\sum_{R_j \in \mathbb{NBS}} P_j$.

Details of the application of the protocol for modeling gp160+CD4+17b complex can be found in supplement.

8.2.2 Model Evaluation Criteria (Scoring Function)

Evaluating Local Stereochemistry, Tertiary Folds, and Contacts We use an ensemble of methods consisting of Verify3D [166], PROCHECK [168], ERRAT [68], ProSA [239], and MolProbity [74]. Note that it is possible for a structure to get perfect scores in local stereochemistry checkers like PROCHECK but have poor tertiary structural folds and get bad scores in global correlation tools like Verify3D, ERRAT, ModEval etc. On the other hand, there are high resolution structures deposited in the PDB database which contain local errors and have poor PROCHECK

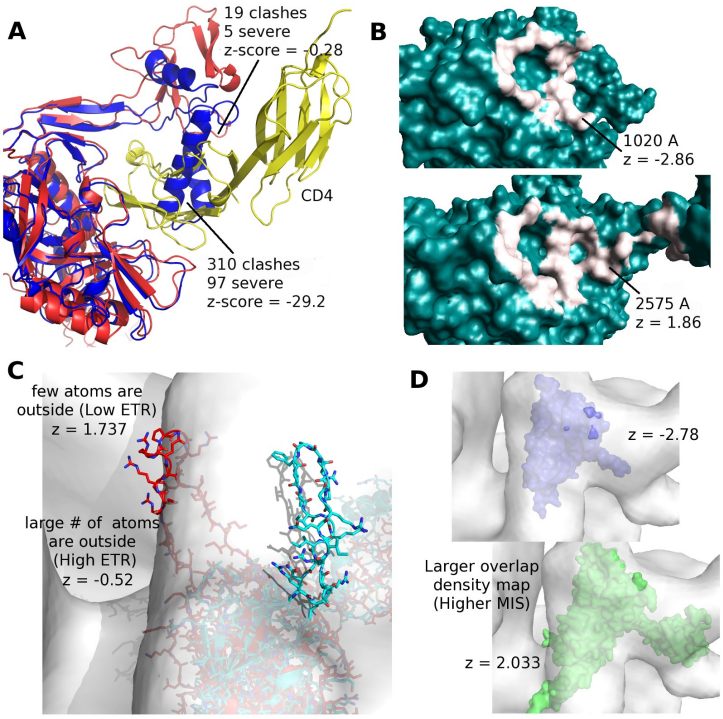


Figure 8.2: **Illustration of scoring terms** In **(A)**, we show two different models of gp120, and their interface with CD4 (yellow). The blue model was generated by Swiss-model [230] using 1G9N:G as template, and our optimized model is shown in red. **(B)** comparison of the binding footprints of CD4 on two different gp120 models. The one on top is a model generated by Swiss-model [230] using the PDB 3IDY as template, the one at the bottom is our final model. **(C)** provides a graphical understanding of external total ratio (ETR) which measures the fraction of atoms of a model lying outside a isosurface of the density map. We used the recommended isovalue (1.0) [161] and rendered with transparency so that atoms lying outside will be distinguishable from those inside. The blue model was generated by Swiss model using 1GC1 chain G as template and the V3 loop is almost complete outside the isosurface. Our optimized model, shown in red, has only a few atoms outside the isosurface. In **(D)**, we compare two models in terms of how much of the density map they are covering. The model on top is actually a crystal structure (1GC1:Chain G) and has poor MIS 53.78 compared to the model below (optimized model) whose MIS is 64.39.

score. So, we consider a model as high quality only if it scores well in each of the validation tools. Note that, PDB evaluation suite ADIT, Modeval [236] and Qmean z-score [34] were used for independent validation the final model, but not used during the scoring and search.

Evaluating Quaternary Contacts/Protein-Protein Interactions When the protein is in complex, then the quaternary structure quality, i.e. the quality of the interface between the proteins must also be considered. We use five scoring terms which can be efficiently computed. The first two terms, clash and severe clash, compute the number of atoms of one protein whose center lies, respectively, too close and inside the VDW volume of any atom of the other (See Figure 8.2A for an example). Good crystal structures typicaly have no severe clashes, and fewer than 10 clashes. The third scoring term, interface area, is computed as the part of the molecular surface of one protein which is within 2Å from any point on the molecular surface of another protein (Figure 8.2B). Interface area can vary a lot depending on the type of the proteins involved and approriate thresholds must be calibrated (see calibration in the Results section and Figure 8.3). The last two terms are statistical residue-residue contact scores, computed based on contact potentials for each residue-residue types reported by Glaser et al. [106], where positive and negative potentials indicate, respectively, higher and lower probabality of the finding such contact on an interface. These scores were applied in [65] and successfully ranked native interfaces, specially for antibody-antigen complexes.

Evaluating Quality of Fitting We used two scoring terms, ETR and MIS, to evaluate the quality of fitting of a model to an EM-map. The external-total ratio (ETR) [195] is defined as $-N_{out}/N$ where N_{out} is the total number of atoms of the model which lies outside a given isocontour of the density map, and N is the total number of atoms. Lower ETR indicates better fitting (see Figure 8.2C). The mutual informa-

tion score (MIS) [235, 262], is given by $MIS = \sum_{x \in B} \sum_{y \in A} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$ where $p(x)$ and $p(y)$ are the percentage of voxels in the volumetric representation of the model B and the density map A that take on intensities equal to x and y respectively and $p(x,y)$ is the percentage of voxels in B with intensity x that are aligned with voxels in A with intensity y. Essentially, MIS is maximized when the model has larger overlap with the map (see Figure 8.2D).

Overall Score We have primarily two different types of score/validation. The first, $s_{internal}$, is a set of scores to evaluate the stereochemistry and tertiary folds using a consensus of several state of the art structure assessment tools. The send set of scores, developed by us, evaluates the ternary interactions and fitting to a density map. The overall score for them are defined as-

$$s_{internal} = g_{(\phi-\psi)} + g_{(all)} + z_{Verify3D} + z_{ProSA} + z_{MolProbity} + z_{ERRAT}$$

and

$$s_{external} = z_{cl} + z_{RCP} + z_{RCN} + z_{IA} + z_{ETR} + z_{MIS}$$

where $g_{(\phi-\psi)}$, $g_{(all)}$, $z_{Verify3D}$, z_{ProSA}, $z_{MolProbity}$, z_{ERRAT}, z_{cl}, z_{RCP}, z_{RCN}, z_{IA}, z_{ETR}, z_{MIS} are PROCHECK g-factors for $\phi - \psi$ angles and for all angles, z-score of Verify3D, ProSA, MolProbity and ERRAT's composite scores, and z-scores of clash, positive residue contact, negative residue contact, interface area, ETR and MIS scores respectively. The z-score for a term X is defined as $(s_X - \mu_X)/\sigma_X$ where s_X is the raw score of the model, and μ_X and σ_X are the mean and standard deviation of the raw scores over all structures, or a benchmark/control set

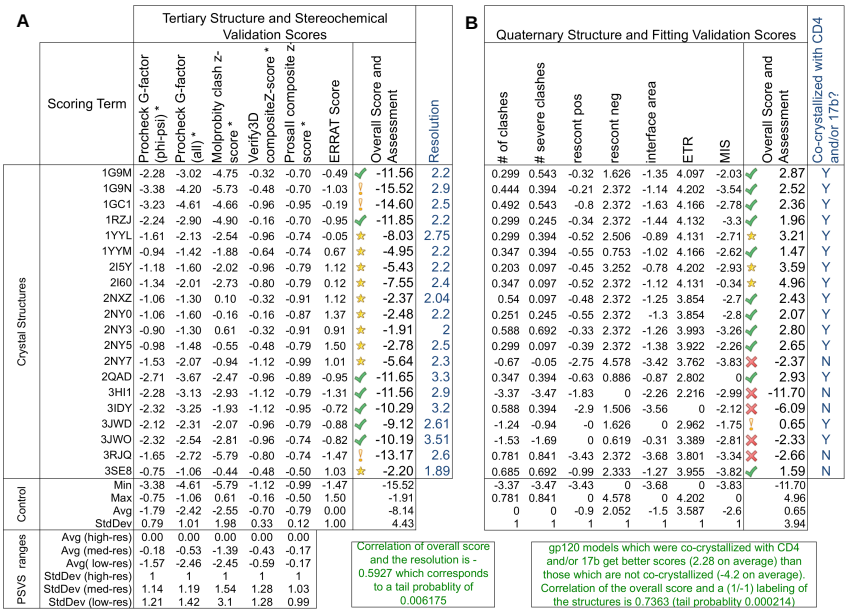
of structures. Details on the control is presented in the next section.

The sum of the z-scores intuitively means that a model have to be better than average in all aspects to be considered as accurate. Notice that some of the scoring terms are complementary to each other and provide check and balance. Poor models may get better score in some terms, but not all. For example, a model of gp120 which penetrates CD4 when placed at the correct orientation (derived based on 1GC1), will get high scores for interface area and positive residue contacts, but will get penalized by clashes and negative residue contacts.

8.2.3 Protocol Calibration for Modeling gp120

We have primarily two different types of scoring terms- a set of terms to evaluate the stereochemistry and tertiary folds using state of the art structure assessment tools, namely Verify3D, PROCHECK, ERRAT, ProSA, ModEval, and MolProbity [68, 74, 166, 168, 236, 239]; and another set of terms, developed by us, to evaluate the ternary interactions and fitting to a density map [10, 65, 106, 235, 262] (see supplement for details). The quality of a structure X for a specific term is defined as $(s_X - \mu_X)/\sigma_X$ where s_X is the raw score of the model, and μ_X and σ_X are the mean and standard deviation of the raw scores over all structures in a control set (see below). The overall quality of the model is a sum of the qualities for each term.

Mean and variance of PROCHECK g-factors, Verify3D, ProSA and MolProbity composite scores over a large set of non-redundant proteins was reported in PSVS [43]. We verified if the distribution of the scores reported in PSVS accu-



A

Scoring Term	Procheck G-factor (phi-psi) *	Procheck G-factor (all) *	Molprobity clash z-score *	Verify3D compositeZ-score *	Prosall composite z score *	ERRAT Score	Overall Score and Assessment	Resolution
1G9M	-2.28	-3.02	-4.75	-0.32	-0.70	-0.49	✓	-11.56	2.2
1G9N	-3.38	-4.20	-5.73	-0.48	-0.70	-1.03	!	-15.52	2.9
1GC1	-3.23	-4.61	-4.66	-0.96	-0.95	-0.19	!	-14.60	2.5
1RZJ	-2.24	-2.90	-4.90	-0.16	-0.70	-0.95	✓	-11.85	2.2
1YYL	-1.61	-2.13	-2.54	-0.96	-0.74	-0.05	★	-8.03	2.75
1YYM	-0.94	-1.42	-1.88	-0.64	-0.74	0.67	★	-4.95	2.2
2I5Y	-1.18	-1.60	-2.02	-0.96	-0.79	1.12	★	-5.43	2.2
2I60	-1.34	-2.01	-2.73	-0.80	-0.79	0.12	★	-7.55	2.4
2NXZ	-1.06	-1.30	0.10	-0.32	-0.91	1.12	★	-2.37	2.04
2NY0	-1.06	-1.60	-0.16	-0.16	-0.87	1.37	★	-2.48	2.2
2NY3	-0.90	-1.30	0.61	-0.32	-0.91	0.91	★	-1.91	2
2NY5	-0.98	-1.48	-0.55	-0.48	-0.79	1.50	★	-2.78	2.5
2NY7	-1.53	-2.07	-0.94	-1.12	-0.99	1.01	★	-5.64	2.3
2QAD	-2.71	-3.67	-2.47	-0.96	-0.89	-0.95	✓	-11.65	3.3
3HI1	-2.28	-3.13	-2.93	-1.12	-0.79	-1.31	✓	-11.56	2.9
3IDY	-2.32	-3.25	-1.93	-1.12	-0.95	-0.72	✓	-10.29	3.2
3JWD	-2.12	-2.31	-2.07	-0.96	-0.79	-0.88	✓	-9.12	2.61
3JWO	-2.32	-2.54	-2.81	-0.96	-0.74	-0.82	✓	-10.19	3.51
3RJQ	-1.65	-2.72	-5.79	-0.80	-0.74	-1.47	!	-13.17	2.6
3SE8	-0.75	-1.06	-0.44	-0.48	-0.50	1.03	★	-2.20	1.89
Min	-3.38	-4.61	-5.79	-1.12	-0.99	-1.47		-15.52	
Max	-0.75	-1.06	0.61	-0.16	-0.50	1.50		-1.91	
Avg	-1.79	-2.42	-2.55	-0.70	-0.79	0.00		-8.14	
StdDev	0.79	1.01	1.98	0.33	0.12	1.00		4.43	
Avg (high-res)	0.00	0.00	0.00	0.00	0.00			
Avg (med-res)	-0.18	-0.53	-1.39	-0.43	-0.17			
Avg(low-res)	-1.57	-2.46	-2.45	-0.59	-0.17			
StdDev (high-res)	1	1	1	1	1			
StdDev (med-res)	1.14	1.19	1.54	1.28	1.03			
StdDev (low-res)	1.21	1.42	3.1	1.28	0.99			

Correlation of overall score and the resolution is -0.5927 which corresponds to a tail probablity of 0.006175

B

# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	Overall Score and Assessment	Co-crystallized with CD4 and/or 17b?
0.299	0.543	-0.32	1.626	-1.35	4.097	-2.03	✓	2.87	Y
0.444	0.394	-0.21	2.372	-1.14	4.202	-3.54	✓	2.52	Y
0.492	0.543	-0.8	2.372	-1.63	4.166	-2.78	✓	2.36	Y
0.299	0.245	-0.34	2.372	-1.44	4.132	-3.3	✓	1.96	Y
0.299	0.394	-0.52	2.506	-0.89	4.131	-2.71	★	3.21	Y
0.347	0.394	-0.55	0.753	-1.02	4.166	-2.62	✓	1.47	Y
0.203	0.097	-0.45	3.252	-0.78	4.202	-2.93	★	3.59	Y
0.347	0.097	-0.52	2.372	-1.12	4.131	-0.34	★	4.96	Y
0.54	0.097	-0.48	2.372	-1.25	3.854	-2.7	✓	2.43	Y
0.251	0.245	-0.55	2.372	-1.3	3.854	-2.8	✓	2.07	Y
0.588	0.692	-0.33	2.372	-1.26	3.993	-3.26	✓	2.80	Y
0.299	0.097	-0.39	2.372	-1.38	3.922	-2.26	✓	2.65	Y
-0.67	-0.05	-2.75	4.578	-3.42	3.762	-3.83	✗	-2.37	N
0.347	0.394	-0.63	0.886	-0.87	2.802	0	✓	2.93	Y
-3.37	-3.47	-1.83	0	-2.26	2.216	-2.99	✗	-11.70	N
0.588	0.394	-2.9	1.506	-3.56	0	-2.12	✗	-6.09	N
-1.24	-0.94	-0	1.626	0	2.962	-1.75	!	0.65	Y
-1.53	-1.69	0	0.619	-0.31	3.389	-2.81	✗	-2.33	Y
0.781	0.841	-3.43	2.372	-3.68	3.801	-3.34	✗	-2.66	N
0.685	0.692	-0.99	2.333	-1.27	3.955	-3.82	✓	1.59	N
-3.37	-3.47	-3.43	0	-3.68	0	-3.83		-11.70	
0.781	0.841	0	4.578	0	4.202	0		4.96	
0	0	-0.9	2.052	-1.5	3.587	-2.6		0.65	
1	1	1	1	1	1	1		3.94	

gp120 models which were co-crystallized with CD4 and/or 17b get better scores (2.28 on average) than those which are not co-crystallized (-4.2 on average). Correlation of the overall score and a (1/-1) labeling of the structures is 0.7363 (tail probablity 0.000214)

Figure 8.3: **Controls, calibration and validation of scoring methods.** Our control or benchmark consist of 20 gp120 structures from the PDB. They are listed by their PDBIDs in the tables. The table in **(A)** details the z-scores for each term in $s_{internal}$. The rows beside 'Control', describe the min, max, mean and standard deviation of the z-scores of the 20 models in the control set. At the bottom of the table, the mean (avg) and standard deviation of scores for models with different resolutions, as reported in [43], are given. On average, the scores for the models in our benchmark correspond to the low-resolution (< 3.5Å) class and agrees with the actual resolutions (shown in the last column of the table). Also, the sum of z-scores (i.e. $s_{internal}$) for individual models in the control set, have a high correlation with corresponding actual resolutions. Hence, $s_{internal}$ is a statistically sound metric for tertiary structural and stereochemical quality. The table in **(B)** reports the interface and fitting based scores ($s_{external}$). Again, notice that the overall score is highly correlated with whether or not the gp120 model is in a co-crystallized state with CD4 and/or 17b (mentioned in the last column of the table). Hence, the validity of the interface and fitting based scores to distinguish correct interface/neighborhood is also established. Finally, the star, check, exclamation and cross icons visually highlight scores which are $\geq \mu + 2\sigma$, $\geq \mu + \sigma$, $\geq \mu$ and $< \mu$ respectively.

rately represent the quality of the structures in our control set (see Figure 8.3A). We found that most of the structures in our benchmark had scores which lie within 2σ from the mean score (μ) for structures in the low-resolution class of PSVS (resolution between $2.5 - 3.5$Å). The actual resolution of the structures in our set (Figure

8.3A) ranges from 1.89 to 3.51, and hence the z-scores and ranges prescribed by PSVS are validated. More importantly, we found that $s_{internal}$ can correctly distinguish between low and high resolution crystal structures within the control set. The Pearson correlation coefficient of $s_{internal}$ and corresponding resolutions across the 20 models is -0.5927 which corresponds to a tail probability of 0.006175. The correlation is statistically significant and hence if the $s_{internal}$ of a model is higher than the average value (-8.14) of the control set, we can accept it, with high confidence, as a high resolution and stereochemically accurate model.

The objective of $s_{external}$ is to distinguish between correct and incorrect interfaces/sites offered to the binding partners (e.g. CD4 and 17b), and correct and incorrect conformations for fitting/alignment to the EM map. For each term in $s_{external}$, we computed the distribution of values observed on a control set of 20 existing crystal structures of gp120 (see Figure 8.3B). The last column in Figure 8.3B shows which gp120 models were co-crystallized with CD4 and 17b and hence have a correct site topology. The correlation between this labeling and $s_{external}$ is 0.7363 and is statistically even more significant.

In conclusion, if a model is rated as high quality under both $s_{internal}$ and $s_{external}$, then the model is indeed high quality with high probability.

8.2.4 Search Protocol Validation

Our multi-resolution docking/fitting algorithm and scoring/ranking scheme was validated by applying it to predict the correct binding interactions with gp120 and other molecules from a large set of co-crystallized structures. We first compute

gp120-17b		First Near-Native Pose		Best Pose	
Receptor (PDBID-Chain)	Ligand (PDBID-Chain)	Rank	RMSD	Rank	RMSD
1G9M-G	1G9M-HL	71	4.1	441	3.3
1G9N-G	1G9N-HL	1	1.3	1	1.3
1GC1-G	1GC1-HL	1	2.4	13	2.1
1RZJ-G	1RZJ-HL	1	1.7	1	1.7
1YYL-G	1YYL-HL	1	2.3	168	1.4
1YYM-G	1YYM-HL	1	0.9	1	0.9
2I5Y-G	2I5Y-HL	1	1.4	1	1.4
2I60-G	2I60-HL	1	2.3	152	1.5
2NXZ-A	2NXZ-CD	7	2.1	19	1.8
2NY0-A	2NY0-CD	1	2	1	2
2NY1-A	2NY1-CD	1	1.3	1	1.3
2NY2-A	2NY2-CD	1	1.5	1	1.5
2NY3-A	2NY3-CD	1	2.2	1190	2.1
2NY4-A	2NY4-CD	1	1.8	31	1.6
2NY5-A	2NY5-CD	1	2.1	1	2.1
2NY6-A	2NY6-CD	56	4.7	276	1.7

Table 8.1: **Performance of multi-resolution docking (F2Dock + PF3Fit) on predicting 17b binding** Result of applying F2Dock+PF3Fit to predict the orientation of 17b w.r.t. gp120. Both 17b and gp120 were extracted from co-crystallized structures available in the PDB. The co-crystallized structure was fitted to the density map. Then, given a randomly transformed 17b and keeping gp120 fixed at the fitted position, the docking protocol predicted the orientation of 17b which maximizes a scoring term similar to $s_{external}$ mentioned in the text. A predicted pose was considered acceptable (near-native) if it was within 4Å RMSD of the fitted position of 17b (before it was randomly moved). The rank of the first such pose, in the ordered list of predictions, and its RMSD is reported in the table. The best pose, on the other hand, is defined as the pose with the lowest RMSD across the first 2000 predictions. Notice that our docking protocol successfully predicted a near-native pose as the top-ranked solution in 13/16 cases. Hence, it is strongly validated that the scoring method clearly distinguishes between native and non-native poses, and the algorithm successfully samples the conformational space to find the native conformation. Also, in most cases, the top solution is the best solution as well.

the best fitting of 1GC1, a complex that includes gp120 core, CD4 and 17b, to the density map EMD5020. Then, for each crystal structure, we computed the best

rigid body transformation which would align the gp120 chain to the fitted gp120. The transformation was applied to the entire structure (including other chains like CD4, 17b etc.). After the alignment, we kept the position of gp120 fixed, and created a copy of the other molecules and applied a random transformation to each. The randomly moved molecule is used as ligand, and the fixed gp120 is used as receptor for our multiresolution docking prediction tool. The position of the other molecule before applying the random transformation is considered the native state. If a predicted pose is within 4Å RMSD from this reference state, then the prediction is considered acceptable.

gp120-cd4		First Near-Native Pose		Best Pose	
Receptor (PDBID-Chain)	Ligand (PDBID-Chain)	Rank	RMSD	Rank	RMSD
1G9M-G	1G9M-C	1	1.3	1	1.3
1G9N-G	1G9N-C	116	3.7	116	3.7
1GC1-G	1GC1-C	1	1.6	1	1.6
1RZJ-G	1RZJ-C	1	1.2	1	1.2
2B4C-G	2B4C-C	4	2.6	4	2.6
2NXZ-A	2NXZ-B	1	1.2	1	1.2
2NY0-A	2NY0-B	1	0.8	1	0.8
2NY1-A	2NY1-B	1	2.4	2	0.9
2NY2-A	2NY2-B	1	1.2	1	1.2
2NY3-A	2NY3-B	1	1.7	1	1.7
2NY4-A	2NY4-B	1	1.5	1	1.5
2NY5-A	2NY5-B	1	1.5	1	1.5
2NY6-A	2NY6-B	4	1.9	4	1.9
2QAD-A	2QAD-B	1	1.9	1	1.9
3JWD-A	3JWD-C	9	3	9	3
3JWO-A	3JWO-C	0	10	0	10

Table 8.2: **Performance of multi-resolution docking (F2Dock + PF3Fit) on predicting CD4 binding** An experiment similar to the one reported in table 8.1 with CD4, instead of 17b. Again, we get a near-native solution at rank 1 in 11/16 cases and within rank 10 in 14/16 cases. 3JWO is the only case where our method could not find any near native solutions.

For each of the complexes, we applied the above procedure and observed the rank and RMSD of the first acceptable prediction. An acceptable solution at high rank indicates that the scoring functions are capable of discriminating between native and non-native poses. A low RMSD indicates that the conformational space is sampled sufficiently.

Our docking protocol successfully predicted a near-native pose as the top-ranked solution in 13/16 cases for gp120-17b interactions and 11/16 cases for gp120-CD4 interactions (with 3 more cases having a correct solution within top 10). In 18 out of the overall 32 cases, our method picked the lowest RMSD solution as the top solution. Hence, it is strongly validated that the scoring method clearly distinguishes between native and non-native poses, and the algorithm successfully samples the conformational space to find the native conformation. Tables 8.1 and 8.2 provide further details.

8.3 Results

8.3.1 Structure of gp120 in Complex with CD4 and 17b

We report the structure of gp120 trimer generated using our calibrated integrative protocol. Please see supplement for detailed report on the outcome of different stages of the protocol. The final structural model consists of a cleaved gp120 bound to CD4, 17b and a peptide model of gp41 (Figure 8.4A) and fits well (Figure 8.4D) into the density map EMD5020 [161]. We use two metrics to quantitatively assess the quality of fitting- excluded total ratio (ETR) and mutual information score (MIS). Low ETR indicates that a smaller fraction of backbone

atoms lie outside a specific boundary representation of the EM model. High MIS indicates that larger portion of the EM model is occupied. Our complete gp120 model has an ETR score of 0.064. The optimized configurations of 17b and CD4 chains also show excellent fitting accuracy with ETRs 0.086 and 0.015 respectively. For comparison, the best rigid body fitting of 1GC1 with the density map have ETR scores of 0.005, 0.112 and 0.03 for the gp120, 17b and CD4 chains respectively. The overlap scores (MIS) for the gp120, 17b and CD4 in the new model are 61.03, 50.88 and 54.99 respectively (compared to 53.85, 49.65 and 53.78 respectively for 1GC1).

Note that the variable regions not only fit well with the density, but also lie on the periphery of the complex (Figures 8.4B-C). This agrees with previous reports in [29, 161, 254] regarding the expected positions of the variable loops in the open conformation (when bound to CD4 and 17b). The general configuration of CD4 and 17b in the trimer also matches well with the EM map, as well as with previously reported models of gp120 co-crystallized with CD4 and 17b. However, in comparison with 1GC1, our model of 17b undergoes a small shearing motion (small tilt and twist w.r.t. the binding site). The footprint on the core remains almost identical, but the light chain comes in contact with the V3 loop (Figure 8.5A). The model of gp41 is based on the one reported in [136]. However, a direct fitting of the model in [136] does not correlate well with EMD5020. Application of our protocol improved the fitting while preserving favorable contacts with gp120. Figure 8.4F in particular shows the difference between the two configurations.

Our model also preserves the binding interactions (residue contacts) at the

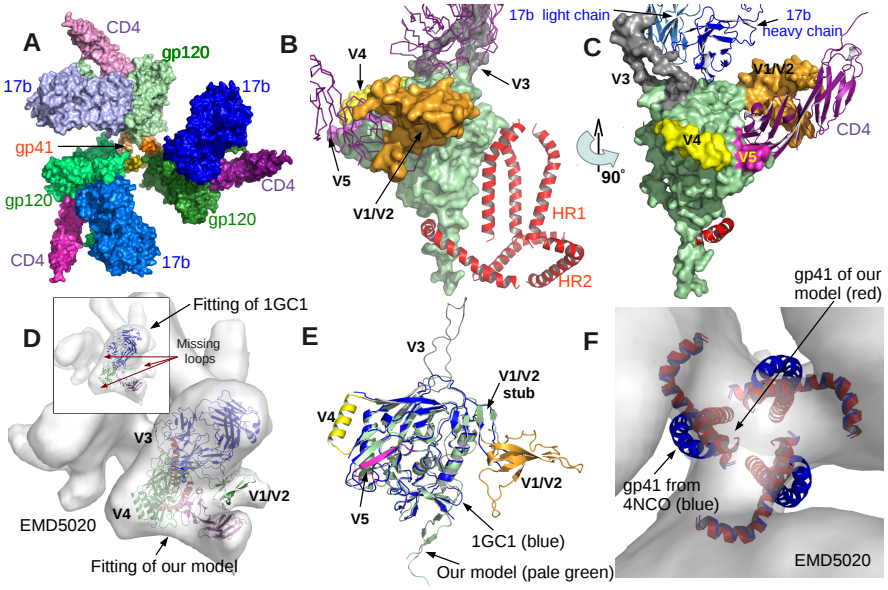


Figure 8.4: **Model Overview (A)** Quaternary architecture of the model shows that three copies of gp120 and gp41 forms the central part of a trimer (3-fold cyclic symmetry). Also three copies of CD4 and 17b are attached to the three gp120s. A small opening through the center along the symmetry axis is also visible. **(B)** shows the model from a direction orthogonal to the symmetry axis. gp120 is rendered using smooth surface representation. The core is colored pale green and the variable loops are highlighted using different colors. The gp41 model occupies the center of the trimer (all three copies are shown). Notice that all the variable regions (except V3) are away from the central area. **(C)** provides a different view of the same. **(D)** shows the fitting of our model into EM density map EMD5020 [161]. Our model include the variable loops which are missing in previously reported x-ray structures of the same complex (e.g. 1GC1 [152] as shown in the inset). **(E)** compares the gp120 model from 1GC1 with our model. Note that the core as well as the V1/V2 stub aligns almost perfectly. Finally, **(F)** compares the gp41 model from 4NCO [136]. Though we used the same models for the two heptad-repeats (HR1 and HR2), to improve the fitting as well as to improve the interface with gp120, the HR1 part in particular was moved closer to the center of the trimer. But there is still an opening as seen in **(A)**.

gp120-CD4 interface as well as places the PHE43 ring of CD4 into the CD4 binding pocket (Figure 8.5D). However, though we used the x-ray model 1GC1 fitted to EMD 5020 as the starting point of our protocol, repeated application of flexible fitting and local refinement docking resulted in relative orientations between gp120-CD4 and gp120-17b, which are slightly different from the corresponding ori-

entations present in 1GC1 (Figure 8.5A). This change of orientation, together with minor side chain movements for energy minimization and stereochemistry corrections, resulted in small changes to binding contacts between gp120 core with CD4 and 17b (see Figure 8.5E-H). Finally, our model also have disulphide bonds at expected locations (between residues 119-205, 218-247, 228-239, 296-331, 378-445 and 385-418).

Please see Discussion for a further comparison of our model with existing crystal structures.

8.3.2 Quality Assessment of the Model

Ramachandran plot analysis by Procheck [168] showed 89.6% residues in most favored, 7.6% in additionally allowed, 1.4% in generously allowed and only 1.4% in disallowed regions (see Figure S18). Overall Procheck g-factor is -0.22 for $\phi - \psi$ angles only and -0.04 for all, both of which is extremely favorable and correspond to high resolution (< 2) structures [43]. In total only 10 bad contacts were reported and 3.6% residues were found to have bad planarity.

ProsaII [239] composite score for the model is 0.76 which is also representative of high resolution structures [43]. MolProbity [74] composite score for the model is 30.44 (z-score -3.70), which in general indicates that a model is in the low resolution range. However, we note that among existing x-ray models of gp120, 1G9M, 1G9N, 1GC1, 1RZJ and 3RJQ all have worse MolProbity scores. Verify3D [166] reports that more than 71% of the residues have a 1D-3D score above 0.2, which is in acceptable range according to Verify3D's guidelines.

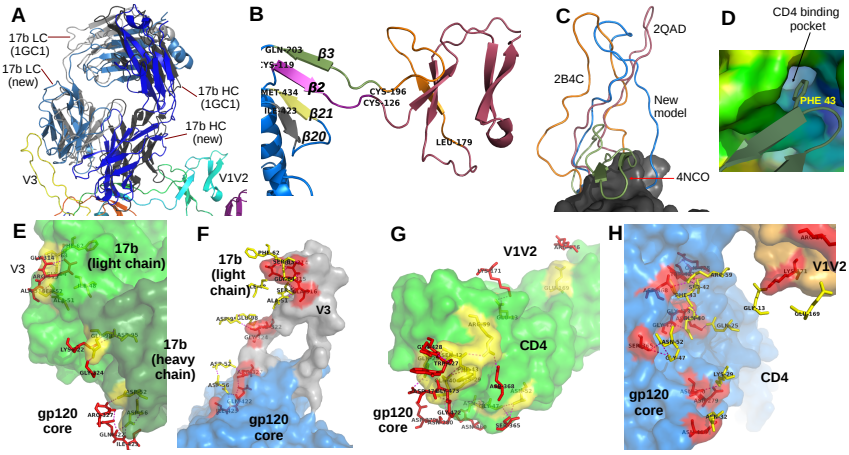


Figure 8.5: **Closer look at the model (A)** We superimposed the position of 17b derived from 1GC1 on our model of 17b to contrast them. The heavy and light chains of our model of 17b is colored dark and light blue respectively. The model from 1GC1 is colored using dark and light gray. In our model, the position of 17b was changed to optimize the fitting of 17b and the V3 loop into the density, while still ensuring favorable interaction between gp120-17b (see text). **(B)** A detailed look at the conformation of the V1V2 loop and stub. Notice the 4 anti-parallel beta sheets between residues 126-178, which is similar to the 4 anti-parallel 'Greek-key' formation reported in [164]. The $\beta 2$ and $\beta 3$ sheets at the stub shows the same orientation w.r.t. $\beta 20$ and $\beta 21$ as reported in existing crystal structures, but is opposite of the models in 4NCO and 3J5M. **(C)** V3 loops from 3 existing structures are compared with our model (the models are superimposed by aligning the cores). The conformation from 3J5M covers the 17b site and also contains beta sheets. Our model is conformationally closest to the V3 of 2B4C. **(D)** shows the nestling of PHE43 ring of CD4 inside the CD4 binding pocket of gp120. Figures **(E)** and **(F)** shows the contacts between 17b and gp120. In these figures, the residues in contact are rendered as sticks. Contact residues belonging to gp120 are colored red, and those belonging to 17b is colored yellow. gp120 itself is colored blue, with the V1V2 and V3 loops highlighted in orange and gray colors. 17b is colored green. The dashed lines indicate polar contacts. The two figures show the same set of contacts from two different perspectives and omits gp120 and 17b respectively. Note that the residues which lie outside the rendered surface actually belongs to the omitted molecule. Figures **(G)** and **(H)** shows the contacts between CD4 and gp120 in a similar manner.

Please refer to supplement, specially Figures S18-S21, for a more detailed analysis of the quality in comparison with existing crystal structures of gp120, and models produced by naive application of homology/theading.

We used the PDB validation software (ADIT), Modeval [236] and Qmean

z-score [34] to provide independent validation of the quality. PDB validation software (ADIT) reports RMS deviation for bond angles at 0.7 degrees and bond length deviation of 0.003Å, both of which is quite acceptable. ModEval [236] predicted an RMSD of 3.378 (for the gp120 chain only). The Qmean z-score was -1.666 which is within the acceptable range for a protein of this size. A plot showing the quality of our model with respect to existing x-ray models in terms of Q-mean z-scores in given in Figure S19 (top). The model, colored by per residue error under the ANOLEA [178] score, is also shown in Figure S19 (bottom).

8.4 Discussion

8.4.1 Significance of the New Model

The new model adds to the current understanding of the interaction of gp120 with CD4 and 17b. None of the previously reported crystal structures of gp120 in complex with either or both of CD4 and 17b include the variable loops V1V2 and V3 (see Figure 8.6B and S1). Among models bound to other partners, only 2B4C contains the V4 loop, and only 2B4C and 2QAD contains V3 loop. The parts of gp120 which is in contact with gp41 (mostly the residues at the start and end of the chain) are also reported only in a few structures (e.g. 3JWD [188]). There are only two recent models (PDBID: 4NCO and 3J5M) of gp120 in complex with antibodies PGV04 and PGT122 [136, 167] which include all the variable regions. However, a study of existing structures revealed that even the structure of the relatively conserved core region of gp120 depend on the binding partner (see Figure 8.6A and Supplement). This is further highlighted when we aligned the gp120 model in

3J5M, which contains the variable loops but is partnered with PGV04, to 1GC1 which does not have the variable loops but is bound with CD4 and 17b. We find that the conformation of the V1/V2 and V3 loops of 3J5M is occluding the binding site of 17b (Figure 8.6C) and hence cannot possibly be the correct configuration of the loops when gp120 is bound to CD4 and 17b. Finally, though there have been previous attempts [115, 266] at modeling gp120 with variable loops, but the protocols depended on manually grafting loop fragments extracted from other crystal structure to the core without rigorous validation, refinement and fitting. Also, direct application of state of the art homology modeling/threading tools produce energetically and stereochemically favorable models, but cannot correctly handle ternary constraints (see Supplement for details). So, we present the first validated high quality model of gp120 in complex with CD4 and 17b.

8.4.2 Comparison of New Model with Existing Atomic Models

To compare our model with existing x-ray structures, we aligned our gp120 model with 42 different x-ray structures available in the Protein Data Bank (PDB) using TM-Align [287]. TM-Align uses a scoring scheme called TM-score [276] which has better accuracy than RMSD in identifying alignments which correctly superimposes the structural motifs. In fact, it was reported that a TM-score higher than 0.5 indicates that two proteins have the same fold. We found that the core of our model has very high similarity with the x-ray models where gp120 is in bound state with CD4 and 17b, and also where gp120 is unliganded. For this class, the average TM score is 0.9545 and average RMSD is 1.575. However, our model is different

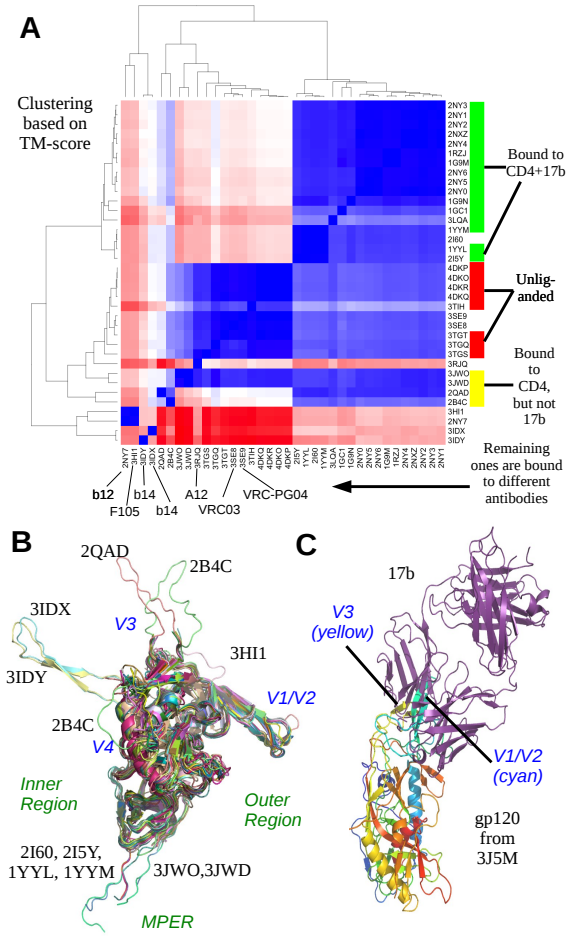


Figure 8.6: **Structural summary of existing gp120 models** **(A)** A heatmap based on TM-score of alignment and corresponding clustering of the gp120 models shows that the clusters are completely correlated with the partners of gp120 in different models; models with same partners or partners with same binding site are clustered together. In **(B)**, all the structures (except 4NCO and 3J5M) have been superimposed. The variable regions are marked out. The membrane proximal extremal region, which comes in contact with gp41 is also shown. The inner and outer regions refers to whether that part is buried/exposed when gp120 forms a trimer, the inner part being closer to the axis of symmetry. From this figure, it is immediately clear that the core is more conserved than the variable regions. Another interesting aspect is the wildly different loop structures of 3IDX, 3IDY and 3HI1. **(C)** superposing the cores of 3J5M and 1GC1 gives us a relative configuration of the gp120 chain of 3J5M and the 17b from 1GC1. The V1V2 and V3 loops of 3J5M is occupying the binding site of 17b and hence is not in a configuration amenable for binding with 17b.

from the cores of 3HI1, 2NY7, 3IDX and 3IDY where gp120 is in bound state with fab F105, b12, b13 and b14. The average TM-score and RMSD in this case are 0.8452 and 2.526 respectively. In Figure 8.4E, we show our model superimposed on the gp120 chain of 1GC1.

The V1V2 region of the new model forms 4 beta sheets between residues 126-178, which is similar to the conformations observed in 4NCO, 3J5M and 3U4E [137, 167, 177] (note that the V1V2 conformation reported in [177] does not include the core). However, the lengths of the beta sheets in our model is somewhat shorter. The orientation of the $\beta 2$ and $\beta 3$ sheets (the V1V2 stub) w.r.t. the $\beta 20$ and $\beta 21$ sheets observed in both 4NCO and 3J5M, are flipped compared to other crystal structures (e.g. 1GC1) containing the V1V2 stubs. Our model however has the same orientation as in 1GC1 (Figure 8.5B). The V3 variable region of our model does not contain any sheets or helices, similar to the models reported in 2B4C and 2QAD and moreover the ternary configuration is also quite similar. However it differs from the PGV04 and PGT122 bound models reported in 3J5M and 4NCO respectively [137, 167], where the V3 region contains two small anti-parallel beta sheets (Figure 8.5C).

8.4.3 CD4-induced Conformational Change of Variable Loops

In both 4NCO and 3J5M, the V1V2 and V3 loops are bundled together and covers the 17b binding site (Figure 8.7A). In 4NCO, we do notice that to allow the binding of PGT122, the V1V2 loop shifts a little bit. Despite this shift, both of these configurations the V1V2 loop remain quite far from PGV04 or CD4 binding

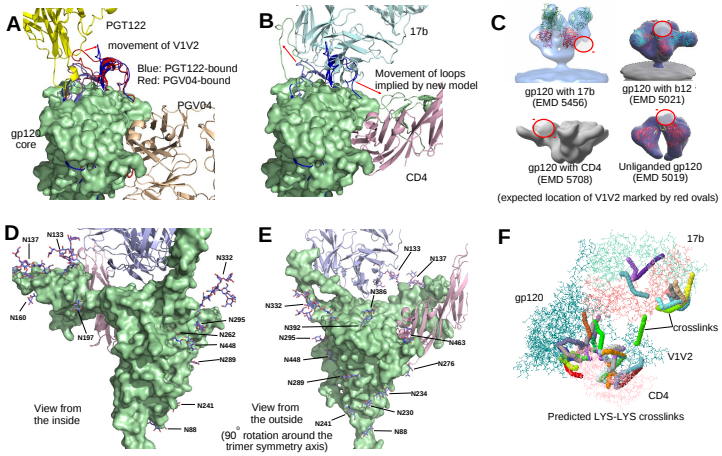


Figure 8.7: **Dynamical implications**. **(A)** Superposition of the gp120 models from 4NCO and 3J5M (blue and red), bound to PGT122 and PGV04 respectively. The V1/V2 and V3 loops occupy the same general location in both cases. But in the PGT122 bound configuration, the V1V2 loop have a noticeable shift to avoid clash with PGT122. In **(B)**, we see that 17b, which binds on the core near the V3 loop has large clash with both V1V2 and V3 loops of 4NCO (blue), and to allow the binding of 17b, both loops must move away to the configuration observed in our model (green). **(C)** EM model of gp120 bound with only 17b (EMD 5456) also shows a similar move of the V1V2 loop. Interestingly, gp120 bound with only soluble CD4 (EMD 5708), also have the V1V2 loop at the same ternary configuration. So, the movement of V1V2 is not only a result of repulsion by 17b, but maybe also due to an attraction towards CD4. On the other hand, b12, which also bind at the same site as CD4, does not allow/require V1V2 to move away from its unliganded configuration, and this may be another mechanism to prevent further binding at the 17b site (by CCR5 for instance). **(D)**-**(E)** shows glycans derived from 1RZJ [126], and 4NCO [167] mapped onto our model. Note that the configurations of the glycans are distal from the CD4 and 17b chains. **(F)** Predicted LYS-LYS crosslinks on the gp120-CD4-17b complex. The CD4bs is heavily crosslinked, and there are also quite a few crosslinks between the V1V2 loop and CD4 (please see Figure S20 for other types of crosslinks).

site. However, this configuration of the loops occludes the expected binding site of 17b (Figure 8.6C). We note that though PGT122 binds near the V3 loop, it has a very small footprint on gp120 itself and the binding is heavily glycan-dependent. On the other hand, 17b has a large footprint on the core of gp120 (as reported in many x-ray models including 1GC1, 1G9M, 1RZJ etc. [126, 152, 153]), as well as on the V3 loop, as seen in our model. In our model the V1V2 loop is pushed off

from the 17b site and squeezed into the space between 17b and CD4 (Figure 8.7B). Such configuration of the V1V2 loop is further validated from EMD5456 [254] and EMD5708 [29] where gp120 is bound with only 17b and only CD4 respectively (Figure 8.7C). However, the configuration is vastly different from unliganded, or b12-bound state seen in EMD5019 and EMD5021 [161] respectively. A similar conclusion was reached by [116] based on different EM models and hydrogen-deuterium exchange experiments. It was reported in [161] that gp120 undergoes a twisting motion around the trimer symmetry axis to expose the CD4 binding site. Our results further implies that motion of V1V2 induced by CD4 actually enables CCR5 binding. Also, EMD5021 and the footprint analysis (below) show that other antibodies might not induce or even prevent a similar motion of the V1V2 loop.

8.4.4 Antibodies Binding at CD4bs Does not Induce the Same Motion of Variable Loops

We computed the footprint of different antibodies whose x-ray structures in bound state with gp120 (core or complete) are available. We transformed the bound gp120-antibody complex such that the gp120 chain aligns with our gp120 model. Alignment was performed using PyMOL [228]. Then, for each antibody, all heavy atoms of our gp120 model were classified as in-contact, clashing and far based on the distance between the atom and the closest atom on the antibody. Figure 8.8 shows the results by coloring gp120 using blue, red and green for the three classes, and also shows the number of atoms in-contact and clashing. The notable aspect is that even though NIH45-46, PGV04, VRC-PG4, VRC03 and b12 bind at the same site as CD4, they clash with the V1V2 loop. Hence, the configuration of

Epitope	Antibody	Source Model	# of atoms in contact	# of atoms in clash
Near CD4bs	CD4	new model	169	0
	NIH45-46	4JDT	161	8
	NIH45-46	3U7Y	241	8
	PGV04	3J5M	189	7
	VRC-PG04	3SE9	241	7
	VRC03	3SE8	355	39
Near 17b binding site	17b	new model	135	0
	48d	4DVR	131	1
	48d	3JWD	172	1
	PGT122	4NCO	70	3
	X5	2B4C	87	2
Other	b12	2NY7	573	117
	b13	3IDX	524	128
	b14	3IDY	507	125
	f105	3HI1	539	156
	PG9	3U2S	90	36

Table 8.3: **Comparison of the footprint of different antibodies on our gp120 model.** The antibodies are grouped by their epitope. The last two columns report the number of atoms of gp120 which comes in contact with the antibody, and the number of atoms of gp120 which clashes with the antibody. Two atoms a and b are considered to be in contact if the distance between their centers is less than r_a+r_b+1 where r_a and r_b are the radii of the atoms. Two atoms are considered to be clashing if the distance between their centers is less than $min(r_a, r_b)$. The contacts and clashes with the antibodies were computed after aligning our gp120 model to the x-ray model containing the antibody bound with gp120 (core or complete). As expected, antibodies that bind to the b13, b14, f105 epitopes have heavy clash with the V1V2 loop, since in our model the loop partially occludes these epitopes. But the interesting aspect is that all antibodies which bind at the CD4 binding site, also have clashes with the V1V2 loop. So, the V1V2 loop must be in a different configuration when gp120 binds with these antibodies. For example, in EMD5021 [161], we see that when b12 binds with gp120, the V1V2 loop stays in a similar ternary configuration as seen in the unliganded state.

V1V2 we found as optimal, is not optimal for those antibodies. This indicates that these neutralizing antibodies not only prevents CD4 binding, and thereby prevent

attachment to the cell membrane, they restrict the motion of V1V2 loop providing a second mechanism to prevent CCR5 binding.

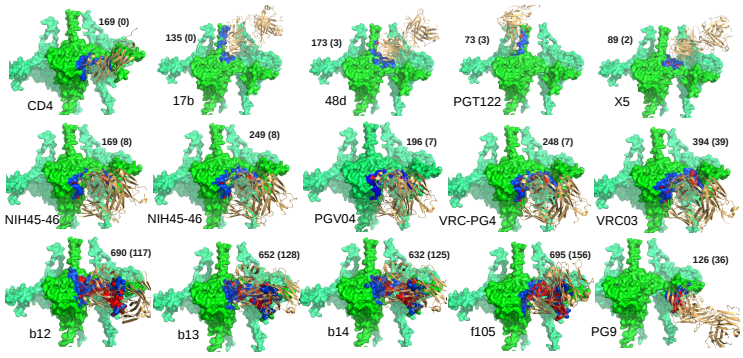


Figure 8.8: **Binding footprints, clashes and kinematic implications**. The footprint of different antibodies on our gp120 model are shown. In each figure, gp120 is colored lime, and the antibody is colored light orange (and rendered as ribbons). The parts of gp120 in contact with the antibody are colored blue, and the parts which have steric clash is colored red. In each figure, the name of the antibody is given. The positions of CD4 and 17b are based on our model. The positions of the rest are derived by aligning the gp120 model, which the antibody was co-crystallized with, to our model of gp120 (please see Table 8.3 for details). The numbers beside the figure report the number of atoms of gp120 which comes in contact, and the numbers inside the braces report the number of atoms of gp120 which have a clash with the antibody. Interestingly, though NIH45-46, VRC03, b12 etc. bind at the same site as CD4, we found that they have clashes with the V1V2 region.

8.4.5 Possible Crosslinks Indicate Clues to Conformational Change

Chemical crosslinking is often used to generate low resolution distance constraints between parts of a protein (or multiple proteins). We used Xwalk [138], a computational tool which mimics crosslinking experiments by calculating the distance between two residues along the surface of the proteins, to identify inter-domain(gp120-CD4 and gp120-17b) crosslinks in our model. We considered only crosslinks between ARG, ASP, GLU and LYS residues whose C-beta atoms were within 25Å of each other (Figure 8.7F and S4). As expected, we observed a high

number of crosslinks between residues at the CD4 binding site with CD4 and 17b binding site with the heavy chain of 17b. However, we also identified a large number of crosslinks between CD4 and the V1V2 region, and a few crosslinks between the light chain of 17b with the V3 region. The crosslinks between 17b-CD4, and 17b-V1V2 were very few. The lack of crosslink constraints between 17b and the V1V2 region, and the presence of high number of crosslink constraints between CD4 and V1V2, provides structural explanation for the conformational motion of the loop, specially the preference of the V1V2 to move away from the 17b binding site (or by extension, the CCR5 binding site).

8.4.6 Modeling Glycans

HIV-1 Env is heavily glycosylated. Though possible glycosylation sites are not difficult to identify, the exact type and configuration of the glycan, which can vary wildly between strains and depending on the bound partner [48], is difficult to model with high-confidence unless finer resolution EM-maps are available. Glycans have recently become targets of many recent antibodies which bind to glycans near V3 loop [136, 149], V1V2 loop [82, 177, 189], gp41-gp120 interface [45, 223] etc. However, glycan dependence of the CD4 and 17b binding interactions is still not clear. We have extracted glycans from 1RZJ for the glycosylation sites on the core and 4NCO for those on the variable loops. The glycans were grafted to their respective locations on the new model (see Figure 8.7(D-E). While all of the glycan sites including the ones on the variable loops seem distal from the CD4 and 17b binding sites in this static model, a more rigorous study of the glycan configurations

is warranted for a better understanding of the possible effect of the glycans on CD4 and 17b binding. For example, recently it was reported that some antibodies like PGT121 which binds far from the CD4bs, still manages to prevent CD4 binding by induced conformational changes to the glycans and variable loops [137]. We are currently exploring new scoring models which would use recently developed databases [132] of glycan structures to effectively quantify the quality of specific glycan configurations and their interactions with the protein chains.

8.5 Detailed Description of Different Stages of the Modeling Protocol Applied to gp120

Quality of Initial Stage Template-based Models Swiss-model [230] and I-TASSER [217] are two state of the art modeling software performing well at CASP challenges. When we tried to generate models using UniProtKB sequence P04578 (the same sequence as 1GC1:Chain G) to produce models which contained the variable regions, the sequence alignment and template identification tools picked 3JWD and 4NCO as templates. However, neither the model produced by Swiss-model or the 5 models produced by I-TASSER scored within 2 standard deviations of the expected $s_{external}$ and $s_{internal}$ values. Then, we generated more models (61 in total) by manually selecting different templates (gp120 cores from different crystal structures). Figure 8.10 shows a superposition of all the models generated in this stage.

In Figure 8.9(A-B) we have plotted the distribution of these models and compare the distribution with the crystal structures in our control set with respect to the range of $s_{external}$ and $s_{internal}$ values. The plots show that the I-TASSER

models are in general poorer quality under both the scoring terms. Swiss models on the other hand have better scores, but $s_{internal}$ is still not close to the crystal structures. Actually there was only one swiss-generated model which was assessed as low quality in terms of $s_{internal}$, and every other model was assessed as unacceptable. Also there was only 1 medium and 1 low quality model in terms of $s_{external}$. Energetically, however, the I-TASSER models have lower solvation energy (Figure 8.9(C)). We noticed that in most of the I-TASSER models, the V1V2 loop conformation brought it close to the core, introducing clashes with CD4, but lowering solvation energy by reducing exposed area. Finally, the average scores of the two sets of models are given in the tables in Fig 8.9(D-F). Detailed score for each model can be found in Figure 8.11-8.14. In these detailed figures, the models are identified by the template used for their core and since Swiss-model reports 5 candidate models, numbers 1-5 are used to distinguish them. Additionally, we also tried to use the bound configuration of gp120, CD4 and 17b (from 1GC1) together as templates to achieve better $s_{external}$ by ensuring fewer clashes with CD4 and 17b. However, these attempts produced even worse results (I-TASSER), or no results at all (Swiss-model).

Quality of Fragments Each of the initial models were decomposed into fragments (core, V1V2, V3, V4, C-termini and N-termini). We found that though an initial model scored poorly, it might contain a fragment which is locally quite feasible and gets a high score when considered on its own. For example, in Figure 8.15, we report the $s_{external}$ score for all V1V2 fragments, and notice that there are

a few acceptable ones. Then the fragments were clustered based on similarity under TM-score [276]. Some clusters for V1V2 and the four selected structures are shown in Figure 1(A-B) of the main text. Similar analyses were also done for the other fragments.

Quality of Models Generated by Fragment Assembly and Refinement Since all the component fragments are already fitted into the density map in their correct relative orientations, assembly does not require any major reconfiguration. However the structures are not stereochemically sound as the bond lengths/angles at the joint are too far from ideal (see Figure 1C in the main text for example). However after local structural refinement and energy minimization, the stereochemical and energetic quality of the models are significantly improved. Also, $s_{external}$ scores are better than initial model, which is unsurprising given the procedure of fragment selection employed in our protocol. Please see Figures 8.9(D-F) for a summary of the scores (in the Figure, these models are referred to as 'spliced models'. Detailed breakdown of scores for each model is given in Figure 8.16. Figure 8.17 shows a superposition of all the spliced models. In the figures (as well in the following text), the models are simply identified using different numbers (and do not carry any other meaning).

We clustered the spliced models based on TM-scores and then selected six models which were in different clusters (see the clusters in Figures 8.18-8.19) and received high $s_{external}$ scores. Two of the models (Model31 and Model56) were rated medium quality, 1 model (Model25) was rated low quality, and the other three

(Model20, Model23 and Model35) were rated unacceptable. The six selected models are shown in Figure 8.20. Note that poorer models were also selected to ensure diversity, and in fact Model35 became high quality after co-optimization further energy minimization (see next section).

Quality of Models After Co-optimization We first applied our docking [65] and fitting [23, 42] protocols to improve the relative configuration of gp120, CD4 and 17b with each other as well as with respect to the EM map EMD5020. As a result, the score for almost all the terms improved significantly (compare the scores reported in Figure 8.16 to the ones in Figure 8.21-8.22). Also, Figures 8.9(A-F) show all the optimized (the term 'final' also used in some of the Figures) models have very good energetics and $s_{internal}$ scores. However, two models (Model20 and Model23) have quite poor $s_{external}$ scores.

Model31 and Model35 both are assessed as high quality in terms of both $s_{internal}$ and $s_{external}$ scores, and considered medium quality in terms of energy. Note that high quality means that they scored better than average crystal structures in our control, and according to the validation and correlation mentioned before, their qualities are equivalent to resolutions better than 3.5Å. However, our binding site analysis (see next section) revealed that Model31 was better than Model35 in terms of the binding interface offered to CD4 and 17b. So, Model31 was was further refined using side-chain repositioning at the interfaces to remove any clashes and improve residue contacts. The quality of this final model is described separately in a later section.

Binding Site Analysis The binding site analysis is performed by docking (using F2Dock [65]) CD4 and 17b with the optimized models, and F2Dock reports the top 1000 possible binding poses. We define the parts of the surface of gp120 which is in contact with the CDR loops of CD4 and 17b in a docking pose as the footprint/site of that particular pose. For each point on the surface of gp120 model, we compute the ratio of the number of poses whose footprint includes the point, and the total number of poses as the probability of that point being on the binding site. We found that model31 showed more binding specificity near the correct CD4 and 17b binding sites compared to model35 (see Figures 8.9(G-J)). Hence, Model31 was chosen as the final model.

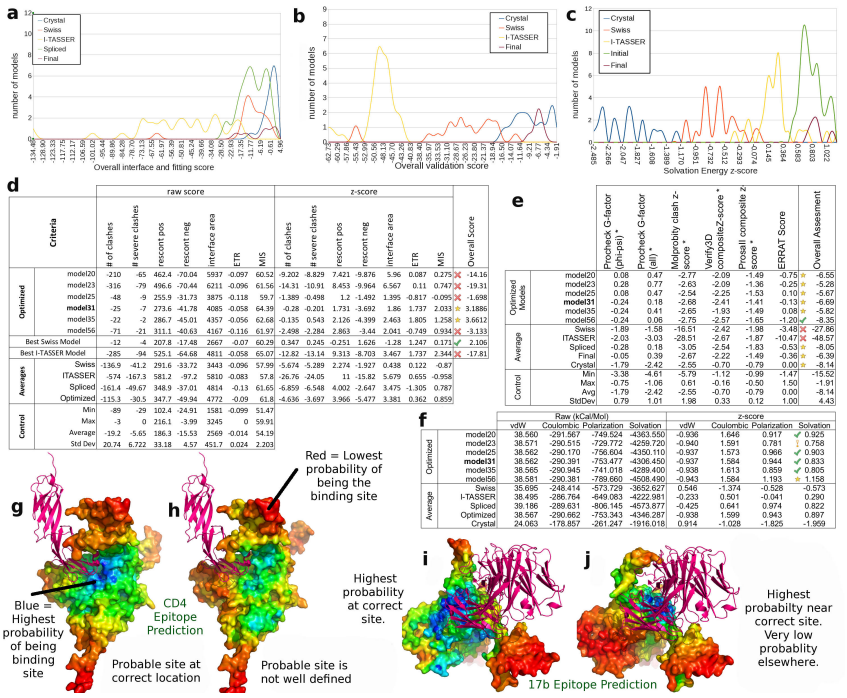


Figure 8.9: **Overview of the modeling excercise (more details in Figures 8.10 to 8.24).** In **(A)-(C)** we plotted the distribution of models generated by direct application of Swiss-model [230] and I-TASSER [217] with different templates, the models made by selected assembly, the optimized models and crystal structures in terms of overall interface and fitting score, overall validation score and solvation energy z-score. The distributions show that the optimized models get, in general, similar scores as crystal structures. Swiss models and spliced model distributions are slightly lower quality. Detailed scores for the 6 models which we selected for optimization is given in **(D)-(F)**, which also reports the average qualities of the models produced by them, as well as the models we generated by selective assembly. The icons in the last column correspond to classification of the models as 'High', 'Medium', 'Low' and 'Unacceptable' using star, check, exclamation and cross symbols. From these tables, we find two models, namely Model31 and Model35 which are consistently ranked high. To chose only one model out of the two, we used our validated docking protocol to predict possible binding sites for CD4 and 17b on both models and compared the result with expected binding site. **(G)** and **(H)** compare the CD4 binding sites predicted on the two models. Each point on the surface of the models are colored by the probability of it being at the binding interface, where blue-red gradient (like a rainbow) is used to show high-low probabilities. Model31 shows higher affinity at the correct site. **(I)** and **(J)** compare the 17b binding sites. Both model have high probabilities near the correct site. Model31 also have slightly lower probability of binding at the b12 epitope as well, which is not desired. Model35 on the other hand have a very focused predicted site, but slightly offset from the correct site. Since Model31 shows good binding preference at the correct sites for both CD4 and 17b, we chose this as our final model. Note that the two models have the same V1V2 and core fragments, but different V3 and V4 fragments.

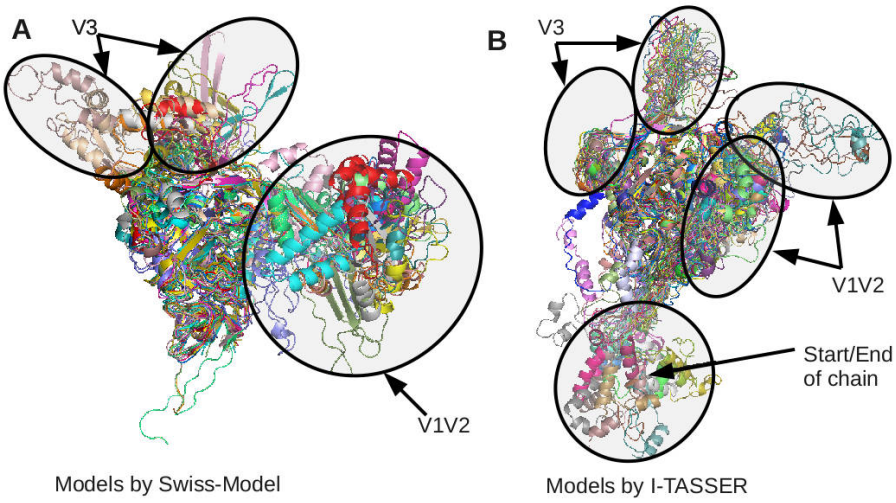


Figure 8.10: **Superposition of initial stage models produced by Swiss-Model [230] and I-TASSER [217].** Notice that the V1V2 loops in most I-TASSER model comes close to the gp120 core and shields the CD4 binding site. Such models score poorly under our $s_{external}$ score which penalizes models that does not offer suitable binding interfaces for CD4 and 17b. Models for the start/end sections of the gp120 chain produced by I-TASSER were also poor in terms of fitting with the density map. See scoring details in Figures 8.11-8.14.

	Criteria	raw score							z-score							Overall Score
		# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	
SWISS Models	1G9M	-147	-46	364.8	-45.16	4437	-0.13	54.53	-6.163	-6.003	4.482	-4.431	2.641	-1.319	-2.441	✗ -13.23
	1G9N	-625	-189	567.1	-119	4728	-0.038	54.93	-29.22	-27.28	10.58	-20.58	3.283	2.583	-2.26	✗ -62.89
	1GC1	-66	-15	266.6	-27.71	3514	-0.111	53.72	-2.257	-1.391	1.521	-0.613	0.596	-0.518	-2.808	✗ -5.469
	1RZJ	-12	-4	207.8	-17.48	2667	-0.07	60.29	0.347	0.245	-0.251	1.626	-1.28	1.247	0.171	✓ 2.106
	1YYL	-183	-47	345	-44.15	3936	-0.037	57.37	-7.9	-6.151	3.882	-4.21	1.53	2.621	-1.152	✗ -11.38
	1YYM	-198	-65	409.9	-41.96	4315	-0.062	59.28	-8.623	-8.829	5.841	-3.731	2.37	1.584	-0.287	✗ -11.67
	2I5Y	-206	-61	390.1	-47.38	4054	-0.053	62.83	-9.009	-8.234	5.244	-4.917	1.793	1.952	1.325	✗ -11.85
	2I60	-13	-8	227.6	-27.71	3073	-0.054	57.15	0.299	-0.35	0.347	-0.613	-0.381	1.921	-1.255	! -0.031
	2NXZ	-17	-6	215.4	-24.12	3007	-0.119	56.82	0.106	-0.052	-0.023	0.173	-0.527	-0.838	-1.402	✗ -2.564
	2NY0	-32	-9	212.7	-20.89	2981	-0.113	56.66	-0.617	-0.498	-0.104	0.88	-0.584	-0.605	-1.476	✗ -3.004
	2NY3	-69	-14	243.8	-34.17	3540	-0.07	55.56	-2.402	-1.242	0.832	-2.026	0.653	1.214	-1.975	✗ -4.946
	2NY5	-23	-6	214	-27.71	2844	-0.072	53.47	-0.183	-0.052	-0.064	-0.613	-0.888	1.162	-2.924	✗ -3.561
	2NY7	-40	-8	139.8	-8.01	1938	-0.178	57.57	-1.003	-0.35	-2.301	3.698	-2.893	-3.366	-1.064	✗ -7.279
	2QAD	-155	-42	328.3	-23.69	4620	-0.103	57.14	-6.549	-5.408	3.379	0.267	3.044	-0.175	-1.258	✗ -6.699
	3HI1	-181	-54	217.5	-24.55	2572	-0.12	61.72	-7.803	-7.193	0.04	0.079	-1.489	-0.887	0.822	✗ -16.43
	3IDY	-17	-5	139.3	-18.03	1954	-0.273	56.65	0.106	0.097	-2.316	1.506	-2.859	-7.383	-1.48	✗ -12.33
	3JW0	-202	-63	361.8	-31.12	3812	-0.071	60.97	-8.816	-8.532	4.391	-1.359	1.256	1.202	0.483	✗ -11.37
	3JWD	-199	-67	357.3	-35.72	3975	-0.064	61.14	-8.671	-9.127	4.253	-2.366	1.616	1.466	0.56	✗ -12.27
	3RJQ	-156	-53	261.1	-24.67	2898	-0.12	61.26	-6.597	-7.044	1.355	0.053	-0.769	-0.885	0.614	✗ -13.27
	3SE8	-196	-62	361.8	-31.12	3989	-0.064	60.82	-8.526	-8.383	4.391	-1.359	1.648	1.466	0.411	✗ -10.35

Figure 8.11: **Detailed $s_{external}$ scores for initial template-based models generated by Swiss-Model [230].** The raw scores report the number of minor and severe clashes, the overall positive and negative residue-contact scores and the interface area, all of which measure the quality of the interface between a gp120 model, with CD4 and 17b. This table also reports the external total ratio (ETR) and mutual information score (MIS), which measure the quality of fitting of a model with the EM density map (EMD5020 [161]. Please see the model evaluation criteria section for details. The z-scores were computed as deviations from the mean score of a set of existing x-ray models where gp120 is in complex with CD4, 17b and other antibodies. The last column shows the sum of z-scores. Finally, the star, check, exclamation and cross icons visually highlight overall scores which are $\geq \mu + 2\sigma$, $\geq \mu + \sigma$, $\geq \mu$ and $< \mu$ respectively (here (μ) and (σ) relate to the observed mean and stddev of overall scores for the complexes in our benchmark). We notice that only one of these initial template-based models have acceptable score.

	Criteria	raw score							z-score							Overall Score
		# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	
I-TASSER models	1G9M_model1	-446	-127	516.8	-186.2	5686	-0.129	59.82	-20.58	-18.05	9.063	-35.29	5.405	-1.26	-0.043	-60.76
	1G9M_model2	-530	-161	608.3	-90.35	5704	-0.106	54.86	-24.63	-23.11	11.82	-14.32	5.445	-0.282	-2.293	-47.38
	1G9M_model3	-605	-168	556.4	-109.9	5765	-0.075	54.11	-28.25	-24.15	10.26	-18.6	5.579	1.014	-2.633	-56.79
	1G9M_model4	-685	-183	690.6	-131.2	6127	-0.093	57	-32.11	-26.38	14.3	-23.26	6.381	0.27	-1.319	-62.12
	1G9M_model5	-351	-93	437.1	-41.17	4973	-0.073	55.67	-16	-12.99	6.66	-3.558	3.827	1.099	-1.925	-22.89
	1G9N_model1	-447	-149	544.9	-77.08	5824	-0.11	57.26	-20.63	-21.33	9.91	-11.42	5.71	-0.474	-1.202	-39.43
	1G9N_model2	-428	-122	529	-62.64	5370	-0.109	60.01	-19.72	-17.31	9.43	-8.257	4.705	-0.431	0.044	-31.53
	1G9N_model3	-718	-197	682.3	-173.9	7022	-0.035	57.97	-33.7	-28.47	14.05	-32.61	8.363	2.735	-0.88	-70.51
	1G9N_model4	-459	-136	511	-55.21	6069	-0.124	65.7	-21.21	-19.39	8.886	-6.631	6.252	-1.069	2.629	-30.53
	1G9N_model5	-384	-129	491.5	-77.32	5730	-0.109	62.17	-17.59	-18.35	8.299	-11.47	5.503	-0.41	1.026	-32.99
	1GC1_model1	-285	-94	525.1	-64.68	4811	-0.058	65.07	-12.82	-13.14	9.313	-8.703	3.467	1.737	2.344	-17.81
	1GC1_model2	-852	-237	700.8	-114.8	6670	-0.061	59.34	-40.16	-34.42	14.61	-19.67	7.584	1.609	-0.258	-70.71
	1GC1_model3	-249	-67	359.3	-48.82	4630	-0.133	59.07	-11.08	-9.127	4.316	-5.232	3.067	-1.43	-0.38	-19.87
	1GC1_model4	-1201	-380	828.2	-242.9	6222	-0.047	56.89	-56.99	-55.69	18.45	-47.69	6.591	2.225	-1.369	-134.5
	1GC1_model5	-274	-67	388.5	-34.96	4855	-0.12	60.66	-12.29	-9.127	5.194	-2.199	3.564	-0.877	0.341	-15.39
	1RZJ_model1	-386	-113	472.3	-76.2	6126	-0.07	58.32	-17.69	-15.97	7.721	-11.22	6.38	1.226	-0.721	-30.28
	1RZJ_model2	-677	-189	628.1	-119.8	5935	-0.054	56.38	-31.72	-27.28	12.42	-20.77	5.956	1.907	-1.603	-61.09
	1RZJ_model3	-624	-178	579	-68.85	5313	-0.061	56.16	-29.17	-25.64	10.94	-9.616	4.578	1.63	-1.703	-48.98
	1RZJ_model4	-968	-283	825.5	-125.2	7704	-0.09	55.54	-45.76	-41.26	18.37	-21.94	9.872	0.376	-1.984	-82.33
	1RZJ_model5	-285	-93	417.8	-76.1	5354	-0.119	59.5	-12.82	-12.99	6.078	-11.2	4.67	-0.856	-0.186	-27.31
	1YYL_model1	-428	-141	470.9	-89.99	5612	-0.078	55.77	-19.72	-20.14	7.677	-14.24	5.241	0.908	-1.877	-42.14
	1YYL_model2	-524	-157	600.2	-78.39	5599	-0.057	55.9	-24.34	-22.52	11.58	-11.7	5.212	1.8	-1.818	-41.79
	1YYL_model3	-509	-171	531.5	-113.3	6115	-0.024	57.37	-23.62	-24.6	9.506	-19.35	6.356	3.203	-1.155	-49.66
	1YYL_model4	-570	-173	617.2	-86.55	6274	-0.097	58.81	-26.56	-24.9	12.09	-13.49	6.708	0.1	-0.5	-46.55
	1YYL_model5	-673	-173	655.8	-92.79	6464	-0.077	58.01	-31.53	-24.9	13.25	-14.85	7.128	0.929	-0.865	-50.84
	2B4C_model1	-444	-117	484.3	-80.67	4931	-0.126	57.15	-20.49	-16.56	8.081	-12.2	3.734	-1.132	-1.252	-39.82
	2B4C_model2	-701	-206	585.7	-118	5657	-0.096	53.52	-32.88	-29.8	11.14	-20.37	5.341	0.143	-2.9	-69.33
	2B4C_model3	-838	-270	681.1	-91.75	6447	-0.043	56.84	-39.49	-39.33	14.01	-14.63	7.089	2.374	-1.393	-71.36
	2B4C_model4	-203	-61	358	-39.93	4520	-0.124	59.39	-8.864	-8.234	4.275	-3.287	2.822	-1.069	-0.237	-14.59
	2B4C_model5	-782	-223	732.6	-110	6937	-0.078	55.87	-36.79	-32.33	15.56	-18.62	8.174	0.908	-1.834	-64.93
	2I60_model1	-622	-183	570.2	-68.09	4783	-0.07	53.85	-29.07	-26.38	10.67	-9.449	3.406	1.248	-2.753	-52.33
	2I60_model2	-699	-210	640.5	-83.56	6455	-0.048	54.21	-32.78	-30.4	12.79	-12.83	7.107	2.183	-2.586	-56.53
	2I60_model3	-813	-220	725.5	-92.3	5530	-0.069	53.32	-38.28	-31.89	15.35	-14.75	5.059	1.29	-2.993	-66.21
	2I60_model4	-421	-132	577.7	-68.92	5434	-0.107	62.86	-19.38	-18.8	10.9	-9.631	4.848	-0.346	1.341	-31.07
	2I60_model5	-1010	-254	819.3	-210.7	6697	-0.06	58.58	-47.78	-36.95	18.18	-40.65	7.643	1.652	-0.606	-98.51

Figure 8.12: **Detailed $s_{external}$ scores for initial template-based models generated by I-TASSER [217].** This table is formatted the same way as the table in Figure 8.11. The scores of these models are even worse. The reason is that I-TASSER works very well for unliganded or tertiary structure modeling, and hence the models produced by them include a configuration of the loops which is stable in anSupplementary unliganded state, but does not favor/expose the binding sites.

		Procheck G-factor (phi-psi) *	Procheck G-factor (all) *	Molprobity clash z-score *	Verify3D compositeZ-score *	Prosall composite z-score *	ERRAT Score	Overall Assesment
Models by Swiss-Model	1G9M	-1.73	-1.66	-10.01	-2.25	-1.53	-2.66	✖ -19.84
	1G9N	-2.28	-2.19	-18.49	-2.09	-1.65	-2.90	✖ -29.60
	1GC1	-2.60	-2.84	-9.92	-2.09	-1.78	-2.20	✖ -21.43
	1RZJ	-2.01	-1.83	-18.68	-2.09	-1.65	-3.02	✖ -29.28
	1YYL	-1.97	-1.42	-14.28	-2.89	-1.61	-4.42	✖ -26.59
	1YYM	-1.22	-0.83	-10.01	-2.57	-1.70	-1.07	✖ -17.40
	2I5Y	-1.65	-1.06	-10.60	-2.73	-1.41	-3.62	✖ -21.07
	2I60	-1.65	-1.30	-13.39	-2.41	-1.82	-3.77	✖ -24.34
	2NXZ	-1.10	-0.47	-7.54	-1.77	-1.61	-0.17	❗ -12.66
	2NY0	-1.22	-0.95	-10.39	-2.09	-1.86	-2.43	✖ -18.93
	2NY3	-1.38	-1.06	-16.84	-2.25	-1.61	-3.12	✖ -26.26
	2NY5	-0.87	-0.53	-22.88	-1.93	-1.90	-2.67	✖ -30.78
	2NY7	-1.93	-1.54	-14.41	-3.53	-2.07	-3.61	✖ -27.09
	2QAD	-2.68	-2.54	-4.65	-1.93	-1.12	-3.62	✖ -16.54
	3HI1	-1.85	-1.42	-13.25	-2.73	-1.65	-3.09	✖ -23.99
	3IDY	-2.36	-2.01	-7.41	-2.73	-1.94	-3.36	✖ -19.81
	3JWD	-2.36	-1.95	-37.99	-2.73	-3.47	-6.20	✖ -54.70
	3JWO	-2.64	-2.31	-15.83	-2.41	-3.64	-5.86	✖ -32.68
	3RJQ	-2.01	-1.77	-21.36	-2.41	-2.07	-5.56	✖ -35.18
	3SE8	-2.36	-1.95	-38.03	-2.73	-3.52	-6.18	✖ -54.77

Figure 8.13: **Structure validation data ($s_{internal}$) for initial template-based models generated by Swiss model [230].** Each column reports the composite score for the models under different model evaluation criteria. Please see the model evaluation criteria section in the supplement, as well as the scoring and validation sections in the main paper for descriptions of the criteria. The overall score was computed as the sum of individual scores. The overall scores, under the same criteria, were also computed for a benchmark of x-ray models of gp120 and the mean, $< \mu$, and standard deviation, $< \sigma$ of those scores were used to classify these models. The star, check, exclamation and cross icons in the table visually highlight overall scores which are $\geq \mu + 2\sigma$, $\geq \mu + \sigma$, $\geq \mu$ and $< \mu$ respectively. All models, except for one, have poor stereochemistry.

		Procheck G-factor (phi-psi) *	Procheck G-factor (all) *	Molprobity clash z-score *	Verify3D compositeZ-score *	Prosall composite z-score *	ERRAT Score		Overall Assesment
Models by I-TASSER	1G9M_model1	-2.05	-3.02	-29.04	-2.41	-2.11	-10.37	✗	-49.00
	1G9M_model2	-1.85	-2.66	-24.69	-2.41	-1.65	-9.18	✗	-42.44
	1G9M_model3	-2.28	-3.37	-40.08	-2.09	-1.74	-11.46	✗	-61.02
	1G9M_model4	-1.89	-2.72	-30.20	-2.73	-2.40	-10.73	✗	-50.67
	1G9M_model5	-1.97	-2.90	-28.53	-2.89	-1.70	-11.35	✗	-49.34
	1G9N_model1	-1.97	-3.02	-26.98	-2.73	-2.03	-10.00	✗	-46.73
	1G9N_model2	-1.97	-3.02	-23.78	-3.05	-2.07	-8.91	✗	-42.80
	1G9N_model3	-1.97	-2.90	-29.12	-2.73	-1.94	-10.64	✗	-49.30
	1G9N_model4	-1.97	-3.02	-25.25	-3.21	-1.78	-9.70	✗	-44.93
	1G9N_model5	-1.85	-3.13	-28.18	-2.57	-1.57	-10.43	✗	-47.73
	1GC1_model1	-1.85	-2.96	-26.13	-2.89	-1.94	-10.34	✗	-46.11
	1GC1_model2	-1.69	-2.54	-29.17	-2.89	-1.45	-11.15	✗	-48.89
	1GC1_model3	-1.77	-2.84	-29.13	-3.05	-1.99	-10.58	✗	-49.36
	1GC1_model4	-1.97	-2.96	-30.18	-2.57	-1.70	-10.63	✗	-50.01
	1GC1_model5	-2.12	-3.02	-25.77	-3.21	-2.11	-10.18	✗	-46.41
	1RZJ_model1	-1.85	-2.90	-26.37	-2.57	-1.70	-9.35	✗	-44.74
	1RZJ_model2	-1.65	-2.48	-27.86	-2.25	-1.86	-10.87	✗	-46.98
	1RZJ_model3	-2.28	-3.19	-27.53	-2.41	-1.53	-11.06	✗	-48.00
	1RZJ_model4	-2.32	-3.55	-36.32	-3.05	-2.56	-11.28	✗	-59.08
	1RZJ_model5	-2.32	-3.25	-25.49	-3.37	-2.11	-10.53	✗	-47.07
	1YYL_model1	-2.16	-3.19	-29.30	-2.25	-1.94	-9.72	✗	-48.56
	1YYL_model2	-1.89	-3.02	-25.56	-2.73	-1.99	-10.53	✗	-45.72
	1YYL_model3	-1.97	-2.84	-31.27	-2.57	-2.07	-11.01	✗	-51.73
	1YYL_model4	-2.28	-3.08	-26.30	-2.73	-1.74	-10.09	✗	-46.22
	1YYL_model5	-2.05	-3.19	-28.91	-3.21	-2.07	-9.30	✗	-48.73
	2B4C_model1	-1.89	-2.90	-25.60	-2.57	-1.65	-9.39	✗	-44.00
	2B4C_model2	-1.85	-2.84	-27.67	-2.09	-1.41	-11.04	✗	-46.90
	2B4C_model3	-2.28	-3.19	-27.12	-2.09	-1.49	-10.15	✗	-46.32
	2B4C_model4	-2.40	-3.61	-39.19	-2.89	-2.48	-12.16	✗	-62.73
	2B4C_model5	-2.60	-3.55	-27.58	-2.25	-1.86	-11.60	✗	-49.44
	2I60_model1	-2.20	-3.25	-27.75	-2.09	-1.65	-10.14	✗	-47.08
	2I60_model2	-1.97	-3.08	-28.61	-3.21	-1.86	-10.99	✗	-49.72
	2I60_model3	-2.05	-3.31	-28.50	-2.25	-1.78	-10.72	✗	-48.61
	2I60_model4	-1.73	-2.42	-24.81	-3.05	-1.90	-9.46	✗	-43.37
	2I60_model5	-2.16	-3.19	-28.53	-2.25	-1.45	-11.32	✗	-48.90

Figure 8.14: **Structure validation data ($s_{internal}$) for initial template-based models generated by I-TASSER [217]** In the same format as Figure 8.13. All models have poor scores.

	Criteria	raw score							z-score							Overall Score	
		# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS		
V1V2 Models Only	**model01**	-24	-10	95.64	-6.82	1124	-0.407	56.23	-0.231	-0.647	-0.131	2.884	-0.169	-6.067	-1.67	✓	-6.032
	model02	-316	-90	301.9	-61.02	2288	-0.217	57.34	-14.31	-12.55	6.086	-8.977	2.409	-4.167	-1.166	✗	-32.68
	model03	-440	-126	382.4	-89.16	3259	-0.193	58.44	-20.29	-17.9	8.512	-15.13	4.559	-3.933	-0.668	✗	-44.86
	model04	-258	-64	223.3	-40.2	2725	-0.273	55.04	-11.52	-8.68	3.716	-4.421	3.376	-4.733	-2.209	✗	-24.47
	model05	-682	-180	495.4	-156.7	3401	-0.133	57.45	-31.96	-25.94	11.92	-29.92	4.874	-3.333	-1.119	✗	-75.48
	model06	-150	-34	150.6	-34.1	2112	-0.247	55.34	-6.308	-4.217	1.526	-3.086	2.02	-4.467	-2.075	?	-16.61
	model07	-622	-184	393.5	-132.8	2295	-0.197	53.31	-29.07	-26.53	8.845	-24.68	2.425	-3.967	-2.996	✗	-75.98
	model08	-332	-114	308.2	-54.2	2574	-0.19	57	-15.09	-16.12	6.276	-7.484	3.042	-3.9	-1.322	✗	-34.59
	model09	-390	-130	330.9	-60.66	2633	-0.133	57.19	-17.88	-18.5	6.96	-8.898	3.174	-3.333	-1.236	✗	-39.72
	model10	-394	-100	320.3	-74.3	2702	-0.127	58.05	-18.08	-14.04	6.64	-11.88	3.325	-3.267	-0.846	✗	-38.14
	model11	-196	-54	163.9	-20.46	1246	-0.227	55.95	-8.526	-7.193	1.927	-0.101	0.102	-4.267	-1.799	?	-19.86
	model12	-236	-78	197	-69.06	2452	-0.23	56.15	-10.46	-10.76	2.924	-10.74	2.771	-4.3	-1.708	✗	-32.27
	model13	0	0	5.32	0	137.2	-0.273	54.89	0.926	0.841	-2.854	4.377	-2.353	-4.733	-2.277	✓	-6.074
	model14	-308	-92	342.8	-67.84	2649	-0.22	58.51	-13.93	-12.85	7.319	-10.47	3.207	-4.2	-0.635	✗	-31.55
	model15	-478	-120	332.9	-47.38	2762	-0.213	59.42	-22.13	-17.01	7.02	-5.992	3.458	-4.133	-0.222	✗	-39.01
	model16	-146	-36	67.92	-33.74	1355	-0.337	58.95	-6.115	-4.515	-0.967	-3.007	0.343	-5.367	-0.436	✗	-20.06
	model17	-648	-184	390.1	-99.58	2960	-0.073	58.12	-30.32	-26.53	8.744	-17.42	3.898	-2.733	-0.815	✗	-65.18
	model18	0	0	14.9	0	235.9	-0.263	62.69	0.926	0.841	-2.565	4.377	-2.134	-4.633	1.263	✓	-1.927
	model19	0	0	23.86	-6.82	338.7	-0.34	52.82	0.926	0.841	-2.295	2.884	-1.907	-5.4	-3.216	✓	-8.167
	model20	-294	-78	226.4	-67.12	2984	-0.217	58	-13.25	-10.76	3.81	-10.31	3.949	-4.167	-0.867	✗	-31.6
	model21	-364	-104	313.5	-74.66	2529	-0.23	56.41	-16.63	-14.63	6.433	-11.96	2.942	-4.3	-1.589	✗	-39.73
	model22	-418	-100	302.8	-74.3	3273	-0.143	57.77	-19.23	-14.04	6.112	-11.88	4.59	-3.433	-0.97	✗	-38.85
	model23	-928	-266	703.7	-159.7	5414	-0.097	57.17	-43.83	-38.73	18.2	-30.57	9.329	-2.967	-1.242	✗	-89.81
	model24	-154	-66	167.3	-27.28	1331	-0.263	58	-6.501	-8.978	2.027	-1.593	0.291	-4.633	-0.868	✗	-20.26
	model25	-198	-54	203.2	-47.74	1495	-0.133	53.96	-8.623	-7.193	3.11	-6.071	0.654	-3.333	-2.701	✗	-24.16
	model26	-258	-78	208.2	-61.02	2498	-0.13	59.35	-11.52	-10.76	3.26	-8.977	2.874	-3.3	-0.255	✗	-28.68
	model27	-408	-124	363.2	-80.76	3142	-0.103	57.49	-18.75	-17.61	7.931	-13.3	4.3	-3.033	-1.097	✗	-41.55
	model28	-326	-110	296.7	-60.3	2919	-0.047	55.63	-14.8	-15.52	5.928	-8.819	3.806	-2.467	-1.941	✗	-33.81
	model29	-502	-158	391.2	-73.94	3964	-0.167	58.53	-23.28	-22.66	8.777	-11.8	6.119	-3.667	-0.627	✗	-47.15
	model30	-602	-152	372.7	-87.94	3498	-0.183	57.53	-28.11	-21.77	8.219	-14.87	5.087	-3.833	-1.08	✗	-56.35
	model31	-234	-76	281.4	-27.28	1677	-0.17	55.56	-10.36	-10.47	5.467	-1.593	1.055	-3.7	-1.975	✗	-21.57
	model32	-398	-120	275.8	-76.68	2754	-0.157	55.84	-18.27	-17.01	5.298	-12.4	3.442	-3.567	-1.848	✗	-44.36
	model33	-664	-210	373.2	-97.08	3442	-0.193	54.29	-31.1	-30.4	8.233	-16.87	4.963	-3.933	-2.55	✗	-71.65
	model34	-930	-286	541.3	-112	4285	-0.163	53.6	-43.92	-41.71	13.3	-20.13	6.831	-3.633	-2.863	✗	-92.13
	model35	-2	0	29.28	0	455.3	-0.177	57.82	0.829	0.841	-2.131	4.377	-1.649	-3.767	-0.947	✓	-2.447
	model36	-454	-136	415.6	-33.38	2757	-0.303	54.34	-20.97	-19.39	9.513	-2.928	3.448	-5.033	-2.529	✗	-37.89
	model37	-24	-4	68.68	-6.82	715.7	-0.897	58.8	-0.231	0.245	-0.944	2.884	-1.072	-10.97	-0.503	?	-10.59
	model38	-208	-60	238.9	-47.38	1493	-0.13	55.03	-9.105	-8.085	4.186	-5.992	0.648	-3.3	-2.217	✗	-23.86
	model39	-10	-6	56.76	-13.64	601.4	-0.19	56.4	0.444	-0.052	-1.303	1.392	-1.325	-3.9	-1.591	✓	-6.337
	model40	-584	-178	393.6	-81.12	2875	-0.18	56.3	-27.24	-25.64	8.848	-13.38	3.708	-3.8	-1.64	✗	-59.14
	model41	-508	-140	339.4	-74.66	3839	-0.133	58.1	-23.57	-19.99	7.216	-11.96	5.843	-3.333	-0.822	✗	-46.62
	model42	-790	-230	547.9	-85.44	3605	-0.133	57.15	-37.17	-33.37	13.5	-14.32	5.325	-3.333	-1.254	✗	-70.63
	model43	-164	-54	205.1	-27.28	1065	-0.167	56.08	-6.983	-7.193	3.167	-1.593	-0.299	-3.667	-1.741	?	-18.31
	model44	-830	-206	563.2	-157.8	4069	0	53.28	-39.1	-29.8	13.96	-30.16	6.352	-2	-3.008	✗	-83.76
	model45	0	0	24.72	-20.1	381.4	-0.423	52.36	0.926	0.841	-2.269	-0.022	-1.812	-6.233	-3.427	?	-12
	model46	-22	-8	25.4	-6.82	411	-0.383	55.79	-0.135	-0.35	-2.248	2.884	-1.747	-5.833	-1.869	✓	-9.298
	model47	-14	-6	31.1	-13.64	605.1	-0.153	54.52	0.251	-0.052	-2.077	1.392	-1.317	-3.533	-2.446	✓	-7.782
	model48	0	0	46.58	0	582.3	-0.553	52.07	0.926	0.841	-1.61	4.377	-1.368	-7.533	-3.557	✓	-7.925
	model49	-18	-4	30.6	-6.82	612.3	-0.32	53.56	0.058	0.245	-2.092	2.884	-1.301	-5.2	-2.881	✓	-8.286
	model50	-10	-2	23.4	-13.64	449.3	-0.363	61.69	0.444	0.543	-2.309	1.392	-1.662	-5.633	0.809	✓	-6.416
	model51	0	0	0	0	0	-0.82	58.02	0.926	0.841	-3.014	4.377	-2.657	-10.2	-0.856	?	-10.58
	model52	-120	-22	158.6	-20.46	2650	-0.327	58.86	-4.861	-2.432	1.767	-0.101	3.21	-5.267	-0.479	✓	-8.162
	model53	0	0	0	0	0	-0.49	53.91	0.926	0.841	-3.014	4.377	-2.657	-6.9	-2.725	✓	-9.152
	model54	0	0	0	0	0	-0.633	54.26	0.926	0.841	-3.014	4.377	-2.657	-8.333	-2.566	?	-10.43
	model55	0	0	0	0	0	-0.857	56.5	0.926	0.841	-3.014	4.377	-2.657	-10.57	-1.546	?	-11.64
	model56	-116	-48	115.9	-13.64	1039	-0.153	58.69	-4.668	-6.3	0.479	1.392	-0.357	-3.533	-0.555	?	-13.54
	model57	-116	-48	115.9	-13.64	1040	-0.157	61.48	-4.668	-6.3	0.479	1.392	-0.355	-3.567	0.714	?	-12.31
	model58	0	0	0	0	0	-0.45	59.88	0.926	0.841	-3.014	4.377	-2.657	-6.5	-0.012	✓	-6.039
	model59	-110	-48	115.9	-13.64	1039	-0.153	60.4	-4.379	-6.3	0.479	1.392	-0.356	-3.533	0.222	?	-12.48
	model60	-116	-48	115.9	-13.64	1039	-0.153	58.69	-4.668	-6.3	0.479	1.392	-0.357	-3.533	-0.555	?	-13.54
	model61	-116	-48	115.9	-13.64	1040	-0.153	60.46	-4.668	-6.3	0.479	1.392	-0.355	-3.533	0.25	?	-12.74

Figure 8.15: **Detailed $s_{external}$ scores for V1V2 fragment models.** Different thresholds were used for the z-score computations since only fragments are scored here. Hence the scores are not directly comparable to the scores in Figures 8.11 and 8.12 for example. But they correctly rank the fragments, and several fragments are identified which are considerably better than other fragments. The selected fragments are shown in bold letters. Note that two clusters with the same folds were not selected.

	Criteria	raw score							z-score							Overall Score
		# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	
Spliced Models	model1	-307	-95	486.7	-55.79	6494	-0.131	59.72	-13.88	-13.29	8.156	-6.758	7.195	-1.366	-0.085	-20.03
	model2	-319	-98	487.4	-55.79	6471	-0.132	64.21	-14.46	-13.74	8.175	-6.758	7.142	-1.387	1.95	-19.07
	model3	-340	-108	487.2	-55.97	6574	-0.178	60.4	-15.47	-15.23	8.169	-6.797	7.372	-3.364	0.223	-25.09
	model4	-342	-106	487.2	-55.97	6543	-0.179	60.44	-15.57	-14.93	8.169	-6.797	7.302	-3.406	0.242	-24.99
	model5	-325	-102	486.5	-62.25	6707	-0.135	59.25	-14.75	-14.33	8.149	-8.171	7.666	-1.536	-0.298	-23.27
	model6	-325	-100	486.5	-62.25	6679	-0.136	67.52	-14.75	-14.04	8.149	-8.171	7.603	-1.579	3.456	-19.33
	model7	-168	-49	314.1	-34.75	4558	-0.136	63.41	-7.176	-6.449	2.951	-2.153	2.908	-1.557	1.588	-9.889
	model8	-167	-51	314.1	-34.75	4543	-0.136	65.57	-7.128	-6.746	2.951	-2.153	2.874	-1.579	2.568	-9.213
	model9	-191	-59	314.6	-34.93	4628	-0.183	60.58	-8.285	-7.936	2.968	-2.193	3.063	-3.576	0.304	-15.66
	model10	-192	-59	314.6	-34.93	4618	-0.184	61.68	-8.334	-7.936	2.968	-2.193	3.04	-3.598	0.803	-15.25
	model11	-181	-56	314	-41.21	4745	-0.14	60.1	-7.803	-7.49	2.948	-3.567	3.322	-1.749	0.084	-14.25
	model12	-184	-55	313.6	-41.21	4753	-0.141	66.93	-7.948	-7.341	2.939	-3.567	3.339	-1.77	3.188	-11.16
	model13	-172	-56	309.4	-31.52	4632	-0.095	66.3	-7.369	-7.49	2.811	-1.447	3.072	0.185	2.899	-7.339
	model14	-172	-56	309.4	-31.52	4619	-0.095	65.9	-7.369	-7.49	2.811	-1.447	3.043	0.164	2.719	-7.569
	model15	-200	-64	310	-31.7	4699	-0.142	60.09	-8.719	-8.68	2.828	-1.486	3.219	-1.834	0.083	-14.59
	model16	-201	-64	310	-31.7	4698	-0.143	64.57	-8.768	-8.68	2.828	-1.486	3.218	-1.855	2.114	-12.63
	model17	-190	-60	309	-37.98	4831	-0.099	65.07	-8.237	-8.085	2.798	-2.86	3.513	-0.006	2.343	-10.53
	model18	-190	-62	309	-37.98	4822	-0.1	66.07	-8.237	-8.383	2.798	-2.86	3.492	-0.027	2.798	-10.42
	model19	-179	-55	421.5	-44.98	5360	-0.099	59.6	-7.707	-7.341	6.188	-4.392	4.684	0.015	-0.139	-8.692
	model20	-180	-52	422.4	-44.98	5371	-0.099	60.43	-7.755	-6.895	6.217	-4.392	4.707	-0.006	0.235	-7.89
	model21	-203	-64	422.2	-45.16	5454	-0.146	60.45	-8.864	-8.68	6.21	-4.431	4.891	-2.004	0.244	-12.63
	model22	-203	-62	423.2	-45.16	5395	-0.147	60.41	-8.864	-8.383	6.241	-4.431	4.761	-2.025	0.229	-12.47
	model23	-188	-54	421.3	-51.44	5555	-0.103	61.36	-8.141	-7.193	6.185	-5.806	5.114	-0.176	0.656	-9.36
	model24	-195	-58	422.9	-51.44	5588	-0.104	61.15	-8.478	-7.788	6.232	-5.806	5.187	-0.197	0.563	-10.29
	model25	-45	-15	278.9	-27.35	3788	-0.103	61.47	-1.244	-1.391	1.892	-0.534	1.203	-0.155	0.709	0.4805
	model26	-42	-14	279.9	-27.35	3743	-0.103	59.28	-1.1	-1.242	1.921	-0.534	1.102	-0.176	-0.284	-0.313
	model27	-69	-20	279.8	-27.53	3823	-0.15	58.59	-2.402	-2.135	1.92	-0.573	1.281	-2.174	-0.6	-4.683
	model28	-68	-21	280.4	-27.53	3798	-0.151	64.74	-2.353	-2.284	1.937	-0.573	1.225	-2.195	2.193	-2.05
	model29	-61	-17	279.7	-33.81	3929	-0.107	59.3	-2.016	-1.688	1.917	-1.948	1.516	-0.346	-0.278	-2.844
	model30	-63	-17	280.1	-33.81	3962	-0.108	59.79	-2.112	-1.688	1.928	-1.948	1.588	-0.367	-0.056	-2.657
	model31	-49	-16	274.3	-24.12	3819	-0.062	58.84	-1.437	-1.54	1.752	0.173	1.272	1.588	-0.486	1.3221
	model32	-53	-18	275.2	-24.12	3835	-0.062	59.04	-1.63	-1.837	1.781	0.173	1.307	1.567	-0.397	0.9627
	model33	-74	-24	275.2	-24.3	3895	-0.109	59.22	-2.643	-2.73	1.779	0.133	1.439	-0.431	-0.315	-2.767
	model34	-74	-25	275.7	-24.3	3886	-0.11	60.08	-2.643	-2.879	1.796	0.133	1.42	-0.452	0.077	-2.547
	model35	-70	-20	274.5	-30.58	4064	-0.066	58.85	-2.45	-2.135	1.759	-1.241	1.814	1.396	-0.482	-1.338
	model36	-66	-20	275.4	-30.58	4023	-0.067	59.91	-2.257	-2.135	1.787	-1.241	1.724	1.375	-9E-04	-0.747
	model43	-182	-57	368.9	-34.75	5025	-0.155	63.99	-7.851	-7.639	4.604	-2.153	3.942	-2.365	1.851	-9.612
	model44	-182	-57	368.2	-34.75	5017	-0.155	66.34	-7.851	-7.639	4.583	-2.153	3.925	-2.386	2.92	-8.603
	model45	-229	-71	369.6	-31.52	4935	-0.191	62.75	-10.12	-9.722	4.625	-1.447	3.742	-3.916	1.287	-15.55
	model46	-223	-69	369	-31.52	4961	-0.192	60.58	-9.829	-9.424	4.606	-1.447	3.8	-3.938	0.303	-15.93
	model47	-200	-59	370.1	-41.21	5261	-0.171	60.85	-8.719	-7.936	4.64	-3.567	4.463	-3.045	0.424	-13.74
	model48	-203	-62	369	-41.21	5238	-0.16	60.31	-8.864	-8.383	4.609	-3.567	4.412	-2.578	0.183	-14.19
	model55	-58	-16	333.2	-27.35	4222	-0.122	58.58	-1.871	-1.54	3.529	-0.534	2.163	-0.962	-0.602	0.1819
	model56	-60	-17	333.4	-27.35	4278	-0.122	61.66	-1.968	-1.688	3.533	-0.534	2.288	-0.984	0.794	1.4419
	model57	-104	-31	333.9	-24.12	4174	-0.158	60.49	-4.09	-3.771	3.549	0.173	2.058	-2.514	0.262	-4.332
	model58	-105	-31	336.7	-24.12	4177	-0.159	61.52	-4.138	-3.771	3.635	0.173	2.065	-2.535	0.73	-3.841
	model59	-77	-21	336.1	-33.81	4456	-0.138	61.86	-2.787	-2.284	3.614	-1.948	2.681	-1.642	0.885	-1.481
	model60	-77	-21	334.6	-33.81	4430	-0.126	59.72	-2.787	-2.284	3.571	-1.948	2.623	-1.154	-0.087	-2.065

Figure 8.16: **Detailed $s_{external}$ scores for models generated by fragment assembly, optimization and energy minimization.** The format as well as the z-score computation model is the same as described in Figure 8.11. Again, the selected models, based on overall scores (last column) and clustering (Figures 8.18-8.19), are highlighted in bold. Note that a couple of poor models also got selected.

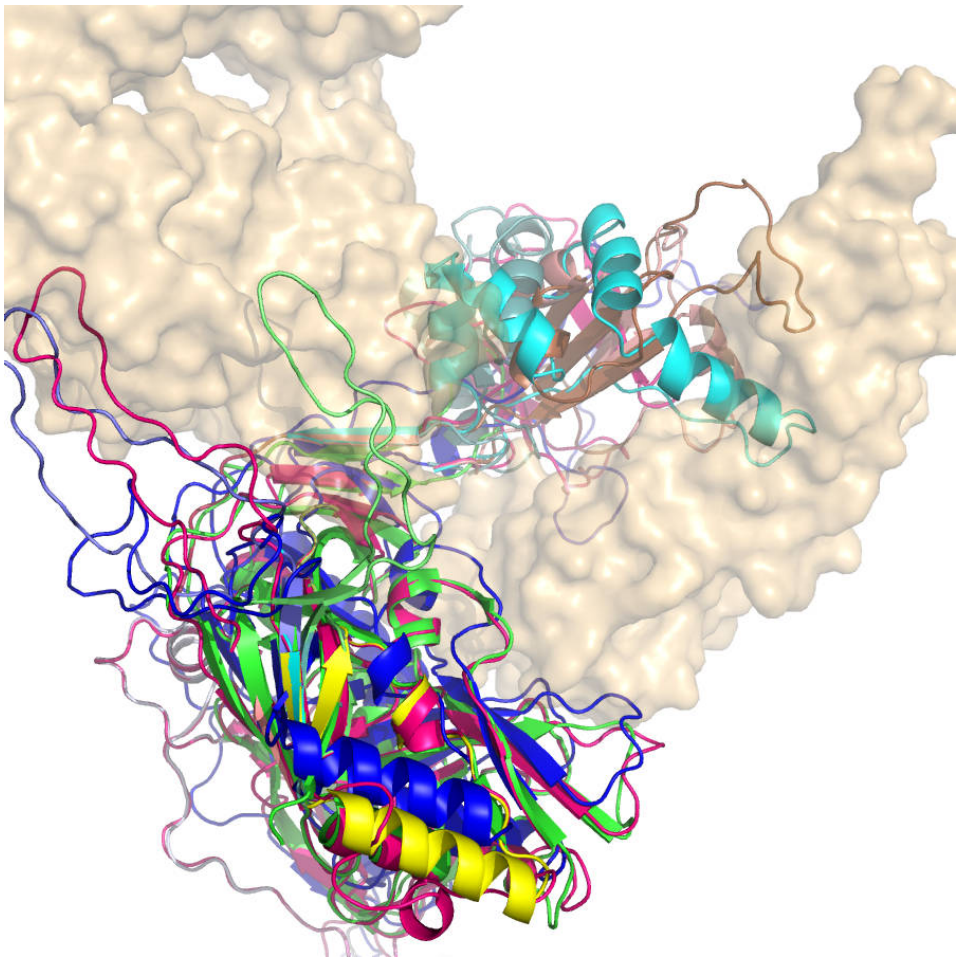


Figure 8.17: **Superposition of spliced models (produced by fragment assembly).**

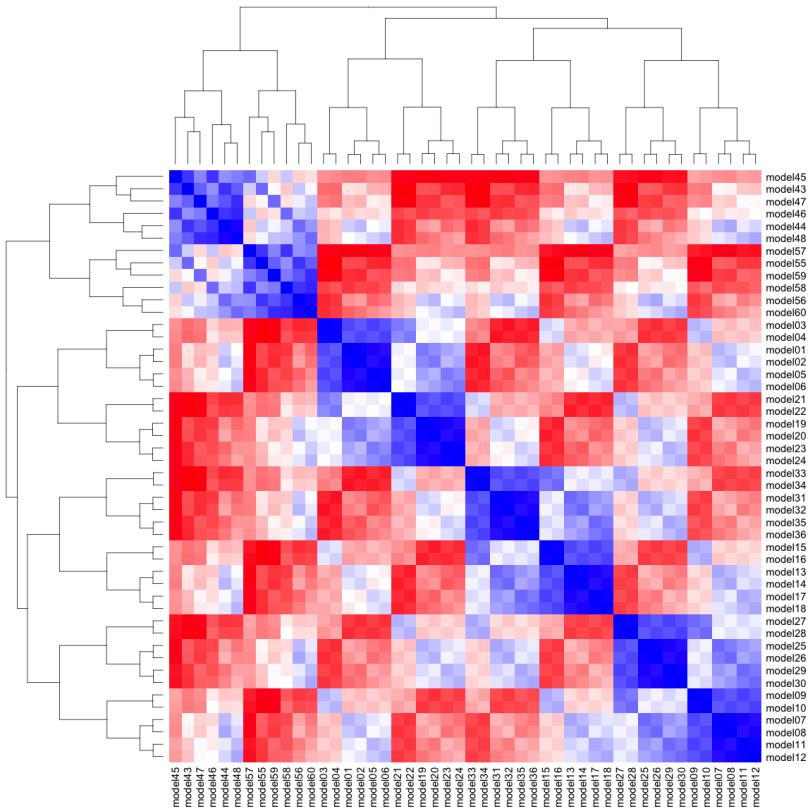


Figure 8.18: **Clustering of the fragment assembly models in terms of their similarity under TM-score [276].**

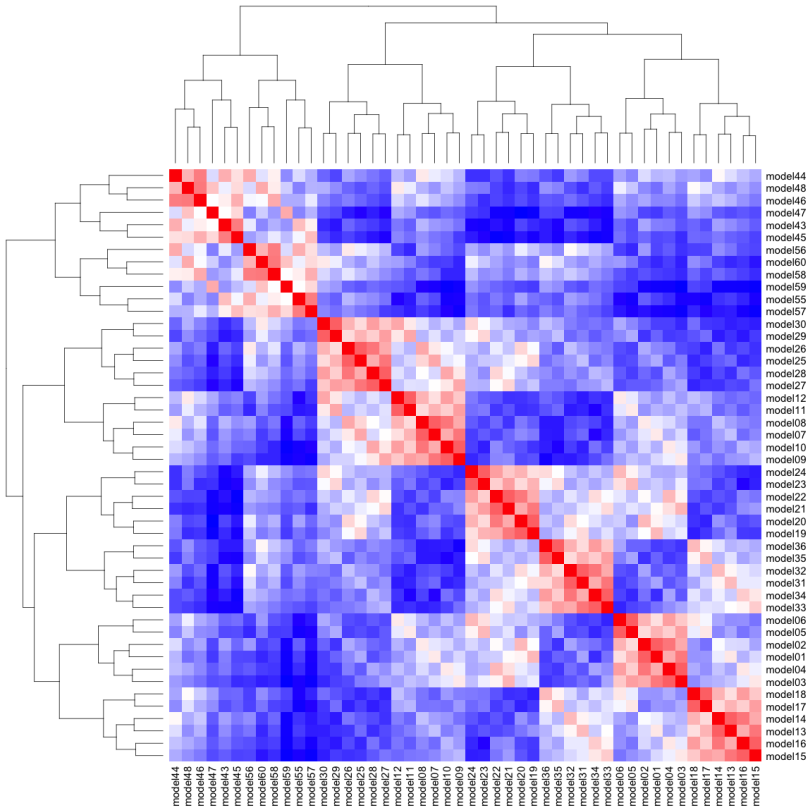


Figure 8.19: **Clustering of the fragment assembly models in terms of their similarity under RMSD (after alignment with PyMOL [228]).**

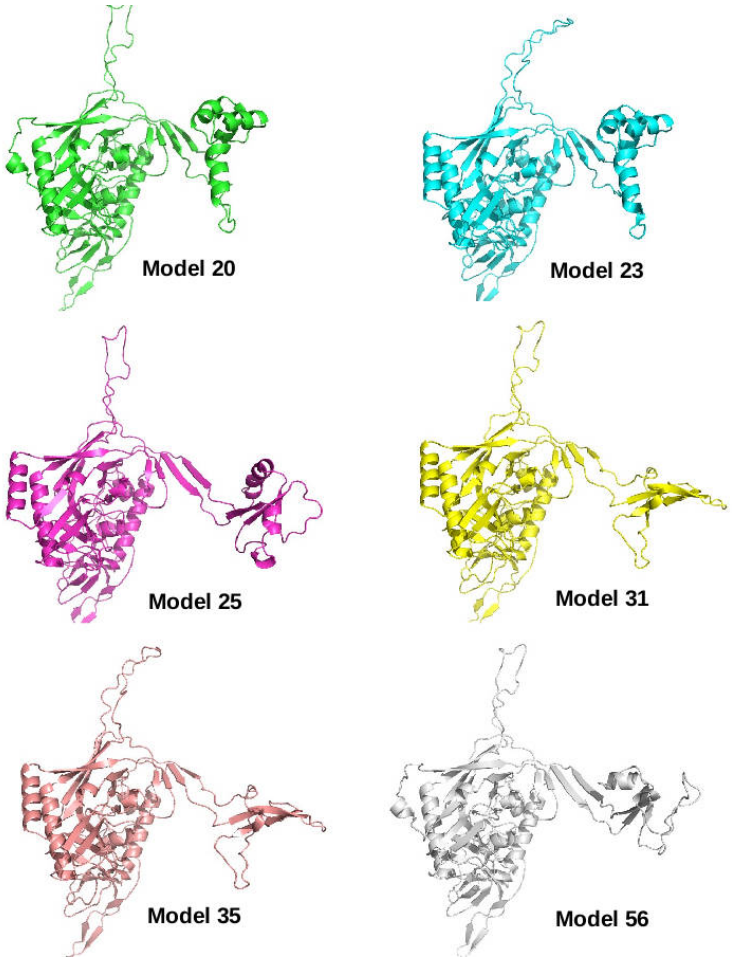


Figure 8.20: **Comparison of the six models selected for co-optimization.**

	Criteria	raw score							z-score							Overall Score
		# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	# of clashes	# severe clashes	rescont pos	rescont neg	interface area	ETR	MIS	
Optimized	model20	-210	-65	462.4	-70.04	5937	-0.097	60.52	-9.202	-8.829	7.421	-9.876	5.96	0.087	0.275	-14.16
	model23	-316	-79	496.6	-70.44	6211	-0.096	61.56	-14.31	-10.91	8.453	-9.964	6.567	0.11	0.747	-19.31
	model25	-48	-9	255.9	-31.73	3875	-0.118	59.7	-1.389	-0.498	1.2	-1.492	1.395	-0.817	-0.095	-1.698
	model31	-25	-7	273.6	-41.78	4085	-0.058	64.39	-0.28	-0.201	1.731	-3.692	1.86	1.737	2.033	3.1886
	model35	-22	-2	286.7	-45.01	4357	-0.056	62.68	-0.135	0.543	2.126	-4.399	2.463	1.805	1.258	3.6612
	model56	-71	-21	311.1	-40.63	4167	-0.116	61.97	-2.498	-2.284	2.863	-3.44	2.041	-0.749	0.934	-3.133
Best Swiss Model		-12	-4	207.8	-17.48	2667	-0.07	60.29	0.347	0.245	-0.251	1.626	-1.28	1.247	0.171	2.106
Best I-TASSER Model		-285	-94	525.1	-64.68	4811	-0.058	65.07	-12.82	-13.14	9.313	-8.703	3.467	1.737	2.344	-17.81

Figure 8.21: **In this table we present a summary of the $s_{external}$ scores for our optimized models.** Notice that both Model31 and Model35 scored very high.

		Procheck G-factor (phi-psi) *	Procheck G-factor (all) *	Molprobity clash z-score *	Verify3D compositeZ-score *	Prosall composite z-score *	ERRAT Score	Overall Assesment
Optimized Models	model20	0.08	0.47	-2.77	-2.09	-1.49	-0.75	-6.55
	model23	0.28	0.77	-2.63	-2.09	-1.36	-0.25	-5.28
	model25	0.08	0.47	-2.54	-2.25	-1.53	0.10	-5.67
	model31	-0.24	0.18	-2.68	-2.41	-1.41	-0.13	-6.69
	model35	-0.24	0.41	-2.65	-1.93	-1.49	0.08	-5.82
	model56	-0.24	0.06	-2.75	-2.57	-1.65	-1.20	-8.35
Average	Swiss	-1.89	-1.58	-16.51	-2.42	-1.98	-3.48	-27.86
	ITASSER	-2.03	-3.03	-28.51	-2.67	-1.87	-10.47	-48.57
	Spliced	-0.28	0.18	-3.05	-2.54	-1.83	-0.53	-8.05
	Final	-0.05	0.39	-2.67	-2.22	-1.49	-0.36	-6.39
	Crystal	-1.79	-2.42	-2.55	-0.70	-0.79	0.00	-8.14
Control	Min	-3.38	-4.61	-5.79	-1.12	-0.99	-1.47	-15.52
	Max	-0.75	-1.06	0.61	-0.16	-0.50	1.50	-1.91
	Avg	-1.79	-2.42	-2.55	-0.70	-0.79	0.00	-8.14
	StdDev	0.79	1.01	1.98	0.33	0.12	1.00	4.43

Figure 8.22: **This table presents the structure validation data ($s_{internal}$) of our optimized models.** All but one model scores very high and hence must have excellent stereochemistry and comparable to x-ray models with resolution between 2.5 to 3.5Å (please see validation section and Figure 6 in the main paper).

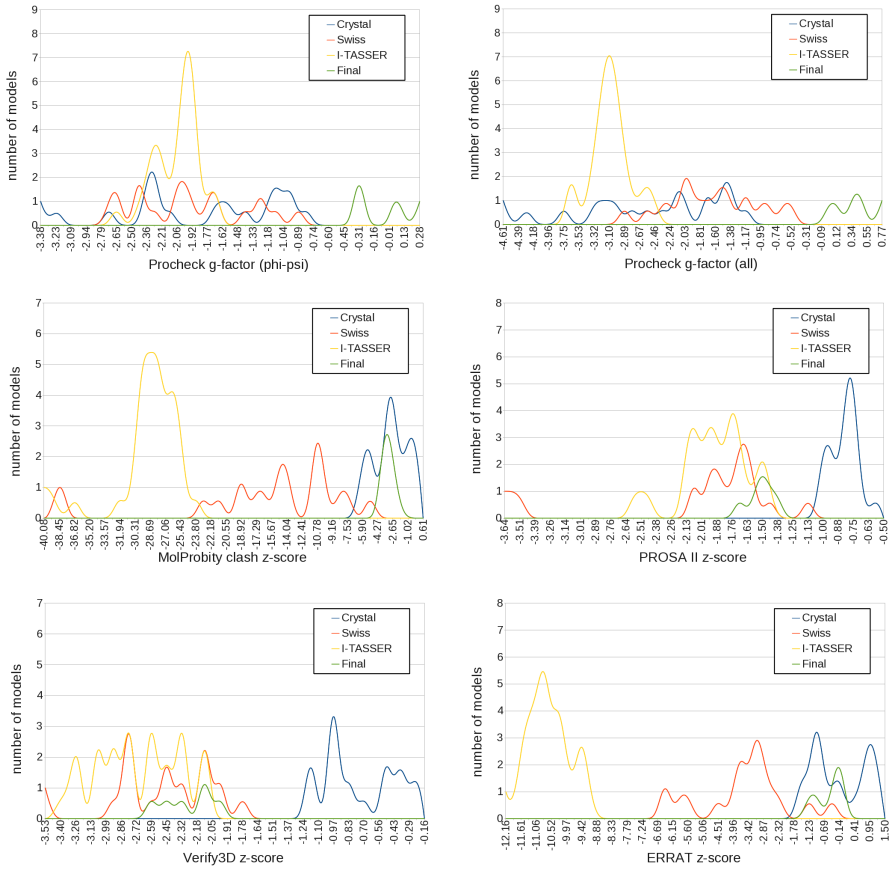


Figure 8.23: **Comparison of the distribution of models for $s_{internal}$ terms.** PROCHECK g-factor ($\phi - \psi$), PROCHECK g-factor (all), MolProbity clash z-score, ProSA II z-score, Verify3D z-score and ERRAT z-score (top to bottom and left to right).

Figure 8.24: **Comparison of the distribution of models for $s_{external}$ terms.** Clash, interface area, positive residue contacts, negative residue contacts, external total ratio and mutual information score z-scores (top to bottom and left to right).

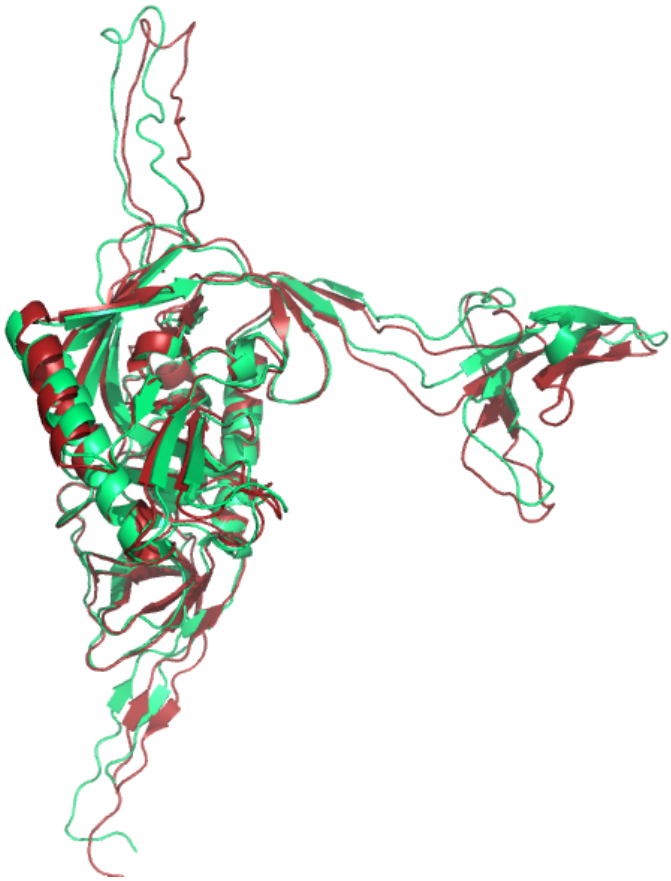


Figure 8.25: **Comparison of Model31 before (red) and after (green) co-optimization and flexible refinement.**

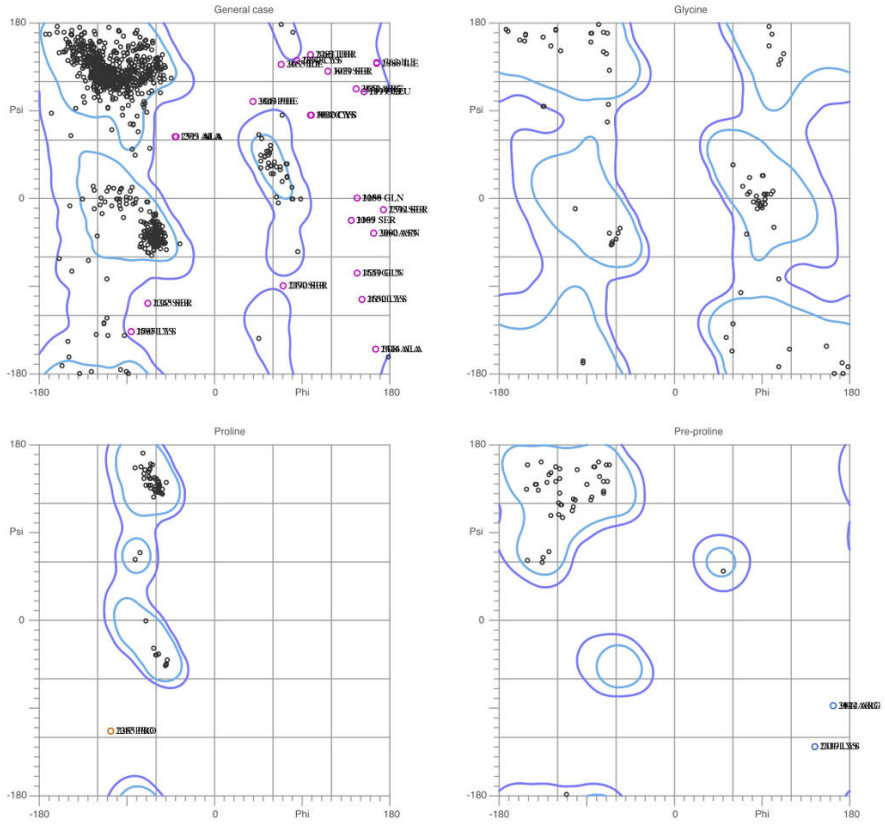


Figure 8.26: **Summary Ramachandran plot of the final model of the trimer.** 89.6% residues are in most favored, 7.6% in additionally allowed, 1.4% in generously allowed and only 1.4% in disallowed regions.

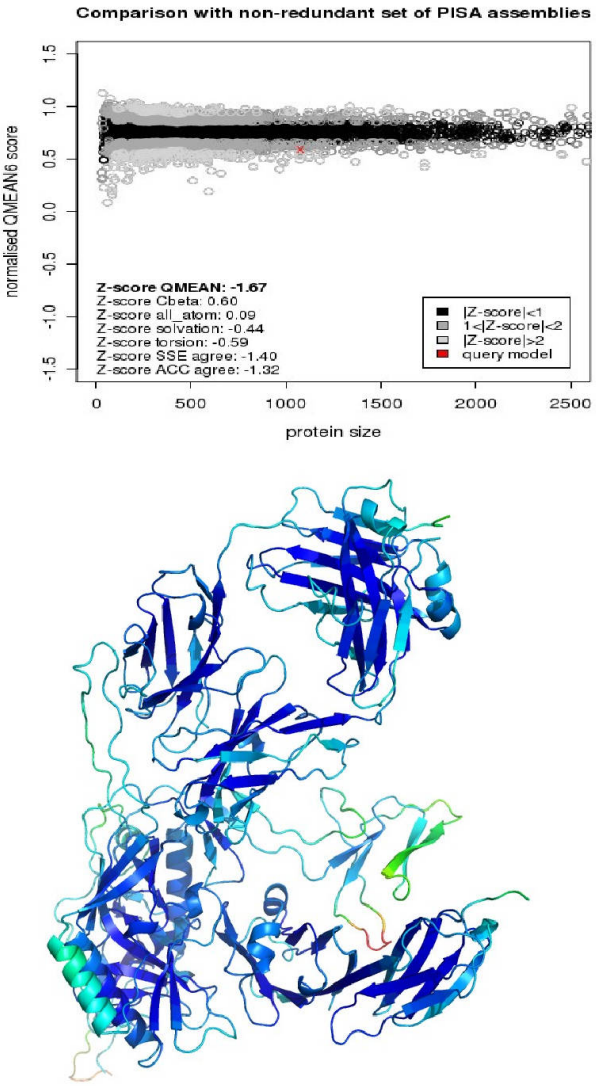


Figure 8.27: **Quality of model in terms of Qmean z-score and ANOLEA score [178]**. (Top) The Qmean z-score [34] of the final model of the trimer of gp120+CD4+17b is -1.666. In the figure, each circle represents an x-ray model and the red cross represents our model. In the plot, our model lies in the range where models have average level of confidence. (Bottom) the surface of the model (1 part of the trimer) is colored based on ANOLEA score [178]. Only small segments have poor quality which is quite expected specially since those regions are far from CD4 and 17b and hence fewer constraints are available to improve their qualities.

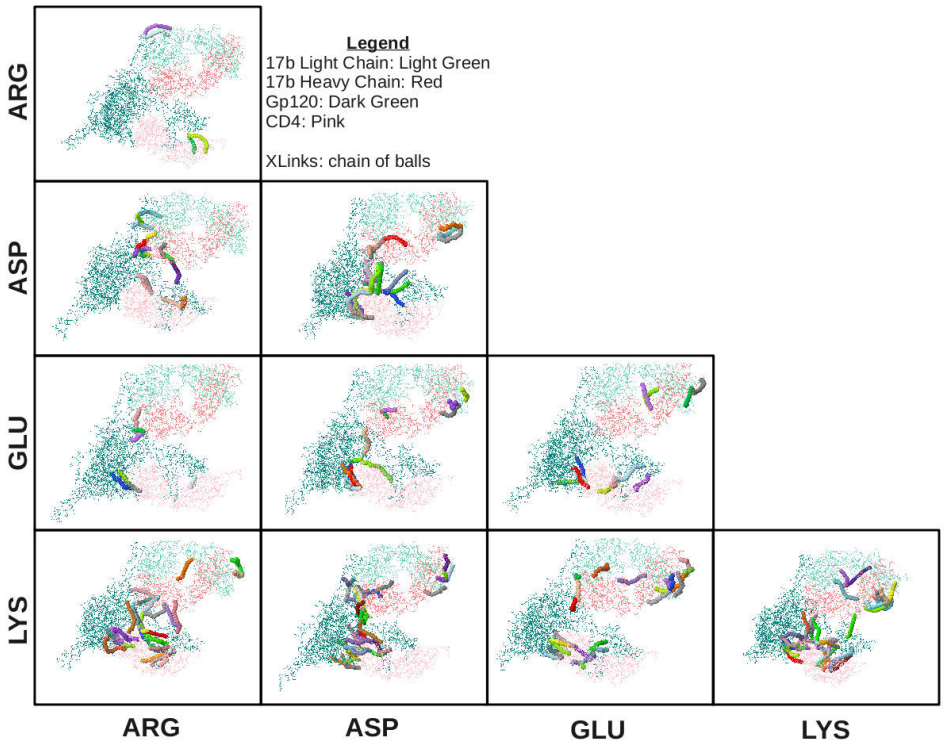


Figure 8.28: **Computationally predicted crosslinks.** The computationally predicted inter-domain (gp120-CD4, gp120-17b and CD4-17b) crosslinks are shown. Each crosslinks shown as a series of contiguous balls, and the actual molecules are shown only using sticks (each stick representing a bond). Colors of the crosslink are only meant to keep a distinction between them, and do not carry any other meaning/interpretation. The figures are arranged like a matrix where each cell represents crosslinks between residues of the types indicated beside the row and column. For example, the second from left figure in the bottom row shows crosslinks between LYS and GLU residue types. We observe a large number of crosslinks between the V1V2 region and CD4, as well as the V3 region and the light chain of 17b; but very few crosslinks between 17b and the V1V2 region.

Chapter 9

Conclusion and Looking Ahead

We have developed efficient data structures and algorithms for maintaining molecules in atomic, smooth surface and volumetric representations, algorithms for updating such models and computing a range of scoring functions for any given configuration. We have also developed scoring functions that mimics real physical attraction-repulsion, and scoring functions based on empirical or knowledge-based observations, and finally applied machine learning techniques to formulate a combination of the scoring functions that showed excellent discriminatoriness in terms of separating good models from bad. Finally, we have characterized the space of possible configurations involving n components. We showed that in generic cases when no other information is known the space is exponential in n. We also proved that if it is known that the structure is symemtric, the space can actually be represented using a constant number of degrees of freedom, irrespective of n. Finally, we also showed that other forms of prior knowledge, like the outline of the complete complex, and/or probably binding sites, can help in restricting the configuration space and also providing more confidence on the predictions. The algorithms, scoring models and also the structures we report in this thesis would greatly accelerate molecular modeling, particularly for viruses and proteins with variable regions. But the algorithms are also useful for designing materials and tempaltes for

building construction, making drug delivery capsules that can self-assemble and disassemble, making naomaterials etc. The data structures we developed are even more generic and will be fundamental in accelerating any computation that involves neighborhood calculation, for example collision detection in computer graphics, mesh separation in graphics and physical simulations, particle systems, molecular dynamics etc. to name a few.

Great!

But we are not done yet. We are only at the first lap in fully automated structure prediction with high accuracy and high confidence. There are many open issues still to be addressed. I have selected two particular issues that I find particularly important and interesting. In this chapter I shall present the problems, their characterization and a first sketch of how to address them.

9.1 Better Models, with High Confidence

Errors cascade in most computations, specially those involving geometry and complicated (linear or non-linear) numerical methods, like the multi-body assembly problem we addressed in this thesis which is a search over high dimensional metric spaces for the optimum of non-convex objective functions. The final predictions of these methods incur uncertainties because the input data itself is sometimes noisy or stochastic, the energy/scoring functions we use are mathematical or statistical models of a natural phenomenon many of whose parameters are either unknown or cannot be modeled efficeintly, the representations and computational approximations introduce more uncertainties in the form of discretization and numerical

errors, and finally, the search space itself is continuous and the finite number of samples we use may not cover it adequately. All of these sources of errors make it improbable that the predicted structure or any other quantity would indeed be the optimal. Previous approaches to deal with this have mainly been to report multiple predicted structures instead of one. But it is not clear how to use such a list, or how large a list is sufficient. We propose to use statistical uncertainty quantification. In particular, for each prediction f of the quantity of interest (QOI), we want to provide a certificate of accuracy in the form of a Chernoff-like tail bound i.e. $Pr[|f - E[f]| > t] < \epsilon$, where $E[f]$ is the expectation of the true value and t and ϵ are two constants.

Whether an adaptive search is performed, or all components are assumed to be rigid, the prediction problem involves discrete sampling of the configurational space, computing the score for each sample and then ranking them. For accurate prediction, we need to bound the error for three issues-

- Given a configuration M which contains uncertainties such that the input data (e.g. positions of the atoms) are not constants, but random variables whose mean and distribution are known- then, if we only considered the means of the variable to compute a scoring term $\mathcal{F}(M_{mean})$ which we want to minimize (without loss of generality), what is the probability that $|\mathcal{F}(M_{mean}) - \mathcal{F}(M_{min})| \leq t1$ where $t1$ is a non-negative constant.

- Given a configuration M (and ignoring the uncertianties in the model) if we evaluate $\mathcal{F}(M)$ using a discrete representation of M as well as a discrete

approximation of $\mathcal{F}$, then can we bound the error $|\mathcal{F}_D(M_D) - \mathcal{F}(M)| \leq t2$, where $\mathcal{F}_D(M_D)$ is the discrete version?

- Given a configurational space $\mathcal{C}$, over which we want to find the configuration $argmax_{M \in \mathcal{C}} \mathcal{F}(M)$, if we only take a discrete subset $\mathcal{D} \subseteq \mathcal{C}$, then what is the guarantee that $dist(argmax_{M \in \mathcal{C}} \mathcal{F}(M) - argmax_{M \in \mathcal{D}} \mathcal{F}(M)) \leq t3$, where $dist$ is a distance metric between configurations, $t3$ is a non-negative constant?

And, then we have to combine these to provide an error bound for the overall protocol.

Another important aspect, in assembly prediction, specially for molecules is to account for the internal flexibility of the components. In this case, the search space does not only contain degrees of freedom for the motion of the components with respect to each other, but also parts of a component can move with respect to the remaining part of the component. For evan small molecules the number of atoms can be much higher than the number of molecules in a complex. We have already seen that for the generic assembly case, the sample space can be $SE(3)^n$, where n is the number of components. If every component/molecule had N domains that move w.r.t. each other, then the total search space can be $SE(3)^{nN}$, which would be beyond the power of current supercomupters, or even the supercomputers of the near future to search exhaustively. So, one needs an adaptive hierarchical representation of the possible motions so that the space can be searched adaptively

while ensuring that the true optima is not missed due to the corase-graining. The problems to consider are-

- Characterizing the possible flexible degrees of freedom, parameterize them, and develope methods to sample the space.
- Develop algorithms and heuristics to clauster/coarse-grain the DOFs (and the underlying structural model), to come up with a hierarchical representation of the flexibility. Develop techniques to parameterize and sample at multiple resolutions.
- Develop an adaptive sampling and search algorithm that can progressively sample at different resolution to adaptively refine predictions

Interestingly, the problem of providing error bounds often boils down to the problem of sampling, or dicretization of the model, the function, or the confiurational space. In the next sections, we address the flexibility and adaptive search from the perspective of sampling, in particular ensuring low discrepancy and low dispersion sampling, since it can be proved that such gurantees on sampling is intimately tied to gurantees on the prediction/evalution.

9.2 Characterizing and Sampling the Configuration Space of Proteins

Let us assume that we want to predict the structure of a protein consisting of n atoms, belonging to m residues. Hence, the optimization function $\mathcal{F}$ depends

on the position (and possibly charge, radii etc.) of n atoms. Assuming, all other properties to be constants, the total number of degrees of freedom would be $3(n-1)$, since the position of one of the atoms can be arbitrarily chosen.

However, proteins have a distinct topology and introduces constraints which nullify most of these degrees of freedom. Please see Section 1.1 for a description of protein structure and composition.First, we describe the so-called internal coordinate system of a protein. The internal coordinate system is represented using the bond lengths, bond angles and dihedral angles of the protein. Bond length is simply the Euclidean distance between the centers of two atoms connected by a covalent bond. If more than two bonds are incident at an atom, then the angle between any two of the bonds, is defined as the bond angle. And finally, given three consecutive bonds between 4 atoms p-q-r-s, the dihedral angle around the bond q-r is defined as the angle between the plane containing p-q-r and the plane containing q-r-s. Though the total number of bond length parameters n_b, bond angle parameters n_a and dihedral angle parameters n_d may be larger than $3(n-1)$, it can be proved that, once a single atom position is fixed, the position of the rest can be expressed using exactly $3(n-1)$ parameters in this coordinate system (i.e. other parameters are dependent). Now, we address the constraints on these parameters, as observed in protein structures in nature.

First, the bond lengths are assumed to be almost constant (between two specific type of atoms), and the bond angle (between three specific types of atoms) have been observed to be almost always the same and hence are considered to be constants. Secondly, the dihedral angles are also fixed in many cases, for example

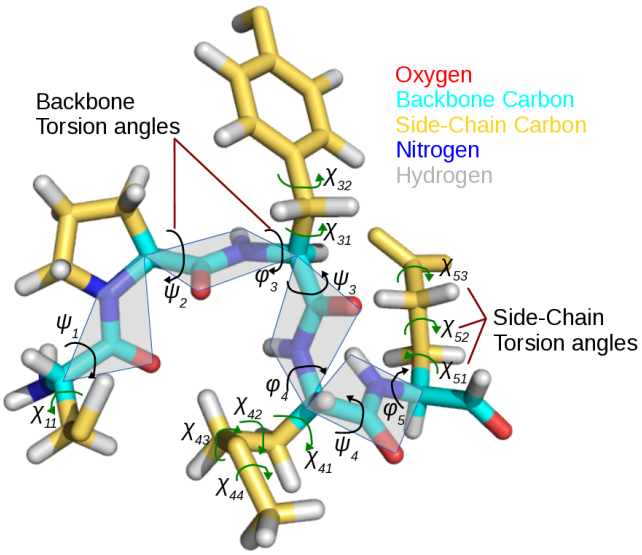


Figure 9.1: The torsion angle degrees of freedom for a protein. The figure shows a small section of a protein and marks all the torsion angles which are in general found to be free degrees of freedom. See the text for more detail.

all carbons (and the Hydrogens connected to them) on aromatic rings lie on the same plane, thereby fixing the dihedral angles around each bond of the ring to a constant (180 degrees). It was also found that the backbone atoms of a residue (as well as the H attached to N, and the O attached to C) all lie on a plane. After these restrictions are applied, we are left with only a small set of dihedral angles (also called torsion angles) which might change. Specifically, there are three bonds incident on every C_α atom that have a free torsion angle, namely ϕ (around N-C_α), ψ (around C_α-C) and χ_1 (around C_α-first carbon on the side-chain). Further, there can be upto 3 more torsion angles in a side-chain (denoted $\chi_2, \chi_3 \ldots$). See Figure 9.1 for examples.

Hence, overall the number of free degrees of freedom comes down to $6m$ at

the most, instead of $3(n-1)$. Hence the space of possible configurations if $O(R^{6m})$. This is still an extremely large and intractable space. In the next section we describe discretization techniques which brings the space of possible configurations down to countable, but still exponential, sets.

Restrictions on the Torsion Angle DOFs First, we quickly review the concept of rotamers for side chain positions. It was observed that the all values between $[-180, 180]$ degrees were not equally likely for the χ angles, but rather, for each residue types very few (usually 3) modes are observed. Based on the observation, the set of possible configuration for each side-chain were discretized into a small set of configurations, each called a rotamer for that type of residue.

Similar, but less specific distribution for ϕ-ψ angles were also found. Essentially, it was found that in different secondary structural motifs, the ratio of the ϕ and ψ angles for the same C_α, clusters close together. If the ϕ and ψ are plotted against each other in a scatter plot (referred to as Ramachandran plot), then separate regions/clusters are seen, each cluster corresponds to a type of secondary structural element. However, the range of values in a cluster is too broad to apply a 'rotamer-like' concept for backbone torsion angles.

In this article, we shall treat both backbone and side-chain torsion angles in the same manner, and though we do not use the concept of rotamers and Ramachandran plots in our discrete sampling protocol, they can easily be incorporated to accept/reject specific samples.

9.2.1 Motion Graph

A protein can naturally be expressed as a graph $M(A, B)$, such that each node $a_i \in A$ represents an atom, and each edge $(a_i, a_j) \in B$ represents a covalent bond between the two atoms a_i and a_j. We can immediately infer that M is necessarily connected and there is no cycles in M that includes an edge with a torsional DOF (after applying the planarity restrictions described above). Hence, for the purpose of motion space parameterization, we can consider M to be a tree.

Now, let us assume that a certain atom $r \in A$ is selected as the root. Now, the configurational space of any other atom $a_i \in A$, with respect to the root, can be computed as the product space of all the torsion angles on the $r - a_i$ path. The torsions are also applied strictly in the correct order. Hence, any configuration of the entire protein parameterized by torsion angles in internal coordinate system, can be mapped to a configuration represented in Cartesian coordinate system by additionally fixing the location the root.

9.2.2 Sampling with Bounded Dispersion

Let $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$ be the set of torsion angles in the motion graph. Also, let each $\theta_i \in \mathbf{S^1}$, where S^1 is the space of 1D rotations.

Let $\mathcal{C}^I$ represents the space of possible configurations $(\mathbf{S^1})^k$ in internal coordinate representation. Let us assume that we choose a finite set $\mathcal{S}^I \subset \mathcal{C}^I$ of samples. Now we would like to prove some properties of $\mathcal{C}^I$. But we need to introduce some definitions first.

Definition 9.2.1 (Metric in ordered product space)**.** *Let $\delta^I(x, y)$ represent the distance between to configurations $x, y \in \mathcal{C}^I$ such that it satisfies the following properties-*

- non-negativity*: $\delta^I(x, y) \geq 0$ for any $x, y \in \mathcal{C}^I$,*
- identity*: $\delta^I(x, x) \geq 0$ for any $x \in \mathcal{C}^I$,*
- symmetry*: $\delta^I(x, y) = \delta^I(y, x)$ for any $x, y \in \mathcal{C}^I$, and*
- triangle inequality*: $\delta^I(x, y) + \delta^I(y, z) \geq \delta^I(x, z)$ for any $x, y, z \in \mathcal{C}^I$.*

We define dispersion of a set of samples.

Definition 9.2.2 (Dispersion)**.** *The dispersion of a set of N samples $\mathcal{S}^I$ in a space $\mathcal{C}^I$ is defined as $d^I(\mathcal{S}^I, \mathcal{C}^I) = \sup_{x \in \mathcal{C}^I} \min_{x_s \in \mathcal{S}^I} \delta^I(x, x_s)$.*

Now we want to prove the following-

Theorem 9.2.1. *For sufficiently large N, it is guaranteed that $d^I_N(\mathcal{S}, \mathcal{C}) \leq \epsilon_d$.*

Dispersion in Cartesian Space We showed before that any sample x in the ordered product of torsion angles space ($\mathcal{C}^I$) can be mapped by a function g^I to a configuration $X = g^I(x)$ in Cartesian space (or represented using Cartesian coordinates). In the rest of the article, we shall use upper case (eg. X) to represent configurations in Cartesian space and lower case (eg. x) to represent configurations parameterized in other spaces.

Let the space of mapped configurations be defined as $\mathcal{C}^C = \{g^I(x) | x \in \mathcal{C}^I\}$. Similarly define the set $\mathcal{S}^C$. We proceed to define a metric $\delta^C(x', y')$ and also dispersion $d_N^C(\mathcal{S}^C, \mathcal{C}^C)$.

Let, for a given set of samples, $d_N^I = d_N^I(\mathcal{S}^I, \mathcal{C}^I)$ and $d_N^C = d_N^C(\mathcal{S}^C, \mathcal{C}^C)$. We have the following result-

Theorem 9.2.2 (adapted from Theorem 6.4 from Niederreiter [186]). *If there exists a value $L \geq 0$ such that $\forall_{x,y \in \mathcal{C}^I} \delta^C(g^I(x), g^I(y)) \leq L\delta^I(x, y)$, then $d_N^C \leq L d_N^I$.*

9.2.3 Bounded Error in Prediction Accuracy

Recall that the protein modeling problem aims to find the configuration $x^* = argmax_{x \in \mathcal{C}^I} \mathcal{F}(X)$, where $X = g^I(x)$. However, with our sampling we simply report the configuration $x^{\mathcal{S}^I} = argmax_{x \in \mathcal{S}^I} \mathcal{F}(X)$. Now we want to bound the error in our prediction.

Definition 9.2.3 (Modulus of continuity). *We define $\omega_{\mathcal{C}^I}(\mathcal{F}, t) = \sup_{x,y \in \mathcal{C}^I \& \delta^I(x,y) \leq t} |\mathcal{F}(X) - \mathcal{F}(Y)|$ as the maximum change of the function value of $\mathcal{F}$ with a change (under δ^I metric) of at most t in the parameter space.*

The following theorem follows trivially from the above definition and Theorem 9.2.1.

Theorem 9.2.3 (adapted from Theorem 6.3 from Niederreiter [186]). *Let, $m^*(\mathcal{F}) = max_{x \in \mathcal{C}^I} \mathcal{F}(X)$ and $m^{\mathcal{S}^I}(\mathcal{F}) = max_{x \in \mathcal{S}^I} \mathcal{F}(X)$. If $(\mathcal{C}^I, \delta^I)$ defines a bounded metric space, then $m^*(\mathcal{F}) - m^{\mathcal{S}^I}(\mathcal{F}) \leq \omega_{\mathcal{C}^I}(\mathcal{F}, d_N^I)$.*

Finally, under some assumptions, the following theorem follows from Theorem 9.2.3.

Theorem 9.2.4. *Let* $\{l_0, l_1, \ldots\}$ *be a set of values corresponding to* $\mathcal{F}(X_0^*), \mathcal{F}(X_1^*), \ldots$ *where* X_i^* *are the local maxima of* $\mathcal{F}$ *such that* $X_0^* = g^I(x^*)$ *and* $\mathcal{F}(X_i^*) \leq \mathcal{F}(X_{i+1}^*)$. *Then, we can guarantee* $\delta^I(x^*, x^{\mathbb{S}^I}) \leq d_N^I$, *iff* $\delta^I(x_0^*, x_i^*) \geq d_N^I \Rightarrow l_0 - l_i \geq \omega_{\mathcal{C}^I}(\mathcal{F}, d_N^I)$

9.3 Hierarchical Modeling of Proteins and Their Configuration Spaces

We define a domain A simply as a set of atoms. A domain A can be partitioned into disjoint domains $A_1, \ldots, A_d$ such that their union is A, or be combined with disjoint domains to form larger domains. We say that a $A_i \subset A_j$ if the set of atoms in A_i is a subset of the set of atoms in A_j.

Here we describe a bottom-up (clustering) idea for generating a hierarchy of domains. We initially consider each atom of the molecule a separate domain. Then, the backbone atoms of a residue are grouped into a single domain, and the side-chain atoms are also grouped into one or more domains depending on the structure and number of free torsion angles of the side-chain (see Figure 9.2. Motion between these domains are defined as hinge motions (3D rotation around a point, in other words SO(3)). Notice that clustering of domains is similar to graph clustering where all nodes of a cluster is collapsed into one super-node, all edges between the nodes of the same super-node are removed, and if there exists one of more edges between nodes belonging to two different clusters, then a super-edge is added between the

corresponding super-nodes.

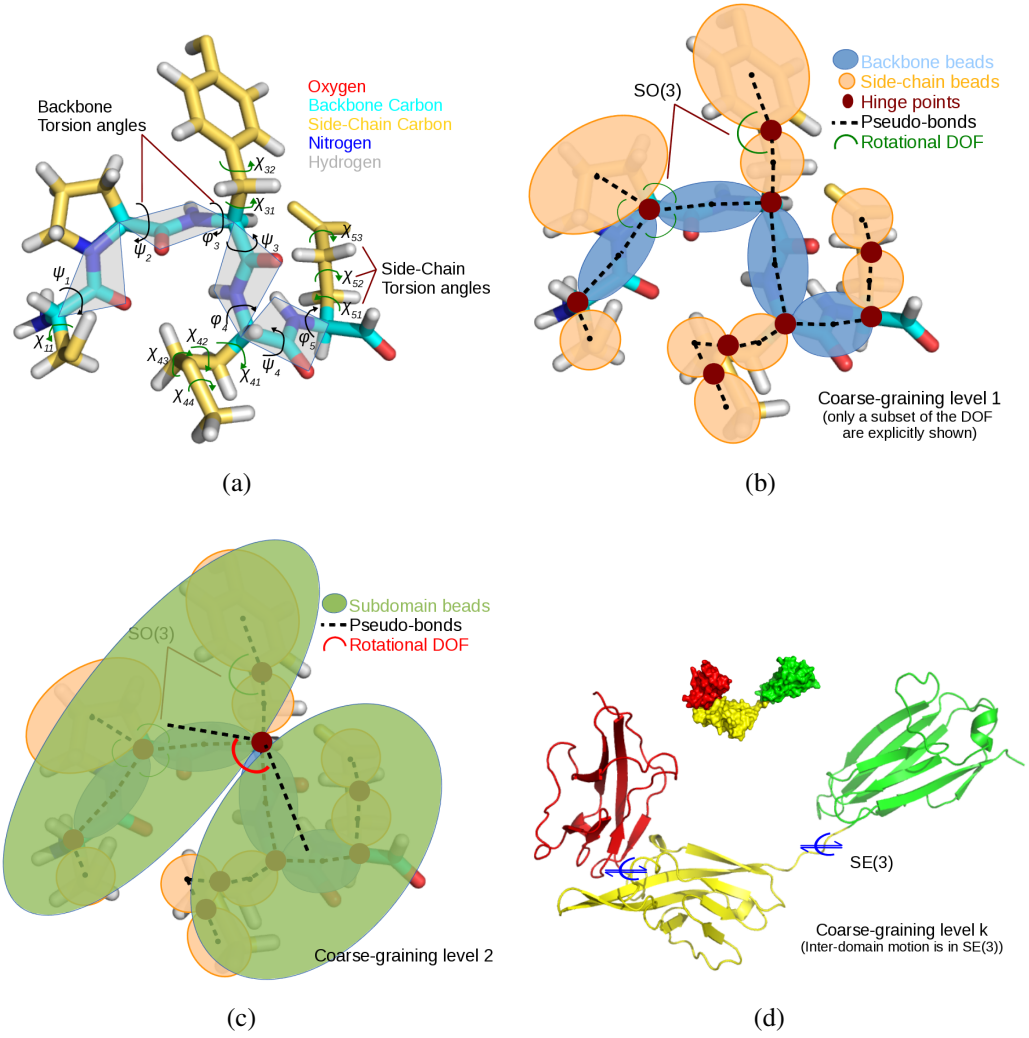


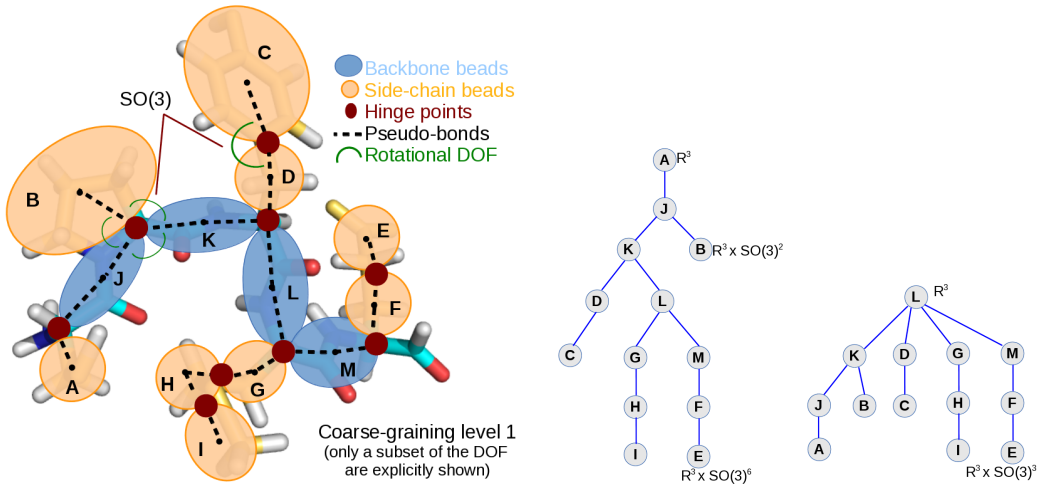
Figure 9.2: Example hierarchical modeling of a protein structure and motion. **(a)** At the lowest level of the hierarchy, each atom represents a domain and the motion is parameterized by the torsion angles. **(b)** We cluster the backbone atoms of a residue into a single domain (bead) and the side chains are grouped into one or more domains. A motion (SO(3)) is defined between two domains if there exists an atom in one that has a covalent bond with an atom in the other domain. **(c)** After another level of clustering, and **(d)** the final level of clustering.

In the next step of clustering, one or more residues become clustered into single domains, and gradually the domains become larger and larger until they correspond to secondary structural motifs, and then larger and larger again until a single domain represents the whole molecule.

9.3.1 Motion Space of Coarse Grained Models

At every stage of the hierarchy, we define a motion graph using the domain at that level as the nodes and the edges representing a relative motion between them. The presence/absence of an edge is determined based on both the topology of a finer level of detail model, and the presence/absence of physical contact between the domains. At very coarse resolutions, where each domain contains multiple secondary structural components, we annotate the motion space between domains using SE(3).

In most cases, the motion graph is a tree. Hence, after fixing a root, the space of possible configurations becomes an ordered product space of $SO(3)$ (see Figure 9.3). Note that the choice of the root determines the order of the product and also the overall complexity of mapping the product space to the Cartesian coordinate space. Finally, in certain cases, the motion graph may not be a tree and in these cases, the space of possible spanning trees of the graph must also be factored into the configurational space definition.



(a) (b) (c)

Figure 9.3: **(a)** A sample coarse-graining step where collections of atoms are grouped into and replaced by larger beads. Each rotational DOF corresponds to the space SO(3). The overall configurational space of the entire protein (in this case, represented as the collection of beads), is a product space. The actual representation of the product space depends on the the order in which the SO(3) spaces are sampled. For example, in **(b)**, the product space is defined by assuming that the bead named A is located in R^3 first, and then the locations of others are resolved with respect to A. In this case, the overall space of motion for I and E are $R^3 x (SO(3))^6$. On the other hand, the choice of using L as the first bead to resolve, leads to a worst case motion space of $R^3 x (SO(3))^3$, as shown in **(c)**.

9.3.2 Sampling with Bounded Dispersion for Coarse Grained Motions

The sampling of the coarse-grained motions is a general case of the atomic resolution motions we introduced before, the only change being the fact that we have a product of $SO(3)$ instead of a product of S^1.

Again, we define a distance metric δ^S in this new product space $\mathcal{C}^S = SO(3) \times SO(3) \times \ldots$, and proceed to define the dispersion in this space as fol-

lows.

Definition 9.3.1 (Dispersion in product of SO(3) space)**.** *The dispersion of a set of N samples $\mathcal{S}^S$ in a space $\mathcal{C}^S$ is defined as $d_N^S(\mathcal{S}^S, \mathcal{C}^S) = \sup_{x \in \mathcal{C}^S} \min_{x_s \in \mathcal{S}^S} \delta^S(x, x_s)$.*

Now we want to prove the following-

Theorem 9.3.1. *For sufficiently large N, it is guaranteed that $d_N^S(\mathcal{S}^S, \mathcal{C}^S) \leq \epsilon_d$.*

Dispersion in Cartesian Space Using the similar notations and arguments introduced before (i.e. g^S maps a configuration $x \in \mathcal{C}^S$ to its Cartesian representation X), we can prove the following theorem.

Theorem 9.3.2. *If there exists a value $L \geq 0$ such that $\forall_{x,y \in \mathcal{C}^S} \delta^C(g^S(x), g^S(y)) \leq L\delta^S(x, y)$, then $d_N^C \leq L d_N^S$.*

9.3.3 Bounded Error in Prediction Accuracy under Coarse Grained Motion

Prediction error is increased when using coarse-grained motions because it does not provide the same level of control and accuracy as fine-grained motions. Note that we assume that only the motion is coarse-grained and the function is evaluated after generating atomic resolution model (in Cartesian space) using the mapping functions. Under this assumption, most of the mechanism introduced before would only require minor adjustments only.

Definition 9.3.2 (Modulus of continuity)**.** *We define $\omega_{\mathcal{C}^S}(\mathcal{F}^S, t) = \sup_{x,y \in \mathcal{C}^S \& \delta^S(x,y) \leq t} |\mathcal{F}(X) - \mathcal{F}(Y)|$ as the maximum change of the function value of $\mathcal{F}$ with a change (under δ^S metric) of at most t in the parameter space.*

The following theorem follows trivially from the above definition and Theorem 9.3.1.

Theorem 9.3.3 (adapted from Theorem 6.3 from Niederreiter [186]). *Let,* $m^*(\mathcal{F}) = max_{x \in \mathcal{C}^S} \mathcal{F}(X)$ *and* $m^{\mathcal{S}^S}(\mathcal{F}) = max_{x \in \mathcal{S}^S} \mathcal{F}(x)$. *If* $(\mathcal{C}^S, \delta^S)$ *defines a bounded metric space, then* $m^*(\mathcal{F}) - m^{\mathcal{S}^S}(\mathcal{F}^S) \leq \omega_{\mathcal{C}^S}(\mathcal{F}^S, d_N^S)$.

Finally, under some assumptions, the following theorem follows from Theorem 9.3.3.

Theorem 9.3.4. *Let* $\{l_0, l_1, \ldots\}$ *be a set of values corresponding to* $\mathcal{F}(X_0^*), \mathcal{F}(X_1^*), \ldots$ *where* X_i^* *are the local maxima of* $\mathcal{F}$ *such that* $X_0^* = g^S(x^*)$ *and* $\mathcal{F}(X_i^*) \leq \mathcal{F}(X_{i+1}^*)$. *Then, we can guarantee* $\delta^S(x^*, x^{\mathcal{S}^S}) \leq d_N^S$, *iff* $\delta^S(x_0^*, x_i^*) \geq d_N^S \Rightarrow l_0 - l_i \geq \omega_{\mathcal{C}^S}(\mathcal{F}, d_N^S)$

9.4 Hierarchical Sampling and Search

Let us assume that we have k levels of coarseness in our hierarchical representation of the structure and motion of the protien. Also assume that level 0 represents the finest level representation where each atom represents a domain, and let level $k-1$ represents the coarsest level representation where the entire protein is a single domain. We shall represent the motion/parameter space for the level i, as $\mathcal{C}^i$.

Now, let us describe a hierarchical search algorithm which leverages from the hierarchical motion/structure representations as well as the bounds on the error at any level of the hierarchy to iteratively refine the configuration of a protein and

get closer and closer to the true maxima $m^*(\mathcal{F})$ of the function $\mathcal{F}$. Recall that $\mathcal{F}$ is applied on X which is generated after mapping a configuration $x \in \mathcal{C}^i$ to Cartesian space using g^i.

Let, $\mathcal{S}^i$ be a set of N samples from the space $\mathcal{C}^i$ with optimal dispersion d_N^i for this space. Also, let $m^i(\mathcal{F})$ be the best scoring sample in $\mathcal{S}^i$.

Definition 9.4.1 (Multi-level distance metric)**.** *We define a distance metric* $\delta^{ij}(x, y)$ *where* $x \in \mathcal{C}^i$ *and* $y \in \mathcal{C}^j$ *as* $\delta^C(g^i(x), g^j(y))$.

Definition 9.4.2 (Finer resolution neighborhood)**.** *We define the finer resolution* ϵ *neighborhood of a configuration* $x \in \mathcal{C}^i$*, as follows-* $\mathcal{N}_\epsilon(x) = \{y | y \in \mathcal{C}^{i-1} \& \delta^{ij}(x, y) \leq \epsilon\}$.

Adaptive Search Algorithm

1. Set Γ as the space $\mathcal{C}^{k-1}$
2. For $i = k - 1$ to 0 do
   (a) Generate Σ as a set of N samples in space Γ with optimal dispersion d_N^Γ.
   (b) Compute x^m as $argmax_{x \in \Sigma} \mathcal{F}(g^i(x))$.
   (c) Set $\Gamma = \mathcal{N}_{d_N^i}(x^m)$.
3. Report $g^0(x^m)$

Theorem 9.4.1 (Correctness of algorithm)**.** *Let* $x^* = argmax_{x \in \mathcal{C}^I} \mathcal{F}(X)$ *be the optimal configuration. Then, for any* i *between* $k-1$ *to* 0, $x^* \in \Gamma$. *In other words, the true solution is never excluded from the current space.*

The above theorem is a direct consequence of Theorem 9.3.4.

Theorem 9.4.2 (Convergence)**.** *Let* $x^* = argmax_{x \in \mathcal{C}^0} \mathcal{F}(X)$ *be the optimal configuration (not necessarily the configuration reported by the algorithm). Let* x^{m^j} *and* x^{m^l} *be two best configurations computed by the algorithm when* i *was set to* j *and* l *respectively. Then, we can guarantee than* $\delta^{j0}(x^{m^j}, x^*) \leq \delta^{l0}(x^{m^l}, x^*)$, *iff* $0 \leq j \leq l \leq k-1$. *In other words, the predicted configuration always gets closer to the true solution.*

A proof of this theorem uses Theorem 9.3.4. It also use the fact that Γ keeps shrinking, and hence with the same N, one should be getting smaller and smaller d_N^Γ, and the cycle continues. I believe this part of the proof requires some reference to dispersion, since dispersion includes a measure on the space and hence is affected by the shrinking of the space. Then we can apply the relationship between dispersion and discrepancy to complete the proof.

9.5 Bounding the Errors in Scoring Function Evaluation

The scoring function $\mathcal{F}(X)$, for most practical applications, is expressed as an integral over a domain representing the surface or the volume of the molecule in the given configuration X. We shall consider two specific examples, one for the problem of fitting a protein configuration to a 3D volumetric map (reconstructed

from cryo-electron microscopy imaging, for instance), and the other for the problem of docking two proteins. We shall briefly describe both of these problems and present typical scoring functions for them.

9.5.1 Example Scoring Functions used in Fitting

Given a volume V, and a configuration X for a protein, the scoring function $\mathcal{F}(X)$ is designed such that it is maximized when the spatial overlap between X and V is maximized. In practice, more complicated functions may be used, but all of them has similar formulation and we can focus only on the spatial correlation/overlap without loss of generality. Let V^X be the volume occupied by X. Now we define two functions $F1(p)$ and $F2(p)$, where p is a point in 3D, as follows-

$F1(p) = 1$ if $p \in V^X$, otherwise 0.

$F2(p) = 1$ if $p \in V$, otherwise 0.

Now, given a fixed V, the scoring function for different configurations of the protein is defined as

$$\mathcal{F}_V(X) = \int_{p \in \mathcal{D}} \int_{q \in \mathcal{D}} F1(p) F2(q) dp dq$$

where $\mathcal{D}$ is the smallest convex subset of R^3 such that $V \subseteq \mathcal{D}$ and $V^X \subseteq \mathcal{D}$.

9.5.2 Example Scoring Functions used in Docking

The docking problem is more complicated than fitting, but has similar scoring paradigm. Given two proteins, the aim of docking is to find the configuration of the two proteins (internally and with respect to each other) so that they come

in contact (docks) in such a way that their interfaces (the region of one which is in contact with the other) are complementary. The complementarity can be computed/expressed using many ways including shape complementarity, electrostatic complementarity, atom-atom or residue-residue contact potentials etc. Here we shall focus on shape complementarity alone, but the formalism we introduce is applicable to all complementary scoring. Let L and R be the two proteins which will henceforth be referred to as the *ligand* and the *receptor*.

Let ρ^X be a smooth surface approximation of the boundary of the region occupied by X. For example, ρ^X is often approximated as a level set of the sum of Gaussians defined at the centers of all the atoms of X. Also let, V^{ρ^X} be the volume bounded by ρ^X.

Now let, the distance between a point p, and a surface ρ^X be defined as $d(p, \rho^X) = \min_{q \in \rho^X} \|p - q\|$, where $\|p - q\|$ is the Euclidean distance between the points p and q.

Affinity function for the ligand

Define V_S^L, or the skin region of L as the set of points p, such that $d(p, \rho^L) \leq r_s$ and $p \in V^{\rho^X}$.

Also define V_C^L, or the core region of L as the set of points p, such that $d(p, \rho^L) > r_s$ and $p \in V^{\rho^X}$.

Now, we define $F^L(p)$ as follows-

$F^L(p) = 1, i$ and 0 respectively, if $p \in V_S^L$, $p \in V_C^L$, and otherwise

where i is the imaginary number $sqrt(-1)$.

Affinity function for the receptor

Define V_S^R, or the skin region of R as the set of points p, such that $d(p, \rho^L) \leq r_s$ and $p \notin V^{\rho^X}$.

Also define V_C^R, or the core region of R as the set of points p, such that $p \in V^{\rho^X}$.

Note that the skin of R is actually outside the boundary of R.

Now, we define $F^R(p)$ as follows-

$F^R(p) = 1, i$ and 0 respectively, if $p \in V_S^L$, $p \in V_C^L$, and otherwise

where i is the imaginary number $sqrt(-1)$.

Scoring

Now we are ready to define the scoring function $\mathcal{F}(X)$. Note that we assume X is the joint (product) configurational space of L and R, and hence given X, it is trivial to map it to L and R.

$\mathcal{F}(X)$ is defined as the real part of $\int_{p \in \mathcal{D}} \int_{q \in \mathcal{D}} F^L(p) F^R(q) dpdq$.

where $\mathcal{D}$ is the smallest convex subset of R^3 such that $V_S^R \subseteq \mathcal{D}$ and $V_S^L \subseteq \mathcal{D}$.

Clearly, $\mathcal{F}(X)$ is maximized when the skins of the ligand and receptor overlap maximally, but the cores do not overlap, hence capturing the state when L and R are close to but not penetrating each other.

Also note that $\mathcal{F}(X)$ is defined almost exactly the same for docking, as it was for fitting.

9.5.3 Bounding the Error

The different scoring functions we discussed in the previous section can simply be described as integrals over continuous domains. Though they included double integral, we shall restrict our discussion bounding the errors of functions to single integrals. However, the concepts are easily translated to multiple integrals.

Let us define the function we want to evaluate as follows- $\mathcal{F}(x) = \int_{\mathcal{D}} f(u)du$. where $\mathcal{D}$ is a convex domain in s dimensional space and f is the integrand.

We shall use a quasi-Monte Carlo scheme to approximate $\mathcal{F}(X)$ as follows- $\mathcal{F}(X) = \int_{\mathcal{D}} f(u)du \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)$ where $\{x_1, x_2, \ldots, x_N\}$ is a set of N samples in $\mathcal{D}$.

Now, we want to prove that if the set of samples satisfy some criteria, then the approximation error $\mathcal{E} = |\int_{\mathcal{D}} f(u)du - \frac{1}{N}\sum_{n=1}^{N} f(x_n)|$ can be bounded. We also want to prove that the approximation converges, in the sense that as $N \to \infty$, the error $\mathcal{E} \to 0$.

9.5.3.1 Discrepancy

Let $\mathcal{I}_s$ denote the x dimensional unit cube, and let λ_s be the s-dimensional Lebesgue measure.

Let $P = \{x_1, x_2, \ldots, x_N\}$ is a set of N samples in $\mathcal{I}_s$. Let B be an arbitrary subset of $\mathcal{I}_s$ and let $\mathcal{B}$ be the set of all arbitrary subsets of $\mathcal{I}_s$.

Now, we define $A(B, P) = |B \cap P|$ as the number of points in the overlap of B and P.

Then, discrepancy of a set of samples P in a space is defined as follows-

$$D_N(\mathcal{B}, P) = \sup_{B \in \mathcal{B}} \left| \frac{A(B,P)}{N} - \lambda_s(B) \right|$$

in other words, a low discrepancy sampling covers any arbitrary subset of the space uniformly.

In general, computing $D_N(\mathcal{B}, P)$ over arbitrary subsets may be difficult, and we use special types of subsets only.

Star discrepancy The star discrepancy is defined as $D_N^*(P)$ for a set of N points $P = \{x_1, x_2, \ldots, x_N\}$, as $D_N(\mathcal{I}^*, P)$ where $\mathcal{I}^*$ is the set of all subintervals of I_s of the form $\prod_{i=1}^{s}[0, u_i)$.

Extreme discrepancy The extreme discrepancy is defined as $D_N(P)$ for a set of N points $P = \{x_1, x_2, \ldots, x_N\}$, as $D_N(\mathcal{I}, P)$ where $\mathcal{I}$ is the set of all subintervals of I_s of the form $\prod_{i=1}^{s}[u_i, v_i)$.

Isotropic discrepancy The isotropic discrepancy is defined as $J_N(P)$ for a set of N points $P = \{x_1, x_2, \ldots, x_N\}$, as $D_N(\mathcal{J}, P)$ where $\mathcal{J}$ is the set of all convex subsets I_s.

Proposition 9.5.1 (Relationship between discrepancies (adapted from Proposition 2.4 of [186]))**.**

- $D_N^*(P) \leq D_N(P) \leq 2^s D_N^*(P)$
- $D_N(P) \leq J_N(P) \leq 4s(D_N(P))^{1/s}$

9.5.3.2 Bounded Variance of a Function

All of the error bounds we shall introduce in the next section depend both on the discrepancy of the sample, in the sense that lower discrepancy guarantees lower error. However, the nature of the function also affects the worst case error. For example if the slope of a 1D function approaches infinity at any point in the domain (e.g. $tan\theta$), then an error guarantee cannot be provided with any finite sample set. So, the error bounds we discuss requires that the functions are Lipschitz, and more specifically are continuous in the domain and have bounded variance. A measure called the modulus of continuity was already introduced before in this article. Here we repeat it again and then discuss different definitions of bounded variance.

Definition 9.5.1 (Modulus of continuity)**.**
For a continuous function f on $\mathbb{I}^s$, the modulus of continuity is defined as $\omega(f,t) = \sup_{u,v\in\mathbb{I}^s \& \delta^S(u,v)\leq t} |f(u) - f(v)|$. In other words, the value of f does not change without bounds if the parameters are close.

Definition 9.5.2 (k dimensional face of $\mathbb{I}^s$)**.**
Given a sequence $i^1, \ldots, i^k$ such that $1 \leq i^1 \leq i^2 \leq \ldots \leq i^k \leq s$, representing indices to k distinct dimensions of $\mathbb{I}^s$, we define a k dimensional face $\phi(i^1, \ldots, i^k)$ of the domain $\mathbb{I}^s$ as the set of all points $(u_1, \ldots, u_s) \in \mathbb{I}^s$ such that $u_j = 1$ for all $j \neq i^1, \ldots, i^k$. [For example, if 1,3 in 3 dimensional space would be a plane.]

For the next two definitions, we assume that not only f is a continuous function over $\mathbb{I}^s$, but also the partial derivatives of f, i.e. $\frac{\delta^s f}{\delta u_1 \delta u_2 \ldots \delta u_s}$, are also continuous in $\mathbb{I}^s$.

Definition 9.5.3 (Variation of f on $\mathfrak{I}^s$ in the sense of Vitaly)**.**

We define the variation of f in the k dimensional face $\phi(i^1, \ldots, i^k)$ of the domain $\mathfrak{I}^s$ in the sense of Vitali as follows-

$$V^k(f, \phi(i^1, \ldots, i^k)) = \int_0^1 \ldots \int_0^1 |\frac{\delta^s f}{\delta u_{i_1} \delta u_{i_2} \ldots \delta u_{i_k}}| du_{i_1} du_{i_2} \ldots du_{i_k}$$

Intuitively, The variation in the sense of Vitali measures the change of f on the k-dimensional face.

Definition 9.5.4 (Variation of f on $\mathfrak{I}^s$ in the sense of Hardy and Krause)**.**

$V(f) = \sum_{k=1}^{s} \sum_{1 \leq i^1 \leq i^2 \leq \ldots \leq i^k \leq s} V^k(f, \phi(i^1, \ldots, i^k))$

Intuitively, variation in the sense of Hardy and Krause represent the variation of f in all possible k-dimensional faces, for all possible k.

Finally, we say that f has bounded variation if $V(f)$ is finite. In the next section (specifically in Theorem 9.5.2) we show that if f has bounded variation, then the intergal $\mathcal{F}(X)$ can be approximated with bounded error.

9.5.3.3 Bounds

If f has bounded variation in $\mathcal{D} \subseteq \mathfrak{I}^s$ then we can bound the approximation error as follows.

Theorem 9.5.2 (Bounded error of intergal over convex domain (adapted from Theorem 2.14 of [186]))**.**

If $\mathcal{D} \subseteq \mathfrak{I}^s$ is convex and f has bounded variation $V(f)$ on $\mathfrak{I}^s$ in the sense of Hardy and Krause, then, for any point set $P = \{x_1, x_2, \ldots, x_N\}$ such that $x_i \in \mathcal{D}$, we have-

$$|\int_{\mathcal{D}} f(u)du - \frac{1}{N}\sum_{n=1}^{N} f(x_n)| \leq (V(f) + |f(1,\ldots,1)|)J_N(P)$$

If we can show on the other hand that f is continuous in $\mathcal{I}^s$, then a possibly tighter bound can be achieved as follows-

Theorem 9.5.3 (Bounded error of intergal over $\mathcal{I}^s$ (adapted from Theorem 2.13 of [186]))**.**
If f is continuous in $\mathcal{I}^s$, then, for any point set $P = \{x_1, x_2, \ldots, x_N\}$ such that $x_i \in \mathcal{I}^s$, we have-

$$|\int_{\mathcal{D}} f(u)du - \frac{1}{N}\sum_{n=1}^{N} f(x_n)| \leq 4\omega(f; (D_N^*)^{1/s})$$

Appendix

Appendix 1

Background on Proteins, their Interactions and Symmetry

1.1 On Protein Structure, Representations and Flexibility

A protein is basically a polypeptide chain, i.e. a chain of peptides. A peptide, or often called an amino acid, consists of an Amine group (NH_2), a central carbon atom (called C_α attached to different residues/side chains R, and another carbon atom (C) part of a Carboxilic acid group (COOH). A sequence of peptides form a chain when a Hydrogen from the amine group of one bonds with the Hydroxil part (OH) of the Carboxilic group to form water, and enabling the N of the first to form a covalent bond with the C of the second; thereby combining the backbones of the peptides into a longer backbone (Figure 1.1). Hence, a peptide chain's backbone is a sequence of N-C_α-C_β-N-C_α-C_β-N...-C_β. Note that all intermediate N is also attached to a Hydrogen, all C_α is attached to a Hydrogen and a residue, and all intermediate C_β is also attached to an Oxygen (O).

There are mainly 20 types of residues (hence, 20 types of amino acids) found in nature. Some of the residues contain more than 20 atoms (Figure 1.2). In the rest of the article, we shall use amino acids, peptides and residues synonymously to mean the collection of atoms on the side-chain as well as the N-C_α-C_β-O atoms

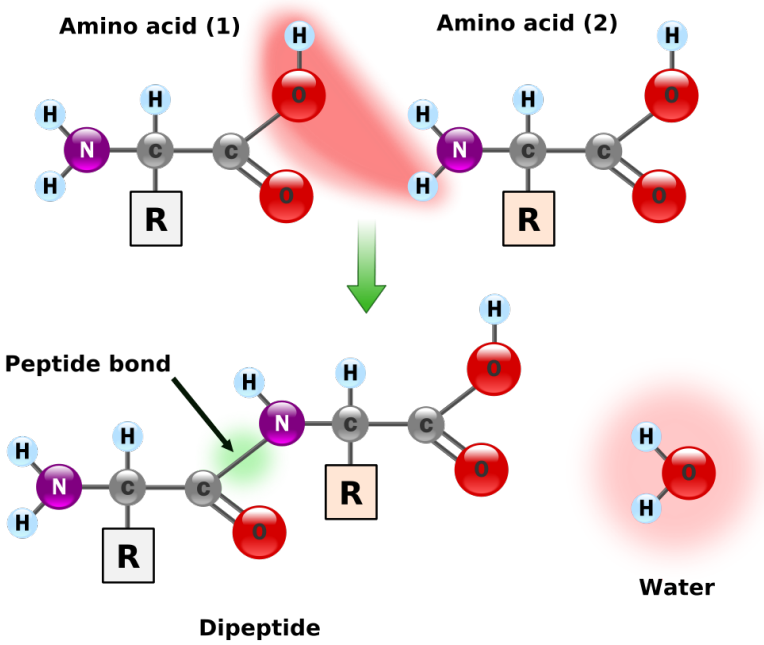


Figure 1.1: Structure of peptides (amino acids), and formation of polypeptide chains (proteins)

attached to the side-chain. When we must distinguish, we shall use the terms 'side-chain' and 'backbone'.

Each amino acid is often also represented using a single letter, and a sequence of such letters is called the primary structure of a protein. The sequence, or the primary structure, is usually much easier to identify, than the exact locations of each atom of the protein. When the location of every atom of a protein (in some coordinate system, or with respect to each other) is defined, then the model is called the tertiary structure of the protein. Sometimes multiple peptide chains bind with each other and form complexes, and in that case, if the atomic coordinates of all the chains are known, then the model is called the ternary structure of the complex.

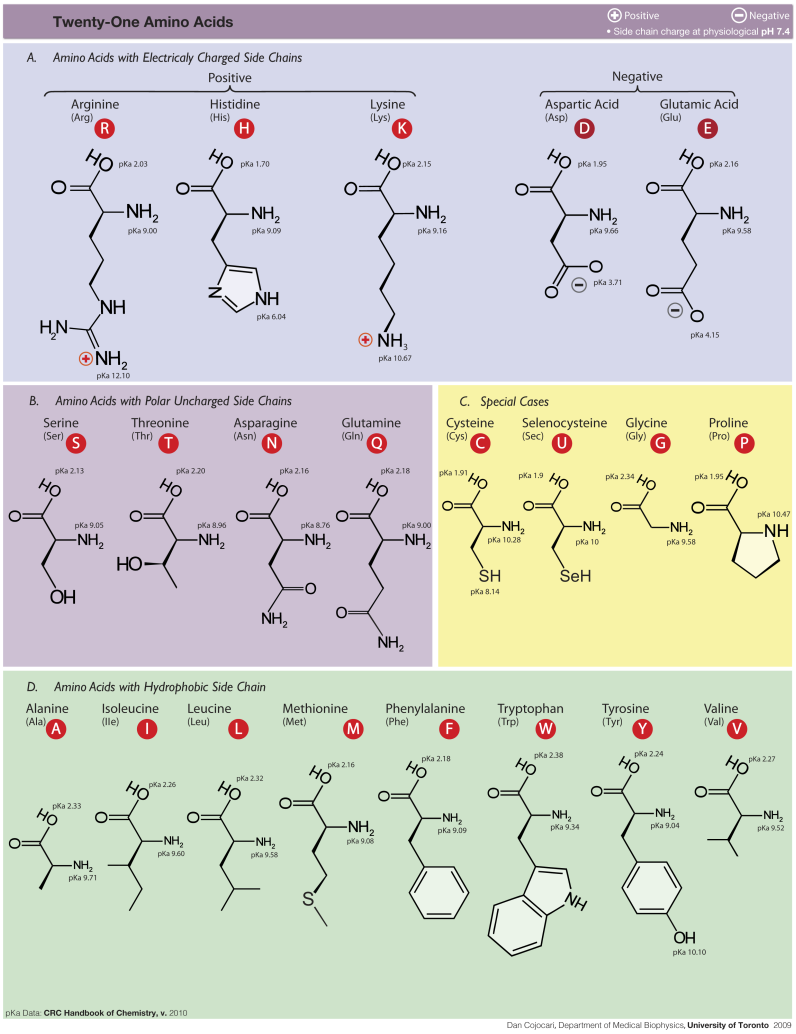


Figure 1.2: List of amino acids and their structures

Finally, proteins have two specific motifs, called the α-helices and the β-sheets that appear as sub-structures. Both of these structures appear by formation of hydrogen bonds between the NH and $C_\beta O$ of the backbone. In an α-helix, the hydrogen bond in formed between the residues i and $i+3$ or $i+4$, and the backbone twists into a helical structure such that each residue corresponds to about a 100 degree turn and

1.5Åpitch. The beta sheet is formed when two sections of the backbone lies parallel (or anti-parallel) to each other to form a planar configuration. Sometimes more than two sections can also lie in parallel. A protein can contain several α-helices or β-sheets, as well as some sections where no apparent template or motif is apparent, such sections are called loops. α-helices, β-sheets and loops are called secondary structural components of a protein.

Experimental Protocols for Elucidating Protein Structure

Resolving the structure of a molecule at the atomic resolution (2Å) is not a trivial problem. Several experimental protocols have been developed, most prominently X-ray Crystallography [146], Nuclear Magnetic Resonance (NMR) [31, 274], and Cryo electron microscopy (EM) [97]. According to the protein data bank repository 85372 atomic resolution structural models have been resolved using x-ray crystallography, 10512 models by nuclear magnetic resonance techniques as of the end of December 2013. X-ray crystallography, first applied for proteins in 1958 to resolve the structure of the Myoglobin molecule [146], works by crystallizing a purified sample of the protein-complex and then analyzing the scattering pattern of x-ray shot at the specimen. However, many proteins, for example proteins with disordered regions are difficult to crystallize, and most insoluble membrane proteins, x-ray crystallography cannot be applied [165], leaving a large portion of the protein landscape beyond its reach. Another drawback of x-ray crystallography is the possibility that the crystallization process induces conformational changes in the protein and the resolved structure is not the natural state of the protein. Nuclear

magnetic resonance (NMR) [31, 274] is applicable to an even smaller set of proteins since the magnetic response from the nuclei of the atoms, which is used to resolve their position and type, tends to decay too quickly for large molecules. The largest structure resolved by NMR [94] is still much smaller than macromolecules like viral capsids and ribosomes. Cryo electron microscopy (EM) is an alternate technique which while does not resolve atomic positions, it is however able to provide a volume occupancy model for a protein-complex in its natural state.

1.2 On Protein Protein Interaction

Protein-protein interactions are governed by a multitude of different weak forces including electrostatics interactions, short range attraction-repulsion, polarization properties of the surface atoms, solvent pH etc. to name a few. Also, the dominant contributor to the affinity between proteins can vary across protein families, and also within families.

In simple terms, a perfect scoring function would always score the true configuration of a protein-protein complex higher than any other configurations of the same proteins. Due to the irregular and highly variable nature of protein-protein interactions, no scoring model designed so far is perfect, and probably they never will be.

The scoring function is usually designed to mimic the biophysical interaction and affinity of the proteins, and is trained based on a benchmark of known solutions. The scoring terms used in docking, molecular dynamics,folding etc. can broadly be classified into two types: physics (free energy) based and knowledge-

based (statistical) scoring. We discuss each of the terms in brief below.

Free Energy Based Scoring

The free energy E of a molecule of a molecule (or a complex) is given by

$$E = E_{\text{MM}} + E_{\text{sol}} - TS \tag{1.2.1}$$

E_{MM} is called the molecular mechanical energy. It represents energy due to the atom-atom interactions among atoms of the molecule(s). E_{sol} is the solvation energy representing the interaction of the molecule(s) with the solvent. The solvent is usually considered as water with some charged ions. T is the temperature and S in the entropy. The change of free energy upon binding or the binding free energy is defined as $\delta(A, B) = E(A + B) - (E(A) + E(B))$, i.e. the difference of the total free energies before and after binding. Since, the binding free energy is highly correlated with the stability of a complex, scoring functions are designed to mimic or approximate the binding free energy.

Now, we discuss each component of the energy-

$$E_{\text{MM}} = \underbrace{E_d + E_\theta + E_\varphi}_{\text{bonded interactions}} + \underbrace{E_{\text{vdw}} + E_{\text{coul}}}_{\text{nonbonded interactions}} \tag{1.2.2}$$

$$E_d = \sum_{bond\ length\ (d)} k_d(d-d_{eq})^2,\ E_\theta = \sum_{bond\ angle\ (\theta)} k_\theta(\theta-\theta_{eq})^2,\ E_\varphi = \sum_{torsion\ (\varphi)} k_\varphi(1-\cos[n(\varphi-\varphi_{eq})]), \tag{1.2.3}$$

$$E_{\text{vdw}} = \sum_i \sum_{j>i} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} \right) \text{ and } E_{\text{coul}} = \sum_i \sum_{j>i} \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}}, \tag{1.2.4}$$

The molecular mechanical energy is decomposed into bonded and non-bonded interaction terms (see Equation 1.2.2). The bonded energy terms (Equation 1.2.3) measure the energy required to deviate from an optimal bonded position (for example, changing the length of a bond). The bonded energy terms are considered constant for rigid body docking and cancels out when computing the binding free energy. However, they must be taken into account for flexible docking. Flexible docking techniques based on molecular or stochastic dynamics must include these terms in their scoring. On the other hand, if the flexible docking is performed using the ensemble docking approach, where a collection of low energy conformations for the molecules are generated and then rigid docking is performed for each pair; then these terms are only considered during the generation of the ensemble.

The non-bonded interactions (Equation 1.2.3) are the most widely used scoring terms in docking. The first term is called the VDW interaction which represents short range attraction (of electron of one atom to the proton of the other) and a very high repulsion if the distance r_{ij} gets too close (representing the positive nuclei of the atoms coming too close). The a_{ij} and b_{ij} are two weights which have been determined based on quantum mechanics for different types of atom-atom pairs. This energy encourages two molecules to come close to each other such that their

shapes complement each other, but does not allow penetration. So, docking softwares sometimes replace this term with a geometric shape complementarity term. The second term of the non-bonded interaction is the long range electrostatic interaction between two molecules. This term is dependent on the charges q as well the distance dependent dielectric $\epsilon(r_{ij})$ of the solvent.

Several models for the solvation energy have been proposed. Some of them take the molecules and ions in the solvent explicitly into consideration. Other models, termed implicit solvent models applies principles from statistical thermodynamics to approximate the energy. Here, we present an implicit solvation energy model called the GBSA model [105].

$$E_{\text{sol}} = \underbrace{E_{\text{cav}} + E_{\text{vdw(s-s)}}}_{\text{nonploar}} + \underbrace{E_{\text{pol}}}_{\text{polar}} \tag{1.2.5}$$

E_{cav} depends on the volume of the protein and the exposed surface area and $E_{\text{vdw(s-s)}}$ is the Van der Waals interaction between exposed atoms and solvent atoms. The polar part E_{pol}, can be approximated using *Generalized Born* (GB) theory [245] as-

$$E_{\text{pol}} = -\frac{\tau}{2} \sum_{i,j} q_i q_j / \sqrt{r_{ij}^2 + R_i R_j e^{-\frac{r_{ij}^2}{4R_i R_j}}}, \tag{1.2.6}$$

where $\tau = 1 - \frac{1}{\epsilon}$, and R_i is the effective Born radius of atom i. There are other models for E_{pol} as well, e.g. [250].

Knowledge Based Scoring

An alternate to free energy based scoring, is to design scoring functions completely based on empirical data. Such models are primarily derived by identifying different properties of the protein-protein interfaces of known complexes. See [281] for a brief survey of knowledge based scoring terms.

A knowledge based scheme is defined based on the observation that hydrophobic residues are more likely to be at the binding interface, since after the binding they get buried and hence does not come into contact with water (solvent). For example, Black and Mould [44] developed a set of hydrophobicity based parameters for each residue and atom types to be used in protein folding studies. The same model can be applied to docking as well.

A similar idea is to simply observe the frequencies at which different residues or atoms appear on protein-protein interfaces. The relative probability of a residue appearing on the interface, given that it appears somewhere in the protein is defined as the interface propensity of the residue. Per-residue interface propensity values were computed in [134] which are based on the relative frequencies of different residues in the interfaces of a set of 63 protein-protein complexes from [133].

Sometimes, specific features can be identified for classes of proteins. For example, it has been observed [1, 169] that each of three specific regions of an antibody make at least one contact with the antigen in an antibody-antigen complex. The regions are: 1) CDR-L3, 2)CDR-H3 and 3) CDR-L1 or CDR-H1. Similarly it is observed that enzyme binding sites contain a sequence of peptides of the forms

$G-X-Y$ or $Y-X-G$ where G is Glycine and X, Y are any two small non-polar residues [277]. Information like these are often used to reward or penalize docking poses for specific type of complexes.

The most popular type of knowledge based scoring is based on defining pairwise contact potentials between residues or atoms. Similar to interface propensities, such potentials are trained based on observed contacts in a benchmark of complexes. Tanaka and Sheraga first proposed a potential like this in [246]. Zhou and Zhou derived distance dependent potentials for 19 different atom types in [289] and modified it to define residue based potentials in [283]. Other models based on similar principles include [182, 201, 212, 251, 252]. We should specially mention that, recently Mayewski developed a multi-body contact potential designed specifically for scoring complexes involving more than two proteins [176].

Finally, other types observations about the protein-protein interfaces exist, which may not directly be applied as scoring terms, but can be used to differentiate between native and non-native configurations. For example, the interface area, interface shape, interface gap, percentage of polar vs nonpolar area etc. have been shown to be remarkably consistent in native protein-protein interfaces. Bahadur et al. studied such statistics for protein-protein interfaces in [11], for protein-RNA interfaces in [13] and for specifically viral capsid proteins in [12].

1.3 On Protein Symmetry

Here we introduce some symmetry classes which are specially relevant for protein complexes, provide examples and introduce some notations.

Cyclic Symmetry

In cyclic symmetry, n copies of the same monomer $P_1, ..., P_n$ forms a complex. In the complex, each monomer P_i is placed by rotation around a single axis by $\frac{360*i}{/}n$ degrees such that the interface between P_i and P_{i+1} is identical for all $i \in [1, n]$. We denote a cyclic symmetry involving n copies of a monomer P as $C_n(P)$. See figure 1.3 for some examples.

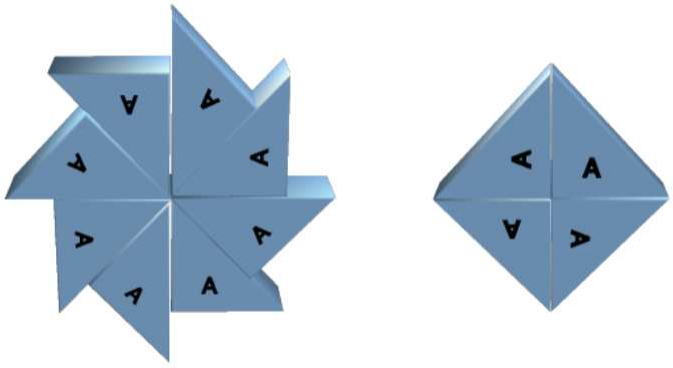


Figure 1.3: Left, C_8 symmetry and Right, C_4 symmetry.

Dihedral Symmetry

If 2 copies of the same monomer P_1, P_2 forms a complex, such that there is a plane p such that P_2 is congruent to P_1 reflected by the plane p. This can also be considered a rotation by 180 degrees around a suitable axis. It is customary to denote 2 bodies in dihedral symmetry as $D_1(P)$. We also use $D_n(P)$ as a notation for $D_1(D_{n-1}(P))$ and D_3 and so on. Often we drop the subscript for D_1 and simply use D. See Figure 1.4 for examples.

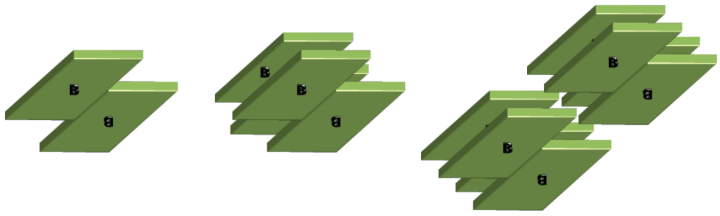


Figure 1.4: Dihedral symmetry. D_1 on the left, D_2 at the middle and D_3 on the right.

Note that this definition of dihedral symmetry is different from the definition in group theory.

Combination of cyclic and dihedral symmetries

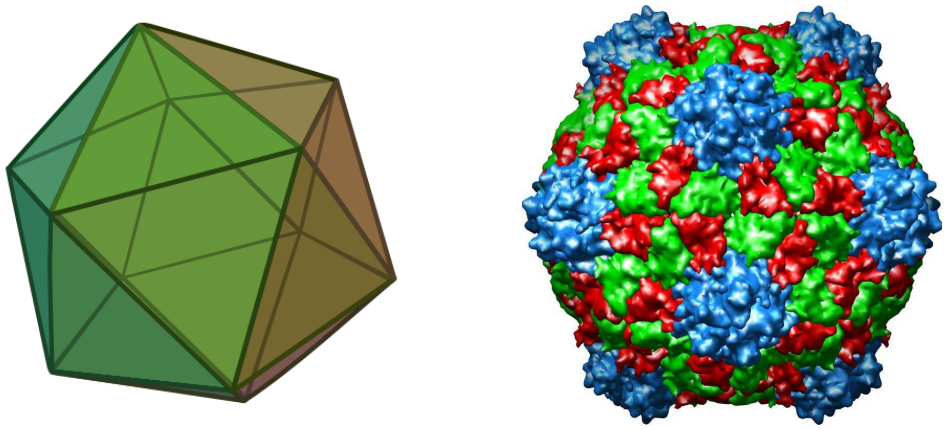
Sometimes combinations of symmetries are found in protein complexes. For example, a protein P may form a dimer and then the dimer is repeated 4 to form a cyclic complex (see Figure 1.5). In such cases, we denote the symmetry as a ordered sequence. For example, the above symmetry will be denoted $C_4(D_1(P))$.

Figure 1.5: Different type of mixed symmetry. (a) A dimer of cyclic symmetric complex (DC_8), (b) Cyclic symmetry of dimers (C_4D).

Icosahedral Symmetry

An icosahedron consists of 20 triangles arranged on a shell such that is 5-fold rotational symmetries at the vertices, 3-fold rotational symmetries at the centers of the triangles and 2-fold rotational symmetries on the edges. We denote this symmetry class as I. See Figure 1.6 for an example.

For icosahedral viruses, if the class of local symmetry is restricted to the Caspar-Klug model only, then we denote it as I_{CK}. See Section 1.4 for details on icosahedral symmetry.



(a) (b)

Figure 1.6: Icosahedral symmetry. (a) A generic icosahedron showing the faces and vertices explicitly. (b) A viral capsid exibiting icosahedral symmetry (Cowpea Mosaic Virus, PDBID 1NY7)

Helical Symmetry

Monomers are arranged on a helix. Each monomers interface with its neighbor relates to a cyclic rotation as well as a shift along the direction of the pitch. We

denote this class as H. See Figure 1.7 for an example.

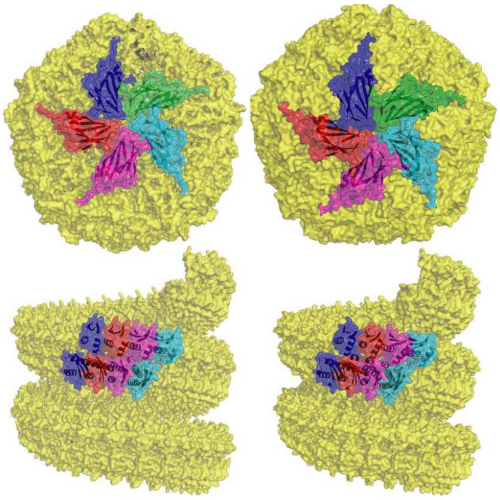


Figure 1.7: Helical symmetry (PDBID:1YJ7). Image taken from [7]

Symmetry involving different proteins

Different types of proteins form an asymmetric complex and then the asymmetric complex is repeated to form a symmetric structure. In such cases, we represent the asymmetric unit formed by the different proteins as the operand of a symmetry operation. For example, for the complex shown in Figure 1.8 would be represented as $C_4(P + Q)$.

1.4 On Viral Capsids

Viruses are microorganisms which are responsible for many diseases in humans as well as many species of animals and plants. Usually viruses have their genetic material (DNA or RNA) encapsulated by a shell formed by proteins. This

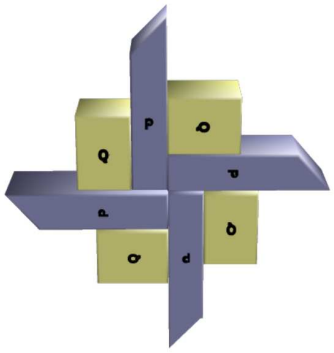
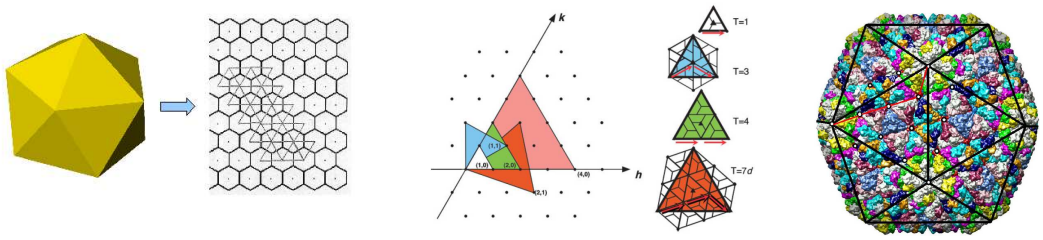


Figure 1.8: Cyclic symmetry where the asymmetric unit consists of two proteins ($C_4(P+Q)$)



(a) (b) (c)

Figure 1.9: (a) The triangles of an icosahedron can be unfolded onto a 2D hexagonal grid, such that corners of the triangles have integer coordinates. (b) Caspar and Klug's idea of using different sized triangles on the hexagonal grid to generate different local symmetry locations on the icosahedron. Corners belonging to only small triangles are locations of 6-fold symmetry and corners belonging to large triangles are locations of 5-fold symmetries. (c) The capsid of Rice Dwarf Virus. The symmetry locations were detected using the Caspar-Klug theorem (image generated using TexMol [15]).

shell is called the viral capsid. Sometimes, the protein shell is also covered by another layer of glycoproteins. Most viruses are actually smaller than a single cell and contain no ribosomes. Since, ribosomes carry out the process of producing proteins based on their template in the RNA, viruses must infiltrate the cell of other

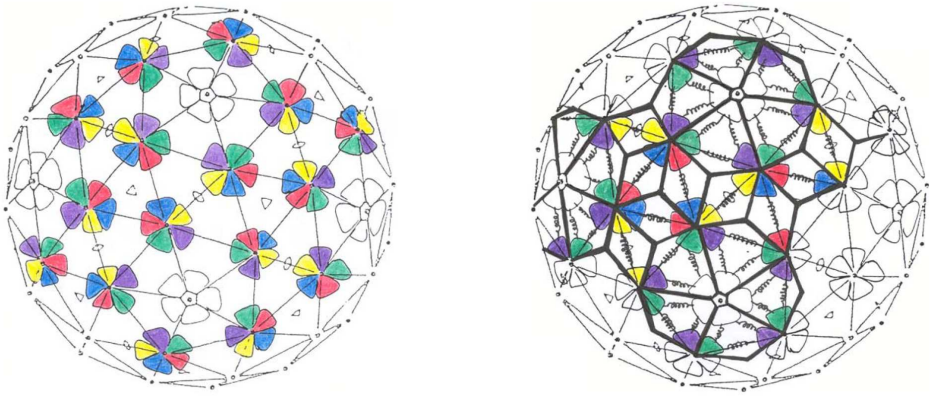
living organism and use their ribosomes to replicate and reproduce. To infiltrate, the virus attaches to the cell membrane and injects the genetic material into the cell. There, several copies of the genetic material are transcripted; copies of the capsid protein are also synthesized. The proteins and the genetic material self assemble to form new viruses which then penetrates the membrane and are ready to infect other cells. The viral capsid is of particular interest to design antiviral drugs because it could potentially prevent the virus from either infiltration or from reproduction (by inhibiting the self-assembly). Using imaging techniques like electron microscopy and x-ray crystallography, it has long been established that most viral capsids are either spherical or helical. it was also observed that the arrangement of the proteins on the spherical capsids exhibit icosahedral symmetry. Several theories and observations have been proposed to explain and characterize this symmetry and organization.

Caspar and Klug [55] proposed an elegant and simple formalism to describe this symmetry. Their idea was that though icosahedral symmetry does not strictly fall under the class of periodic crystallographic symmetry, but the local interfaces observed by each protein on the capsid is exactly the same, a concept which was termed as quasi-symmetry. Assuming that the capsid is a simple icosahedron with 20 triangular faces and 12 vertices with 5-fold symmetry, they showed a way to unfold the triangles onto a 2D hexagonal grid such that the vertices of the triangles have integer coordinates on the hexagonal grid (Figure 1.9(a)). Their method enforced quasi symmetry by requiring that the proteins are placed on each corner of each triangles. So, under the most basic representation, a capsid have 60 pro-

teins. However, there are capsids with a much larger number of proteins on them. Caspar and Klug explain the structure of larger proteins by increasing the size of the triangles such that the corners still have integer coordinates. Hence, these large triangles would contain many smaller (unit sized) triangles. If a large triangle has two corners at $(0, 0)$ and (h, k) coordinates, then the number of small triangles in it is exactly $h^2 + k^2 + hk$. This number is called the T-number. So, according to Caspar-Klug theory, a capsid has exactly $60T$ triangles, 12 vertices with 5-fold symmetry corresponding to corners of large triangles, and $10(T - 1)$ locations of six-fold symmetry corresponding to corners of small triangles which are not corners of large triangles (Figure 1.9(b)). This simple theory successfully models most of the observed spherical viral capsids (for example, (Figure 1.9(c))).

However, Caspar-Klug theory fails to explain the capsid structure of a some viruses in the papilloma, papova and polyoma families. For example, the Simian Virus 40 (of Papova family) is predicted to have T=7 capsid i.e 420 proteins with 12 locations with 5-fold symmetry and 60 locations with 6-fold symmetry, but in reality it has only 360 proteins and 72 locations with 6-fold symmetry (Figure 1.10(a))[160]. Another example is the inner layer of the capsid of the Blue-tongue virus which has 120 proteins or T=2, a fact considered impossible under the Caspar-Klug model.

Recently, Twarock et al. proposed a new model with the assumption that the local symmetries (the arrangement of proteins inside a large triangle) are also governed by icosahedral symmetry [143, 145, 193, 256–258]. The study of aperiodic crystal symmetries show that lattices with 5-fold symmetry in 3D can be generated



(a) (b)

Figure 1.10: (a) The capsid of the Polyoma virus cannot be tiled correctly using the Caspar-Klug model. The locations and numbers of the proteins are wrong. (b) But it can be correctly tiled using a set of kite and rhomb tiles. [fig/AssemblyProblem taken from [256]]

by projection of a regular lattice of 6D. More specifically, a projection from the root system D_6 can be used to generate point samples in 3D which follow icosahedral symmetry. However, Twarock et al. showed that specific affine extensions of the icosahedral root system H_3 can also be used to generate such point samples. The sampled points are the locations of 5, 6 and 3 fold symmetries. They also propose a set of tiles which can be used to cover the capsid surface by connecting the sampled points. The set of tiles include Caspar-Klug's equilateral triangles as well as kite, dart and rhombus tiles. Similar to Caspar-Klug theory, these set of tiles also place proteins at equivalent positions i.e. corners with equal angles. There are also specific rules on which tiles can can share which of their edges with each other. These additional set of tiles can be successfully used to explain the structures and symmetry locations of the capsids for which Caspar-Klug theory fails (Figure

1.10(b)).

Among other relevant research, Zandi et al. [259, 282] analyzed the capsid structure by assuming that the building blocks are circular disks and showed that the most energetically stable organization of such disks has icosahedral symmetry. This study explains why icosahedral capsids are so prevalent in nature. Recently, Mannige and Brooks [172] analyzed the structures of different families of icosahedral viruses and determined a class of viral capsids which they call 'canonical capsids'. These canonical capsids can be tiled by the Caspar-Klug model. Using simple topological constraints, it was also shown that in canonical viruses, each prototile (protein) have exactly 5 interfaces with other prototiles. The same authors also studied the distribution of dihedral angles at six-fold and 5-fold symmetry locations [171] and empirically established that the dihedral angles at the five fold locations are exactly the same, but the dihedral angle at six fold locations increase with their distance from five fold locations. This study suggests that larger capsids have more probability of looking like icosahedrons with 20 flat triangles, and smaller capsids would look like perfect spheres (where the proteins are arranged in icosahedral symmetry).

The special class of multi-protein docking problem introduced in this thesis can predict the interactions of proteins within a tile, and also predict the inter-tile interaction i.e. the rules governing the tile-tile contacts. The solution to this multi-protein docking problem coupled with the mathematical formalism of the viral tiling theory also allows us to predict all possible structure of complete and intermediate states of a capsid.

Bibliography

[1] Antibody-antigen contacts. http://www.bioinf.org.uk/abs/allContacts.html.

[2] The pdb file format. http://www.wwpdb.org/docs.html.

[3] PDB2PQR: An automated pipeline for the setup, execution, and analysis of Poisson-Boltzmann electrostatics calculations. http://pdb2pqr.sourceforge.net/.

[4] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71(1-3):5–22, 1996.

[5] Frank Alber, Svetlana Dokudovskaya, Liesbeth M Veenhoff, Wenzhu Zhang, Julia Kipper, Damien Devos, Adisetyantari Suprapto, Orit Karni-Schmidt, Rosemary Williams, Brian T Chait, and et al. Determining the architectures of macromolecular assemblies. *Nature*, 450(7170):683–694, 2007.

[6] C David Andersson, Elin Thysell, Anton Lindström, Max Bylesjö, Florian Raubacher, and Anna Linusson. A multivariate approach to investigate docking parameters effects on docking performance. *Journal of Chemical Information and Modeling*, 47(4):1673–1687, 2007.

[7] I. Andre, P. Bradley, C. Wang, and D. Baker. Prediction of the structure of symmetrical protein assemblies. *PNAS*, 104(45):17656–17661, 2007.

[8] N. Andrusier, R. Nussinov, and H.J. Wolfson. FireDock: fast interaction refinement in molecular docking. *Proteins*, 69(1):139–159, 2007.

[9] Iris Antes, Christian Merkwirth, and Thomas Lengauer. Poem: Parameter optimization using ensemble methods: application to target specific scoring functions. *Journal of Chemical Information and Modeling*, 45(5):1291–1302, 2005.

[10] R P Bahadur and M Zacharias. The interface of protein-protein complexes: analysis of contacts and prediction of interactions. *Cellular and molecular life sciences : CMLS*, 65(7-8):1059–1072, 2008.

[11] Ranjit Prasad Bahadur, Pinak Chakrabarti, Francis Rodier, and Joël Janin. A dissection of specific and non-specific protein-protein interfaces. *Journal of Molecular Biology*, 336(4):943–955, 2004.

[12] Ranjit Prasad Bahadur, Francis Rodier, and Joël Janin. A dissection of the protein-protein interfaces in icosahedral virus capsids. *Journal of Molecular Biology*, 367(2):574–590, 2007.

[13] Ranjit Prasad Bahadur, Martin Zacharias, and Joël Janin. Dissecting protein–rna recognition sites. *Nucleic Acids Research*, 36(8):2705–2716, 2008.

[14] C. Bajaj, R. A. Chowdhury, and M. Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. *Bioinf.*, 27(1):55–62, 2011.

[15] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. TexMol: Interactive visual exploration of large flexible multi-component molecular complexes. In *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, 2004.

[16] C. Bajaj, H. Y. Lee, R. Merkert, and V. Pascucci. NURBS based B-rep models for macromolecules and their properties. In *Proceedings of the 4th ACM Symposium on Solid Modeling and Applications*, pages 217–228, New York, NY, USA, 1997. ACM.

[17] C. Bajaj, H. Y. Lee, R. Merkert, and V. Pascucci. Nurbs based b-rep models for macromolecules and their properties. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 217–228. ACM Press, 1997.

[18] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Multiresolution molecular shapes. Technical report, TICAM, Univ. of Texas at Austin, Dec. 1999.

[19] C. Bajaj, V. Pascucci, A. Shamir, R. Holt, and A. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[20] C. Bajaj, G.-L. Xu, and Q. Zhang. Higher-order level-set method and its application in biomolecular surfaces construction. *Journal of Computer Science and Technology*, 23(6):1026–1036, November 2008.

[21] C. Bajaj and W. Zhao. Fast molecular solvation energetics and forces computation. Technical Report ICES TR-08-20, The University University of Texas at Austin, 2008.

[22] C. Bajaj and W. Zhao. Fast molecular solvation energetics and force computation. *SIAM Journal on Scientific Computing*, 31(6):4524–4552, 2010.

[23] Chandrajit Bajaj, Benedict Bauer, Radhakrishna Bettadapura, and Antje Vollrath. Nonuniform fourier transforms for rigid-body and multi-dimensional rotational correlations. *SIAM Journal of Scientific Computing*, 35(4):B821–B845, 2013.

[24] Chandrajit Bajaj, Rezaul A. Chowdhury, and Muhibur Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*, pages 259–270, 2009.

[25] Chandrajit Bajaj, Rezaul A. Chowdhury, and Muhibur Rasheed. A dynamic data structure for flexible molecular maintenance and informatics. *Bioinformatics*, 27(1):55–62, 2010.

[26] Chandrajit Bajaj, Rezaul A. Chowdhury, and Vinay Siddavanahalli. F2Dock: Fast Fourier Protein-Protein Docking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1):45–58, 2011.

[27] Chandrajit Bajaj, Antonino Favata, Andrea Micheletti, Paolo Podio-Guidugli, and Muhibur Rasheed. A multiscale comparative analysis of viral capsids'

normal modes. *Manuscript*, 2014.

[28] Chandrajit Bajaj and Wenqi Zhao. Fast molecular solvation energetics and forces computation. *SIAM Journal on Scientific Computing*, 31(6):4524–4552, 2010.

[29] Alberto Bartesaghi, Alan Merk, Mario J Borgnia, Jacqueline L S Milne, and Sriram Subramaniam. Prefusion structure of trimeric HIV-1 envelope glycoprotein determined by cryo-electron microscopy. *Nature structural & molecular biology*, 20(12):1352–7, 2013.

[30] S.S. Batsanov. Van der Waals radii of elements. *Inorganic Materials*, 37:871–885(15), September 2001.

[31] Ad Bax and Mitsuhiko Ikura. An efficient 3d nmr technique for correlating the proton and15n backbone amide resonances with the ??-carbon of the preceding residue in uniformly15n/13c enriched proteins. *Journal of Biomolecular NMR*, 1(1):99–104, 1991.

[32] E. Ben-Zeev, A. Berchanski, A. Heifetz, B. Shapira, and M. Eisenstein. Prediction of the unknown: Inspiring experience with the CAPRI experiment. *Proteins: Structure, Function, and Bioinformatics*, 52(1):41–46, 2003.

[33] E. Ben-Zeev and M. Eisenstein. Weighted geometric docking: incorporating external information in the rotation-translation scan. *Proteins: Structure, Function, and Bioinformatics*, 52(1):24–27, 2003.

[34] Pascal Benkert, Marco Biasini, and Torsten Schwede. Toward the estimation of the absolute quality of individual protein structure models. *Bioinformatics (Oxford, England)*, 27(3):343–350, 2011.

[35] Alexander Berchanski and Miriam Eisenstein. Construction of molecular assemblies via docking: Modeling of tetramers with d2 symmetry. *Proteins: Structure, Function, and Bioinformatics*, 53(4):817–829, 2003.

[36] Alexander Berchanski, Dadi Segal, and Miriam Eisenstein. Modeling oligomers with cn or dn symmetry: application to capri target 10. *Proteins*, 60(2):202–206, 2005.

[37] Alexander Berchanski, Boaz Shapira, and Miriam Eisenstein. Hydrophobic complementarity in protein-protein docking. *Proteins*, 56(1):130–42, 2004.

[38] B. Berger and P. W. Shor. On the mathematics of virus shell assembly. Technical report, 1994.

[39] B. Berger, P. W. Shor, L. Tucker-Kellogg, and J. King. Local rule-based theory of virus shell assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 91(16):7732–7736, 1994.

[40] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Res*, 28(1):235–42, 2000.

[41] R. Bettadapura, M. Rasheed, A. Volrath, and C. Bajaj. Pf2fit: Polar fast fourier alignment of atomistic structures with 3d electron microscopy. *Submitted to PLOS Computational Biology*, 2014.

[42] Radhakrishna Bettadapura, Chandrajit Bajaj, and Antje Vollrath. PF3Fit: Hierarchical flexible fitting in 3d em. Technical Report 12-18, 2012.

[43] Aneerban Bhattacharya, Roberto Tejero, and Gaetano T Montelione. Evaluating protein structures determined by structural genomics consortia. *Proteins*, 66(4):778–795, 2007.

[44] S.D. Black and D.R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Analytical Biochemistry*, 193:72–82, 1991.

[45] Claudia Blattner, Jeong Hyun Lee, Kwinten Sliepen, Ronald Derking, Emilia Falkowska, Alba Torrents de la Peña, Albert Cupo, Jean-Philippe Julien, Marit van Gils, Peter S. Lee, and et al. Structural delineation of a quaternary, cleavage-dependent epitope at the gp41-gp120 interface on intact HIV-1 Env trimers. *Immunity*, 40(5):669–80, 2014.

[46] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[47] A. Bondi. van der waals volumes and radii. *The Journal of Physical Chemistry*, 68(3):441–451, 1964.

[48] Camille Bonomelli, Katie J. Doores, D. Cameron Dunlop, Victoria Thaney, Raymond A. Dwek, Dennis R. Burton, Max Crispin, and Christopher N. Scanlan. The glycan shield of HIV is predominantly oligomannose independently of production system or viral clade. *PLoS ONE*, 6(8):1–7, 2011.

[49] A.M.J.J. Bonvin. Flexible protein-protein docking. *Current opinion in structural biology*, 16(2):194–200, 2006.

[50] S. F. Boys. Electronic wave functions. I. a general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 200(1063):542–554, 1950.

[51] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[52] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[53] Adrian A Canutescu and Roland L Dunbrack. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein science*, 12(5):963–972, 2003.

[54] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *Journal of Computational Chemistry*, 26(16):1668–1688, 2005.

[55] D. L. Caspar and A. Klug. Physical principles in the construction of regular viruses. In *Cold Spring Harbor Symposium on Quantitative Biology*, volume 27, pages 1–24, 1962.

[56] Julio Castrillon-Candas, Vinay Siddavanahalli, and Chandrajit Bajaj. Nonequispaced fourier transforms for protein-protein docking. ICES Report 05–44, The University of Texas at Austin, October 2005.

[57] Kunal N. Chaudhury, Yuehaw Khoo, Amit Singer, and David Cowburn. Global registration of multiple point clouds using semidefinite programming. *arXiv preprint*, 2013.

[58] Jingyi Chen, Fusayo Saeki, Benjamin J. Wiley, Hu Cang, Michael J. Cobb, Zhi Yuan Li, Leslie Au, Hui Zhang, Michael B. Kimmey, Xingde Li, and et al. Gold nanocages: Bioconjugation and their potential use as optical imaging contrast agents. *Nano Letters*, 5(3):473–477, 2005.

[59] Rong Chen, Li Li, and Zhiping Weng. Zdock: An initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Genetics, Special Issue: CAPRI - Critical Assessment of PRedicted Interactions . Issue Edited by Joel Janin*, 52(1):80–87, May 2003.

[60] Rong Chen and Zhiping Weng. Docking unbound proteins using shape complementarity, desolvation, and electrostatics. *Proteins: Structure, Function, and Genetics*, 47(3):281–294, 2002.

[61] Rong Chen and Zhiping Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function, and Genetics*, 51(3):397–408, 2003.

[62] Taeg Sang Cho, Shai Avidan, and William T. Freeman. A probabilistic image jigsaw puzzle solver. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 183–190. IEEE, 2010.

[63] Yoonjoo Choi and Charlotte M Deane. FREAD revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, 78(6):1431–1440, 2010.

[64] Gaurav Chopra, Nir Kalisman, and Michael Levitt. Consistent refinement of submitted models at CASP using a knowledge-based potential. *Proteins*, 78(12):2668–2678, 2010.

[65] Rezaul Chowdhury, Muhibur Rasheed, Donald Keidel, Maysam Moussalem, Arthur Olson, Michel Sanner, and Chandrajit Bajaj. Protein-protein docking with f2dock 2.0 and gb-rerank. *PLoS ONE*, 8(3):e51307, 2013.

[66] Rezaul Alam Chowdhury and Chandrajit Bajaj. Algorithms for faster molecular energetics, forces and interfaces. ICES report 10–32, Institute for Computational Engineering & Science, The University of Texas at Austin, August 2010.

[67] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir, and M. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Computational Geometry*, 5(2):99–160, 1990.

[68] C Colovos and T O Yeates. Verification of protein structures: patterns of nonbonded atomic interactions. *Protein science*, 2(9):1511–1519, 1993.

[69] Stephen R Comeau and Carlos J Camacho. Predicting oligomeric assemblies: N-mers a primer. *Journal of Structural Biology*, 150(3):233–244, 2005.

[70] Stephen R. Comeau, David W. Gatchell, Sandor Vajda, and Carlos J. Camacho. Cluspro: an automated docking and discrimination method for the prediction of protein complexes. *Bioinformatics*, 20:45–50, January 2004.

[71] M. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 1983.

[72] Harold Scott Macdonald Coxeter. *Regular Polytopes (3rd edition)*. Dover Publications, 1973.

[73] Rhiju Das, Ingemar André, Yang Shen, Yibing Wu, Alexander Lemak, Sonal Bansal, Cheryl H Arrowsmith, Thomas Szyperski, and David Baker. Simultaneous prediction of protein folding and docking at high resolution. *Proceedings of the National Academy of Sciences of the United States of America*, 106(45):18978–18983, 2009.

[74] Ian W Davis, Andrew Leaver-Fay, Vincent B Chen, Jeremy N Block, Gary J Kapral, Xueyi Wang, Laura W Murray, W Bryan Arendall, Jack Snoeyink, Jane S Richardson, and et al. MolProbity: all-atom contacts and structure validation for proteins and nucleic acids. *Nucleic acids research*, 35(Web Server issue):W375–W383, 2007.

[75] Erik Demaine. Advanced data structures (6.897) lecture notes (MIT), Spring 2005. http://courses.csail.mit.edu/6.897/spring05/lec/lec09.pdf.

[76] Tao Deng, Ming-Li Yu, Guang Hu, and Wen-Yuan Qiu. The architecture and growth of extended platonic polyhedra. *Communications in Mathematical and in Computer Chemistry*, 67:713–730, 2012.

[77] A. Desmyter, S. Spinelli, F. Payan, M. Lauwereys, L. Wyns, S. Muyldermans, and C. Cambillau. Three camelid vhh domains in complex with porcine pancreatic alpha-amylase. inhibition and versatility of binding topology. *Journal of Biological Chemistry*, 277(26):23645–23650, 2002.

[78] R. Diamond. A real-space refinement procedure for proteins. *Acta Crystallographica Section A*, 27:436–452, 1971.

[79] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker. Pdb2pqr: an automated pipeline for the setup, execution, and analysis of poisson-boltzmann electrostatics calculations. *Nucleic Acids Research*, 32:665–667, 2004.

[80] Cyril Dominguez, Rolf Boelens, and Alexandre M J J Bonvin. Haddock: a protein-protein docking approach based on biochemical or biophysical information. *Journal of the American Chemical Society*, 125(7):1731–1737, 2003.

[81] A G Donchev, V D Ozrin, M V Subbotin, O V Tarasov, and V I Tarasov. A quantum mechanical polarizable force field for biomolecular interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(22):7829–7834, 2005.

[82] Katie J Doores and Dennis R Burton. Variable loop glycan dependency of the broad and potent HIV-1-neutralizing antibodies pg9 and pg16. *Journal of virology*, 84(20):10510–10521, 2010.

[83] P. Steinhardt D.P. DiVincenzo. *Quasicrystals: The state of the Art*. World Scientific, 1999.

[84] Dina Duhovny, Ruth Nussinov, and Haim J. Wolfson. Efficient unbound docking of rigid molecules. In *Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, WABI '02, pages 185–200, London, UK, UK, 2002. Springer-Verlag.

[85] B. S. Duncan and A. J. Olson. Approximation and characterization of molecular surfaces. *Biopolymers*, 33:219–229, 1993.

[86] Bruce S. Duncan and Arthur J. Olson. Approximation and characterization of molecular surfaces. *Biopolymers*, 33:219–229, 1993.

[87] H. Edelsbrunner and E. Mucke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[88] M Eisenstein, I Shariv, G Koren, A A Friesem, and E Katchalski-Katzir. Modeling supra-molecular helices: extension of the molecular surface recognition algorithm and application to the protein coat of the tobacco mosaic virus. *Journal of Molecular Biology*, 266(1):135–143, 1997.

[89] K.M. ElSawy, A. Taormina, R. Twarock, and L. Vaughan. Dynamical implications of viral tiling theory. *Journal of Theoretical Biology*, (252):357–369, 2008.

[90] Juan Esquivel-Rodríguez, Yifeng David Yang, and Daisuke Kihara. Multi-lzerd: multiple protein docking for asymmetric complexes. *Proteins*, 80(7):1818–33, 2012.

[91] Narayanan Eswar, Ben Webb, Marc A Marti-Renom, M S Madhusudhan, David Eramian, Min-Yi Shen, Ursula Pieper, and Andrej Sali. Comparative protein structure modeling using MODELLER. *Current protocols in protein science*, Chapter 2:Unit 2.9, 2007.

[92] E. Eyal and D. Halperin. Dynamic maintenance of molecular surfaces under conformational changes. In *SCG '05: Proceedings of the 21st Annual Symposium on Computational Geometry*, pages 45–54, 2005.

[93] E. Eyal and D. Halperin. Improved maintenance of molecular surfaces using dynamic graph connectivity. *Algorithms in Bioinformatics*, pages 401–413,

2005.

[94] Jocelyne Fiaux, Eric B Bertelsen, Arthur L Horwich, and Kurt Wuthrich. Nmr analysis of a 900k groel groes complex. *Nature*, 418(6894):207–211, 2002.

[95] András Fiser and Andrej Sali. ModLoop: automated modeling of loops in protein structures. *Bioinformatics (Oxford, England)*, 19(18):2500–2501, 2003.

[96] P. W. Fowler and K. M. Rogers. Spiral codes and goldberg representations of icosahedral fullerenes and octahedral analogues. *Journal of Chemical Information and Computer Sciences*, 41(1):108–111, 2001.

[97] Joachim Frank. *Three-dimensional electron microscopy of macromolecular assemblies: visualization of biological molecules in their native state*. Oxford University Press US, 2006.

[98] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993.

[99] H Freudenthal and B. L. van der Waerden. Over een bewering van euclides ("on an assertion of euclid"). *Simon Stevin (in Dutch)*, 25:115–128, 1947.

[100] Joseph S. Fruton. *Proteins, Enzymes, Genes: The Interplay of Chemistry and Biology*. Yale University Press, 1999.

[101] H. A. Gabb, R. M. Jackson, and M. J. E. Sternberg. Modelling protein docking using shape complementarity,electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106–120, 1997.

[102] Henry A. Gabb, Richard M. Jackson, and Michael J. E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106–120, 1997.

[103] R. R. Gabdoulline and R. C. Wade. Analytically defined surfaces to analyze molecular interaction properties. *J. Mol. Graph.*, 14:341–353, 1996.

[104] J.I. Garzon, J.R. Lopez-Blanco, C. Pons, J. Kovacs, R. Abagyan, J. Fernandez-Recio, and P. Chacon. FRODOCK: a new approach for fast rotational protein-protein docking. *Bioinformatics*, 25(19):2544–2551, 2009.

[105] M. Gilson, M. Davis, B. Luty, and J.A. McCammon. Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation. *Journal of Physical Chemistry*, 97:3591–3600, 1993.

[106] Fabian Glaser, David M. Steinberg, Ilya A. Vakser, and Nir Ben-Tal. Residue frequencies and pairing preferences at protein-protein interfaces. *Proteins: Structure, Function, and Genetics*, 43:89–102, 2001.

[107] David Goldberg, Christopher Malon, and Marshall Bern. A global approach to automatic solution of jigsaw puzzles. In *Proceedings of the eighteenth annual symposium on Computational geometry*, SCG '02, pages 82–87, New York, NY, USA, 2002. ACM.

[108] Michael Goldberg. A class of multi-symmetric polyhedra. *Tohoku Mathematical Journal*, pages 104–108, 1937.

[109] Rafael Gozalbes, Laurence Simon, Nicolas Froloff, Eric Sartori, Claude Monteils, and Romuald Baudelle. Development and experimental validation of a docking strategy for the generation of kinase-targeted libraries. *Journal of Medicinal Chemistry*, 51(11):3124–3132, 2008.

[110] J. A. Grant, M. A. Gallardo, and B. T. Pickup. A fast method of molecular shape comparison: A simple application of a Gaussian description of molecular shape. *Journal of Computational Chemistry*, 17(14):1653–1666, 1996.

[111] J. A. Grant and B. Pickup. A Gaussian description of molecular shape. *Journal of Phys. Chem.*, 99:3503–3510, 1995.

[112] J.J. Gray. High-resolution protein-protein docking. *Current opinion in structural biology*, 16(2):183–193, 2006.

[113] Branko Grunbaum and G. Shepard. *Tilings and Patterns*. Dover Publications, 2013.

[114] Shengyin Gu, Patrice Koehl, Joel Hass, and Nina Amenta. Surface-histogram: A new shape descriptor for protein-protein docking. *Proteins: Structure, Function, and Bioinformatics*, 80(1):221–238, 2012.

[115] Yongjun Guan, Marzena Pazgier, Mohammad M Sajadi, Roberta Kamin-Lewis, Salma Al-Darmarki, Robin Flinko, Elena Lovo, Xueji Wu, James E

Robinson, Michael S Seaman, and et al. Diverse specificity and effector function among human antibodies to HIV-1 envelope glycoprotein epitopes exposed by CD4 binding. *PNAS*, 110(1):E69–78, 2013.

[116] Miklos Guttman, Natalie K. Garcia, Albert Cupo, Tsutomu Matsui, Jean-Philippe Julien, Rogier W. Sanders, Ian A. Wilson, John P. Moore, and Kelly K. Leeemail. CD4-induced activation in a soluble HIV-1 Env trimer. *Structure*, 22(7):974–984, 2014.

[117] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[118] M. F. Hagan and D. Chandler. Dynamic pathways for viral capsid assembly. *Biophysical Journal*, 91:42–54, 2006.

[119] D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. In *SCG '94: Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 113–122, 1994.

[120] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. Principles of docking: an overview of search algorithms and a guide to scoring functions. *Proteins: Structure, Function, and Bioinformatics*, 47(4):409–443, 2002.

[121] Yaser Hashem, Amedee des Georges, Jie Fu, Sarah N Buss, Fabrice Jossinet, Amy Jobe, Qin Zhang, Hstau Y Liao, Robert a Grassucci, Chandrajit Bajaj, and et al. High-resolution cryo-electron microscopy structure of the trypanosoma brucei ribosome. *Nature*, 494(7437):385–9, 2013.

[122] Alexander Heifetz, Ephraim Katchalski-Katzir, and Miriam Eisenstein. Electrostatics in protein–protein docking. *Protein Science*, 11(3):571–587, 2002.

[123] Charlotte Helgstrand, Sanjeev Munshi, John E Johnson, and Lars Liljas. The refined structure of nudaurelia capensis omega virus reveals control elements for a t = 4 capsid maturation. *Virology*, 318(1):192–203, 2004.

[124] Csaba Hetényi, Gábor Paragi, Uko Maran, Zoltán Timár, Mati Karelson, and Botond Penke. Combination of a modified scoring function with two-dimensional descriptors for calculation of binding affinities of bulky, flexible ligands to proteins. *Journal of the American Chemical Society*, 128(4):1233–1239, 2006.

[125] Guang Hu and Wen-Yuan Qiu. Extended goldberg polyhedra. *Communications in Mathematical and in Computer Chemistry*, 59:585–594, 2008.

[126] Chih-chin Huang, Miro Venturi, Shahzad Majeed, Michael J Moore, Sanjay Phogat, Mei-Yun Zhang, Dimiter S Dimitrov, Wayne A Hendrickson, James Robinson, Joseph Sodroski, and et al. Structural basis of tyrosine sulfation and VH-gene usage in antibodies that recognize the HIV type 1 coreceptor-binding site on gp120. *PNAS*, 101(9):2706–2711, 2004.

[127] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006.

[128] W. Huang and H. Liu. Optimized grid-based protein-protein docking as a global search tool followed by incorporating experimentally derivable restraints. *Proteins*, 80(3):691–702, 2012.

[129] Howook Hwang, Thom Vreven, Joel Janin, and Zhiping Weng. Protein-protein docking benchmark version 4.0. *Proteins: Structure, Function, and Bioinformatics*, 78(15):3111–3114, 2010.

[130] Yuval Inbar, Hadar Benyamini, Ruth Nussinov, and Haim J. Wolfson. Prediction of multimolecular assemblies by multiple docking. *Journal of Molecular Biology*, 349(2):435 – 447, 2005.

[131] Joël Janin, Kim Henrick, John Moult, Lynn T. Eyck, Michael J. E. Sternberg, Sandor Vajda, Ilya Vakser, and Shoshana J. Wodak. CAPRI: A Critical Assessment of PRedicted Interactions. *Proteins*, 52(1):2–9, 2003.

[132] Sunhwan Jo, Hui Sun Lee, Jeffrey Skolnick, and Wonpil Im. Restricted N-glycan conformational space in the PDB and its implication in glycan structure modeling. *PLoS Computational Biology*, 9(3):1–10, 2013.

[133] Susan Jones and Janet M. Thornton. Principles of protein-protein interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 93(1):13–20, 1996.

[134] Susan Jones and Janet M. Thornton. Analysis of protein-protein interaction sites using surface patches. *Journal of Molecular Biology*, 272(1):121–132, 1997.

[135] T A Jones and L Liljas. Structure of satellite tobacco necrosis virus after crystallographic refinement at 2.5 a resolution. *Journal of Molecular Biology*, 177(4):735–767, 1984.

[136] Jean-Philippe Julien, Albert Cupo, Devin Sok, Robyn L Stanfield, Dmitry Lyumkis, Marc C Deller, Per-Johan Klasse, Dennis R Burton, Rogier W Sanders, John P Moore, and et al. Crystal structure of a soluble cleaved HIV-1 envelope trimer. *Science*, 342(6165):1–12, 2013.

[137] Jean Philippe Julien, Devin Sok, Reza Khayat, Jeong Hyun Lee, Katie J. Doores, Laura M. Walker, Alejandra Ramos, Devan C. Diwanji, Robert Pejchal, Albert Cupo, and et al. Broadly neutralizing antibody PGT121 allosterically modulates cd4 binding via recognition of the HIV-1 gp120 V3 base and multiple surrounding glycans. *PLoS Pathogens*, 9(5):1–15, 2013.

[138] Abdullah Kahraman, Lars Malmström, and Ruedi Aebersold. Xwalk: computing and visualizing distances in cross-linking experiments. *Bioinformatics (Oxford, England)*, 27(15):2163–2164, 2011.

[139] Ezgi Karaca, Adrien S J Melquiond, Sjoerd J De Vries, Panagiotis L Kastritis, and Alexandre M J J Bonvin. Building macromolecular assemblies by information-driven docking: introducing the haddock multibody docking server. *Molecular cellular proteomics MCP*, 9(8):1784–1794, 2010.

[140] Gunilla B Karlsson Hedestam, Ron A M Fouchier, Sanjay Phogat, Dennis R Burton, Joseph Sodroski, and Richard T Wyatt. The challenges of eliciting

neutralizing antibodies to HIV-1 and to influenza virus. *Nature reviews. Microbiology*, 6(2):143–155, 2008.

[141] E Katchalski-Katzir, I Shariv, M Eisenstein, A A Friesem, C Aflalo, and I A Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, 1992.

[142] Ephraim Katchalski-Katzir, Isaac Shariv, Miriam Eisenstein, Asher A. Friesem, Claude Aflalo, and Ilya A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, 1992.

[143] T. Keef. *A Mathematical Approach to Cross-Linked Structures in Viral Capsids: Predicting the Architecture of Novel Containers for Drug Delivery*, volume 4287, pages 239–249. Springer, 2006.

[144] T. Keef, C. Micheletti, and R. Twarock. Master equation approach to the assembly of viral capsids. *Journal of Theoretical Biology*, (242):713–721, 2006.

[145] T. Keef and R. Twarock. Affine extensions of the icosahedral group with applications to the three-dimensional organisation of simple viruses. *Journal of Mathemaical Biology*, 3(59):287–313, 2009.

[146] J C KENDREW, G BODO, H M DINTZIS, R G PARRISH, H WYCKOFF, and D C PHILLIPS. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662–666, 1958.

[147] Reza Khayat, Jeong Hyun Lee, Jean-Philippe Julien, Albert Cupo, Per Johan Klasse, Rogier W Sanders, John P Moore, Ian a Wilson, and Andrew B Ward. Structural characterization of cleaved, soluble HIV-1 envelope glycoprotein trimers. *Journal of virology*, 87(17):9865–72, 2013.

[148] Per Johan Klasse, Rafael S Depetris, Robert Pejchal, Jean-Philippe Julien, Reza Khayat, Jeong Hyun Lee, Andre J Marozsan, Albert Cupo, Nicolette Cocco, Jacob Korzun, and et al. Influences on trimerization and aggregation of soluble, cleaved HIV-1 SOSIP envelope glycoprotein. *Journal of virology*, 87(17):9873–85, 2013.

[149] Leopold Kong, Jeong Hyun Lee, Katie J Doores, Charles D Murin, Jean-Philippe Julien, Ryan McBride, Yan Liu, Andre Marozsan, Albert Cupo, Per-Johan Klasse, Simon Hoffenberg, Michael Caulfield, C Richter King, Yuanzi Hua, Khoa M Le, Reza Khayat, Marc C Deller, Thomas Clayton, Henry Tien, Ten Feizi, Rogier W Sanders, James C Paulson, John P Moore, Robyn L Stanfield, Dennis R Burton, Andrew B Ward, and Ian A Wilson. Supersite of immune vulnerability on the glycosylated face of HIV-1 envelope glycoprotein gp120. *Nature Strucutral and Molecular Biology*, 20(7):796–803, 2013.

[150] Julio A. Kovacs, Pablo Chacon, Yao Cong, Essam Metwally, and Willy Wriggers. Fast rotational matching of rigid bodies by fast fourier transform acceleration of five degrees of freedom. *Acta Crystallographica, Biological Crystallography*, D59(8):1371–1376, 2003.

[151] D. Kozakov, R. Brenke, S.R. Comeau, and S. Vajda. PIPER: An FFT-based protein docking program with pairwise potentials. *Proteins: Structure, Function, and Bioinformatics*, 65(2):392–406, 2006.

[152] P D Kwong, R Wyatt, J Robinson, R W Sweet, J Sodroski, and W A Hendrickson. Structure of an HIV gp120 envelope glycoprotein in complex with the CD4 receptor and a neutralizing human antibody. *Nature*, 393(6686):648–659, 1998.

[153] P D Kwong, R Wyatt, J Robinson, R W Sweet, J Sodroski, and W A Hendrickson. Structure of an HIV gp120 envelope glycoprotein in complex with the CD4 receptor and a neutralizing human antibody. *Nature*, 393(6686):648–659, 1998.

[154] K. Lasker, A. Sali, and H. J. Wolfson. Determining macromolecular assembly structures by molecular docking and fitting into an electron density map. *Proteins: Structure, Function, and Bioinformatics*, 78:3205–3211, 2010.

[155] K Lasker, M Topf, A Sali, and H J Wolfson. Inferential optimization for simultaneous fitting of multiple components into a cryoem map of their assembly. *Journal of Molecular Biology*, 388(1):180–194, 2009.

[156] Keren Lasker, Friedrich Förster, Stefan Bohn, Thomas Walzthoeni, Elizabeth Villa, Pia Unverdorben, Florian Beck, Ruedi Aebersold, Andrej Sali, and Wolfgang Baumeister. Molecular architecture of the 26S proteasome holocomplex determined by an integrative approach. *PNAS*, 109(5):1380–7, 2012.

[157] B. Lee and F. Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379–400, 1971.

[158] Julian Lee, Dongseon Lee, Hahnbeom Park, Evangelos A Coutsias, and Chaok Seok. Protein loop modeling by using fragment assembly and analytical loop closure. *Proteins*, 78(16):3428–3436, 2010.

[159] Li Li, Rong Chen, and Zhiping Weng. Rdock: Refinement of rigid-body protein docking predictions. *Proteins: Structure, Function, and Genetics*, 53(3):693–707, 2003.

[160] R.C. Liddington, Y. Yan, J. Moulai, R. Sahli, T.L. Benjamin, and S.C. Harrison. Structure of simian virus 40 at 3.8-a resolution. *Nature*, 354:278–284, 1991.

[161] J. Liu, A. Bartesaghi, M. J. Borgnia, G. Sapiro, and S. Subramaniam. Molecular architecture of native HIV-1 gp120 trimers. *Nature*, 455(7209):109–113, 2008.

[162] Shiyong Liu and Ilya A. Vakser. DECK: Distance and environment-dependent, coarse-grained, knowledge-based potentials for protein-protein docking. *BMC*

Bioinformatics, 12(280), 2011.

[163] Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 3d polyomino puzzle. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 157:1–157:8, New York, NY, USA, 2009. ACM.

[164] Mark K. Louder, John A. Crump, Wayne C. Koff, Peter D. Kwong, Saidi H. Kapiga, Sanjay Phogat, Jeffrey C. Boyington, Barton F. Haynes, Andrew B. Ward, Mattia Bonsignori, and et al. Structure of HIV-1 gp120 V1/V2 domain with broadly neutralizing antibody PG9. *Nature*, 480(7377):336–343, 2011.

[165] K Lundstrom. Structural genomics for membrane proteins. *Cellular and molecular life sciences : CMLS*, 63(22):2597–2607, 2006.

[166] R Luthy, J U Bowie, and D Eisenberg. Assessment of protein models with three-dimensional profiles. *Nature*, 356(6364):83–85, 1992.

[167] Dmitry Lyumkis, Jean-Philippe Julien, Natalia de Val, Albert Cupo, Clinton S Potter, Per-Johan Klasse, Dennis R Burton, Rogier W Sanders, John P Moore, Bridget Carragher, and et al. Cryo-EM structure of a fully glycosylated soluble cleaved HIV-1 envelope trimer. *Science*, 1484, 2013.

[168] M. W. MacArthur, D. S. Moss, R. A. Laskowski, and J. M. Thornton. PROCHECK: a program to check the stereochemical quality of protein structures. *Journal of Applied Crystallography*, 26(2):283–291, 1993.

[169] R.M. MacCallum, A.C.R. Martin, and J.M. Thornton. Antibody-antigen interactions: contact analysis and binding site topography. *Journal of Molecular Biology*, 262(5):732–745, 1996.

[170] Jeffrey G. Mandell, Victoria A. Roberts, Michael E. Pique, Vladimir Kotlovyi, Julie C. Mitchell, Erik Nelson, Igor Tsigelny, and Lynn F. Ten Eyck. Protein docking using continuum electrostatics and geometric fit. *Protein Engineering*, 14(2):105–113, 2001.

[171] Ranjan V. Mannige and Charles L. Brooks. Geometric considerations in virus capsid size specificity, auxiliary requirements, and buckling. *Proceedings of the National Academy of Sciences*, May 2009.

[172] Ranjan V. Mannige, Charles L. Brooks, and III. Tilable nature of virus capsids and the role of topological constraints in natural capsid design. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 77(5):051902–9, 2008.

[173] R.V. Mannige and C.L. Brooks 3rd. Geometric considerations in virus capsid size specificity, auxiliary requirements, and buckling. *Proc Natl Acad Sci U S A*, 2009.

[174] A.C.R. Martin. Protein sequence and structure analysis of antibody variable domains. *Antibody engineering*, pages 33–51, 2010.

[175] Nelson L. Max and Elizabeth D. Getzoff. Spherical harmonic molecular surfaces. *IEEE Computer Graphics & Applications*, 8:42–50, 1988.

[176] Stefan Mayewski. A multibody, whole-residue potential for protein structures, with testing by monte carlo simulated annealing. *Proteins*, 59(2):152–169, 2005.

[177] Jason S McLellan, Marie Pancera, Chris Carrico, Jason Gorman, Jean-Philippe Julien, Reza Khayat, Robert Louder, Robert Pejchal, Mallika Sastry, Kaifan Dai, and et al. Structure of HIV-1 gp120 V1/V2 domain with broadly neutralizing antibody PG9. *Nature*, 480(7377):336–343, 2011.

[178] F Melo, D Devos, E Depiereux, and E Feytmans. ANOLEA: a www server to assess protein structures. *Int. Conf. Intelli. Sys. Mol. Biol.*, 5:187–190, 1997.

[179] J. Mintseris, B. Pierce, K. Wiehe, R. Anderson, R. Chen, and Z. Weng. Integrating statistical pair potentials into protein complex prediction. *Proteins: Structure, Function, and Bioinformatics*, 69(3):511–520, 2007.

[180] Julian Mintseris, Kevin Wiehe, Brian Pierce, Robert Anderson, Rong Chen, Joël Janin, and Zhiping Weng. Protein-protein docking benchmark 2.0: an update. *Proteins*, 60(2):214–6, 2003.

[181] Julie C. Mitchell. Personal Communication, University of Wisconsin - Madison.

[182] S Miyazawa and R L Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *Journal of Molecular Biology*, 256(3):623–644, 1996.

[183] Garrett M. Morris, David S. Goodsell, Ruth Huey, and Arthur J. Olson. Distributed automated docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. *Journal of Computer-Aided Molecular Design*, 10:293–304, 1996.

[184] C. W. Mortensen, R. Pagh, and M. Pătraçcu. On dynamic range reporting in one dimension. In *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 104–111, 2005.

[185] Atsushi Nakagawa, Naoyuki Miyazaki, Junichiro Taka, Hisashi Naitow, Akira Ogawa, Zui Fujimoto, Hiroshi Mizuno, Takahiko Higashi, Yasuo Watanabe, Toshihiro Omura, and et al. The atomic structure of rice dwarf virus reveals the self-assembly mechanism of component proteins. *Structure London England 1993*, 11(10):1227–1238, 2003.

[186] H. Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[187] Jorge Nocedal and Stephen Wright. *Numerical Optimization (2nd ed.)*. Berlin, 2nd edition, 2006.

[188] Marie Pancera, Shahzad Majeed, Yih-En Andrew Ban, Lei Chen, Chih-chin Huang, Leopold Kong, Young Do Kwon, Jonathan Stuckey, Tongqing Zhou, James E Robinson, and et al. Structure of HIV-1 gp120 with gp41-interactive region reveals layered envelope architecture and basis of conformational mobility. *PNAS*, 107(3):1166–1171, 2010.

[189] Marie Pancera, Syed Shahzad-Ul-Hussan, Nicole a Doria-Rose, Jason S McLellan, Robert T Bailer, Kaifan Dai, Sandra Loesgen, Mark K Louder, Ryan P Staupe, Yongping Yang, and et al. Structural basis for diverse N-glycan recognition by HIV-1-neutralizing V1-V2-directed antibody PG16. *Nature structural and molecular biology*, 20(7):804–13, 2013.

[190] Ralph Pantophlet and Dennis R Burton. Gp120: target for neutralizing HIV-1 antibodies. *Annual review of immunology*, 24:739–769, 2006.

[191] G. Papaioannou, E.A. Karabassi, and T. Theoharis. Virtual archaeologist: Assembling the past. In *IEEE Computer Graphics and Applications*, volume 21(2), pages 53–59, 2001.

[192] Georgios Papaioannou and Evaggelia-Aggeliki Karabassi. On the automatic assemblage of arbitrary broken solid artefacts. *Image and Vision Computing*, 21(5):401—412, 2003.

[193] J. Patera and R. Twarock. Affine extensions of noncrystallographic coxeter groups and quasicrystals. *Journal of Physics*, A35:1551–1574, 2002.

[194] G.S. Pawley. Plane groups on polyhedra. *Acta Crystallographica*, 15:49–53, 1961.

[195] Eric F. Pettersen, Thomas D. Goddard, Conrad C. Huang, Gregory S. Couch, Daniel M. Greenblatt, Elaine C. Meng, and Thomas E. Ferrin. UCSF Chimera–a visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25(13):1605–12, 2004.

[196] Tuan A Pham and Ajay N Jain. Parameter estimation for scoring protein-ligand interactions using negative training data. *Journal of Medicinal Chemistry*, 49(20):5856–5868, 2006.

[197] B. Pierce and Z. Weng. Structure prediction of protein complexes. *Computational Methods for Protein Structure Prediction and Modeling*, pages 109–134, 2007.

[198] Brian Pierce, Weiwei Tong, and Zhiping Weng. M-zdock: a grid-based approach for cn symmetric multimer docking. *Bioinformatics*, 21(8):1472–1478, 2005.

[199] Brian G Pierce, Yuichiro Hourai, and Zhiping Weng. Accelerating protein docking in ZDOCK using an advanced 3d convolution library. *PLoS ONE*, 6(9):e24657, 2011.

[200] M M Pierce, C S Raman, and B T Nall. Isothermal titration calorimetry of protein-protein interactions. *Methods (San Diego, Calif.)*, 19(2):213–221, 1999.

[201] Carles Pons, David Talavera, Xavier De La Cruz, Modesto Orozco, and Juan Fernandez-Recio. Scoring by intermolecular pairwise propensities of exposed residues (sipper): a new efficient potential for protein-protein docking. *Journal of Chemical Information and Modeling*, 51(2):370–377, 2011.

[202] Helmut Pottmann, Qixing Huang, Bailin Deng, Alexander Schiftner, Martin Kilian, Leonidas Guibas, and Johannes Wallner. Geodesic patterns. *ACM*

Trans. Graph., 29:43:1–43:10, July 2010.

[203] D. C. Rapaport, J. E. Johnson, and J. Skolnick. Supramolecular self-assembly: molecular dynamics modeling of polyhedral shell formation. *Computer Physics Communications*, 121-122:231–235, 1999.

[204] M. Rasheed and C. Bajaj. Characterization, enumeration and construction of *Almost-regular* polyhedra. Technical Report 14-22, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, Texas, USA.

[205] M. Rasheed and C. Bajaj. Predicting symmetric spherical shell assemblies. Technical Report 14-24, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, Texas, USA.

[206] M. Rasheed, R. Bettadapura, and C. Bajaj. Structure of HIV spike protein gp120 including all variable loops in complex with CD4 and 17b through computational modeling, fitting and validation. *Submitted to Structure*, 2014.

[207] M. Rasheed, A. Rand, and C. Bajaj. Maintaining flexible molecular surfaces using augmented dynamic octrees. Technical Report 14-23, Institute of Computational Engineering and Sciences, The University of Texas at Austin, Austin, Texas, USA.

[208] M. Rasheed, Q. Yuan, and C. Bajaj. Learning optimized scoring models for protein-protein docking. Technical Report 14-25, Institute of Computational

Engineering and Sciences, The University of Texas at Austin, Austin, Texas, USA.

[209] P. Rasmus and R. Flemming. Cuckoo hashing. *Journal of Algorithms*, 51(2), 2004.

[210] D. V. S. Ravikant and R. Elber. Pie - efficient filters and coarse grained potentials for unbound protein-protein docking. *Proteins*, 78:400–419, 2010.

[211] D V S Ravikant and Ron Elber. Pie-efficient filters and coarse grained potentials for unbound protein-protein docking. *Proteins*, 78(2):400–419, 2010.

[212] D V S Ravikant and Ron Elber. Pie-efficient filters and coarse grained potentials for unbound protein-protein docking. *Proteins*, 78(2):400–419, 2010.

[213] F. Richards. Areas, volumes, packing, and protein structure. *Ann. Rev. of Biophysics and Bioengineering*, 6:151–176, 1977.

[214] David W. Ritchie. *Parametric Protein Shape Recognition*. Phd thesis, Departments of Computer Science & Molecular and Cell Biology, University of Aberdeen, King's College, Aberdeen, UK, September 1998.

[215] David W. Ritchie and Graham J.L. Kemp. Protein docking using spherical polar Fourier correlations. *Proteins: Structure, Function, and Genetics*, 39(2):178–194, 2000.

[216] D.W. Ritchie. Recent progress and future directions in protein-protein docking. *Current Protein and Peptide Science*, 9(1):1–15, 2008.

[217] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-TASSER: a unified platform for automated protein structure and function prediction. *Nature Protocols*, 5(4):725–738, 2010.

[218] Daniel Russel, Keren Lasker, Ben Webb, Javier Velázquez-Muriel, Elina Tjioe, Dina Schneidman-Duhovny, Bret Peterson, and Andrej Sali. Putting the pieces together: integrative modeling platform software for structure determination of macromolecular assemblies. *PLoS Biology*, 10(1):e1001244, 2012.

[219] Mahmut Samil Sagiroglu and Aytul Ercil. A texture based matching approach for automated assembly of puzzles. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 03*, ICPR '06, pages 1036–1041, Washington, DC, USA, 2006. IEEE Computer Society.

[220] Rogier W. Sanders, Ronald Derking, Albert Cupo, Jean Philippe Julien, Anila Yasmeen, Natalia de Val, Helen J. Kim, Claudia Blattner, Alba Torrents de la Pena, Jacob Korzun, and et al. A next-generation cleaved, soluble HIV-1 Env trimer, BG505 SOSIP.664 gp140, expresses multiple epitopes for broadly neutralizing but not non-neutralizing antibodies. *PLoS Pathogens*, 9(9):1–20, 2013.

[221] M. Sanner, A. Olson, and J. Spehner. Fast and robust computation of molecular surfaces. In *Proceedings of the 11th Annual Symposium on Computational Geometry*, pages 406–407. ACM Press, 1995.

[222] M F Sanner, A J Olson, and J C Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, 1996.

[223] Louise Scharf, Johannes F. Scheid, Jeong Hyun Lee, Anthony P. West Jr., Courtney Chen, Han Gao, Priyanthi N.P. Gnanapragasam, Rene Mares, Michael S. Seaman, Andrew B. Ward, Michel C. Nussenzweig, and Pamela J. Bjorkman. Antibody 8ANC195 reveals a site of broad vulnerability on the HIV-1 envelope spike. *Cell Reports*, 7(3):785–795, 2014.

[224] Doris Schattschneider. *M.C. Escher, Visions of Symmetry*. Henry Holt and Company, 1992.

[225] Stan Schein and James Maurice Gayed. Fourth class of convex equilateral polyhedron with polyhedral symmetry related to fullerenes and viruses. *Proceedings of the National Academy of Sciences of the United States of America*, 111(8):2920–5, 2014.

[226] Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. Geometry-based flexible and symmetric protein docking. *Proteins*, 60(2):224–231, 2005.

[227] Dina Schneidman-Duhovny, Andrea Rossi, Agustin Avila-Sakar, Seung Joong Kim, Javier Velázquez-Muriel, Pavel Strop, Hong Liang, Kristin A Krukenberg, Maofu Liao, Ho Min Kim, and et al. A method for integrative structure determination of protein-protein complexes. *Bioinformatics Oxford England*, 28(24):3282–3289, 2012.

[228] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.3r1. August 2010.

[229] R. Schwartz, P.W. Shor, P.E.J. Prevelige, and B. Berger. Local rules simulation of the kinetics of virus capsid self-assembly. *Biophysics Journal*, 75:2626–2636, 1998.

[230] Torsten Schwede, Jürgen Kopp, Nicolas Guex, and Manuel C Peitsch. SWISS-MODEL: An automated protein homology-modeling server. *Nucleic acids research*, 31(13):3381–5, 2003.

[231] Peter Schwerdtfeger, Lukas Wirz, and James Avery. Program fullerene: A software package for constructing and analyzing structures of regular fullerenes. *Journal of Computational Chemistry*, 34(17):1508–1526, 2013.

[232] Markus H J Seifert. Optimizing the signal-to-noise ratio of scoring functions for protein–ligand docking. *Journal of Chemical Information and Modeling*, 48(3):602–612, 2008.

[233] M Senechal. *Quasicrystals and Geometry*. Cambridge University Press, 1996.

[234] Weilong Shang, Jie Liu, Jie Yang, Zhen Hu, and Xiancai Rao. Dengue virus-like particles: Construction and application. *Applied Microbiology and Biotechnology*, 94(1):39–46, 2012.

[235] M. Shatsky, R.J. Hall, S.E. Brenner, and R.M Glaeser. A method for the alignment of heterogeneous macromolecules from electron microscopy. *Journal of Structural Biology*, 166:67–78, 2008.

[236] M Shen. Statistical potential for assessment and prediction of protein structures. *Protein Science*, 15(11), 2006.

[237] G. C. Shephard. Convex polytopes with convex nets. *Mathetical Processdings of the Cambridge Philosophical Society*, 78:389–403, 1975.

[238] Jinjun Shi, Alexander R. Votruba, Omid C. Farokhzad, and Robert Langer. Nanotechnology in drug delivery and tissue engineering: From discovery to applications. *Nano Letters*, 10(9):3223–3230, 2010.

[239] M J Sippl. Recognition of errors in three-dimensional structures of proteins. *Proteins*, 17(4):355–362, 1993.

[240] M. Sitharam and M. Agbandje-McKenna. Modeling virus self-assembly pathways: avoiding dynamics using geometric constraint decomposition. *Journal of Computational Biology*, 6(13):1232–1265, 2006.

[241] Mark Thomas Smith, Anna K. Hawes, and Bradley Charles Bundy. Reengineering viruses and virus-like particles through chemical functionalization strategies. *Current Opinion in Biotechnology*, 24(4):620–626, 2013.

[242] Ryan Smith, Roderick E Hubbard, Daniel A Gschwend, Andrew R Leach, and Andrew C Good. Analysis and optimization of structure-based virtual

screening protocols. (3). new methods and old problems in scoring function design. *Journal of molecular graphics modelling*, 22(1):41–53, 2003.

[243] R Stanfield, E Cabezas, A Satterthwait, E Stura, A Profy, and I Wilson. Dual conformations for the HIV-1 gp120 V3 loop in complexes with different neutralizing fabs. *Structure (London, England : 1993)*, 7(2):131–142, 1999.

[244] N. F. Steinmetz, T. Lin, G. P. Lomonossoff, and J. E. Johnson. Structure-based engineering of an icosahedral virus for nanomedicine and nanotechnology. *Current Topics in Microbiology and Immunology*, 327:23–58, 2009.

[245] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *Journal of Americal Chemical Society*, 112:6127–6129, 1990.

[246] S Tanaka and H A Scheraga. Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. *Macromolecules*, 9(6):945–950, 1976.

[247] C. J F Ter Braak. A markov chain monte carlo version of the genetic algorithm differential evolution: Easy bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249, 2006.

[248] Reiji Teramoto and Hiroaki Fukunishi. Structure-based virtual screening with supervised consensus scoring: evaluation of pose prediction and enrichment factors. *Journal of Chemical Information and Modeling*, 48(4):747–754, 2008.

[249] Elina Tjioe, Keren Lasker, Ben Webb, Haim J Wolfson, and Andrej Sali. Multifit: a web server for fitting multiple protein structures into their electron microscopy density map. *Nucleic Acids Research*, 39(Web Server issue):W167–W170, 2011.

[250] H. Tjong and H. X. Zhou. GBr6: A parameterization-free, accurate, analytical generalized born method. *Journal of Physical Chemistry B*, 111(11):3055–3061, 2007.

[251] D Tobi and R Elber. Distance-dependent, pair potential for protein folding: results from linear optimization. *Proteins*, 41(1):40–46, 2000.

[252] Dror Tobi and Ivet Bahar. Optimal design of protein docking potentials: efficiency and limitations. *Proteins*, 62(4):970–981, 2006.

[253] M Topf, K Lasker, B Webb, H Wolfson, W Chiu, and A Sali. Protein structure fitting and refinement guided by cryo-em density. *Structure London England 1993*, 16(2):295–307, 2008.

[254] Erin E H Tran, Mario J. Borgnia, Oleg Kuybeda, David M. Schauder, Alberto Bartesaghi, Gabriel A. Frank, Guillermo Sapiro, Jacqueline L S Milne, and Sriram Subramaniam. Structural mechanism of trimeric HIV-1 envelope glycoprotein activation. *PLoS Pathogens*, 8(7):37, 2012.

[255] Charles W. Trigg. An infinite class of deltahedra. *Mathematics Magazine*, 51(1):55–57, 1978.

[256] R. Twarock. A tiling approach to virus capsid assembly explaining a structural puzzle in virology. *Journal of Theoretical Biology*, 226:477–482, 2004.

[257] R. Twarock. The architecture of viral capsids based on tiling theory. *Journal of Theoretical Medicine*, 6:87–90, 2005.

[258] R. Twarock. Mathematical virology: A mathematical physicist's approach to the protein stoichiometry and bonding structure of viral capsids. In *Proceedings of the 9th International Conference on Symmetry Methods in Physics*, pages 113–120, 2005.

[259] P. van der Schoot and R. Zandi. Kinetic theory of virus capsid assembly. *Physical Biology*, 4:296–304, 2007.

[260] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov, and et al. Charmm general force field: A force field for drug-like molecules compatible with the charmm all-atom additive biological force fields. *Journal of Computational Chemistry*, 31(4):671–690, 2010.

[261] A. Varshney, F. P. Brooks Jr., and W. V. Wright. Computing smooth molecular surfaces. *IEEE Computer Graphics Applications*, 14(5):19–25, 1994.

[262] Daven Vasishtan and Maya Topf. Scoring functions for cryoEM density fitting. *Journal of Structural Biology*, 174:333–343, 2011.

[263] Javier Velázquez-Muriel, Keren Lasker, Daniel Russel, Jeremy Phillips, Benjamin M Webb, Dina Schneidman-Duhovny, and Andrej Sali. Assembly of

macromolecular complexes by satisfaction of spatial restraints from electron microscopy images. *PNAS*, 109(46), 2012.

[264] R. Voorintholt, M. T. Kosters, G. Vegter, G. Vriend, and W. G. Hol. A very fast program for visualizing protein surfaces, channels and cavities. *Journal of Molecular Graphics*, 7(4):243–245, December 1989.

[265] J. A. Vrugt, C.J.F. ter Braak, C.G.H. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon. Accelerating markov chain monte carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, 10(3), 2009.

[266] Wenbo Wang, Jianhui Nie, Courtney Prochnow, Carolyn Truong, Zheng Jia, Suting Wang, Xiaojiang S Chen, and Youchun Wang. A systematic study of the N-glycosylation sites of HIV-1 envelope protein on infectivity and antibody-mediated neutralization. *Retrovirology*, 10:14, 2013.

[267] Benjamin Webb, Keren Lasker, Dina Schneidman-Duhovny, Elina Tjioe, Jeremy Phillips, Seung Joong Kim, Javier Velázquez-Muriel, Daniel Russel, and Andrej Sali. Modeling of proteins and their assemblies with the integrative modeling platform. *Methods In Molecular Biology Clifton Nj*, 781:377–397, 2011.

[268] Scott J. Weiner, Peter A. Kollman, David A. Case, U. Chandra Singh, Caterina Ghio, Guliano Alagona, Salvatore Profeta, and Paul Weiner. A new

force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society*, 106:765–784, 1984.

[269] J. Weiser, P. S. Shenkin, and W. C. Still. Fast, approximate algorithm for detection of solvent-inaccessible atoms. *Journal of Computational Chemistry*, 20(6):588–596, 1999.

[270] J. Weiser, P.S. Shenkin, and W.C. Still. Approximate atomic surfaces from linear combinations of pairwise overlaps (LCPO). *Journal of Computational Chemistry*, 20(2):217–230, 1999.

[271] J. Weiser, A. A. Weiser, P. S. Shenkin, and W. C. Still. Neighbor-list reduction: Optimization for computation of molecular van der Waals and solvent-accessible surface areas. *Journal of Computational Chemistry*, 19(7):797–808, 1998.

[272] Dan Willard. Log-logarithmic worst-case range queries are possible in space n. *Information Processing Letters*, 17(2):81–84, 1983.

[273] Sitao Wu, Jeffrey Skolnick, and Yang Zhang. Ab initio modeling of small proteins by iterative TASSER simulations. *BMC biology*, 5:17, 2007.

[274] K Wuthrich. The way to nmr structures of proteins. *Nature structural biology*, 8(11):923–925, 2001.

[275] X. Xia and Y. Xia. Gold nanocages as multifunctional materials for nanomedicine. *Front. Phys.*, 9(3):378–384, 2014.

[276] Jinrui Xu and Yang Zhang. How significant is a protein structure similarity with TM-score = 0.5? *Bioinformatics (Oxford, England)*, 26(7):889–895, 2010.

[277] B.X. Yan and Y.Q. Sun. Glycine residues provide flexibility for enzyme active sites. *Journal of Biological Chemistry*, 272(6):3190, 1997.

[278] Yifeng David Yang, Changsoon Park, and Daisuke Kihara. Threading without optimizing weighting factors for scoring function. *Proteins*, 73(3):581–596, 2008.

[279] T. You and D. Bashford. An analytical algorithm for the rapid determination of the solvent accessibility of points in a three-dimensional lattice around a solute molecule. *Journal of Computational Chemistry*, 16(6):743–757, 1995.

[280] Z. Yu, M. P. Jacobson, and R. A. Friesner. What role do surfaces play in GB models? a new-generation of surface-generalized Born model based on a novel Gaussian surface for biomolecules. *J. Comput. Chem.*, 27:72–89, 2006.

[281] Martin Zacharias. Scoring and refinement of predicted protein-protein complexes. In Martin Zacharias, editor, *Protein-protein complexes. Analysis, Modeling and Drug Design*, pages 236–271. Imperial College press, 2010.

[282] R. Zandi, D. Reguera, R.F. Bruinsma, W.M. Gelbart, and Rudnick J. Origin of icosahedral symmetry in viruses. In *Proceedings of the National Academy*

of Science of the USA, volume 101, pages 15556–15560, 2004.

[283] Chi Zhang, Song Liu, Hongyi Zhou, and Yaoqi Zhou. An accurate, residue-level, pair potential of mean force for folding and binding based on the distance-scaled, ideal-gas reference state. *Protein Science*, 13(2):400–411, 2004.

[284] Q. Zhang, M. Sanner, and A.J. Olson. Shape complementarity of protein–protein complexes at multiple resolutions. *Proteins: Structure, Function, and Bioinformatics*, 75(2):453–467, 2009.

[285] Qin Zhang, Radhakrishna Bettadapura, and Chandrajit Bajaj. Macromolecular structure modeling from 3dem using volrover 2.0. *Biopolymers*, 97(9):709–731, 2012.

[286] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer Aided Geometric Design*, 23:510–530, 2006.

[287] Yang Zhang and Jeffrey Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

[288] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces. In *Proceedings of the ACM symposium on Solid and physical modeling*, pages 297–302, 2007.

[289] Hongyi Zhou and Yaoqi Zhou. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Science*, 11(11):2714–2726, 2002.

[290] A. Zlotnick. To build a virus capsid: An equilibrium model of the self assembly of polyhedral protein complexes. *Journal of Molecular Biology*, 241:59–67, 1994.

[291] A. Zlotnick. Theoretical aspects of virus capsid assembly. *Journal of Molecular Recognition*, 18:479–490, 2005.

[292] A. Zlotnick, R. Aldrich, J. M. Johnson, P. Ceres, and M. J. Young. Mechanism of capsid assembly for an icosahedral plant virus. *Virology*, 277:450–456, 2000.

Vita

Md Muhibur Rasheed was born in Bangladesh, the son of Md Abdur Rasheed and Mahmuda Khatun. He completed his undergraduate in 2005 from Bangladesh University of Engineering and Technology (BUET) majoring in Computer Science and Engineering. After briefly working at Stochastic Logic where he developed software for financial instruments, he joined BUET as a lecturer. Muhibur completed his M.Sc. Engg. from BUET in 2008, and came to The University of Texas at Austin to start his PhD research, which culminates with this dissertation. Muhibur is married and the father of two wonderful children.

Permanent address: muhibur@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.