RNA 3D Structure Analysis and Validation, and Design Algorithms for Proteins and RNA

by

Swati Jain

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: __________________________

Approved:

Jane S. Richardson, Co-Supervisor

Bruce. R. Donald, Co-Supervisor

David C. Richardson

Jack Snoeyink

Jack Keene

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate Program in Computational Biology and Bioinformatics in the Graduate School of Duke University 2015
ABSTRACT

RNA 3D Structure Analysis and Validation, and Design Algorithms for Proteins and RNA

by

Swati Jain

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: ___________________
Approved: ___________________

______________________
Jane S. Richardson, Co-Supervisor

______________________
Bruce. R. Donald, Co-Supervisor

______________________
David C. Richardson

______________________
Jack Snoeyink

______________________
Jack Keene

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate Program in Computational Biology and Bioinformatics in the Graduate School of Duke University 2015
RNA, or ribonucleic acid, is an important biological macromolecule that is a critical part of a variety of cellular processes. The well known functions of RNA molecules include their role in the central dogma of molecular biology, acting as carriers of genetic information in the form of mRNAs, and then assisting in translation of that information to protein molecules as tRNAs and rRNAs. In recent years, many other kinds of non-coding RNAs have been found, like miRNAs and siRNAs, that are important for gene regulation. RNA molecules, called ribozymes, are also known to catalyze many biochemical reactions. Functions carried out by these recently discovered RNAs, coupled with the traditionally known functions of tRNAs, mRNAs, and rRNAs in the central dogma of molecular biology, make RNA molecules even more crucial and essential components in biology.

Most of the functions mentioned above are carried out by RNA molecules associating themselves with proteins to form Ribonucleoprotein (RNP) complexes, e.g. the ribosome or the splicesosome. RNA molecules also bind a variety of small molecules, such as metabolites, and their binding can turn on or off gene expression. These RNA-protein complexes and small molecules binding RNA are increasingly being recognized as potential therapeutic targets for drug design. The technique of computational structure-based rational design is a technique that has been successfully used for designing drugs and inhibitors for protein function, but its potential has not been tapped for design of RNA or RNA-protein complexes. Understanding the
features of RNA three dimensional structure and developing new and improved algorithms for protein and RNA design are both equally important for this purpose.

This document details my thesis work which is divided into two main parts. The first part consists of characterizing and analyzing RNA three dimensional structure, in order to develop new methods for RNA validation and refinement, and new tools for rebuilding currently solved RNA structures. I extended the previous work on assembling non-redundant datasets of high-quality RNA crystal structures (RNA09 and RNA11), and I analyzed the range of values occupied by the RNA backbone and base dihedral angles to improve methods for RNA structure correction, validation, and refinement in MolProbity and PHENIX. I rebuilt and corrected the pre-cleaved structure of the HDV ribozyme and parts of the 50S ribosomal subunit to demonstrate the potential use of new tools and techniques to improve RNA structures and help crystallographers to make better and correct biological interpretations. I also extended the previous work of characterizing RNA backbone conformers by the RNA Ontology Consortium (ROC) to define new conformers from those found in the larger RNA11 dataset, supplemented by ERRASER runs that optimize data points to add new conformers or improve cluster separation.

The second part of my thesis work consists of develops novel algorithms for structure-based protein design when interactions between certain distant residue pairs are neglected and the design problem is represented by a sparse residue interaction graph. I analyzed the sequence and energy differences caused by using sparse residue interaction graphs (using the protein design package OSPREY), and I collaborated to develop a novel branch-decomposition based dynamic programming algorithm, called BWM*, that returns the Global Minimum Energy Conformation (GMEC) for sparse residue interaction graphs much faster than the traditional A* search algorithm. As the final step, I used the results of my analysis of the RNA base dihedral angle and implemented RNA structural flexibility in OSPREY to enable
it to design RNA and RNA-protein interfaces.
## Contents

Abstract iv

List of Tables xii

List of Figures xiii

List of Abbreviations and Symbols xvi

Acknowledgements xviii

1 Introduction 1

2 Background: RNA Structure 3

2.1 RNA Structure Basics 3

2.1.1 RNA base 4

2.1.2 RNA Secondary and tertiary structure 5

2.1.3 RNA Backbone 7

2.2 RNA Structure Validation 10

2.2.1 MolProbity Structure Validation Server 12

2.2.2 RNA Specific Validation in Molprobity 16

2.3 RNA correction and refinement 18

2.3.1 Refinement with PHENIX 18

2.3.2 RNA Rotator 18

2.3.3 RNABC 19

2.3.4 RCrane 19
### 3 RNA Ribose Pucker and Base $\chi$ Angle: Analysis for structure validation and refinement

- Non-redundant high-quality dataset of RNA crystal structures
- RNA09 dataset
- RNA11 dataset
- Local residue-level filtering for data analysis

- Ribose Pucker
  - Backbone $\delta$ Dihedral Angle
  - $P_{\text{perp}}$ Distance

- Base orientation and $\chi$ Dihedral Angle
  - Sequence and Pucker Specific $\chi$ angle ranges
  - Changes to $\chi$ angle targets in PHENIX

### 4 RNA Structure Correction: 50S Ribosomal Subunit and HDV Ribozyme

- 50S Ribosomal Subunit Structure Correction
  - Why Correct Ribosome Structures
  - Structure Corrections

- Hepatitis Delta Virus Ribozyme Structure Correction
  - Rebuild of C75U-inhibited uncleaved structure
  - Impact of the rebuild on catalytic mechanism interpretations
  - Was the pre-cleaved conformation crystallized?

### 5 Revising the list of RNA backbone conformers

- Method
  - Data Filtering
  - Using ERASER
5.2 Updating existing clusters ........................................... 67
  5.2.1 Upgrading the wannabes to main consensus list ............... 67
  5.2.2 Updating existing clusters ..................................... 68
  5.2.3 Motif based backbone conformer correction .................. 70
5.3 Finding new backbone conformer clusters .......................... 71

6 Background: Computational Structure-based Protein Design and
  Algorithms ................................................................. 74
  6.1 General Background ............................................. 74
    6.1.1 GMEC vs. Ensemble-based design ............................ 77
    6.1.2 Provable vs. Heuristic Algorithms .......................... 78
  6.2 OSPREY details .................................................. 79
    6.2.1 Inputs to OSPREY ............................................ 80
    6.2.2 Dead-end elimination ....................................... 81
    6.2.3 A* search ................................................... 82
    6.2.4 Side-chain and Backbone Flexibility ....................... 84
    6.2.5 K* score ................................................... 85
  6.3 Protein Design with Sparse Residue Interaction Graph ........... 86
    6.3.1 Definitions related to Sparse Residue Interaction Graph ... 87

7 A Critical Analysis of Protein Design with Sparse Residue Interac-
  tion Graphs .............................................................. 90
  7.1 Energy Bounds .................................................... 91
  7.2 Computational Methods .......................................... 94
    7.2.1 Sparse A* .................................................. 94
    7.2.2 Computational Experiments .................................. 95
  7.3 Results ............................................................ 97
    7.3.1 Core Redesign ............................................... 99
7.3.2 Boundary Redesign .............................................. 99
7.3.3 Surface Redesign .............................................. 102
7.3.4 Large Long-range interactions neglected by distance cutoffs . 105
7.4 Novel use of Sparse Graphs and Ensemble-based provable algorithms 107

8 BWM and BWM*: Novel Algorithms for Protein Design with Sparse Residue Interaction Graphs in OSPREY 111

8.1 Definitions and Previous Work .................................. 113
8.1.1 Branch-Decomposition and Branch-Width ....................... 113
8.1.2 Initial GMEC-based BWM Algorithm ............................ 113

8.2 Improvements to the GMEC-based BWM Algorithm .............. 116
8.2.1 Energy calculation at a tree edge ................................. 117
8.2.2 The back-tracking procedure ................................... 119
8.2.3 Final Retrieval of the sparse GMEC .............................. 122
8.2.4 New Time Complexity of BWM Algorithm ....................... 124

8.3 Results .................................................................. 126

8.4 BWM*: Ensemble-based algorithm for sparse residue interaction graph 129
8.4.1 Algorithm: Preprocessing ....................................... 129
8.4.2 Algorithm: Enumeration ........................................ 131

9 Implementation of RNA structure flexibility in OSPREY 133

9.1 RNA Design Background ........................................... 133
9.1.1 RNA Structural Flexibility and Design .......................... 134

9.2 RNA $\chi$ angle flexibility in OSPREY ................................ 135

9.3 Testing the software .................................................. 136
9.3.1 Experiments ......................................................... 136
9.3.2 Results .............................................................. 139

10 Conclusions and Future Directions 142
List of Tables

3.1 Non-redundant dataset Statistics ................................................. 24
4.1 MolProbity (version 4.1) statistics for the 50S ribosomal subunit .... 40
4.2 MolProbity (version 4.1) statistics for the C75U-inhibited HDV ribozyme structure .............................................................. 44
4.3 Metal ion - oxygen atoms distances in HDV ribozyme active site ... 52
4.4 Different "pre-cleaved" structures of the HDV Ribozyme ................. 56
5.1 Wannabe Conformers promoted to main list ................................. 68
7.1 Sequence differences between GMECs for boundary designs .......... 101
7.2 Sequence differences between GMECs for surface designs ............ 103
7.3 Results of core, boundary, and surface design on same structure ... 106
7.4 Rank for Boundary Designs ....................................................... 108
7.5 Rank for Surface Designs ......................................................... 109
8.1 Run time comparison of A*, Sparse A*, and BWM ....................... 128
9.1 List of RNA $\chi$ angle Rotamers .............................................. 138
9.2 Rotamer Recovery for RNA residues designed .......................... 139
List of Figures

2.1 Overview of RNA structure ........................................... 4
2.2 Base-pairing edges of nitrogenous bases .......................... 6
2.3 Two ribose pucker conformations in RNA .......................... 8
2.4 RNA Backbone Suite ................................................. 9
2.5 RNA backbone atoms at different resolutions .................... 11
2.6 Base vs. backbone contacts in RNA structure ...................... 13
2.7 MolProbity structure validation visual markups .................. 15
2.8 Long and short Pperp distance. ..................................... 17
3.1 Resolution vs. Number of Files .................................... 25
3.2 Residue Distribution for RNA11 .................................... 26
3.3 Plot of the backbone $\delta$ dihedral angle ......................... 27
3.4 Plot of the backbone $\delta$ dihedral angle ......................... 28
3.5 Pperp distance vs. the $\delta$ dihedral angle ....................... 30
3.6 Pucker Specific $\chi$ angle ranges .................................. 31
3.7 Sequence-specific $\chi$ angle distribution for unfiltered and filtered data 32
3.8 Purine and Pyrimidine Pucker $\chi$ ................................. 33
3.9 Original $\chi$ angle parameters for refinement ..................... 34
3.10 New $\chi$ angle targets ............................................. 35
4.1 RNA Rotator correction of Suite 542 ............................. 39
4.2 RNA Rotator Correction of Suite 559 ............................. 39
9.1 Unrecovered Rotamer in 1CSL .......... 140
List of Abbreviations and Symbols

Symbols

Å  Angstrom (1X10^{-10} meter)
α  Backbone Alpha dihedral angle (O3’-P-O5’-C5’)
β  Backbone Beta dihedral angle (P-O5’-C5’-C4’)
γ  Backbone Gamma dihedral angle (O5’-C5’-C4’-C3’)
δ  Backbone Delta dihedral angle (C5’-C4’-C3’-O3’)
ε  Backbone Epsilon dihedral angle (C4’-C3’-O3’-P)
ζ  Backbone Zeta dihedral angle (C3’-O3’-P-O5’)
χ  Dihedral angle across glycosidic bond (O4’-C1’-N9-C4 for purines, O4’-C1’-N1-C2 for pyrimidines)
°  Angle Degree

Abbreviations

BWM  Branch-Width Minimization
DEE  Dead-end Elimination
DNA  DeoxyriboNucleic Acid
dsRNA double-stranded RNA
ERRASER Enumerative Real-space Refinement ASisted by Electron density under Rosetta
GMEC  Global Minimum Energy Conformation
H-bond Hydrogen Bond
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDV</td>
<td>Hepatitis Delta Virus</td>
</tr>
<tr>
<td>kb</td>
<td>kilo base</td>
</tr>
<tr>
<td>MD</td>
<td>Molecular Dynamics</td>
</tr>
<tr>
<td>miRNA</td>
<td>micro RNA</td>
</tr>
<tr>
<td>mRNA</td>
<td>messenger RNA</td>
</tr>
<tr>
<td>ncRNA</td>
<td>non-coding RNA</td>
</tr>
<tr>
<td>NDB</td>
<td>Nucleic acid Data Bank</td>
</tr>
<tr>
<td>NMR</td>
<td>Nuclear Magnetic Resonance</td>
</tr>
<tr>
<td>OSPREY</td>
<td>Open Source Protein REdesign for You</td>
</tr>
<tr>
<td>PDB</td>
<td>Protein Data Bank</td>
</tr>
<tr>
<td>PHENIX</td>
<td>Python-based Hierarchical Environment for Integrated Xtallography</td>
</tr>
<tr>
<td>RBD</td>
<td>RNA Binding Domain</td>
</tr>
<tr>
<td>RNA</td>
<td>RiboNucleic Acid</td>
</tr>
<tr>
<td>RNP</td>
<td>RiboNucleo-Protein Complex</td>
</tr>
<tr>
<td>rRNA</td>
<td>ribosomal RNA</td>
</tr>
<tr>
<td>siRNA</td>
<td>small interfering RNA</td>
</tr>
<tr>
<td>snRNA</td>
<td>small nuclear RNA</td>
</tr>
<tr>
<td>tRNA</td>
<td>transfer RNA</td>
</tr>
<tr>
<td>UTR</td>
<td>Un-Translated Region of mRNAs</td>
</tr>
<tr>
<td>vdW</td>
<td>van der Waals</td>
</tr>
<tr>
<td>WC</td>
<td>Watson-Crick</td>
</tr>
</tbody>
</table>
Acknowledgements

The journey of my thesis work and, in general, life in the United States for the past seven years would not have been so enjoyable and educational without a lot of people, who I will attempt to thank here for all that they have done for me.

I was really lucky to have a wonderful set of advisors in Jane and Dave Richardson, and Bruce Donald. All of them have been excellent mentors, encouraging me to find my own path and passion, and providing necessary guidance along the way. As I see my future working in the field of RNA structure-based design, it does not get any better than to get an opportunity to learn about structure in the Richardson lab, and about design algorithms in the Donald lab. Apart from the actual science, I have learned a lot from them about how to be a good scientist, and how to be a better person. Jane, Dave, and Bruce, I thank you for all your guidance and support.

I would like to thank my committee members, Jack Snoeyink and Jack Keene for keeping me on track with my research and for their valuable insights and suggestions. I would especially like to thank Jack Keene and his post-doc, Kyle Mansfield, who gave me an opportunity to do a wet-lab rotation in his lab. Anything and everything I know about biology and wet-lab experiments is because of them.

I would like to thank all past and present members of both the Richardson and the Donald lab for being helpful and supportive throughout the seven years, for long lab meetings, and conference trips. From the Richardson lab, I would especially like to thank Gary Kapral and Laura Murray for allowing me to follow them around and
teaching me about RNA structure, Jeff Headd for all the times I have troubled him by asking naive questions about PHENIX and refinement, and Steven Lewis for all the help with ERRASER and Rosetta. From the Donald lab, I would especially like to thank Kyle Roberts and Pablo Gainza who know anything and everything there is to know about protein design, Ivelin Georgiev for his collaboration with the BWM project, Anna Yershova for being my mentor during my rotation in the Donald lab, and Jonathan Jou for being an excellent collaborator, and handling all the necessities of paper submission and revision while I was busy working on my thesis.

I would like to thank Terry Oas, Dave Richardson, and Carol Richardson for being supportive and helping me out when I first came to the United States. My first few months here were a complete culture shock, and they made it a lot easier for me to get a grip on the situation. Special thanks to Terry for ordering a veggie pizza without mushrooms for the SBB seminars for the past seven years.

I would like to thank all my friends I made in Durham for all their support over the past years. I don’t think any of us would have made it this far if we did not have each other. I would like to thank all members of the charity organization I work with (Association for India’s Development) who gave me a reason to be passionate about something other than my work and research.

Last but not least, I would like to thank my family, especially my parents and my sister. I know it broke their hearts to send me so far away from themselves.

Thank you all very much!!
This thesis is divided into two parts.

The first part (Chapters 2-5) describes my work on to study and characterize RNA three-dimensional structure. Chapter 2 provides the essential background on RNA structure, MolProbity RNA structure validation techniques developed by Richardson lab, and RNA correction tools used in the subsequent chapters. Chapter 3 describes the hand-annotated, quality-conscious, non-redundant dataset of RNA crystal structures, RNA09 and RNA11, and using these datasets to study the range of backbone dihedral angle $\delta$, and base dihedral angle $\chi$. Chapter 4 describes my work on correcting and rebuilding modeling errors in the 2.4 Å structure of 50S ribosomal subunit (PDB ID: 3CC2) from H. marismortuii, and the 2.45 Å structure of the cis-acting pre-cleaved ribozyme (PDB ID: 1VC7) from Hepatitis Delta Virus. Chapter 5 provides details on extending the RNA Ontology Consortium study on RNA backbone conformers and using the larger RNA11 dataset to find new RNA backbone conformers.

The second part (Chapters 6-8) describes my work on using sparse residue interaction graphs for computational structural based protein design. Chapter 6 provides
background on computational structure-based protein design, OSPREY protein design package developed in the Donald lab, and how to generate sparse residue interaction graphs. Chapter 7 describes the effects of using sparse residue interaction graphs in protein design, and how they vary depending on how the sparse graph is generated and what residues are being designed. Chapter 8 describes an ensemble-based, dynamic programming algorithm for protein design with sparse residue interaction graph, its implementation in OSPREY, and its comparison with the A* search algorithm used for conformation enumeration in OSPREY. This work is in collaboration with a current and a former student in the Donald lab, Jonathan D. Jou and Ivelin Georgiev.

Finally, connecting the two parts together, Chapter 9 describes the implementation of RNA design and $\chi$ angle flexibility in OSPREY.
This chapter provides details of RNA 3 dimensional structure, and RNA structure validation and correction tools. Section 2.1 covers the basics of RNA structure, Section 2.2 talks about MolProbity structure validation server and its components, and Section 2.3 describes the various tools used for RNA structure correction. This chapter introduces terminology that is used in this thesis and serves as background for Chapters 3, 4, and 5. This chapter is partly adapted from Jain S, Richardson DC, Richardson JS. (2015) Computational Methods in RNA Structure Validation and Improvement. Book Chapter 7, Book: Structures of large RNA molecules and their complexes. Methods in Enzymology (Woodson and Allain) Book Series. In Press.

2.1 RNA Structure Basics

RNA molecule is a polymer chain that consists of individual monomers called nucleotides. Each nucleotide has three main components (Fig. 2.1): a phosphate group that links the different nucleotides together, the ribose sugar, and one of the 4 nitrogenous bases, adenine (A), guanine (G), uracil (U), or cytosine (C). The phosphate group is electron rich and negatively charged at physiological pH, leading to RNA
Figure 2.1: The base, ribose sugar, and phosphate group that makes up nucleotide \( i \). The bond connecting the ribose sugar to the base is called the glycosidic bond, and the dihedral angle across that bond is the \( \chi \) angle.

Chains being inherently negatively charged. An important aspect of the RNA backbone is that the ribose sugar has a O2’ oxygen atom, a feature that differentiates RNA molecules from DNA molecules where the ribose sugar is deoxygenated at 2’ position.

2.1.1 RNA base

Out of the 4 nitrogenous bases, adenine and guanine are purines, with two fused rings (residue on the left in Fig. 2.2), and uracil and cytosine are pyrimidines, with a single 6-membered ring (residue on the right in Fig. 2.2). Occasionally modified version of these four bases are also found in RNA molecules, and they often have specific functional importance. The three edges of a base, the Watson-Crick (WC) edge, the Hoogsteen edge, and the Sugar edge, are decorated with hydrogen-bond donors and acceptors that interact with other bases to form base pairs (Fig. 2.2). The strongest base-pairs are formed between guanine and cytosine (with 3 H-bonds), and adenine and uracil (thymine in DNA) (with 2 H-bonds) as the hydrogen-bond donor
and acceptor atoms on their respective WC edges are complimentary to each other. This complimentary base pairing that allows specific recognition of RNA (or DNA) bases is the fundamental concept that forms the basis of many (but certainly not all) biological functions of RNA molecules. Other base pair types are also found in RNA structures (Leontis and Westhof, 2001; Leontis et al., 2002), but are less common.

The nitrogenous base is connected to the ribose sugar via the glycosidic bond. The dihedral angle across the glycosidic bond is called the $\chi$ angle (O4'-C1'-N9-C4 for purines and O4'-C1'-N1-C2 for pyrimidines), and it determines the orientation of the base with respect to the backbone of the residue. The $\chi$ angle is known to adopt two main conformations, the anti conformation, where the bulky group of the bases (the six membered ring for purines and O group of the pyrimidines) points away from the ribose sugar; and the syn conformation where the bulky group points towards the sugar. The anti conformation is significantly more populated than the syn conformation. A variation of the anti conformation called the high-anti conformation is also observed in some structures (Saenger, 1984).

2.1.2 RNA Secondary and tertiary structure

Most biologically relevant RNA molecules are single stranded and can fold back on themselves to form complex three dimensional structures. Base pairs formed between complimentary bases provides the foundation for higher-order structure of a RNA molecule, and constitute its secondary structure. A series of continuous base pairs in a RNA strand can stack on each other to form an A-form helix, also known as a stem. The A-form helix has a narrower major groove and a wider minor groove than the B-form of a DNA double helix. DNA can adopt A-form, but RNA cannot adopt B-form, due to the presence of the O2' atom on the ribose sugar of RNA residues. On average, more than half of the residues in a RNA molecule form helical structures. In between the base-paired helical regions are other secondary structure
Figure 2.2: The three main edges of the nitrogenous bases highlighted in a GC Watson-Crick base pair. The hydrogen bonding heavy atoms highlighted by atom-colored balls.

elements like stem loops, bulges, and internal loops. When the base-pairing in two or more stems is not conventionally nested, the base pairs form a knot like structure called a pseudoknot. Pseudoknots are often found in functionally important regions and line binding pockets in RNA structures.

Three dimensional structure of RNA molecules with specific xyz coordinates for its atoms is known as its tertiary structure. Interactions between the secondary structure elements result in RNA molecules folding into complex and intricate three dimensional structures. Helices co-axially stacking on each other result in residues far apart in sequence coming closer and interacting with each other. Non-helical bases from different loops and bulges can base pair with each other, or can wedge into stacks of continuous nucleotides forming intercalation motifs. Base triples involving more
than two bases hydrogen bonding via multiple edges are also frequently observed in RNA structures (Abu Almakarem et al., 2012). A common example of a base triple is an A-minor motif, where unpaired adenine residues insert themselves into the minor groove of helices, forming hydrogen bonds with both bases and the backbone. The phosphate group and the O2’ atom on the backbone are capable of making hydrogen bonds and salt bridges that also contribute to tertiary interactions within an RNA molecule. Metals also play an important role in folding the RNA molecule, interacting with the negative charge on the RNA backbone and they also coordinate with oxygens on the bases.

2.1.3 RNA Backbone

The RNA backbone with the negatively charged phosphate group and the O2’ atom on the ribose ring forms a key component of RNA structure. Not only do interactions made by the backbone contribute to the overall fold of the RNA molecule, it is also important from a functional and biological perspective. A significant portion of RNA interactions with proteins and various ligands and drugs is made by backbone atoms. For instance, about 20% of all RNA-protein interactions are to the O2’ atom of the RNA backbone. RNA backbone is also involved in most types of RNA catalysis.

Ribose Pucker

The ribose sugar in RNA residues adopts one of the two main ribose pucker conformations, the C3’-endo or the C2’-endo (Fig. 2.3), with about 80% of RNA residues adopting the C3’-endo pucker conformation. The δ backbone dihedral angle (C5’-C4’-C3’-O3’) is known to be indicative of the ribose pucker, with the δ angle value being around 84° for C3’-endo pucker, and around 145° for C2’-endo pucker conformations.
RNA backbone conformers

The RNA backbone is highly flexible, with 6 degrees of freedom (dihedral angles $\alpha - \zeta$). The individual dihedral angles sample a large range of values, but when analyzed together, describe a discrete set of conformations that the RNA backbone can adopt. RNA backbone has been shown to be rotameric (Murray et al., 2003), a study that was extended by the RNA Ontology Consortium to describe 54 distinct conformations commonly adopted by the RNA backbone (Richardson et al., 2008). These backbone conformers are described in terms of sugar-to-sugar unit of the RNA backbone called a suite, because the dihedral angles within a suite are better correlated than the dihedral angles between phosphates of a traditional nucleotide. Each suite $i$ consists of 7 dihedral angles: $\delta$, $\epsilon$, and $\zeta$ from the hemi-nucleotide $i$-1, and $\alpha$, $\beta$, $\gamma$, and $\delta$ from the hemi-nucleotide $i$ (Fig. 2.4). The 54 backbone conformers were identified using the data from a quality-conscious, non-redundant dataset of RNA crystal structures (RNA05), and distinct clusters in 7 dimensions
of the suite were identified as an individual conformer. Each conformer has a two character name, a number describing the conformation of hemi-nucleotide i-1, and a letter describing the conformation of hemi-nucleotide i. For example, 1a represents the RNA backbone conformation in a standard A-form helix, 1[ is the most common intercalation conformation, 1g is the starting conformer for the GNRA tetraloop motif, and 5z and 4s are two of the three conformers that describe the backbone of an S-motif.

Using the two-character names, any three-dimensional RNA structure or a lo-
cal structural motif can now be represented as a string of backbone conformer names, called a *suitestring*. Many commonly occurring structural motifs have consensus suitestrings, for example kink-turns (7r6p2[0a]), S-motif (5z4s#a), GNRA tetraloops (1g1a1a1c). Non-1a backbone conformers are found in the regions of RNA molecules interacting with proteins and a variety of ligands and drugs. The backbone conformer representation of the RNA backbone is used for RNA structure validation (Section 2.2.2) and correction (Chapter 4). Details on existing backbone conformers are provided in Chapter 5, along with finding new ones.

2.2 RNA Structure Validation

With the importance of RNA molecules in biological functions coming to light, there is an increasing need to understand the structure-function relationship of RNA molecules. This has led to a recent surge in the determination of three-dimensional structures of RNA molecules, with more than half of the known RNA structures being solved after 2010, a trend that is likely to continue. The most common technique used for determination of RNA structures is X-ray crystallography. The most important measure of the quality of the X-ray structure is the *resolution* of the crystal structure, that gives an indication of how far the atoms have to be in three-dimensional space for them to be told apart in the electron density. A lower absolute value indicates higher-resolvability, therefore higher the resolution the better. However, determining good-quality high-resolution structures of RNA molecules, especially for the backbone with its large number of degrees of freedom, is a very difficult task. This is further complicated by the fact that RNA molecules almost always bind with protein to assemble into large RNP (ribonucleoprotein) complexes that are inherently difficult to crystalize. Therefore, it should not come as a surprise that only 85 crystal structures in the PDB that contain RNA residues are better than or at 1.5 Å resolution, with a majority of them being in the range 2.4-4 Å.
Figure 2.5: An example of a backbone conformer 1a from (a) PDB ID: 2Q1O at 1.1 Å, (b) PDB ID: 3CC2 at 2.4 Å, and (c) PDB ID: 3R8T at 3.0 Å resolution. The 2Fo-Fc density is shown at 1.8 $\sigma$ as grey mesh, and at 5.3 $\sigma$ as purple mesh.

The first step in building the initial RNA model into the electron density is the placement of helical segments that can be identified reliably at a wide range of resolutions, because the phosphates can be seen in the regular double spiral. Next steps are to connect these helical segments in sequence and to position individual atoms into the electron density. This task is relatively easy at resolutions where the position of each individual base and backbone atom can be clearly seen in the electron density (Fig. 2.5(a)). This becomes increasingly difficult as resolution decreases. However, even at a poorer resolution, the position of the flat nitrogenous base and the electron-rich phosphorus atom can be reliably identified in the electron density (although it may be difficult to judge the identity of the base). In contrast, the density for the ribose is a blob and a round tube for the atoms between the phosphorus and the ribose, with no way to tell the position of the individual atoms of the ribose or the zig-zag of the backbone (Figs. 2.5(b) and (c)). The bump for the O2’ atom that informs about the pucker of the ribose also tends to disappear at low resolutions. This disparity in the quality of the data for different components of the structures results in the conformations and interactions of the base being modeled more accurately than the backbone, which is often modeled with physically impossible conformations.
For cases where electron density is alone not sufficient to build an accurate and high-quality model, one has to rely on prior knowledge about the features and characteristics of RNA structure. Our only source of information, however, are the previously solved RNA structures that are themselves riddled with modeling errors. Structures can be filtered based on global measures like resolution and the agreement of the model-calculated and the observed diffraction data ($R$-factor and $R_{\text{free}}$ (Brunger, 1992)), but as noted earlier, not all regions of a structure are equally trustworthy. Therefore to avoid the vicious circle of inaccuracy feeding inaccuracy, techniques that will help to sieve out reliable and accurate information from all the available data are needed.

The above requirement is filled by macromolecular structure validation that identifies modeling errors at the level of individual atoms and residues that provides information on local model quality. It serves the two-fold purpose of effectively mining the already solved structures for trustworthy data, and of preventing modeling errors in new structures. The following section provides details on the MolProbity structure validation server developed in the Richardson lab, with focus on RNA structure related validation.

### 2.2.1 MolProbity Structure Validation Server

MolProbity is a macromolecular structure validation web server that is geared towards structure improvement and correction (Davis et al., 2007; Chen et al., 2010b). The web server carries out a thorough analysis of the structure to provide a residue-level error report, presented in a sortable multi-criterion chart. The visual indicators for the all-atom contacts and various outliers detected by MolProbity, shown in Fig. 2.7, highlight the favorable interactions made by atoms and the problematic areas in the structure that need improvement. The visual 3D markup of outliers can
Figure 2.6: All-atom contacts shown for residues from PDB ID: 3R8O. vdW contacts shown as blue and green dots, H-bonds as green-dotted pillows, steric clashes (overlap of $\geq 0.4$ Å) as hot pink spikes. (a) Base-Base contacts (b) Backbone-backbone contacts.

be viewed on the web and can also be downloaded as a multi-criterion kinemage file that can be viewed in the visualization software KiNG (Chen et al., 2009b).

The first step in the MolProbity structure validation pipeline is to add hydrogen atoms to the given macromolecule. Hydrogen atoms contain only a single electron, and that too is shared with their covalently-bonded heavy atom. Hence they do not diffract X-rays very well, and it is impossible to determine their position in crystal structures based on density alone (except in very high sub-atomic resolution structures). Because of these reasons, hydrogen atoms are often not considered during structure building and refinement, and their coordinates are not deposited in the PDB. However, they constitute more than half the atoms of any molecule, and contribute significantly to atomic interactions (vdW interactions and H-bonds). Therefore it is imperative to add hydrogens to the structures before carrying out any validation analysis. MolProbity uses an automated program called reduce to add hydrogen atoms to a given structure. The hydrogen atom is added according to geometry of the covalent bonds, and rotatable positions are optimized by considering atomic interactions like steric and H-bonds. The hydrogen atoms are added
at electron-cloud position for X-ray structures, and at nuclear positions for NMR structures (Deis et al., 2013). Apart from adding hydrogen atoms, Reduce is also capable of carrying out automated fix-ups for the side-chains of Asn, Gln, and His, by flipping the amide group of Asn and Gln, and the ring of His by 180°, in the case that they are fitted backwards.

Next step in the validation pipeline is carrying out all-atom contact analysis, which is done by a program called PROBE. A probe ball of radius 0.25 Å is rolled over the vdW surface of a given atom, called the source atom. If the probe touches any other atom’s surface (not including covalently bonded atoms), that atom is recorded as a potential contact called the target atom. If the source and the target’s surfaces do not overlap, the interaction is characterized as a good vdW contact and indicated as blue and green dots. If the surfaces overlap, the source and the target atom are checked for a potential H-bond donor acceptor pair. If the criterion is met, the H-bond is indicated as sea-green dots on the caps of the vdW spheres of the atoms that overlap. Non H-bond overlaps are indicated as yellow dots for overlap of < 0.4 Å, and as hot pink spikes for overlaps of ≥ 0.4 Å. The latter represent impossible steric clashes and are almost always a result of modeling errors. MolProbity assigns each structure a clashscore, calculated as the number of clashes per 1000 atoms, and a percentile for the clashscore calculated based on structures of similar resolution.

Apart from the above analysis, RNA and protein chains are also checked for optimal covalent-bond geometry. Any residue with bond-length or a bond-angle value deviating by more than 4 σ from the mean value is flagged as a geometry outlier. The bond-length outlier spirals and bond-angle outlier fans (Fig. 2.7) are colored red if they are too-long and blue if they are too-short. For RNA chains, the bond-length and bond-angle values used for this analysis are sugar pucker specific.

The protein backbone is analyzed to check if residues deviate from the φ, ψ distribution of the Ramachandran plot, and if the Cβ atom is positioned incorrectly
Figure 2.7: Visual markups for all-atom contacts and various outliers flagged by structure validation server MolProbity.

relative to the Cα atom (Lovell et al., 2003). Recently a functionality to report cis-peptides has been added that will also flag any peptides that deviate significantly from planarity (Williams and Richardson, 2015). Protein side chains are analyzed for rotamerlicity, and if any side-chain has χ dihedral angles that deviate significantly from ideal rotamer conformations (Lovell et al., 2000) are flagged as rotamer outliers. The protein chain is given a MolProbity score relative to other structures at similar resolutions taking into account the above mentioned validation criteria. A new method called Cα Based Low-resolution Annotation Method, or CaBLAM (Williams et al., 2013) has been developed for protein backbone geometry validation at poor resolution (lower than 2.5 Å). CaBLAM detects errors and gives recommendations
for probable protein secondary based on the more reliable Cα backbone trace.

### 2.2.2 RNA Specific Validation in Molprobity

As mentioned earlier, determining the ribose sugar pucker from the electron density alone becomes increasingly difficult at lower resolutions. This often results in crystallographers modeling RNA residues as C3'-endo pucker by default, as more than 80% of all RNA residues are C3'-endo. This results in a large number of incorrect sugar pucker conformations being found in RNA structures, usually accompanied by other modeling errors, like geometry outliers and steric clashes. MolProbity uses a simple and reliable way to detect the pucker of a RNA residue from the two components of RNA structures that can be relatively well placed in the electron density: the electron-dense phosphorus atom and the base. The glycosidic bond, that connects the usually well defined base to the thickened density of the ribose ring, is surprisingly well modeled. If a perpendicular is dropped from the 3’ phosphorus atom to the extended line of the glycosidic bond vector, the length of the perpendicular, called the *Pperp distance*, is highly correlated with the pucker of the ribose ring, long for C3'-endo pucker and short for C2'-endo pucker, as shown in Fig. 2.8. The Pperp distance cutoff used to distinguish between the two puckers is 2.9 Å, i.e. Pperp distance $\geq 2.9$ Å for C3'-endo pucker, and $< 2.9$ Å for C2'-endo pucker (Jain et al., 2014). The automated method for determining whether the modeled pucker of a RNA residue is correct or not is to calculate the Pperp distance and the backbone dihedral angle $\delta$ and to check if the Pperp distance and the $\delta$ angle indicate the same pucker conformation. If yes, the modeled pucker for that RNA residue is correct, otherwise the residue is flagged as a pucker outlier (magenta cross in Fig. 2.7), which should be corrected. In addition, the residues are also flagged if the $\epsilon$ dihedral angle value is outside the preferred range (155° – 310°).

For automated backbone conformer assignment in MolProbity, a program called
Suitename has been developed. Suitename takes input as the 7 backbone dihedral angles, assigns the suite one of the twelve $\delta - \gamma - \delta$ bins (C3’ or C2’ for $\delta$ angles and three categories of minus, plus, and trans for $\gamma$ angle) and then assigns a backbone conformer to the suite within that bin. It also provides a value called the suiteness for each suite (a number between 0 and 1, 0 for outliers), which indicates how far from the center of the backbone cluster this particular suite is. Any suite which does not belong to the 54 backbone conformers is flagged as a backbone conformer outlier, denoted by !!. In addition, if any of the seven dihedral angles of a suite is outside the preferred range of angle values listed in (Richardson et al., 2008), the suite is flagged as triaged.
2.3 RNA correction and refinement

Detecting modeling errors and validating RNA structures is important, but it is most useful when the error detection is coupled with error correction. The following section describes the various tools developed to correct, rebuild, and refine RNA structures that I have used in Chapters 4 and 5.

2.3.1 Refinement with PHENIX

PHENIX or Python-based Hierarchical ENivornment for Integrated Xtallography is a macromolecule structure building and refinement software that is widely used for RNA structure refinement (Adams et al., 2010, 2011). It incorporates MolProbity validation tools into the refinement pipeline and is integrated with structure correction and visualization tools like Coot and KiNG. This provides the user with the capability of validating their structure, rebuilding and correcting it if necessary, and feeding it right back into refinement. PHENIX uses the Pperp test (described in Section 2.2.2) to determine the pucker for RNA residues, and accordingly sets the pucker-specific target values for refinement of bond lengths, bond angles, and dihedral angles.

2.3.2 RNA Rotator

RNA Rotator is a tool available in the visualization software KiNG (Chen et al., 2009b), for manual correction of errors in the RNA structure. It allows the user to select a suite of interest and manually edit all the 7 backbone dihedral angles and the two $\chi$ angles associated with the suite. It also allows the user to choose from a list of backbone conformers to replace the selected suite with the ideal suite. Further, the tool also provides the suiteness value of the suite as well as all-atom contact analysis dots in real time as the user is changing the dihedral angles, providing essential feedback about the quality of the refit. The atoms on which the molten suite is
kept superimposed can be adjusted, which helps in maintaining the covalent bond connections and also allows the user to superimpose on the parts of the structure that are correct or trustworthy (e.g. the base or the phosphorus atom). The electron density can be visualized to guide the fitting process. This is a powerful tool to gain insights and build a certain level of familiarity with the RNA structures, while correcting the errors.

2.3.3 RNABC

RNABC is an automated tool developed to correct errors in RNA structures (Wang et al., 2008). It keeps the base and the phosphate position fixed and tries to build the rest of the backbone of a specified dinucleotide using forward kinematics. The pucker of two ribose rings in the dinucleotide is either specified by the user or is determined by the Pperp test (described in Section 2.2.2). During the rebuilding stage, all possible conformations are scored based on steric-clashes, pucker, and geometry terms. The program outputs clusters of possible alternate conformations for the user to choose from. However, RNABC does not take fit to the electron density or the known RNA backbone conformers into account during the rebuilding process. The program is often able to correct pucker outliers and the associated steric clashes.

2.3.4 RCrane

RCrane is a plugin, available in the structure refinement package Coot (Emsley et al., 2010) for semi-automated building of RNA structure into electron density (Keating and Pyle, 2012). It is partially based on the RNA backbone parameters $\eta'$ (P-C1'-P-C1') and $\theta'$ (C1'-P-C1'-P), that are modified version of the two pseudo dihedral angles $\eta$ and $\theta$ used to describe RNA backbone conformations in a coarse-grained approach (Duarte and Pyle, 1998). These dihedrals are shown to be related to the all-atom RNA backbone representation as described by backbone suite conformers,
though the relationship is not one-to-one (Keating and Pyle, 2010). The tool takes as input the electron density, and asks the user to construct a trace for the nucleotides to be built. This is done by calculating the highest intensity peaks in the electron density and allowing the user to choose the position of the phosphorus atom and the C1’ atom from these density peaks (within 10 Å), and adjust them (by rotation and translation) if necessary. After the initial trace is built, the P and C1’ positions are fixed, and automatic selection of the backbone conformers (based on $\eta'$, $\theta'$, $P_{\text{perp}}$, C1’-C1’ and P-P distances) is done, followed by individual coordinate minimization, taking into account electron density. The user is then presented with the list of possible conformations (along with their scores) to choose from for each built nucleotide. This technique can also be used for correction of errors in the RNA structure, given the input model and the electron density, as selected nucleotides can be rebuilt.

2.3.5 ERRASER

All the correction and rebuilding tools described above require some input from the user, either in identifying errors or in the actual correction or building process. A new tool called Enumerative Real-space Refinement ASisted by Electron density under Rosetta, or ERRASER (Chou et al., 2012a,b) has been developed for completely automated RNA structure correction. It utilizes capabilities in PHENIX and MolPro-bity for identification of modeling errors, and a step-wise assembly (SWA) procedure to re-build each residue by enumerating many conformations covering all build-up paths, taking into account the fit of the model to the electron density (Das and Baker, 2007; Mandell et al., 2009). ERRASER can be used to rebuild whole RNA structures (where error detection is done automatically), or single residues as specified by the user, in which case ERRASER returns its top 10 distinct conformations for the user to choose from. The rebuilding process in ERRASER consists of three steps: First, ERRASER minimizes all torsion angles and all backbone bond-lengths and bond-angles
using the Rosetta energy function, which includes an electron density correlation score. Second, PHENIXs MolProbity-style RNA validation tools are used to identify errors (geometry, pucker, and unrecognized backbone conformations) in the minimized model. These residues, as well as residues with large rms deviation (>2 Å) between their original position and the minimized position, are identified as residues to be rebuilt. The second step is skipped if the user has specified a particular residue to be rebuilt. Third, the residues from step two are rebuilt one at a time, and then minimized again. This tool has proven to be most effective in automatic detection and error correction in RNA structures.
This chapter provides details on my work on assembling non-redundant dataset of RNA crystal structure and its use for RNA structure refinement and validation in PHENIX and MolProbity. Section 3.1 provides details on assembling the RNA09 and RNA11 non-redundant datasets, Section 3.2 describes the analysis of the ribose pucker that led to a change in parameters used in MolProbity for detecting ribose pucker outliers, and Section 3.3.1 describes the analysis of the base $\chi$ angle that led to a change in the target parameters for $\chi$ angle in PHENIX.

3.1 Non-redundant high-quality dataset of RNA crystal structures

As discussed in Chapter 2, deriving useful information from the existing structures of RNA molecules is important to build accurate and reliable models. It is imperative to make sure that the information derived be accurate and not biased by either modeling errors or towards RNA molecules for which a larger number (as compared to other RNA molecules) of structures are available, e.g. RNA helices and ribosomal subunits. Therefore a dataset needs to be constructed that is quality-conscious and
non-redundant to represent different types of RNA molecules and interesting and non-ideal structures.

Richardson lab had earlier assembled two datasets of RNA crystal structures, *RNA03* (Murray et al., 2003) and *RNA05* (Richardson et al., 2008), that were used to develop methods for validation of the ribose pucker and defining the RNA backbone conformers. Growing interest in RNA structural biology and improvement in structure determination techniques have led to a recent acceleration in solving RNA structures. Therefore there is a constant need for updating the RNA structure datasets to include, and take advantage of these new structures. The following sections describe my work on the same.

### 3.1.1 RNA09 dataset

Laura Murray, a former student in the Richardson lab, began the initial task of updating the RNA05 dataset, the work that I completed. Crystal structures containing RNA residues, with a resolution of 3 Å or better, and released on or before December 31st, 2009 at the Protein Data Bank (PDB) were considered as candidates for the high-quality RNA structure dataset. The structures were divided into functional and structural categories based on their RNA and protein molecules; these categories include ribosomal subunits, tRNAs, mRNAs, aptamers and riboswitches, ribozymes, different RNA binding proteins, siRNAs and miRNAs, viral RNAs, and duplexes and multiplexes. Within each category, the structures were selected based on their resolution, validation statistics generated by MolProbity 3.2 (Davis et al., 2007; Chen et al., 2010b), and number of RNA residues. Apo RNA structures were preferred over structures bound to protein or ligand (in case all other scores were similar). In structures with multiple copies in the asymmetric unit, the chain selection was made depending on the validation statistics of each individual chain. The final *RNA09* dataset consisted of 277 files, with close to 19,000 RNA residues (Table 3.1). The final list
of PDB files, along with chain IDs, divided into different structural and functional categories can be downloaded from http://kinemage.biochem.duke.edu/rnadb/.

Table 3.1: The number of files and number of RNA residues in the three datasets RNA05, RNA09, and RNA11

<table>
<thead>
<tr>
<th></th>
<th>RNA05</th>
<th>RNA09</th>
<th>RNA11</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of files</td>
<td>171</td>
<td>277</td>
<td>311</td>
</tr>
<tr>
<td>No. of residues</td>
<td>9486</td>
<td>18546</td>
<td>24856</td>
</tr>
</tbody>
</table>

3.1.2 RNA11 dataset

To further update RNA09, I considered crystal structures containing RNA residues, released on or before November 11\textsuperscript{th}, 2011 at the PDB. The resolution cutoff was still kept at 3 Å. The selection process was the same as described for RNA09, and mainly consisted of replacing older structures with newer higher resolutions ones. In addition, some new structures were added for which either structures had not been available, or had not met the 3 Å resolution cutoff. This included the 70S ribosomal subunits from \textit{E.coli}, a RNA nano square, and RNA residues in complex with CRISPR endonuclease, pseudo-U synthetase, or dead-box proteins. The resulting RNA11 dataset consists of 311 PDB files, and contains about 2.5 times the number of residues in RNA05 and 1.5 times more than RNA09 (Table 3.1).

3.1.3 Local residue-level filtering for data analysis

Fig. 3.1 compares the resolution distribution of crystal structures in the RNA09 and RNA11 datasets, and Fig. 3.2 shows the percentage of residues in the RNA11 dataset from different types of RNA structures. There is an especially useful expansion of the RNA11 dataset at 1.5-2 Å resolution range. However, a majority of the structures still are within the 2 - 3 Å range, and about 60% of the residues in the dataset come from ribosomal subunits, that are also from structures with resolution poorer than
2.4 Å. As mentioned in Chapter 2, this is the resolution range where modeling errors begin to creep up, making the analysis of the data prone to errors. However, strict filtering at the structure level (based on resolution, R, and \( R_{\text{free}} \)) does not leave enough data for the analysis. Therefore to achieve a balance between the amount of data available and the quality and accuracy of data, filtering out data at the residue-level is required.

![Figure 3.1: Number of files for each resolution bin in the RNA09 and RNA11 dataset.](image)

For crucial quality filtering at the residue-level for data analysis with the RNA09 and RNA11 datasets, hydrogens were added and optimized in the full PDB files using Reduce (Word et al., 1999a). The hydrogen atoms were added at the nuclear positions for files in RNA09, and at electron-cloud positions for files in RNA11 (Deis et al., 2013). Subsequently, all non-RNA chains, waters, ligands, and metals were removed, along with additional copies of the RNA molecule. DNA residues within the RNA
chain were treated as modified bases. To get residue-level validation information, the following programs were run on each PDB file: Probe (Word et al., 1999b) for all-atom contact analysis like steric-clashes and hydrogen-bonds, and Dangle to calculate the dihedral angles and Pperp distance (see Section 3.2.2).

3.2 Ribose Pucker

3.2.1 Backbone $\delta$ Dihedral Angle

The $\delta$ backbone dihedral angle (C5’-C4’-C3’-O3’) is the angle that encodes the pucker of the ribose ring (Section 2.1.3). The ranges of $\delta$ angle values previously allowed for the two pucker conformations, as determined using the RNA05 dataset, were $55^\circ - 110^\circ$ for C3’-endo pucker and $120^\circ - 175^\circ$ for C2’-endo pucker (Richardson et al., 2008). Taking advantage of the larger size of the RNA09 dataset, we wanted to revisit the range of allowed $\delta$ angle values for potential revision. Because $\delta$ is a
Figure 3.3: Backbone dihedral angle $\delta$ for 7292 residues from the RNA09 dataset. The $\delta$ angle value was rounded to the nearest integer to count the number of residues with a particular $\delta$ angle value.

backbone dihedral angle, we filtered the residues based on the backbone atom steric clashes and B factors: residues with any backbone atom clashing with any other RNA atom or with a B factor $\geq 60$ were filtered out, along with any modified bases. 7296 residues remain after this step. Fig 3.3 plots the number of residues at each $\delta$ angle value. Only 4 of the $\delta$ angle values lie outside the $50^\circ - 170^\circ$ range shown in the figure. These 4 residues were not considered for further analysis since they all contain bond-length, bond-angle, or chirality outliers.

As expected, the $\delta$ angle plot is clearly bimodal with distinct ranges for C3'-endo and C2'-endo pucker and a clear minimum in between. To determine the range of $\delta$ values for each pucker conformation, we looked at specific examples that lie in the tail of the two peaks. The two residues in the range $50^\circ - 60^\circ$ ($\delta$ values at 52.2$^\circ$ and 58.3$^\circ$) were both geometry outliers and the modeled ribose sugars were almost flat. The two residues in the range $165^\circ - 170^\circ$ ($\delta$ values at 167.8$^\circ$, and 168.1$^\circ$) are all modeled as C3'-exo pucker conformation (presumably incorrectly), are mostly geometry outliers, and the resolution of the structures is not high enough for the electron density to
Figure 3.4: Backbone dihedral angle $\delta$ for 10304 residues from the RNA11 dataset. The $\delta$ angle value was rounded to the nearest integer to count the number of residues with a particular $\delta$ angle value.

support the unusual conformation. Looking at the examples from the range of values $105^\circ - 125^\circ$, there are a total of 51 examples. Almost all examples are either modeled as O4’-endo or C1’-exo conformations or else the ribose rings are almost flat; most are also geometry outliers. This includes the 7 residues that do come from high resolution structures (better than 2 Å), and in all but one case (PDB ID: 1FUF, 1.7 Å, residue B 20), either the electron density at that particular residue is not sufficient to support the unusual conformation, or the fit of the model to the density can be improved by a rebuild to a more standard conformation. Overall, this range of $\delta$ angle values did not contain any clear examples of C3’-endo or C2’-endo pucker. Therefore we defined the allowed range of $\delta$ angle values to be $60^\circ - 105^\circ$ for C3’-endo pucker and $125^\circ - 165^\circ$ for C2’-endo pucker. Out of 7296 quality-filtered residues, the percentages in the new C3’-endo and C2’-endo $\delta$ angle ranges are 88.43% and 10.76% respectively, with only 0.88% lying outside the allowed ranges.

We repeated the analysis with the RNA11 dataset. As the number of residues in the RNA11 dataset are more than in RNA09, we applied an additional residue-level
filtering, taking out residues that had any backbone atoms with a bond-length or a bond-angle more than $4 \sigma$ away from the mean values. Fig. 3.4 plots the $\delta$ values for the 10307 residues left after the filtering process. There were only three residues with $\delta$ values $< 60^\circ$ or $> 165^\circ$, and the number of residues in between $105^\circ - 125^\circ$ were reduced to 28. The percentages of residues in the C3'-endo and C2'-endo $\delta$ angle ranges increased to 88.95% and 10.76% respectively, with only 0.3% lying outside the allowed ranges. Therefore the RNA11 dataset also supports the new allowed $\delta$ angle ranges.

3.2.2 $P_{\text{perp}}$ Distance

Using the larger RNA11 dataset, we revisited the empirical correlation between the $P_{\text{perp}}$ distance and the $\delta$ dihedral angle, and how it changes with additional residue-level filtering. Fig. 3.5 shows the $P_{\text{perp}}$ distance for standard RNA residues (excluding any modified bases) with the $\delta$ dihedral angle in the C3'-endo pucker and C2'-endo pucker ranges (described in Section 3.2.1), as the following residue-level filters are applied sequentially: 1) residue with any backbone atom clashing with any other RNA atom, or residue with $\epsilon$ backbone dihedral angle outside the allowed range ($155^\circ - 310^\circ$), 2) residue with any backbone atom with B factor $\geq 60$, and 3) residues with bond-angle and bond-length outliers. What can be seen from the figure is that the correlation of the $P_{\text{perp}}$ distance with the $\delta$ angle improves as additional residue-level filters are applied. When all three filters are applied simultaneously (Fig. 3.5(d)), there are no residues with $\delta$ angle in the C2'-endo range and $P_{\text{perp}}$ distance $\geq 2.9$ Å, and only 5 points with $\delta$ angle in the C3'-endo range and $P_{\text{perp}}$ distance $< 2.9$ Å. None of these 5 points are modeled as C3'-endo puckers. Hence 2.9 Å remains a good cutoff to distinguish between the two pucker conformations.
3.3 Base orientation and $\chi$ Dihedral Angle

The definitions and rules regarding the $\chi$ angle values for the anti and the syn conformation are unclear in the literature. The IUPAC nomenclature describes the range of $\chi$ angles as anti: $180^\circ \pm 90^\circ$ and syn: $0^\circ \pm 90^\circ$, and this is consistent with the accepted chemical nomenclature for defining torsion angle ranges. However, according to this notation, the high-anti conformation ($\chi$ angle near $270^\circ$) partly...
Figure 3.6: Pucker-specific \( \chi \) angle distribution. Here and in the later figure, the \( \chi \) angle was rounded off to the nearest integer to count the total number of residues with a particular \( \chi \) value. The radius of the ball at each \( \chi \) value represents the number of residues in log scale. (a) 7091 residues from the RNA09 dataset: 6197 C3'-endo residues and 711 C2'-endo residues. (b) 9930 residues from the RNA11 dataset: 8,830 C3'-endo residues and 1,077 C2'-endo residues.

falls under the syn conformation range.

3.3.1 Sequence and Pucker Specific \( \chi \) angle ranges

For studying the base orientation and the range of \( \chi \) angle values using the RNA09 and RNA11 datasets to analyze its dependence on base type and ribose pucker, we applied residue-level filters on the pucker and the base atoms. Residues with 1) any base atom clashing with any other RNA atom, 2) any base atom with B factor \( \geq 40 \), or 3) a ribose pucker outlier (according to the Pperp test described in Section 3.2.2) were filtered out, along with any modified bases. We used a stricter cutoff for the B factor for base atoms than for backbone atoms (40 vs. 60), as the base is less flexible and can be placed into the electron density more easily than the individual atoms of the RNA backbone. The number of residues left after applying the above filters was 9,930. Fig. 3.6 shows the \( \chi \) angle distribution for residues with
a C3'-endo pucker (orange) and C2'-endo pucker (green). The figure clearly shows narrowing of the major peaks and the high-anti peaks between the two puckers, with the number of residues in high-anti and syn conformations more for C2'-endo pucker than C3'-endo pucker. Separating the residues into purines (Fig. 3.8(a)) and pyrimidines (Fig. 3.8(b)) shows that the syn and the high-anti conformations are extremely rare for pyrimidines, with only 3 high-resolution (< 2 Å) examples for each of those conformations without modeling errors.

Overall, purines populate a larger region of the $\chi$ angle space than pyrimidines (Fig. 3.7(a)). This seems counter-intuitive because purine bases are larger than pyrimidine bases. However, the smaller 5-membered ring is closer to the RNA backbone in case of purines, as opposed to the larger 6-membered ring in pyrimidines. Therefore one reason for pyrimidines to populate a smaller region of the $\chi$ angle space could be the backbone sterically hindering the possible base orientations. Indeed, if the $\chi$ angle distribution is plotted without applying any residue-level filters (Fig. 3.7(b)), both purine and pyrimidine bases occupy almost the entire $\chi$ angle space.
Figure 3.8: Pucker-specific $\chi$ angle distribution for 9,930 residues divided based on base-type. (a) Only purines: 4,909 C3'-endo residues and 665 C2'-endo residues. (b) Only pyrimidines: 3,944 C3'-endo residues and 412 C2'-endo residues.

space, indicating that certain regions are only populated in the unfiltered data. This also highlights the importance of applying residue-level filters on the data for such an analysis, to make sure we are looking at actual trends, rather than being thrown off by artifacts or decoys.

3.3.2 Changes to $\chi$ angle targets in PHENIX

Fig. 3.9 shows the target parameters used for the $\chi$ angle in refining RNA structures in refinement packages PHENIX (previous to my analysis) and X-PLOR. In contrast to the range of $\chi$ angles in the RNA11 dataset, the target parameters used in PHENIX (black arrows in Fig. 3.9(a)) cover the anti and the syn $\chi$ angle ranges, but are neither pucker or sequence specific. In addition, the targets seem to correspond very well to the peak for the C2'-endo range. This suggests that these parameters may have been borrowed from the targets used in DNA structure refinement. The target parameters used in X-PLOR are pucker-specific, with different targets for purines and pyrimidines (magenta and blue arrows respectively in Fig. 3.9(b)) that agree well
with the data from RNA11. However, X-PLOR has no targets corresponding to the syn range.

Based on the $\chi$ angle analysis from Section 3.3.1, the target parameters for $\chi$ angle need to be both pucker and sequence specific. The black arrows in Fig. 3.10 show the new target parameters implemented in PHENIX, with three peaks for purine residues (Fig. 3.10(a)) covering the anti, high-anti, and the syn regions, and one peak covering the anti region for pyrimidine residues (Fig. 3.10(b)).
Figure 3.10: New χ angle targets (shown with arrows) that are sequence and pucker specific, implemented in PHENIX. (a) Targets for Purine (A, G) residues, (b) Targets for Pyrimidine (C, U) residues.
RNA Structure Correction: 50S Ribosomal Subunit and HDV Ribozyme

This chapter provides details of my work on correcting and rebuilding RNA structures. Section 4.1 describes the corrections in crystal structure of the 50S ribosomal subunit from H. marismortui (PDB ID: 3CC2) that improved its MolProbity validation statistics. Section 4.2 describes the rebuild of the crystal structure of the mutant-inhibited cis-acting HDV ribozyme (PDB ID: 1VC7), and the analysis that changed the interpretation of the catalytic mechanism made from the structure. The rebuilt structure is deposited in the PDB with ID 4PRF, and text in the section is partially adapted from Kapral GJ†, Jain S†, Noeske J, Doudna JA, Richardson DC, Richardson JS (2014). “New tools provide a second look at HDV ribozyme structure, dynamics, and cleavage”. *Nucleic Acids Research* 42(20): 12833-12846. († joint first authors)

4.1 50S Ribosomal Subunit Structure Correction

4.1.1 Why Correct Ribosome Structures

Ribosomes are one of the largest and most important RNP complexes present in the cells. The first three dimensional structures of bacterial ribosomal subunits were
determined in 2000, revealing that the ribosome is actually a ribozyme, with RNA residues catalyzing the formation of the peptide bond. Since then, a number of ribosome structures have been solved, from different organisms, in different stages of translation and bound to a variety of inhibitory drugs. These structures not only provide insights on the mechanism of translation, but also the atomic details of the interaction of the ribosome with various drugs, that can now be utilized to design new inhibitors. Because the determination of these 3D structure resulted in the above-mentioned breakthroughs in the study and functioning of the ribosomes, they were awarded the Nobel Prize in Chemistry in 2009.

Determining the 3D structure of ribosomes is an extremely difficult task, especially because they are large structures, containing more than 3000 RNA residues. Because of the shear number of both protein and RNA residues involved, it is virtually impossible to build a 3D model that is without any modeling errors. However, there are various reasons why one would want ribosome structures to be as accurate as possible. Residues from ribosomal fragments and subunits form a majority of the residues in the various RNA structural datasets (more than 60% for RNA09 and RNA11 mentioned in Chapter 3) used for studying and characterizing RNA structures. Ribosomal subunits (unfortunately without any residue-level filters) are often used as starting libraries for fragment based RNA structure building. Previously solved ribosome structures are used as molecular replacement models for refining and solving new ribosome structures. Correcting the existing structures of ribosomes will help to prevent the modeling errors propogating from one structure to another. The following section describes the corrections made to the structure of the 50S ribosomal subunit from *H. morismortui*. 
4.1.2 Structure Corrections

The 50S ribosomal subunit, PDB ID: 3CC2 (Blaha et al., 2008), is solved at 2.4 Å resolution, and is one of the highest resolution ribosome structures available. The structure has a R value = 18.27% and R\textsubscript{free} = 21.48% as calculated using Phenix 1.9-1642. The MolProbity statistics (shown in Table 4.1) for this structure are remarkably good for a structure of this size and resolution, with a clashscore placing it in the 92\textsuperscript{nd} percentile for its resolution range. The two chains for the 23S and 5S RNA have some pucker, clash, and backbone conformer outliers that show scope for improvement, along with rotamer and Ramachandran outliers in the protein chains. Residues in the 23S and the 5S RNA chains were corrected using the RNA Rotator and ERRASER tools mentioned in Section 2.3. Some of these RNA corrections are described below.

Fig. 4.1 and Fig. 4.2 show two similar corrections made to suite 542 and suite 559 respectively using the RNA Rotator tool. Both suite 542 and 559 had similar internal clashes, with the C5' atom and its two hydrogen atoms clashing with both the base and the backbone. Changing the \( \alpha \), \( \beta \), and \( \gamma \) backbone dihedral angles and superimposing the new suite on the 5' and 3' ends and the base atoms fixed the internal clash, and both suites changed from a \( !! \) to a \( 1a \) conformation.

Fig. 4.3 shows an incorrectly modeled kink-turn with the suite identities shown in the figure. The first suite in the kink-turn is a \( !! \) conformer outlier, its backbone clashing with other residues in the kink-turn. ERRASER was run on that residue (1603), and the top-scoring conformation it returned fixed the steric clash and changed suite 1603 from \( !! \) to \( 7r \), making the kink-turn backbone conformation consistent with the consensus suitestring for a kink turn motif (\( 7r6p2[0a] \)). Note that ERRASER almost always returns valid suite conformers, although they are not explicitly part of its target function.
Figure 4.1: (a) The originally modeled backbone of suite 542 clashes in both directions, with backbone of 541 and base of 542 (clusters of hotpink spikes), and suite 542 is a backbone conformer outlier (!!). (b) After clash correction using RNA Rotator, suite 542 adopts a valid \(1a\) conformer.

Figure 4.2: (a) The originally modeled backbone of suite 559 and 560. Suite 559 has an internal clash (clusters of hotpink spikes), and suite 559 is a backbone conformer outlier (!!). (b) After clash correction using RNA Rotator and refinement, suite 559 adopts a valid \(1a\) conformer, and the small clash of suite 560 is corrected.
Table 4.1: MolProbity Statistics for the original 50S subunit (PDB ID: 3CC2) and its rebuilt version. The traffic light colors highlight the good (green), marginal (yellow) and bad (red) validation statistics.

<table>
<thead>
<tr>
<th>MolProbity Statistics</th>
<th>Original</th>
<th>Rebuilt</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>All-Atom Contacts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>11.25</td>
<td>2.85</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Poor Rotamers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>2.85</td>
<td>11.25</td>
<td>&gt;98%</td>
</tr>
<tr>
<td>Ramachandran outliers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>0.62%</td>
<td>3.78%</td>
<td>&lt;0.05%</td>
</tr>
<tr>
<td>Ramachandran favored</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>94.60%</td>
<td>95.23%</td>
<td>&gt;98%</td>
</tr>
<tr>
<td>MolProbity score</td>
<td>2.38</td>
<td>1.77</td>
<td></td>
</tr>
<tr>
<td>Protein Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(\beta) deviations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>0.06%</td>
<td>0.00%</td>
<td>0%</td>
</tr>
<tr>
<td>Bad bonds</td>
<td>11/29840</td>
<td>0/29840</td>
<td>0%</td>
</tr>
<tr>
<td>Bad angles</td>
<td>16/40219</td>
<td>0/40219</td>
<td>&lt;0.1%</td>
</tr>
<tr>
<td>Nucleic Acid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrong Sugar Puckers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>1.15%</td>
<td>0.59%</td>
<td>0%</td>
</tr>
<tr>
<td>Bad backbone Conformations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>8.84%</td>
<td>9.16%</td>
<td>&lt;5%</td>
</tr>
<tr>
<td>Bad bonds</td>
<td>254/68881</td>
<td>263/68881</td>
<td>0%</td>
</tr>
<tr>
<td>Bad angles</td>
<td>72/107416</td>
<td>2/107416</td>
<td>&lt;0.1%</td>
</tr>
</tbody>
</table>

Fig. 4.4 shows the correction done to a S-motif in the 5S ribosomal chain. Suite 77 is the first suite of the S-motif, but has an internal clash (backbone of residue 77 clashing with ribose ring of residue 76), and is a !!. Using the RNA Rotator tool, suite 77 was changed to the ideal 5z backbone conformer, then adjusted to fit the electron density and maintain the 5’ and 3’ covalent connections. The correction held through refinement, and the backbone conformation now is consistent with the consensus suitestring of 5z4s#a.

For correcting errors in the protein chains, the structure was run through a program called Autofix (Headd et al., 2009), that is used to automatically correct backward-fit side chains (more complex than NQH flips) for leucine, threonine, arginine, and valine. Other protein corrections included flipping a few peptides over to fit
Figure 4.3: (a) Suites 1603-1606 form a kink-turn motif, but suite 1603 was modeled as a backbone conformer outlier (!!), badly clashing with the 1605 backbone. (b) ERASER has made large rotations in the backbone between ribose and 5' P to resolve the clash and rebuild suite 1603 as a $7r$ conformer.

Figure 4.4: S-motif in the 5S ribosomal subunit. (a) The originally modeled suite 77 has an internal clash and is modeled as a !!, and suite 78 is 4s. (b) After clash correction using RNA Rotator and refinement, suite 77 adopts a valid $5z$ conformer. The suitestring for the S-motif is now $5z4s#a$. 
the electron density better, and rebuilding residues 74-83 in chain L to correct a register shift. The protein corrections were done by former students in the Richardson lab, Vincent Chen, and Jeff Headd.

The rebuilt structure after the above corrections to the RNA and protein chains was refined in **PHENIX**. The refined structure has $R = 17.32\%$ and $R_{free} = 21.40\%$. Table 4.1 shows the MolProbity statistics for final refined structure. Once the rewrite of **ERRASER** to see protein as well as RNA is available (see Chapter 10), my corrections will join new ones, and the rebuilt 50S structure will be redeposited on the PDB.

4.2 Hepatitis Delta Virus Ribozyme Structure Correction

Hepatitis Delta Virus (HDV) is a small virus that causes the disease Hepatitis D in humans. The 1.7 kb RNA genome of the virus contains a stretch of 85 nucleotides, a self-cleaving RNA called the **HDV ribozyme** (Sharmeen et al., 1988; Kuo et al., 1988; Wu et al., 1989). The self-cleaving activity of the ribozyme is absolutely essential for viral replication and viral particle assembly inside the host cells, as the cleavage reaction separates the linear RNA strand containing multiple copies of the HDV genome (replicated via the rolling-circle mechanism) into single genome-sized strands. The cleavage reaction proceeds by a nucleophilic attack by the 2' hydroxyl of the residue at position -1 (residues are numbered with respect to the cleavage site) on the phosphate group of the following residue, that generates a cyclic phosphate at the RNA product 3' end and a hydroxyl group at its 5' end. Either divalent or monovalent cations are required for the cleavage reaction; however, the reaction rate in saturating Mg$^{2+}$ has been found to be about 3000 times higher than the reaction rate in 1M Na$^{+}$ (Nakano et al., 2000; Veeraraghavan et al., 2011). Residue C75 is absolutely essential for the cleavage reaction, and mutating it to any other base, especially uracil or guanine, kills the reaction (Tanner et al., 1994; Perrotta and
The first crystal structure of the cleaved ribozyme was solved in 1998, followed by structures of uncleaved, mutant-inhibited, and ion-complexed forms. In collaboration with the Doudna and Cate lab at UC Berkeley, the Richardson lab rebuilt the cleaved (PDB ID: 1CX0) (Ferré-D’Amaré et al., 1998) and the mutant-inhibited uncleaved (PDB ID: 1VC7) (Ke et al., 2004) structure of the HDV ribozyme, and compared the rebuilt structures to a higher-resolution deoxy-inhibited structure (PDB ID: 3NKB) (Chen et al., 2010a). The rebuilt structures are deposited in the PDB as 4PR6 (cleaved) and 4PRF (mutant-inhibited uncleaved), and the study is published as (Kapral et al., 2014). Gary Kapral (former student in the Richardson lab) worked on rebuilding the 1CX0 cleaved structure, and I worked on rebuilding the mutant-inhibited uncleaved 1VC7 structure, the details of which are described in the following sections.

4.2.1 Rebuild of C75U-inhibited uncleaved structure

In order to inhibit the cleavage reaction and obtain structural information for the pre-cleaved state of the HDV ribozyme, the catalytic residue C75 was mutated to a uracil. The structure of this mutant was solved with a variety of metal ions bound in the active site. The strontium ion bound structure (PDB ID: 1VC7) contains two additional residues (the upstream nucleotides) as compared to the cleaved structure: adenine at position -2 and uracil at position -1, with the cleavage site between residue -1 and 1 (Ke et al., 2004). The RNA chain is bound to the RNA binding domain of the spliceosomal protein U1A (U1A-RBD) in the crystal structure, as a part of the P4 stem-loop was replaced with the target sequence of U1A-RBD to promote crystallization.

Solved at 2.45 Å resolution, the 1VC7 structure has R = 23.91% and R_{free} = 26.40%, as calculated by PHENIX (version 1.8.4-1496) with anisotropic B factors from
Table 4.2: MolProbity Statistics for the original C75U-inhibited structure (PDB ID: 1VC7) and its rebuilt version (PDB ID: 4PRF). The rebuilding process improved the clashscore from 61\textsuperscript{st} percentile to 100\textsuperscript{th} percentile within its resolution range (2.45Å ± 0.25Å). The traffic light colors highlight the good (green), marginal (yellow) and bad (red) validation statistics. The total number of RNA bonds and angles is larger in the rebuilt structure as it includes alternate conformations. Table made for (Kapral et al., 2014)

<table>
<thead>
<tr>
<th>MolProbity Statistics</th>
<th>PDB ID: 1VC7 (original)</th>
<th>PDB ID: 4PRF (rebuilt)</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>All-Atom Contacts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clashscore</td>
<td>21.69</td>
<td>0.76</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Poor Rotamers</td>
<td>1</td>
<td>0</td>
<td>&lt;0.05%</td>
</tr>
<tr>
<td>Ramachandran outliers</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Ramachandran favored</td>
<td>91</td>
<td>92</td>
<td>&gt;98%</td>
</tr>
<tr>
<td>MolProbity score</td>
<td>1.93</td>
<td>0.74</td>
<td></td>
</tr>
<tr>
<td>Protein Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cβ deviations</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Bad bonds</td>
<td>6/773</td>
<td>0/773</td>
<td>0%</td>
</tr>
<tr>
<td>Bad angles</td>
<td>2/1036</td>
<td>0/1036</td>
<td>&lt;0.1%</td>
</tr>
<tr>
<td>Nucleic Acid Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wrong Sugar Puckers</td>
<td>12</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Bad Backbone Conformations</td>
<td>23</td>
<td>11</td>
<td>≤5%</td>
</tr>
<tr>
<td>Bad bonds</td>
<td>6/1760</td>
<td>0/1791</td>
<td>0%</td>
</tr>
<tr>
<td>Bad angles</td>
<td>29/2792</td>
<td>0/2692</td>
<td>&lt;0.1%</td>
</tr>
</tbody>
</table>

the original TLS refinement. The MolProbity validation statistics for the structure (shown in Table 4.2) show a lot of potential for improvement. The structure has a clashscore of 21.7, which puts it in the 61\textsuperscript{st} percentile for its resolution range, while the bound U1A-RBD protein has a 95\textsuperscript{th} percentile on MolProbity score. The RNA chain has 12/74 ribose pucker outliers, 23/74 unrecognized backbone conformations, 6 bond-length and 29 (1%) covalent bond-angle outliers > 4σ, 3 of which are > 10σ. Fig. 4.5(a) shows the outlier markup on the 1VC7 3D structure.

Before starting the rebuilding process, the original 1VC7 structure was refined with PHENIX (version dev-1314) for three macrocycles. Coordinates, individual B factors, and occupancies were refined, and the weight of geometry vs. diffraction
Figure 4.5: MolProbity style outlier markup for the C75U-inhibited HDV Ribozyme structure. The RNA backbone is in black and the protein main chain is in yellow. The outliers are marked as follows: steric clashes as pink spikes, ribose pucker outliers as magenta crosses, bond-length outliers as red and blue spirals, bond-angle outliers as red and blue fans, and rotamer outliers as gold side chains. (a) The original structure (PDB ID: 1VC7) and (b) The rebuilt structure (PDB ID: 4PRF).
Figure 4.6: (a) Superposition of the three crystal structures of the HDV ribozyme: rebuilt cis-acting C75U-inhibited (PDB ID: 4PRF in gold), rebuilt cleaved (PDB ID: 4PR6 in blue), and trans-acting deoxy-inhibited (PDB ID: 3NKB, ribozyme in green, substrate strand in light green). Active-site metal ions for the two inhibited structures are also shown in corresponding colors. The region surrounding the metal ions forms the active site. (b) Sequences of the three HDV ribozyme crystal structures, aligned with the full-length ribozyme. Residues discussed individually are highlighted in cyan and are in bold. Part of the P4 stem-loop was replaced by residues shown in smaller italics (U1A hairpin sequence motif in 1CX0/4PR6 and 1VC7/4PRF, and GAAA loop in 3NKB) to aid crystallization. These residues (146-159 in 1CX0/4PR6 and 1VC7/4PRF, 50-53 in 3NKB) do not have a corresponding sequence number in the full-length HDV ribozyme. Figure made for (Kapral et al., 2014)

Data terms (wxc_scale) was set to 0.1. Then, ERRASER was run on entire structure, followed by individual residue refits wherever necessary. After all the rebuilds and correction, the structure was refined in PHENIX (version 1.8.4-1496) with two TLS groups, one for the RNA chain and one for the protein chain. The final rebuilt structure has \( R = 19.80\% \) and \( R_{\text{free}} = 25.13\% \), an improvement of 4.11% and 1.27% respectively. The MolProbity validation statistics for the final structure can be seen in Table 4.2. All geometry and pucker outliers are corrected and the number of unrecognized suite conformers is reduced from 23 to 11. The clashscore improved.
dramatically, placing the rebuilt structure at 100th percentile for its resolution range. Fig. 4.5(b) shows the improvement in the outlier markup on the 4PRF 3D structure. The following sections describe the different corrections made to the structure, including a description of the individual refits done to some residues using the tools described in Chapter 2. The residues are numbered according to the full-length HDV ribozyme (Fig. 4.6(b)) unless otherwise specified.

**Overall Pucker and Backbone Conformations Correction**

The original 1VC7 structure had 12 ribose pucker outliers, all of which were corrected in the initial ERRASER run on the entire structure, including the 10 ribose puckles that were modeled incorrectly as either C3’-endo or O4’-endo. Fig. 4.7(a) shows residue 152 incorrectly modeled as a C3’-endo pucker, leading to deviations in geometry and steric-clashes with the phosphate group of the next residue. O2’ is modeled out of the 2Fo-Fc density, and there is a large peak of positive difference density (blue mesh) near the residue. Fig. 4.7(b) shows the ERASER returned residue remodeled as C2’-endo pucker and correction of all other modeling errors. O2’ moves into 2Fo-Fc density, getting rid of the positive difference density peak.

The pucker corrections led to significant changes in the backbone conformers for a number of suites. 11 suites changed from !! to a recognized backbone conformer: 1g for 1, 1b for 20, 8d for 21, 1[ for PDB residue 150, 6g for PDB residue 151, 0a for PDB residue 153, 5p for PDB residue 155, 4g for PDB residue 156, 2o for 75, 6p for 76, and 2a for 77. PDB Suite 152 changed from one intercalation conformer, 1m, to another, 1[, and suite 74 changed from 1a to 1b as the ribose pucker for both these residues was corrected from C3-endo to C2-endo. Suite 41 was incorrectly modeled as 1e, but after its ribose pucker was corrected, it is now a !!. Residue 27 and 79 were both modeled as C3-endo and were not flagged as ribose pucker outliers, but after rebuilding, both puckers changed to C2-endo, as they fit the density better.
This resulted in 2 suites moving from one backbone conformer to another: 7d to 7p for 27, and 1c to 4n for 80.

Alternate conformations for A16

Fig. 4.8(a) shows the model and electron density for residue A16 in the original structure, which has a water clashing with the base. Visual examination of the density suggested that a different conformation of the base would fit the density better, along with deletion of the clashing water. Single-residue ERRASER was run on A16 and the fourth conformation retuned was chosen as the alternate conformation, after some manual rebuilding with RNA Rotator. Modeling of either the original or the new conformation alone gave a positive difference density peak, but using both conformations as alternates (Fig. 4.8(b)) eliminates the difference density and
matches the 2Fo-Fc density better for both base and backbone. The occupancies for the original conformation (altA) and the new conformation (altB) are 0.51 and 0.49 respectively.

Figure 4.8: Base and backbone of residue A16 from the C75U-inhibited HDV ribozyme structure, with 2Fo-Fc density at 1σ contour level. H-bonds are shown as green-dotted pillows and steric clashes as hot pink spikes. Nitrogen atoms forming H-bonds are shown as blue balls. (a) A16 in the original 1VC7 structure and original density. The base clashes with a water molecule (peach ball). (b) The two alternate conformations for A16 in the rebuilt structure 4PRF and new density. Figure made for (Kapral et al., 2014)

Flipping of G25 to form a reverse wobble base pair

In the original structure, the residue U20 forms one H-bond with the Hoogsteen edge of residue G25, and the RNA backbone model for residues C24, G25, and C26 contains steric clashes and pucker outliers (Fig. 4.9(a)). Initial refinement and the subsequent ERRASER run on the full structure corrected the pucker and steric clashes, and ERRASER flipped the G25 base from the anti to the syn conformation, to form a G25:U20 reverse wobble base pair. However, the above changes led to the base of C24 moving out of density. Therefore, the coordinates of residues 24-26 were reverted back to the coordinates from the initial refined structure (before running ERRASER), and single-residue ERRASER was run on C24 and G25 to fix the model errors. The Rosetta minimized conformation of C24 and the second conformation retuned by
single-residue ERRASER on G25 fixed the pucker outliers and also replicated the G25:U20 reverse wobble base pair (Fig. 4.9(b)). The WC edge of G25 in the reverse wobble base pair makes two H-bonds with U20 (one more than the Hoogsteen edge in the old conformation), and O6 of G25 now coordinates the metal ion in the active site. Therefore the reverse wobble base pair was preferred in the rebuilt structure, despite the electron density for G25 base not being clear enough to differentiate between the old and the new conformations.

Electron density for the backbone of residues 25-27 is also minimal and fragmentary, and both the original and the rebuilt structures show small negative difference-
density peaks for the backbone of G25 and/or neighboring residues. This suggests that this region of the structure is flexible and can adopt multiple conformations. However, fitting both the Hoogsteen and the reverse wobble base pair as alternates, or reducing the occupancy of the reverse wobble, worsened either the R values or the clashscore, and was therefore not accepted.

Active Site

Fig. 4.10(a) shows the active site in the original 1VC7 structure with many modeling errors, including bad geometry, steric clashes, and pucker outliers. ERRASER run on the entire structure was able to fix all the modeling errors in the active site, including the pucker outliers of residue 74-76. The active site in the rebuilt structure is shown in Fig. 4.10(b). The corrections to the residues in the active site introduced changes in the interaction of the metal ion (Sr$^{2+}$ here) bound in the active site with the neighboring oxygen atoms. The two close ligands to the metal ion O2 of U20 and O4 of U75 move closer to the ligand, although the interaction between Sr$^{2+}$ and O4 of U75 is a little strained at 2.388 Å. OP2 of C22 stays at about 5 Ådistance and points toward the metal, and OP1 of U23 shifts closer, to 3.6 Å. O5' of G1 and OP2 of U-1, move away from the metal ion by almost 1 Å; at > 6 Å away, they are perhaps too far for interaction with the metal ion. These two interactions are replaced by OP2 of G1, which is available for interaction with the metal ion at 4.2 Å, and O6 of the flipped G25 that now faces the metal ion at 4.1 Å. The waters in the active site are not visible in the electron density, hence it is difficult to reconstruct the entire network of ligands in the active site. Table 4.3 gives the distances between the metal ion and the neighboring oxygen atoms in the original and the rebuilt structure.

The upstream nucleotides in the original structure, and especially the critical backbone between U -1 and G1, show steric clashes, geometry outliers, and a too-short (clashing) H-bond between the base of U -1 and the backbone of C3 (Fig. 4.11(a)).
Figure 4.10: The active site of the C75U-inhibited ribozyme, with the catalytic metal ion (grey ball). The gold balls highlight the oxygen atoms in the metal ion neighbourhood, and the dotted lines indicate metal ion interactions. H-bonds are shown as green-dotted pillows, steric clashes as hot pink spikes, pucker outliers as magenta crosses, bond-length outliers as red and blue spirals, and bond-angle outliers as red and blue fans. (a) Original structure (1VC7). The active site has a number of modeling errors. (b) The rebuilt structure (4PRF) active site, with all modeling errors corrected. Figure made for (Kapral et al., 2014)

Table 4.3: The distances of the metal ion (Sr$^{2+}$) from the neighboring oxygen atoms are given (in Å) in the original (PDB ID: 1VC7) and the rebuilt (PDB ID: 4PRF) C75U-inhibited structures of the HDV ribozyme. Table made for (Kapral et al., 2014)

<table>
<thead>
<tr>
<th>Atom name</th>
<th>PDB ID: 1VC7 (original)</th>
<th>PDB ID: 4PRF (rebuilt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>O4 of U75</td>
<td>2.638</td>
<td>2.388</td>
</tr>
<tr>
<td>O2 of U20</td>
<td>3.13</td>
<td>3.086</td>
</tr>
<tr>
<td>OP1 of U23</td>
<td>4.16</td>
<td>3.624</td>
</tr>
<tr>
<td>OP2 of C22</td>
<td>5.076</td>
<td>4.981</td>
</tr>
<tr>
<td>O6 of G25</td>
<td>4.886 (facing away)</td>
<td>4.128</td>
</tr>
<tr>
<td>OP2 of G1</td>
<td>3.891</td>
<td>4.22</td>
</tr>
<tr>
<td>OP2 of U-1</td>
<td>5.221</td>
<td>6.227</td>
</tr>
<tr>
<td>O5' of G1</td>
<td>5.328</td>
<td>6.267 (facing away)</td>
</tr>
</tbody>
</table>
The improved model of the upstream nucleotides (returned by the erraser run on the entire structure) from the rebuilt structure is shown in Fig. 4.11(b). The N3 of U-1 and the non-bridging OP2 atom of C3 are now at an optimal H-bonding distance. Suite G1 moves from !! to 1g, and suite U-1 is no longer an $\epsilon$ dihedral angle outlier and has an internal H-bond. However, the backbone for A-2 and U-1 of the new conformation is less well fit to the electron density than the old conformation. Using RCrane, I manually built a different conformation for residues A-2 and U-1, that placed the base and the phosphate of A-2 in each other’s density and visually fit the electron density better, but refinement with that conformation resulted in 0.3% higher $R_{free}$. Reducing the occupancy of the upstream nucleotides or omitting the base and/or the phosphate of A-2 from the model did not help to improve the R values. Hence, the RCrane conformation for the upstream nucleotides was not accepted.

**Figure 4.11**: The position of the upstream nucleotides (A-2 and U-1) in the active site of the C75U-inhibited structure in 2Fo-Fc electron density map (contour levels 1$\sigma$ as grey mesh, 3$\sigma$ as purple mesh). (a) Original structure (1VC7) and original density. Modeling errors are highlighted. Suite 1 is an unrecognized backbone conformation (!!). (b) The rebuilt structure (4PRF) and new density. All modeling errors are corrected and suite 1 is now 1g. Figure made for (Kapral et al., 2014)
Protein Corrections

The U1A-RBD protein chain in the 1VC7 structure had some modeling errors that were also corrected. The corrections were done using the side-chain Rotator and the backrub tool in KiNG (Chen et al., 2009b). The corrections were validated by comparison with a higher resolution structure of the isolated U1A-RBD chain bound to its cognate RNA sequence (1.92 Å, PDB ID: 1URN) (Oubridge et al., 1994), and the 1CX0/4PR6 structure of the cleaved HDV ribozyme.

Asn9 side chain in 1VC7 clashes with the backbone nitrogen of His10. A combination of changes in the side chain dihedrals of Asn9 and a small backrub shift (Davis et al., 2006) to His10 is enough to move the nitrogen away from the Asn9 side chain, fixing the clash. The new conformation is similar to the one in 1URN and 1CX0/4PR6.

In the original 1VC7 structure, the side chain of Arg47 clashed with the carboxyl group of Glu19 and a nearby water molecule (Fig. 4.12(a)). The carboxyl group of Glu19 fits the electron density well and also makes a good H-bond with the water. To alleviate the clash, the guanidinium group of Arg47 was flipped, changing the rotamer from 'mmm180' to 'mtp180' (m, p and t mean minus, plus, and trans χ angles). The new rotamer now fits the density better and makes good H-bonds with both Glu19 and the water molecule (Fig. 4.12(b)). The new conformation for Arg47 is similar to the one found in 1URN.

The Leu69 in 1VC7 clashed with local backbone. Refinement merely nudged the clash and rotamer scores just past the cutoff values (better for clashes, worse for rotamers), with no significant net improvement. In contrast, changing to the 'tt' rotamer removes both clashes entirely, puts the χ angles in a highly favoured range, and matches the 1URN conformation.
Figure 4.12: Arg47, Glu19 and nearby water molecule (peach ball) in the protein chain of the C75U-inhibited structure in the 2Fo-Fc electron density map at 1.2σ contour level. The H-bonded heavy atoms of Arg47 and Glu19 are highlighted as atom-coloured balls. (a) mmm180 rotamer of Arg47 in the original structure (1VC7) clashes with both the water and the backbone of Glu19. (b) mtp180 rotamer of Arg47 in the rebuilt structure (4PRF) now makes H-bonds with both the water and the backbone of Glu19. Figure made for (Kapral et al., 2014)

4.2.2 Impact of the rebuild on catalytic mechanism interpretations

The 2.3 Å resolution 1CX0 crystal structure of the cleaved HDV ribozyme revealed that C75 is located in the active site and near the 5’ hydroxyl, providing evidence that RNA residues can participate in general acid-base catalysis; a notion also supported by a study linking C75 to proton transfer (Shih and Been, 2001). Various groups have done structural and biochemical studies to pin down the exact role of C75 in the cleavage reaction, and the interpretations of these studies have often been contradictory to each other. Some early experiments using imidazole to rescue the cleavage reaction for a cytosine to a uracil mutation suggested that C75 acts as a general base (Perrotta et al., 1999). The C75U-inhibited 1VC7 structure of the HDV ribozyme (along with other related structures solved at 2.2–3.4 Å resolution (Ke et al., 2004)) succeeds in inhibiting the reaction and has two additional nucleotides upstream of the cleavage site. Based on the position of the upstream nucleotides of the original structure, the authors proposed a catalytic mechanism with C75 acting as
a general base, with a hinge motion around the bond O3’-P positioning the O2’, the phosphate group, and the N3 atom on C75 for catalysis. Other biochemical results supported a general-acid role for C75 (Nakano et al., 2000; Perrotta et al., 2006; Das and Piccirilli, 2005). Molecular dynamics simulations with the C75U-inhibited structure as a starting model also provided conflicting results. One study reported spontaneous rotation of the residue at position -1 to support the general-base role for C75 (Banáš et al., 2008), and their efforts to simulate a general-acid role for C75 failed. Another study reported a similar spontaneous rotation but supported the general-acid role for C75 (Lee et al., 2011).

Table 4.4: The three different versions of the "pre-cleaved" structures of the HDV ribozyme, along with the inhibition strategy, and the cleavage mechanism they support.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Inhibition</th>
<th>Upstream nucleotides</th>
<th>Mechanism</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1VC7</td>
<td>C75U Mutation</td>
<td>Partial density for 2 residues</td>
<td>General-base role of C75</td>
<td>Cis-acting</td>
</tr>
<tr>
<td>4PRF</td>
<td>C75U Mutation</td>
<td>Partial density for 2 residues</td>
<td>Partial General-acid role of C75</td>
<td>Cis-acting, Rebuilt</td>
</tr>
<tr>
<td>3NKB</td>
<td>Deoxy-substitution of -1 residue</td>
<td>No density. Modeled based on hammerhead ribozyme</td>
<td>General-acid role of C75</td>
<td>Trans-acting</td>
</tr>
</tbody>
</table>

Supporting the general-acid role for C75, the structure of the trans-acting HDV ribozyme was solved at 1.9 Å resolution (PDB ID: 3NKB) with the cleavage reaction inhibited by removing the 2’ hydroxyl on the substrate strand to prevent the nucleophilic attack (Chen et al., 2010a). No significant electron density was observed for the single nucleotide upstream of the cleavage site. Therefore, to propose a model for the catalytic mechanism, the hammerhead ribozyme cleavage site (Martick et al., 2008) was superimposed on that of the HDV ribozyme, implying a similar reaction mechanism. This proposal is also consistent with a recent computational study that analyses metal ion contributions to the C75 general-acid mechanism (Ganguly et al., 2010b).
2011), and supported by further biochemical studies (Chen et al., 2013; Thaplyal et al., 2013). The current consensus in the field is that C75 acts as a general acid (Ferré-D’Amaré and Scott, 2010), with the hydrated metal ion acting as the general base.

To find out if the structural improvements made to 1VC7 had any impact on the interpretation about the catalytic mechanism, it was necessary to compare the new 4PRF structure with the higher resolution 3NKB structure. The details of the comparison are provided below.

Support for changing from general-base role to general-acid role

In order to compare the model of the upstream nucleotides in the mutated vs. the deoxy-inhibited structures, I recapitulated the model of U -1 and the scissile phosphate discussed in (Chen et al., 2010a), but not deposited, by superimposing the residue G1 of 3NKB and the equivalent residue (residue 7) from the structure of the hammerhead ribozyme (PDB ID: 2OEU (Martick et al., 2008)). The modeling resulted in a significant steric-clash that was reduced after a round of refinement, and the resulting structure looks like the figure in the paper, and hence was used for all further analysis. The modeled position of U -1 and the scissile phosphate in 3NKB based on the hammerhead ribozyme makes three key interactions in the structure (Fig. 4.13). The base of U -1 stacks on the base of U23, and both O2' of U -1 and OP2 of the scissile phosphate coordinate the Mg^{2+} ion. OP2 of the scissile phosphate had been shown to be important for the cleavage reaction and its interaction with the coordinated Mg^{2+} has been confirmed experimentally (Jeoung et al., 1994; Thaplyal et al., 2013). This conformation also shows the potential of a H-bond between the N3 atom of C75 and the O5' of G1, which is observed in the cleaved structure 1CX0/4PR6. These interactions support the general-acid role of C75.
The C75U-inhibited structure 1VC7/4PRF1VC7 has the crucial C75U mutation to stop the cleavage reaction, and has a Sr$^{2+}$ ion bound in the active site instead of a Mg$^{2+}$. In the original 1VC7 structure, the Sr$^{2+}$ ion was reported to be coordinated by the O5’ of G1, and that suggested the general-base role for C75. However, O5’ of G1 faces away from the metal ion (with a distance of > 6 Å) in the rebuilt 4PRF structure, and OP2 (pro-Rp) of the scissile phosphate is now unambiguously available for interaction with the metal ion. This change in oxygen atom interacting with the metal ion, which now is similar to the one modeled in the deoxy-inhibited 3NKB structure, changes the interpretation of the catalytic mechanism from general-base role for C75 to general-acid role. However, the conformation of U-1 and G1 in 4PRF do not make the other two interactions present in 3NKB, metal ion coordination by O2’ of U-1 and the potential H-bond between N3 of C/U75 and O5’ of G1. O2’ of U-1 is at a distance of about 9 Å and facing away from the metal ion, and interaction between N3 and O5’ is blocked by the scissile phosphate. This makes the rebuilt 4PRF structure only partially suggestive of the general-acid role for C75.

Role of the reverse G25:U20 wobble base pair

G25 and U20 are located in the active site and interact with groups essential for catalysis. Several studies have found that mutating one or both these residues leads to significant decrease in the catalytic activity of the ribozyme, suggesting that these residues play a functional role in the catalytic mechanism (Perrotta and Been, 1996; Chen et al., 2013; Lévesque et al., 2012). Also, residues G25 and U20 are conserved in catalytically active HDV-like ribozymes of various species, including the human CPEB3 gene (Webb et al., 2009; Webb and Lupták, 2011; Salehi-Ashtiani et al., 2006). G25:U20 reverse wobble base pair is present in the deoxy-inhibited 3NKB structure with unambiguous electron density, and is now modeled in the cleaved 4PR6 and the C75U-inhibited 4PRF structures of the HDV ribozyme as well. Molecular
dynamic simulations starting from the original cleaved 1CX0 and the deoxy-inhibited structure 3NKB show that this reverse GU wobble base pair is stable both in the presence and absence of the metal ion (Veeraraghavan et al., 2011). However, this base pair was not observed in the simulations starting from the structure 1SJ3 (one of the related structures to 1VC7 (Ke et al., 2004)), resulting in the conclusion that the reverse wobble base pair is not compatible with the C75U-inhibited structures. This is incorrect as the rebuilt structure 4PRF does have this reverse wobble base pair, and it is possible that the simulations were not able to flip G25 (along with the accompanying backbone changes) to make the required hydrogen bonds.

However, biochemical and NMR studies suggest that the formation of the GU reverse wobble base pair is a post-cleavage event (Lévesque et al., 2012; Tanaka et al., 2002), and hence are inconsistent with the presence of this base pair in the C75U-inhibited 4PRF structure. As previously mentioned in Section 4.2.1, this region in
both the original and rebuilt C75U-inhibited structures is fragmentary, and not suf-
ficient to distinguish between the original Hoogsteen and the rebuilt reverse wobble base pair. The rebuilt 4PRF structure shows both positive and negative difference-density peaks that corresponds to the Hoogsteen conformation of G25, but modeling both conformations as alternates worsened the R values or the clashscore. A likely scenario therefore is that residues 25-27 are flexible and can adopt multiple conformations, including the reverse GU wobble base pair, which is the alternative stabilized after the cleavage reaction.

### 4.2.3 Was the pre-cleaved conformation crystallized?

In order to get structural information for the pre-cleaved state of the HDV ribozyme, significant changes must be made to the molecule to prevent catalysis. It is hoped that the changes made to the molecule will inhibit the reaction enough and not lead to significant changes in the structure of the molecule. The two crystal structure 1VC7/4PRF and 3NKB inhibited the cleavage reaction by a C75U mutation and removing the 2' hydroxyl on the substrate strand respectively. In addition crystallization of those these structures was promoted by replacing the nonessential P4 stem-loop with known interaction motifs: the target sequence for U1A-RBD in 1VC7/4PRF structure, and a GNRA tetraloop in the 3NKB structure. The sequence gap in the trans-acting 3NKB form shifts the P1 and P2 stems somewhat (green spline, Fig. 4.6(a)), and it also contains an additional base pair in the P2 stem. As a result the constructs of the two attempted pre-cleaved structures are quite different.

Fig. 4.14 shows the relative positions of the upstream nucleotides A -2 and U -1 in the rebuilt C75U-inhibited 4PRF and the modeled U -1 in the deoxy-inhibited 3NKB (based on the active site of the hammerhead ribozyme) as superimposed on the residue G1. The position of the upstream nucleotides in the active site is very
different in the two structures, with the two conformations almost perpendicular to each other. Modeling of U-1 conformation from 4PRF in 3NKB resulted in overlap of the upstream nucleotides with rest of the ribozyme. Likewise, modeling of U-1 conformation from 3NKB into 4PRF resulted in steric clashes with rest of the ribozyme, and also left the density of the upstream nucleotides unaccounted for. This indicates that the two conformations for the residue U-1 cannot be present in the same structure.

**Figure 4.14:** Upstream nucleotides (gold) in the rebuilt C75U-inhibited structure (4PRF) and the modeled upstream nucleotide U-1 and scissile phosphate (green) in the trans-acting deoxy-inhibited structure (3NKB), as superimposed on the base of residue G1. The positions of the corresponding metal ions in the active site are also shown. The two models for the upstream nucleotides are almost perpendicular to each other, and the two metal positions are nearly 4 Å apart. Figure made for (Kapral et al., 2014)

As noted earlier, there is no clear density in the 2Fo-Fc map for U-1 in the deoxy-inhibited 3NKB structure. This lack of density was partly attributed to the dynamics and flexibility of the U-1 residue, and partly to the lowered occupancy of U-1 due to a fraction of the substrate strand being cleaved in the crystal. The three interactions made by the modeled U-1 and the scissile phosphate in the 3NKB structure (Fig. 4.13) make a good binding site for U-1, and should be strong enough
to hold U-1 in place, and the 2Fo-Fc map should have had decent density for the residue. If the modeled conformation is correct, the former scenario seems unlikely. Although the modeled conformation is claimed to be partially supported by some positive difference density is seen in the region, there is also an equal amount of negative difference density in that region (Fig. 4.15), so those weak peaks cannot be taken as enough evidence to support the modeled U-1 conformation. Therefore, the second scenario of the upstream nucleotides being cleaved while sitting in the crystal (10-50% as reported in (Chen et al., 2010a)) seems much more plausible, in spite of the deoxy substitution. In addition, after rebuilding the cleaved 1CX0 structure, the 3NKB structure is now more similar to the rebuilt cleaved 4PR6 structure, though it has the intact metal site.

**Figure 4.15:** The position of G1, modeled U-1 and the scissile phosphate in 3NKB, based on the active site of the 2OEU hammerhead ribozyme structure, shown with Fo-Fc difference density. Blue mesh shows the positive difference density peaks, and the orange mesh shows the negative difference density peaks, both at 3σ contour level. Figure made for (Kapral et al., 2014)

The 2Fo-Fc map for the rebuilt C75U-inhibited structure 4PRF shows electron
density for at least the backbone of the upstream nucleotides, so the C75U mutation indeed succeeded in producing an uncleaved structure. However, the coordination of the Sr\(^{2+}\) ion by the O4 atom of U75 is an artifact of the C75U mutation, as an exocyclic amino group replaces the O4 atom in the wild-type ribozyme. The hydrogen bond between N3 of C/U75 (the atom identified as the general-acid in (Das and Piccirilli, 2005)) and O5' of G1 is also blocked by the scissile phosphate. In addition, the U-1 O2' is still 9 Å from the metal, and the position and orientation of the scissile bond is even less appropriate for catalysis than in the original 1VC7 structure. Therefore, it is very likely that the C75U mutation distorted the active site, so that the upstream nucleotides in 4PRF do not represent the catalytic position, a thought also discussed in (Chen et al., 2010a).

Biochemical studies on the C75U mutant ribozyme show that the ribozyme can still carry out the self-cleavage reaction when either external reagents substitute for C75 or the ribozyme is modified such that the general-acid role of C75 is not needed (Das and Piccirilli, 2005). This suggests that while C75U disrupts the active site by eliminating the interactions made by C75 (to position itself for catalysis), the mutant ribozyme is not destabilized or misfolded. A molecular dynamics study starting from the original 1VC7 structure (restoring the C75 residue and the proper metal ion position) under different protonation states of C75 report a spontaneous rotation of the upstream nucleotides to make the required interactions to support the general-acid role of C75 (Lee et al., 2011). However, attempts to model this rotated position in the rebuilt structure failed. Therefore, a possibility is that the spontaneous rotation of the upstream nucleotides occurs only during the cleavage, making it difficult to capture in a crystal structure.

The above analysis confirms the likelihood that the position of upstream nucleotides in 4PRF do not represent the catalytic position, and highlights that the upstream nucleotide is not convincingly intact in 3NKB (although the modeled U
-1 position seems very plausible). Therefore neither strategy to inhibit the cleavage reaction and capture a molecular representation of the HDV ribozyme substrate has succeeded in achieving a good mimic of the pre-cleavage active site.

This study highlights the role these new tools can play in assisting crystallographers to make well-supported biological conclusions by helping build reliably accurate models, and how important it is to have the structure as accurate as possible in order to make correct biological conclusions.
Revising the list of RNA backbone conformers

This chapter describes my work on revising the RNA backbone conformers, defined by the RNA Ontology Consortium (ROC) in 2008. Section 5.1 describes the dataset and methods used, Section 5.2 provides details on updating the existing backbone conformers, and Section 5.3 provides details on the new backbone conformers found.

As described in Section 2.1.3, the ROC identified discrete set of conformations that the RNA backbone is most likely to adopt (Richardson et al., 2008). The backbone conformer list consisted of 54 RNA backbone conformers (46 clear conformers and 8 wannabes), described in terms of the seven backbone dihedral angles of the suite (the sugar-to-sugar unit of the RNA backbone, which thus contains two hemi-nucleotides Fig. 2.4). These backbone conformers were established in 2008, using the RNA05 dataset. However, in the larger RNA11 dataset (Table 3.1), about 19% of the suites do not belong to any of the known 54 backbone conformers (denoted by !!), as can be seen in Fig. 5.1. Hence we wanted to revisit the backbone conformers using the larger dataset, to update the list and find new backbone conformers that may have been missed because of the smaller dataset.
5.1 Method

5.1.1 Data Filtering

The RNA11 dataset was used to study the RNA backbone conformers here. A suite was filtered out if any one of its two residues met the following conditions: 1) any residue with a backbone atom of Bfactor ≥ 60, 2) any residue with a pucker outlier, 3) any modified residue, and 4) any residue whose backbone atom clashed with the base or the backbone atom of residue ± 1 in sequence. After the filtering step, the number of suites left was 9153. Dangle was run to calculate the suite dihedral angles. Suites were divided into 12 $\delta-\delta-\gamma$ bins based on the pukers of the two ribose sugars ($\text{C3'}-\text{C3'}$, $\text{C3'}-\text{C2'}$, $\text{C2'}-\text{C3'}$, and $\text{C2'}-\text{C2'}$) represented by $\delta$, and the $\gamma$ angle ($\text{m}$ (minus), $\text{p}$ (plus), and $\text{t}$ (trans)). The visualization program KiNG was used to visualize the conformer clusters in seven dimensions. 8% of the filtered data did not belong to any of the known backbone conformers (Fig. 5.1).

**Figure 5.1:** Percentage of 1a, non-1a, and ! suites in the unfiltered and filtered data from the RNA11 dataset. Total number of suites in the unfiltered data: 24335. Total number of suites in the filtered data: 9153
5.1.2 Using ERRASER

One method which we have used to help with the backbone conformer analysis is single-residue correction using ERRASER (described in Section 2.3). Since the scoring function used in ERRASER is based on electron density and an energy function that does not explicitly include the backbone conformers, conformations returned by ERRASER can be used to confirm newly found conformer clusters. To visualize the top 10 conformations returned by ERRASER, I wrote a script that would allow us to visualize the 10 returned conformations along with the all-atom contacts each made with the rest of the structure. This, along with the Rosetta scores and fit to the electron density, can be used to select the best of the 10 returned conformations.

5.2 Updating existing clusters

5.2.1 Upgrading the wannabes to main consensus list

Out of the 54 conformers, 46 were part of the main consensus list, and 8 were called wannabes. These 8 clusters either did not have enough examples or a good resolution example (better than 2 Å) to confirm that they were genuine clusters. I looked at the examples of these clusters in the filtered RNA11 dataset.

2u conformer was a wannabe in the 22t bin. There are six examples of this cluster in the filtered dataset, three of them from good resolution structures. Another high resolution point, 2XNZ A48 at 1.59 Å (local electron density is sketchy), moved from a to this cluster after ERRASER was run on it. Because of the high-resolution examples and a moving into this cluster, it was promoted to be part of the main consensus list.

5r conformer was a wannabe in the 32m bin. In addition to having more points in the RNA11 dataset, ERRASER moves some of the nearby points to this cluster. Therefore, this conformer was promoted to be part of the main consensus list.
**2g** conformer in the bin 23p, **2z** conformer in the 22p bin, **5p** conformer in 32p bin, **5n** conformer in 33t bin, were wannabes. All 4 of these clusters have more examples in the RNA11 dataset, along with examples from high resolution structures and good electron density. Therefore, these four clusters were promoted to the main consensus list.

Table 5.1: The list of wannabes that were promoted to the main consensus conformer list, along with the number of examples in each cluster and its representative suite.

<table>
<thead>
<tr>
<th>Bin</th>
<th>Conformer</th>
<th>No. of Examples</th>
<th>Representative suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>33t</td>
<td>5n</td>
<td>13</td>
<td>2XNZ A54 1.59 Å</td>
</tr>
<tr>
<td>32p</td>
<td>5p</td>
<td>13</td>
<td>3PDR X101 1.85 Å</td>
</tr>
<tr>
<td>32m</td>
<td>5r</td>
<td>15</td>
<td>3D2S F2 1.7 Å</td>
</tr>
<tr>
<td>23p</td>
<td>2g</td>
<td>16</td>
<td>3PDR X130 1.85 Å</td>
</tr>
<tr>
<td>22p</td>
<td>2z</td>
<td>15</td>
<td>3D2S E5 1.7 Å</td>
</tr>
<tr>
<td>22t</td>
<td>2u</td>
<td>7</td>
<td>1EHZ A48 1.93 Å</td>
</tr>
</tbody>
</table>

Overall, 6 out of the 8 wannabes were promoted to the main consensus list, for a total of 52 backbone conformers in the main consensus list. Table 5.1 provides the number of examples and the representative suite for each of these six backbone conformers.

5.2.2 **Updating existing clusters**

The **2o** cluster in the 22m bin had 7 points in the filtered data. One of the points 1SDS F209 comes from a high-resolution structure, but was a bit further away from the other points in the cluster in the ζ dimension. The first model returned by single residue ERRASER returned a conformation that remains **2o**, much closer to the other points in the cluster. ERRASER run on another point from 22t bin which was originally a !! also moves to the 2o cluster. So overall the new cluster has a total of eight points, much closer together in space. The new representative example would be 3NKB B62 at 1.9 Å resolution. Similarly, running ERRASER on other !!
points leads them to moving into other clusters like 2{ and 4p in 22p, 2h in 23t, 2a and 8d in 23p, 1o in 32m, 1t in 32t, and 3b in 32p.

Three backbone conformer clusters 0i, 6n, and 6j in 23t bin are indistinguishable in six out of the seven dihedral angles, except the \( \beta \) angle (Fig. 5.2(a)). In the RNA11 dataset, however, the boundary between them in the \( \beta \) dimension is not very clear. Running single-residue ERRASER on the points in the filtered data between the three clusters made the separation between the clusters much cleaner, especially between 6n and 6j (Fig. 5.2(c)). In addition, a few points from \(!!\) moved to each of the three clusters, increasing the number of points in them.

**Figure 5.2:** (a) 0i, 6n, and 6j backbone conformers in bin 23t in the RNA11 dataset. (b) The original points (small gray balls) connected to their ERRASER results. (c) Final cluster boundaries are much more cleaner.

1b and 1f are two backbone conformers in the 32p bin. 1f is the most common conformation for an intercalation motif and 1b is the most common conformation leading into the C2’ pucker suites. Though these two conformers have very distinct
and characteristic structural features, it was difficult to separate them in the initial study. Hence an artificial line in the $\beta$ dimension was created to separate them, but this led to some examples of one functional conformer being counted in the other. Even in the RNA11 dataset, the examples at the boundary of $1b$ and $1l$ are more often than not in the wrong cluster. Running ERRASER on these mis-classified boundary examples leads most of them to move into the correct cluster, making the separation between them much cleaner. For this analysis, I was assisted by the work of Jeremy Norte, a high-school student that I mentored in Fall 2013.

The $4d$ conformer cluster in 23p bin contains examples that have the characteristic hydrogen bond between O2’ of residue i-1 and OP2 atom of residue i+1. Most of the examples in this cluster are from suite 59 in tRNA structure. However, another cluster in the 23p bin, $8d$ contains some examples with the above hydrogen bond. These examples were not included in the $4d$ conformer cluster, because there was no clear boundary separating all the H-bonded suites from the suites that did not contains this H-bond. In the larger RNA11 filtered dataset, $4d$ cluster contained some examples that did not have the characteristic H-bond. Running single-residue ERRASER on suites in $8d$ with the H-bond and on suites in $4d$ without the H-bond led to a clear separation between the H-bonded and the non H-bonded suites in the $\beta$ dimension. Therefore the new definition of $4d$ includes all suites with the characteristic H-bond, and all other suites are in the $8d$ cluster.

5.2.3 Motif based backbone conformer correction

As mentioned in Section 2.1.3, the advantage of using the two-character backbone conformer names is that three-dimensional structures can be represented as a string of the two-character names, called the suitestring. Besides tetra-loops, the two most common structural motifs that have a consensus backbone conformer suitestring are the S-motif (consensus suitestring $5z4s\#a$) and the kink-turn motif (consensus
suitestring 7r6p20a. However, I found some examples of the S-motif and the kink-turn motif in the RNA11 dataset where one or more of the suites were backbone conformer outliers !!. Running single-residue erraser on these !! suites led most of them to move to the correct backbone conformer. This resulted in addition of points to the 5z, 4s, 7r, and 6p clusters and cleaner cluster boundaries.

5.3 Finding new backbone conformer clusters

The 54 backbone conformers identified in the initial 2008 study did not cover all regions of the 7 dimensional dihedral angle space. With the new RNA11 dataset new conformer clusters were identified in some of those regions.

A new cluster was identified in the 22m bin. In the filtered dataset, it has six data points, all but one from ribosomal subunits. The one example not from the ribosomal subunit is from 3SKI 2.3 Å resolution, and is a good fit to the density. In addition, a single residue erraser run on suite A2491 from 3R8T moves here from 22p !!. This gives us the confidence that this cluster is a genuine conformer. Since there is no high resolution example, this will be a wannabe cluster. The name of the cluster is 4o.

Two new clusters were identified in the 22t bin. For most of the examples in one of the clusters, the ribose sugars are perpendicular to each other and bases are far apart. The backbone of all examples overlap very closely onto each other. In addition some points from the 22p !! region moved to this cluster when single residue erraser was run on them. This cluster has enough high-resolution examples to be part of the main list. The second cluster has 7 examples, with one additional suite moving here from 22p. These eight examples form a tight cluster in the 7 dimensional dihedral space, but all of them are from ribosomal subunits. Therefore this cluster will be considered a wannabe. The one letter code for the second hemi-nucleotide for both the above clusters, with $\alpha - \beta - \gamma - \delta$ of p t t 2’, was not defined in the 2008
study. To name these two clusters, we had to come up with a new letter representing the hemi-nucleotide conformation for this $\alpha - \beta - \gamma - \delta$ bin: the letter $v$. Therefore, the name of the main backbone conformer is $6v$ and the wannabe is $8v$.

Overall with the addition of all the new conformer clusters, there are now 53 confirmed backbone conformers and 4 wannabes.

**Figure 5.3**: Suite 8 (chain B) from a HIV-1 RNA duplex bound to antibody ribostamycin (PDB ID: 3C3Z, 1.5 Å). The 2Fo-Fc density shown at 1.2 $\sigma$ as grey mesh and at 3.2 $\sigma$ as purple mesh. The water molecules are shown as mustard balls, and hydrogen bonds shown as green dot pillows. This suite is clearly well fit, but is an isolated !! in the 33t bin.

When looking for new backbone conformers and making sure that they are real, one of the main criteria is to look for one or two examples that are from high-resolution structures (usually better than 2 Å), that make good contacts, and have no modeling errors. In the RNA11 dataset, there are a few !! suites that meet the above criteria. Running ERRASER on these examples does not return any valid alternate conformation as they fit the electron density very well (for instance, see
Fig. 5.3). But these examples are isolated in the seven dimensional dihedral space, without any examples near them in the filtered RNA11 dataset. The electron density combined with lack of modeling errors is strong evidence that these conformations are valid. Some may be held in a relatively disfavored conformation by an unusual arrangement of the rest of the structure, usually at a binding or active site. But most are probably just rare because there is more conformational freedom in stretched-out loops. Looking at the unfiltered RNA11 data, and running ERRASER on the suites in the vicinity of these isolated high-resolution examples may potentially correct whatever errors made those suites get filtered out, and result in some of these high-resolution examples to gather a cluster and be listable as real backbone conformers.
6

Background: Computational Structure-based Protein Design and Algorithms

This chapter provides the background on computational structure based protein design and defines terms that are used in Chapters 7 and 8. Section 6.1 provides the general background on protein design, Section 6.2 provides the details on the algorithms used in the protein design software osprey, and Section 6.3 describes the use of sparse residue interaction graphs in protein design.

6.1 General Background

Design of new protein structures from scratch without a close relationship to an existing sequence/structure example is called *de novo protein design* (Richardson and Richardson, 1989). Although the first attempts at de novo protein design were not very successful (Hecht et al., 1990), the findings contributed immensely to the field of protein design in general. These studies taught us about protein structure and folding, importance of negative design (designing against a specific property), and that a unique structure is harder to achieve than a stable structure (Richardson et al.,
More recent attempts at de novo design tend to be more successful (Offredi et al., 2003; Kuhlman et al., 2003; Richardson and Richardson, 2002), but not yet reproducibly enough to be useful for practical purposes.

**Protein redesign** starts from existing protein molecules and attempts to design new proteins, starting from existing ones, with the objective of either folding to a desired three-dimensional structure, binding new targets, or modifying catalysis to achieve a specific new function—is called protein redesign. Protein redesign process is usually accomplished by predicting the optimal sequence to achieve the desired function using rational computational structure-based (design based on three-dimensional protein structures) protein design algorithms, that requires searching over large sequence and conformational spaces. Protein redesign algorithms were first used to redesign a new sequence to fold to a desired structure (Dahiyat and Mayo, 1996, 1997), but is now an emerging field with a wide variety of applications (Donald, 2011). It can help in the advancement of basic sciences by probing the principles of protein folding and stability (Keating et al., 2001), to-in the advancement of translational medical research by altering biological functions (Hellinga and Richards, 1991; Marvin and Hellinga, 2001; Shifman and Mayo, 2002; Looger et al., 2003; Chen et al., 2009a; Bolon and Mayo, 2001) and designing drugs and antibodies that can have therapeutic applications (Lippow et al., 2007; Gorczynski et al., 2007; Rudicell et al., 2014; Georgiev et al., 2014; King et al., 2014; Roberts et al., 2012).

The inputs to a protein redesign algorithm consist of a given input structure, the conformational and sequence search space, and a method to score the different candidate sequences and conformations. The input structure consists of three-dimensional coordinates for all protein atoms (including hydrogens) determined by either X-ray crystallography or NMR. The search space consists of a set of amino acids for the residues that are allowed to mutate during the design process, along with a model for protein backbone and side-chain flexibility. Most protein redesign algorithms keep
the bond lengths and bond angles fixed. Side-chain flexibility is modeled using a discrete set of frequently-observed, low-energy conformations called rotamers (Lovell et al., 2000), that can also be backbone dependent (Dunbrack and Karplus, 1993; Shapovalov and Dunbrack, 2011). Rotamers can be modeled as rigid, or as continuously flexible over a defined space (Gainza et al., 2012). Protein backbone is often kept rigid, but can be modeled as flexible by either designing over multiple structures of the same protein, or allowing the backbone $\phi$ and $\psi$ dihedral angles to change during the design process, either directly (Georgiev and Donald, 2007), or indirectly as a consequence of other “moves” (Georgiev et al., 2008a; Hallen et al., 2013). Although time consuming, modeling the backbone as flexible allows the structure to breathe and make necessary accommodations that are often accompanied by mutating the native sequence (Gainza et al., 2013; Babor et al., 2011).

The scoring of different conformations is done using an energy function. The energy functions needs to be accurate and capable of ranking sequences based on how well they would fold to the desired structure. The most accurate energy functions are based on quantum mechanical calculations, but these are seldom suitable for design algorithms as they are time-consuming to compute. Design algorithms have to evaluate an exponential number of candidate structures and computing sophisticated energy functions for each candidate conformation becomes prohibitive. For computational feasibility, energy functions are usually simplified to consist of pair-wise energy terms: i.e. the energy of a conformation can be calculated by adding the interaction energy between all-atom residue pairs of the protein. Pair-wise energy functions can be broadly categorized into physics-based potential energy functions that consist of electrostatic and vdw energy terms (Brooks et al., 2009; Cornell et al., 1995), and statistical knowledge-based energy functions that consist of terms derived from conformations in already solved protein structures (Kuhlman and Baker, 2004; Russ and Ranganathan, 2002). Each design algorithm tends to use its own ver-
sion of an energy function, which sometimes contains both physics-based and knowledge-based energy terms.

6.1.1 GMEC vs. Ensemble-based design

The goal of a protein redesign algorithm is to predict the optimal protein sequence (or a list of top ranking sequences) that will achieve the desired function. Often this is done the desired function is achieved by finding the conformation that minimizes the energy function, called the Global Minimum Energy Function (GMEC), and reporting the sequence of the GMEC as the optimal sequence. However, a protein does not exist as a single static structure, but is intrinsically flexible and continuously undergoes modest but significant conformational changes. It is important to make sure that these alternate conformations of a protein are also stable and low-energy, and if the case might be, bind to the desired target. Therefore, a much more realistic representation of proteins is as a thermodynamic ensemble of low-energy conformations for the region being redesigned.

Designing based on an ensemble of structures ensures that the design algorithm does not fall into the trap of giving a high score to a sequence that can adopt only a single low-energy conformation and all other alternate conformations being high-energy or unstable. Studies have shown that protein redesign algorithms using an ensemble of low-energy structures result in better design predictions (Tzeng and Kalodimos, 2012; Lilien et al., 2005). However, this does result in increased running times for design algorithms, as it has to enumerate a larger set of conformations and score each sequence independently, leading to the quintessential trade-off between accuracy and speed of generating the results.
6.1.2 Provable vs. Heuristic Algorithms

As mentioned above, a major challenge for protein design algorithms is to efficiently enumerate conformations and search over the sequence and conformational search space. Identification of the GMEC, using a pair-wise energy function, rigid backbone and rotamer libraries is known to be NP-Hard (Pierce and Winfree, 2002). Incorporation of realistic molecular flexibility and enumeration of multiple conformations adds to the complexity of the design problem, leading to increased run times. Due to this increase in complexity, numerous heuristic techniques are used to reach locally optimal solutions (Koehl and Delarue, 1994; Kuhlman and Baker, 2000; Desmet et al., 2002; Jones, 1994; Leaver-Fay et al., 2011). Although heuristic algorithms can return a large number of acceptable solutions quickly, they have no guarantees on the accuracy of the generated solutions. On the other hand, provably optimal algorithms (algorithms that guarantee to find the optimal solution for the given input model and assumptions) make use of additional time and memory to provide guarantees on the quality of the solutions found. In addition, ensemble-based provable algorithms and can generate a “gap-free” list of low-energy conformations, (i.e. “gap-free” list of “low energy” conformations means that the algorithm is capable of generating all conformations with energy < the “low-energy” limit from the GMEC) for representing the protein structure ensemble (Gainza et al., 2013; Gordon and Mayo, 1999; Hong et al., 2009; Kingsford et al., 2005).

To make the modeling of biological concepts computationally feasible, various assumptions are made in the inputs (energy function, molecular flexibility, starting structures) given to the design algorithm. These assumptions only approximate the underlying biological rules and may sometimes result in the failure of the designed protein to carry out its predicted function. When such a failure occurs, it is important to distinguish it from the case when the design algorithm does not find the best
solution. Using provable algorithms for design eliminates the latter possibility, and clearly indicates the success or failure of the assumptions made by the inputs. This is especially important in the field of protein design, with new techniques being developed continually, and a proper evaluation of current state of art is important to make further progress. However, provable algorithms run more slowly than most heuristic algorithms, rendering large scale designs and sophisticated modeling of molecular flexibility infeasible. Improving provable algorithms to reduce their time and memory requirements, and designing new algorithms can provide better results with finite resources.

6.2 OSPREY details

The Donald lab has developed an ensemble-based protein design software package called Open Source Protein Redesign for You or OSPREY (Gainza et al., 2013). It uses provable algorithms that guarantee to return the optimal solution for the given set of input parameters for a design problem. OSPREY has been successfully used in a variety of design problems, including redesign of the adenylation domain of Non-ribosomal peptide synthetase enzyme to switch its specificity from Phe to Leu (Chen et al., 2009a; Stevens et al., 2006), design of inhibitors for leukemia-associated proteins (Gorczynski et al., 2007) and peptide inhibitors to interrupt a crucial protein-protein interaction associated with cystic fibrosis (Roberts et al., 2012), design of antibodies and nanobodies against HIV-I (Rudicell et al., 2014; Georgiev et al., 2014, 2012), and prediction of resistance mutations of di-hydro folate reductase enzyme in *S. aureus* (Frey et al., 2010; Reeve et al., 2015).

The flow of the algorithms in OSPREY is given in Fig. 6.1. OSPREY takes as input a starting protein structure, a model for protein flexibility, a rotamer library, and an energy function. It first runs Dead-End Elimination (DEE) to provably prune rotamers not part of the GMEC (Section 6.2.2). The lab has also developed
provable algorithms to incorporate protein side-chain and backbone flexibility in the design process (Section 6.2.4). After the pruning stage, A* search is used to enumerate conformations in increasing order of energy (Section 6.2.3), which are used to calculate an \( \epsilon \)-approximation to the partition function, which in turn is used to calculate the \( K^* \) score to approximate the binding constant \( K_a \) (Section 6.2.5).

**Design Software - OSPREY**

![Diagram of the OSPREY design software workflow]

**Figure 6.1**: The workflow of the OSPREY design software.

### 6.2.1 Inputs to OSPREY

The input structure to OSPREY specifies the three-dimensional coordinates of all atoms, including hydrogens, in the protein. A set of mutable residues, along with mutations allowed at each of those residues also needs to be specified. Side-chain flexibility is modeled using the Lovell Penultimate rotamer library with a total
of 152 rotamers (Lovell et al., 2000). The energy function used by OSPREY is a pair-wise energy function, using either the AMBER (Cornell et al., 1995) or the CHARMM (Brooks et al., 2009) molecular force fields to model the electrostatic and vDW interaction terms. Solvation energy is modeled using the EEF1 implicit solvation model (Lazaridis and Karplus, 1999). Bond lengths and bond angles are kept rigid and not scored in the energy function.

Using the pair-wise energy function in OSPREY, the energy $E$ of a conformation $c$ for a protein design problem with $n$ mutable residue positions can be written as:

$$E(c) = E_{temp} + \sum_{i=1}^{n} (E(c_i) + \sum_{j>i} E(c_i, c_j))$$  \hspace{1cm} (6.1)

where $E_{temp}$ is the energy of the portion of the protein structure that is kept static during the design process (called the template energy), $E(c_i)$ is the energy of the conformation at residue $i$ with the protein template (called the intra-rotamer energy), and $E(c_i, c_j)$ is the interaction energy between conformations at residue $i$ and $j$ (called the inter-rotamer energy).

### 6.2.2 Dead-end elimination

Dead-end elimination (DEE) is a provable algorithm that is used in OSPREY to reduce the search space for the design problem. Using a pair-wise energy function, DEE is used to prune out rotamers at each residue position that are provably not part of the GMEC (Desmet et al., 1992). Let $\mathcal{r}_i$ be one possible rotamer at residue position $i$ for a protein design problem with $n$ mutable residue positions. Rotamer $\mathcal{r}_i$ can never be part of the GMEC if the minimum energy contributed by it to the GMEC is greater than the maximum energy contributed by any other rotamer at residue $i$. Therefore, rotamer $\mathcal{r}_i$ can be pruned if there exists a rotamer $\mathcal{t}_i$ for residue $i$ that satisfies the following condition:
If all conformations within an energy window $E_w$ of the GMEC are to be returned, Eq. 6.2 can be modified to prune rotamers that will provably not be part of the GMEC or any conformation within $E_w$ of the GMEC.

\[
E(r_i) + \sum_{j \neq i}^{n} \min_s E(r_i, s_j) > E(t_i) + \sum_{j \neq i}^{n} \max_s E(t_i, s_j) + E_w
\]  

(6.3)

Various extensions to the original DEE pruning criteria are used in osprey for effective and efficient pruning of rotamers (Goldstein, 1994; Pierce et al., 2000; Lasters et al., 1995; Gordon and Mayo, 1998). DEE is run multiple times (as pruning at one residue depends on the available rotamers at other residues) until no further rotamers can be pruned. If more than one conformation is left after DEE, then conformations are enumerated by the A* search algorithm (see Section 6.2.3). Running DEE before enumerating conformations is an efficient way to reduce the search space, as DEE is run in polynomial-time of $O(n^2 q^3)$, where $q$ is the maximum number of rotamers allowed per residue position.

6.2.3 A* search

If more than one conformation is left after DEE pruning is finished, the conformations need to be enumerated to find the GMEC. This is done by the A* search algorithm in osprey. A* is a branch and bound search algorithm that guarantees that the first conformation it returns is the GMEC (Leach and Lemon, 1998). A* expands a conformation search tree, with each level in the tree representing a residue position. Each node $p$ at level $i$ in the tree represents a partial conformation, where the first $i$ residues have been assigned a particular rotamer, and residues $i+1$ to $n$ have yet to
be assigned a rotamer. Each partial conformation is scored based on a function $f(p)$, that is the sum of the energy contributed by the rotamers that have been assigned ($g(p)$), and a lower bound on the energy contributed by the unassigned rotamers ($h(p)$).

$$f(p) = g(p) + h(p) \quad (6.4)$$

Let $p_x$ be the assigned rotamer at residue position $x$ in the partial conformation at node $p$, and $q_j$ be a possible rotamer at residue position $j$ whose rotamer assignment is not finalized, then

$$g(p) = E_{templ} + \sum_{x=1}^{i} (E(p_x) + \sum_{g>x}^{i} E(p_x,p_y)) \quad (6.5)$$

$$h(p) = \sum_{j=i+1}^{n} \min_{q_j}(E(q_j)) + \sum_{x=1}^{i} E(q_j,p_x) + \sum_{k=j+1}^{n} \min_{q_k}(E(q_j,q_k)) \quad (6.6)$$

The algorithm maintains a priority queue of all nodes in the tree, ordered based on the value of the function $f$. At every step, the node with the lowest value of function $f$ (the head node) is removed from the queue. If the head node is at level $i$, child nodes are created for each possible rotamer at residue position $i + 1$, value of function $f$ of each of those nodes is calculated and they are inserted back into the priority queue.

If the level of the head node is $n$, then all residues have been assigned a rotamer and hence the conformation at this node is complete. The value of function $f$ for this node is the actual energy for the complete conformation. As the function $f$ is a lower bound on the actual energy of this conformation, the energy of the this complete conformation is lower than the lower bound on all conformations in the priority queue, making this conformation the GMEC for the design problem. After
A* returns the GMEC, it can continue to enumerate a gap-free list of conformations in increasing order of energy. Parallelized version of A* search (Georgiev et al., 2006) and implementation of A* search on GPU (Zhou et al., 2014) within OSPREY have also been made available.

6.2.4 Side-chain and Backbone Flexibility

A known practice in protein design is to energy minimize the GMEC obtained as the final result of the design process. Traditional DEE equations (Eq. 6.2) treat side-chain rotamers as rigid and discrete, and therefore can incorrectly prune rotamers that may be part of the minimized GMEC. A high-energy, unfavorable rotamer that is pruned by DEE can minimize into a low-energy conformation. In order to prevent this and to make sure that no rotamer that can minimize into a favorable low-energy conformations is pruned, DEE needs to be made minimization aware.

OSPREY uses a modified version of the DEE equation to account for minimization of rotamers during the design process. The minimized DEE, or minDEE (Georgiev et al., 2008b) allows each $\chi$ angle on the protein side chain to minimize within a voxel of $\pm 9^\circ$ from the ideal rotamer position. The minimum and maximum value of the intra-rotamer energy (for every rotamer), and the inter-rotamer energy (for every pair of rotamers) is calculated within this voxel, and is used to prune rotamers that are provably not part of the minimized GMEC. After the minDEE step, A* enumerates conformations in increasing order of the lower-bound on their minimized energy, that are then energy minimized and a running record of the minimized GMEC is kept. When the lower-bound on the minimized energy of the latest conformation enumerated by A* exceeds the energy of the current minimized GMEC, A* search terminates and the minimized GMEC is returned.

Using the minDEE criterion instead of traditional DEE prevents the pruning of any rotamer part of the minimized GMEC, but leads to a reduction in the number
of rotamers that are pruned, increasing the search space. A modified version of the minDEE criterion, called the \textit{iMinDEE} (Gainza et al., 2012) has recently been implemented in \textsc{osprey} that provides better pruning, while maintaining all the provable guarantees. The \textit{iMinDEE} study also shows that there are large sequence differences between the GMEC obtained by accounting for minimization during the design process and the GMEC obtained by doing a rigid-rotamer design and minimizing the structure afterwards.

\textsc{osprey} also models continuous flexibility of the protein backbone during the design process. A modified version of DEE, \textit{BD} (Georgiev and Donald, 2007) accounts for small variations in the $\phi$ and $\psi$ angles to allow small multiple correlated shifts across multiple residues, and \textit{BRDEE} (Georgiev et al., 2008a) allows the protein backbone to undergo backrub-like local motions during the design process. A method called \textit{deeper} (Hallen et al., 2013) is also available in \textsc{osprey} that accounts for perturbations in both the protein backbone and side chain.

6.2.5 \textit{K*} score

To approximate the binding constant $K_a$, \textsc{osprey} calculates what is called a \textit{K* score} (Lilien et al., 2005). For a protein $P$ and a ligand $L$, the \textit{K*} score is calculated as:

\begin{equation}
K^* = \frac{q_{PL}}{q_{P}q_{L}}
\end{equation}

where $q_{PL}$ is the Boltzmann-weighted partition function for the bound protein-ligand complex, and $q_{P}$ and $q_{L}$ are the partition functions for the unbound protein and the ligand respectively. Each partition function is calculated using the low-energy thermodynamic ensemble of conformations returned by the A* search algorithm for the bound complex and the unbound partners. For example, the partition function $q_{P}$ for the unbound protein is calculated as:
where $p$ is one conformation in the ensemble, $E_p$ is the energy (or the minimized energy) of conformation $p$, $R$ is the gas constant, and $T$ is the temperature.

The computed $K^*$ score is guaranteed to be an $\epsilon$-approximation to the binding constant $K_a$, with $\epsilon$ being specified by the user. Since the conformations are Boltzmann-weighted to calculate the partition function, the lower the energy of the conformation, the higher its contribution to the partition function. As A* search enumerates a gap-free list of conformations in increasing order of energy, it can generate conformations until the computed $K^*$ score is $\epsilon$-accurate. This $K^*$ score is used to rank protein sequences or ligands, with a higher $K^*$ score considered to have higher binding affinity.

### 6.3 Protein Design with Sparse Residue Interaction Graph

Protein design algorithms have to evaluate a vast search space, which increases exponentially with the number of design positions, mutations and incorporation of realistic molecular flexibility. To get good design results with limited resources, it is important for design algorithms to methodically reduce the search space, without compromising on the quality and accuracy of design predictions. One way to do this is to use sparse residue interaction graphs for protein design, described below.

Most protein design algorithms use pair-wise energy functions to score protein conformations. Any such protein design problem can be represented by a residue interaction graph, where the nodes represent residues, and edges represent the interaction between residues. The energy functions usually consist of distance dependent terms to model vdw and electrostatic interaction between residue pairs, and the interaction energy decreases with increasing distance between residues. Therefore, it
is possible to neglect interaction energies between distant residues and not add them to the overall energy of a conformation. This leads to eliminating edges between these negligibly-interacting residues from the residue interaction graph to construct a sparse residue interaction graph.

Whether explicitly acknowledged or not, the concept of sparse residue interaction graphs is ubiquitous in the field of protein design, whether explicitly acknowledged or not. Many design algorithms apply appropriate distance cutoffs implicitly in the energy function (with different cutoffs for different kinds of energies calculated) (Jones, 1994; Koehl and Delarue, 1994; Desjarlais and Handel, 1995; Jiang et al., 2000; Desmet et al., 2002; Kortemme et al., 2003; Kingsford et al., 2005; Leaver-Fay et al., 2011; Privett et al., 2012). As the number of interacting residue pairs is less than all pairs of mutable residues, using sparse residue interaction graphs reduces the effective search space considerably. The conformations of these non-interacting residue pairs do not directly depend on each other, and can be determined independently. Protein design algorithms have been developed that take explicit advantage of this substructure introduced by sparse residue interaction graphs (Canutescu et al., 2003; Leaver-Fay et al., 2004; Xu and Berger, 2006; Krivov et al., 2009; Xu, 2005).

Section 6.3.1 describes the background definitions related to sparse residue interaction graphs (Georgiev, 2009; Donald, 2011) that are used in Chapters 7 and 8, which provide details about my work on analyzing the consequences of using sparse residue interaction graphs in protein design and developing novel ensemble-based algorithms for protein design with sparse residue interaction graphs.

6.3.1 Definitions related to Sparse Residue Interaction Graph

For a protein design with \( n \) mutable residues, let \( i \) and \( j \) denote two residues, and for a given conformation \( c \), let \( E(c, c) \) be the pairwise energy between them. The energy of a given the conformation \( c \) is represented in Eq. 6.9, when the self-energy
Figure 6.2: (a) Cobrotoxin protein (PDB ID: 1V6P) with the wild-type side chains of the 8 core mutable residues shown in blue. (b) Design problem in (a) represented as a full residue interaction graph where all pairs of residues interact. (c) Design problem in (a) represented as a sparse residue interaction graph generated with a distance cutoff $\delta = 8 \text{ Å}$.

of the residues can be included in the pairwise energy terms.

$$E(c) = \sum_{j \geq 1} E(c_i, c_j).$$  \hspace{1cm} (6.9)

Such a protein design problem can be represented as an undirected complete graph $G = (V, E)$, where $V$ represents all mutable residue positions ($|V| = n$), and $E$ represents all possible pairwise residue interactions ($|E| = \binom{n}{2}$). I will refer to the graph $G$ as a full residue interaction graph, the energy of any conformation calculated using Eq. (6.9) as its full energy, and the protein conformation that minimizes the full energy over all possible rotamer assignments as the full GMEC.

Let $E'$ be the set of edges deleted from $G$ to generate the sparse residue interaction graph $G' = (V, E - E')$. I will use the method described in (Georgiev, 2009) to generate the sparse residue interaction graph. If $r_i$ and $s_j$ represent rotamers $r$ and $s$ at residues $i$ and $j$ respectively, an edge between residue $i$ and $j$ is deleted from the graph $G$ if it meets any of the following two conditions:

- $d_{\text{min}}(i, j) > \delta$, where $d_{\text{min}}(i, j) = \min_{r,s} d(r_i, s_j)$ is the closest Euclidean distance between any two atoms in residues $i$ and $j$ when all rotamer combinations for
these two residues are considered; $\delta$ is the distance cutoff parameter;

- $E_{\text{max}}(i, j) < \alpha$, where $E_{\text{max}}(i, j) = \max_{r,s} |E(r_i, s_j)|$ is the maximum energy (in absolute value) between residues $i$ and $j$ when all rotamer combinations for these two residues are considered; $\alpha$ is the interaction energy cutoff parameter.

With appropriate distance and energy cutoffs, the residue interaction graph $G' = (V, E - E')$ is a sparse graph with the edges $E'$ deleted. Eq. (6.9) for the energy of a given conformation $c$ can now be re-written as:

$$E'(c) = \sum_{j>i} E(c_i, c_j) - \sum_{j>i, (i,j) \in E'} E(c_i, c_j). \quad (6.10)$$

I will refer to the energy of any conformation as calculated using Eq. (6.10) as its sparse energy, and the protein conformation that minimizes the sparse energy over all possible rotamer assignments as the sparse GMEC. Fig. 6.2 shows the full and sparse residue interaction graphs for a protein design problem with 8 mutable residues.
A Critical Analysis of Protein Design with Sparse Residue Interaction Graphs

This chapter provides details on my work to study the pros and cons of using sparse residue interaction graphs in protein design, as defined in Chapter 6. Section 7.1 consists of the proofs of the upper bounds on the energy difference between the GMECs of the full and the sparse residue interaction graphs, Section 7.2 describes my changes to the energy function in the A* search algorithm, which now can be the Sparse A* algorithm used to return the sparse GMEC and a gap-free list of conformations in order of sparse energy; Section 7.3 provides the analysis of the energy and sequence differences caused by using different distance and energy cutoffs for protein design problems with core, boundary and surface residues; and Section 7.4 describes a novel approach where sparse residue interaction graphs combined with ensemble-based design algorithms can be used to generate the full GMEC. This work is in collaboration with a Jonathan Jou, a graduate student in the Donald lab.

As mentioned in Chapter 6, using sparse residue interaction graphs reduces the effective search space for a protein design problem considerably. However, the energies omitted by deleting the edges between negligibly interacting residues can add
up, causing differences in sequence of the GMEC returned or the rankings of the top sequences. While small differences might not be detrimental, larger differences in energies and sequences can lead to design algorithms returning a protein sequence that may not have the desired function. Therefore there is a trade off between reducing the search space and guaranteeing the accuracy and quality of the computed GMEC. In spite of the widespread use of sparse residue interaction graphs in protein design, the effects of this trade-off have not been analyzed.

There are two main questions to be answered when using sparse residue interaction graphs. First, can the energy difference caused by the neglected pair-wise interactions between the full and the sparse GMEC be bounded? Second, how do the sequence and energy differences between the full and the sparse GMEC depend on the different distance and energy cutoffs and on the types of residue involved in the design problem. This study aims to answer the above two questions. Section 7.1 gives the proof of an upper bound on the energy difference between the full and the sparse GMEC, and Sections 7.2 and 7.3 describe computational experiments, and their results, on analyzing the differences between the full and the sparse GMEC.

7.1 Energy Bounds

As described in Section 6.3.1, given a distance cutoff $\delta$ and an energy cutoff $\alpha$, a sparse residue interaction graph $G'$ can be generated by deleting the set of edges $E'$ from the full residue interaction graph $G$. Because certain interaction energies are omitted from the sparse residue interaction graph, the full GMEC $c^*$ (conformation with minimum full energy in Eq. 6.9) and the sparse GMEC $c'$ (conformation with minimum sparse energy in Eq. 6.10) may be different. It would be useful to upper bound the energy difference between the full and the sparse GMEC. If $r_i$ and $s_j$ represent rotamer $r$ and $s$ at residues $i$ and $j$ respectively, let $E'_{max}$ and $E'_{min}$ be defined as the respective sums of the maximal and minimal energies contributed by
each edge \( e \in E' \) eliminated from \( G \).

\[
E'_{\text{max}} = \sum_{\substack{j > i \in E' \setminus \{i,j\}}} \max_{r,s} E(r_i, s_j).
\]  
\( (7.1) \)

\[
E'_{\text{min}} = \sum_{\substack{j > i \in E' \setminus \{i,j\}}} \min_{r,s} E(r_i, s_j).
\]  
\( (7.2) \)

The following two lemmas bound the difference in the full and sparse energies of the full and the sparse GMEC.

**Lemma 1.** Given a full residue interaction graph \( G \), sparse residue interaction graph \( G' \), full GMEC \( c^* \), and sparse GMEC \( c' \), the difference in full energy of \( c^* \) and \( c' \) is bounded as \(|E(c') - E(c^*)| \leq E'_{\text{max}} - E'_{\text{min}}\).

**Proof.** By definition, full GMEC \( c^* \) has the minimum full energy and the sparse GMEC \( c' \) has the minimum sparse energy. Hence,

\[
E(c^*) \leq E(c').
\]  
\( (7.3) \)

\[
E'(c') \leq E'(c^*).
\]  
\( (7.4) \)

Define \( \delta'(c^*) = \sum_{\substack{j > i \in E' \setminus \{i,j\}}} E(i, j) \) to be the difference between the full energy and sparse energy of the full GMEC \( c^* \). \( \delta'(c') \) is defined accordingly. By definition both \( \delta'(c^*) \) and \( \delta'(c') \) lie in between \( E'_{\text{max}} \) and \( E'_{\text{min}} \). The sparse energy can now be expressed in terms of the full energy as follows:

\[
E'(c^*) = E(c^*) - \delta'(c).
\]  
\( (7.5) \)

\[
E'(c') = E(c') - \delta'(c').
\]  
\( (7.6) \)

Substituting Eq. (7.5) and Eq. (7.6) in Eq. (7.4), we obtain

\[
E(c') - \delta'(c') \leq E(c^*) - \delta'(c).
\]  
\( (7.7) \)
Rearranging,
\[ E(c') - E(c^*) \leq \delta'(c') - \delta'(c). \] (7.8)
Substituting \( \delta'(c) \) by \( E_{\text{min}}' \), and \( \delta'(c') \) by \( E_{\text{max}}' \) maintains the above inequality:
\[ E(c') - E(c^*) \leq E_{\text{max}}' - E_{\text{min}}'. \] (7.9)
As \( E(c') - E(c^*) \) will always be positive (from Eq. (7.3)), which means
\[ |E(c') - E(c^*)| \leq E_{\text{max}}' - E_{\text{min}}'. \] (7.10)

**Lemma 2.** Given a full residue interaction graph \( G \), a sparse residue interaction graph \( G' \), the full GMEC \( c^* \) and sparse GMEC \( c' \), the difference in sparse energy of \( c^* \) and \( c' \) is bounded by the relationship \( |E'(c^*) - E'(c')| \leq E_{\text{max}}' - E_{\text{min}}' \).

**Proof.** By definition, the full GMEC \( c^* \) has the minimum full energy and the sparse GMEC \( c' \) has the minimum sparse energy. Hence,
\[ E(c^*) \leq E(c'). \] (7.11)
\[ E'(c') \leq E'(c^*). \] (7.12)
Define \( \delta'(c^*) = \sum_{(i,j) \in E'} E(i,j) \) to be the difference between the full energy and sparse energy of the full GMEC \( c^* \). \( \delta'(c') \) is defined accordingly. By definition both \( \delta'(c^*) \) and \( \delta'(c') \) lie in between \( E_{\text{max}}' \) and \( E_{\text{min}}' \). The sparse energy can now be expressed in terms of the full energy as follows:
\[ E(c^*) = E'(c^*) + \delta'(c^*). \] (7.13)
\[ E(c') = E'(c') + \delta'(c'). \] (7.14)
Substituting Eq. (7.13) and Eq. (7.14) in Eq. (7.11), we obtain
\[ E'(c^*) + \delta'(c^*) \leq E'(c') + \delta'(c'). \] (7.15)
Substituting $\delta'(c^*)$ by $E'_{\text{min}}$, and $\delta'(c')$ by $E'_{\text{max}}$ maintains the inequality:

$$E'(c^*) + E'_{\text{min}} \leq E'(c') + E'_{\text{max}}.$$  \hfill (7.16)

Rearranging,

$$E'(c^*) - E'(c') \leq E'_{\text{max}} - E'_{\text{min}}.$$  \hfill (7.17)

As from Eq. (7.12), $E'(c^*) - E'(c')$ will always be positive, which means

$$|E'(c^*) - E'(c')| \leq E'_{\text{max}} - E'_{\text{min}}.$$  \hfill (7.18)

7.2 Computational Methods

7.2.1 Sparse A*

To analyze the effects of using distance and energy cutoffs, to generate the sparse GMEC, the energy calculation method in A* was modified to calculate the sparse energy of a conformation (Eq. 6.10). This variation of the A* search algorithm is called Sparse A*. Because of the modified energy function, Sparse A* evaluates conformations in order of sparse energy (instead of the full energy in traditional A* search), and can now be used for protein design with sparse residue interaction graphs. As Sparse A* retains all the guarantees provided by the A* search algorithm (because the algorithm is unmodified), the first conformation returned by Sparse A* is the sparse GMEC, and it can also return a gap-free list of conformations within $E_w$ of the sparse GMEC in increasing order of sparse energy. When $E_w \geq E'_{\text{max}} - E'_{\text{min}}$, the upper bound on the difference in sparse energy between the full and the sparse GMEC as calculated by Lemma 2, the gap-free list is guaranteed to contain the full GMEC.

Lemma 3. Given a protein design problem with n residues, the gap-free list of conformations generated by Sparse A* is guaranteed to contain the full GMEC if
$E_w \geq E'_\text{max} - E'_\text{min}$, and can be found in additional $O(kn^2)$ time, where $k$ is the number of conformations generated.

**Proof.** After the Sparse A* run ends and it returns $k$ conformations within the specified energy window $E_w$, the full energy for each of these $k$ conformations is calculated by adding back the energies contributed by each edge $e \in E'$ deleted from the full residue interaction graph (see Section 6.3.1). This is done in $O(kn^2)$ time. Next, the conformation with the minimum full energy (full GMEC) within the list of $k$ conformations is found in $O(k)$ time. □

### 7.2.2 Computational Experiments

To analyze the effects of using distance and energy cutoffs, Sparse A* was implemented in the Donald lab’s protein design algorithm package OSPREY v2.1 (Gainza et al., 2013). Computational experiments were performed on the following design problems to generate the full and the sparse GMEC, and subsequently the energy and sequence differences were analyzed.

- **Core designs:** 62 protein design problems, with number of mutable residues ranging from 4-15 (each residue allowed to mutate to 5-10 amino acids) were used as described in (Gainza et al., 2012).

- **Boundary designs:** PDB files for protein structures used in (Gainza et al., 2012) were run through Naccess (Hubbard and Thornton, 1993) to calculate the relative accessible surface area (RSA) for each residue. The residues with RSA between 20-50% were classified as boundary residues. Terminal residues, residues forming di-sulphide bonds, and prolines were not designed. 46 design problems were chosen and at most 20 residues were designed in each case. The residues were allowed to mutate to their wild-type identities and all amino acids except proline.

95
- **Surface designs:** PDB files for protein structures used in (Gainza et al., 2012) were run through Naccess (Hubbard and Thornton, 1993) to calculate the relative accessible surface area (RSA) for each residue. The residues with RSA between 50-80% were classified as surface residues. Terminal residues, residues forming di-sulphide bonds, and prolines were not designed. 28 design problems were chosen and at most 20 residues were designed in each case. The residues were allowed to mutate to their wild-type identities and all polar and charged amino acids.

In all experiments, the DEE pruning stage was followed by either A* with $E_w = 0$ to get the full GMEC, or the following two steps to generate the sparse GMEC and gap-free list of conformations: 1) Sparse residue interaction graph generation and calculation of energy bounds $E_b$ (Lemma 2) using a user-defined distance cutoff $\delta$ or energy cutoff $\alpha$, and 2) Sparse A* run with $E_w = E_b$ to generate the sparse GMEC and a gap-free list of conformations. Sparse A* was run until all conformations within $E_w$ of the sparse GMEC were returned, or until the number of conformations returned was 10,000. For each design problem, Sparse A* was run four times using the following distance or energy cutoffs:

- $\delta = 8$ Å;
- $\delta = 7$ Å;
- $\alpha = 0.1$ kcal/mol;
- $\alpha = 0.2$ kcal/mol.

All computations were performed on Intel Xeon processor nodes with number of cores ranging from 8-48, and processor speeds ranging from 2.40-2.66GHz. Each core design was given 10GB of memory, and each boundary and surface design was given
Figure 7.1: Number of unpruned conformations left after DEE vs. the percentage of edges deleted from the residue interaction graph. Two data points are plotted for each design problem: with distance cutoff $\delta=7$ Å (blue), and energy cutoff $\alpha=0.2$ kcal/mol (red). (a) 62 core design problems, (b) 46 boundary design problems, and (c) 28 surface design problems.

30 GB. The energy function consisted of the Amber van der Waals and electrostatic terms (Cornell et al., 1995) and the EEF1 pairwise implicit solvation model (Lazaridis and Karplus, 1999). A distance-dependent dielectric of 6 and a solvation energy scaling factor of 0.05 were used. The atomic van der Waals radii were scaled by a factor of 0.95. All designs were done keeping the backbone fixed and modeling side-chain flexibility using the modal values of rotamers from the Penultimate rotamer library (Lovell et al., 2000).

7.3 Results

Fig. 7.1 shows the number of unpruned conformations left after DEE against the percentage of edges deleted to construct the sparse residue interaction graph with distance cutoff $\delta=7$ Å, or energy cutoff $\alpha=0.2$ kcal/mol for core, boundary, and surface design problems. Using the same distance cutoff deletes a larger percentage of edges in boundary and surface designs than core designs. This is not unexpected, as the cores of protein structures are generally compact with tightly packed residues, whereas distances between residues in the boundary regions or on protein surfaces are
Figure 7.2: (a) 62 core protein design problems, (b) 46 boundary design problems, and (c) 28 surface design problems. Design problems where A* returned the full GMEC and Sparse A* returned the sparse and the full GMEC are shown in green. Design problems where A* ran out of memory (30GB) before returning the full GMEC and Sparse A* returned the sparse GMEC are shown in orange. Design problems where both A* and Sparse A* ran out of memory (30GB) before returning any conformation are shown in red.

larger. Using an energy cutoff deletes a larger percentage of edges than the distance cutoff in core designs, however this trend is reversed in boundary and surface designs with the energy cutoff deleting a smaller percentage of edges. This suggests that even though residues in the core of the protein are tightly packed, interaction energy between each residue pair is small, and hence even neglecting interaction energies less than 0.2 kcal/mol (less than the typical van der Waals interaction energy of 0.5-1 kcal/mol) results in omitting a large fraction (45-80%) of interacting residue pairs. This is not necessarily true for boundary or surface designs, where using the same energy cutoff results in deleting a smaller fraction (mostly less than 50%) of edges.

Out of the 62 core, 46 boundary, and 28 surface protein redesign problems, A* returned the full GMEC and Sparse A* returned the sparse GMEC (with all four distance and energy cutoffs given in Section 7.2) for all 62 core, 21 boundary, and 12 surface design problems, shown as green dots in Fig. 7.2. These design problems were used to study the effects of different distance and energy cutoffs. Three kinds
of differences were studied between the full and the sparse GMEC for each of the distance and energy cutoffs: rotamer, sequence, and energy. The following subsections discuss the details of this analysis. The protein design problems for which both A* and Sparse A* ran out of memory (red points) and for which only Sparse A* run finished (orange squares) are discussed in Section 7.4.

7.3.1 Core Redesign

For 62 core designs, Sparse A* with both distance cutoffs returned the sparse GMEC identical to the full GMEC for all 62 design problems, and in 59 design problems with energy cutoff $\alpha=0.1$ kcal/mol, and in 58 design problems with energy cutoff $\alpha=0.2$ kcal/mol. For the 4 core problems where the full GMEC was different from the sparse GMEC, the full GMEC was the second conformation returned by Sparse A*, and the energy difference between the full and the sparse GMEC was less than the energy cutoff of 0.2 kcal/mol. Out of these 4 design problems, two had sequence differences between the full and the sparse GMEC: the human sulphite oxidase cytochrome b5 domain (PDB ID: 1MJ4) and bacterial iron-sulphur protein (PDB ID: 3A38) had single amino acid differences between the full and the sparse GMEC (residue 50 for 1MJ4 and residue 26 for 3A38). In both cases, serine in the full GMEC was replaced by alanine in the sparse GMEC. Except for these two cases, distance and energy cutoffs did not have sequence changing effects on the GMEC returned for core designs.

7.3.2 Boundary Redesign

Unlike core designs, the number of boundary design problems where the sparse GMEC was identical to the full GMEC was higher for energy cutoffs than distance cutoffs, as shown in Fig. 7.3(a). The energy cutoff of $\alpha=0.1$ kcal/mol gave the best results, returning the sparse GMEC identical to the full GMEC in 19 out of the
Figure 7.3: Data is shown for 21 boundary design problems, for each of which Sparse A* was run with the following cutoffs: distance cutoff $\delta = 8 \text{ Å}$, $\delta = 7 \text{ Å}$, energy cutoff $\alpha = 0.1 \text{ kcal/mol}$ and $\alpha = 0.2 \text{ kcal/mol}$. Number of mutable residues in each design problem ranged from 10-20. (a) Number of design problems where full GMEC and sparse GMEC are identical (purple), and where the sequences of the full GMEC and sparse GMEC are identical (cyan). (b) Percentage of edges deleted from the residue interaction graph vs. the full energy difference between full GMEC and sparse GMEC. (c) Number of residues with different amino acids between the full GMEC and the sparse GMEC. $y$-axis value of 0 indicates that the sequences of the full GMEC and the sparse GMEC are identical.

21 boundary design problems. For problems where the full GMEC and the sparse GMEC were different, the full energy and sequence difference between the full and the sparse GMEC are larger for distance cutoffs than for energy cutoffs, (Figs. 7.3(b) and (c)), with the distance cutoffs introducing sequence differences in total of 24 residues as compared to 11 residues with energy cutoffs (Table 7.1). For a single design problem, this number can be as high as 6 (C-terminal domain of the Rous Sarcoma Virus capsid protein, PDB ID: 3G21), which is more than one-third of the 15 mutable residues for that design problem.
Table 7.1: Sequence differences between the full and the sparse GMEC for boundary design problems, for distance cutoff $\delta = 7$ Å, energy cutoff $\alpha = 0.2$ kcal/mol. Sequence differences common to both cutoffs are shown in black.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Amino acid in full GMEC</th>
<th>Amino acid in sparse GMEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1IQZ</td>
<td>Asp</td>
<td>Lys</td>
</tr>
<tr>
<td></td>
<td>Lys</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Phe</td>
<td>Lys</td>
</tr>
<tr>
<td>1PSR</td>
<td>Glu</td>
<td>Gln</td>
</tr>
<tr>
<td>1UCS</td>
<td>Ser</td>
<td>Ala</td>
</tr>
<tr>
<td></td>
<td>Met</td>
<td>Asn</td>
</tr>
<tr>
<td></td>
<td>Met</td>
<td>Arg</td>
</tr>
<tr>
<td>1VBW</td>
<td>Met</td>
<td>Glu</td>
</tr>
<tr>
<td>2B97</td>
<td>His</td>
<td>Lys</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Lys</td>
<td>Glu</td>
</tr>
<tr>
<td></td>
<td>Met</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Ile</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>His</td>
</tr>
<tr>
<td>2CS7</td>
<td>Asp</td>
<td>Arg</td>
</tr>
<tr>
<td>2DSX</td>
<td>Lys</td>
<td>Ile</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Arg</td>
</tr>
<tr>
<td>2ZXY</td>
<td>Met</td>
<td>Asp</td>
</tr>
<tr>
<td>3DNJ</td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Phe</td>
<td>Asn</td>
</tr>
<tr>
<td>3FIL</td>
<td>Leu</td>
<td>Met</td>
</tr>
<tr>
<td>3HFO</td>
<td>Ile</td>
<td>Arg</td>
</tr>
<tr>
<td>3G21</td>
<td>Tyr</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>Glu</td>
</tr>
<tr>
<td></td>
<td>Trp</td>
<td>Asp</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>Met</td>
</tr>
<tr>
<td>1C75</td>
<td>Trp</td>
<td>Phe</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Met</td>
</tr>
</tbody>
</table>
7.3.3 Surface Redesign

Similar to boundary design problems, the energy cutoffs fare better than distance cutoffs in returning the sparse GMEC identical to the full GMEC, although the difference in performance is much more prominent. The sparse GMEC is identical to the full GMEC in 10 out of the 12 surface design problems for energy cutoff $\alpha = 0.1 \text{ kcal/mol}$, and only for 1 out of 12 for distance cutoff $\delta = 7 \text{ Å}$ (Fig. 7.4(a)). Sequence differences between the full and the sparse GMEC occur even though the energy differences between these GMECs are small. Unlike boundary designs, where in a few cases the energy cutoffs introduced more sequence differences between the
full and the sparse GMEC, the energy cutoff of $\alpha=0.1$ kcal/mol has less or an equal number of sequence differences than distance cutoffs in all 12 surface design problems, as shown in Fig. 7.4(c). Overall, distance cutoffs introduced sequence differences in a total of 25 residues, as compared to 9 residues for energy cutoffs (Table 7.2).

Table 7.2: Sequence differences between the full and the sparse GMEC for surface design problems, for distance cutoff $\delta = 7$ Å, energy cutoff $\alpha = 0.2$ kcal/mol. Sequence differences common to both cutoffs are shown in black.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Amino acid in full GMEC</th>
<th>Amino acid in sparse GMEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C75</td>
<td>Lys</td>
<td>Glu</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Lys</td>
</tr>
<tr>
<td>1UCS</td>
<td>Glu</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>Glu</td>
</tr>
<tr>
<td>2CS7</td>
<td>Lys</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td>2FMA</td>
<td>Glu</td>
<td>Arg</td>
</tr>
<tr>
<td>2HLR</td>
<td>Tyr</td>
<td>Arg</td>
</tr>
<tr>
<td>2J8B</td>
<td>Glu</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Lys</td>
</tr>
<tr>
<td></td>
<td>Asp</td>
<td>Glu</td>
</tr>
<tr>
<td></td>
<td>Arg</td>
<td>His</td>
</tr>
<tr>
<td>2O9S</td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>His</td>
<td>Tyr</td>
</tr>
<tr>
<td>2RH2</td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td>3FIL</td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Asp</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>Lys</td>
<td>Arg</td>
</tr>
<tr>
<td>1IQZ</td>
<td>Tyr</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>His</td>
<td>Arg</td>
</tr>
<tr>
<td></td>
<td>Glu</td>
<td>Thr</td>
</tr>
<tr>
<td></td>
<td>Lys</td>
<td>Arg</td>
</tr>
<tr>
<td>1V6P</td>
<td>Glu</td>
<td>His</td>
</tr>
<tr>
<td></td>
<td>His</td>
<td>Arg</td>
</tr>
</tbody>
</table>

Since distances between residues on the surface of the protein are larger as compared to boundary or core regions, Sparse A* was run for the 12 surface designs.
Figure 7.5: Distribution of amino-acid types for residues in the full GMEC but which were mutated in the sparse GMEC, for boundary and surface protein design problems with distance cutoff $\delta=7$ Å and energy cutoff $\alpha=0.2$ kcal/mol.

problems again with a distance cutoff $\delta=10$ Å. This resulted in the sparse GMEC identical to the full GMEC for 2 more design problems, Extracellular Domain of the Type II BMP Receptor (PDB ID: 2HLR) and Human signal recognition homology 3 domain (PDB ID: 2O9S), taking the total number to 5, although this the still only half of the number with energy cutoff $\alpha=0.1$ kcal/mol. Apart from the above two design problems, the number of residues with different amino acids in the sparse and the full GMEC reduces only for bacterial oxidized ferredoxin protein (PDB ID: 1IQZ), and the maximum number of 5 for human CD59 glycoprotein (PDB ID: 2J8B) does not change.
7.3.4 Large Long-range interactions neglected by distance cutoffs

The above results suggest that using distance cutoffs can neglect long range interactions between residue pairs, and cause significant sequence differences between the GMECs returned. Using a larger distance cutoff (10 Å) did little to improve the results. Neglecting these long range interactions tends to have a larger effect on boundary and surface designs than core designs. To investigate this further, the maximum energy (in absolute value) contributed over all rotamers pairs by each of the edges deleted using distance and energy cutoffs were analyzed. To eliminate any uncertainties, the results of core, boundary, and surface designs on the same protein structure were used for this analysis. For the 6 protein structures shown in Fig. 7.6, using distance cutoffs in boundary and surface designs can delete edges from the residue interaction graph with larger interaction energies, as compared to using energy cutoffs. The opposite occurs for core designs. This is consistent with the fact that both the distance cutoff $\delta=7$ Å and energy cutoff $\alpha=0.2$ kcal/mol had similar results for core designs, but for boundary and surface designs, the number of residues with different amino acids between the full and the sparse GMEC is larger for the distance cutoff than for the energy cutoff, except for bacterial cytochrome C-553 protein (PDB ID: 1C75). However, even though using energy cutoffs does not always return the sparse GMEC identical to the full GMEC, their effects do not vary hugely between boundary and surface designs (Table 7.3). By definition, energy cutoffs only delete edges with energy contributions within the defined limit, but this is not true for edges deleted by distance cutoffs whose energy contributions can vary arbitrarily from being very small (0.05 kcal/mol) to being very large (almost 0.9 kcal/mol).

The above analysis indicates that the effects of distance cutoffs range from introducing no sequence differences in core designs, to sequence differences in almost all surface designs. However, the effects of energy cutoffs are similar across core,
Figure 7.6: The maximum (in absolute value) value of interaction energy for each deleted edge with distance cutoff $\delta=7$ Å (blue) and energy cutoff $\alpha=0.2$ kcal/mol (red), for core, boundary, and surface designs for 6 protein structures.

Table 7.3: Sequence differences between the full and the sparse GMEC, and the rank of the full GMEC in the gap-free list of conformations returned by Sparse A* for selected core, boundary, and surface designs with distance cutoff $\delta=7$ Å, and energy cutoff $\alpha=0.1$ kcal/mol.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Rank of full GMEC Core $\delta=7 \alpha=0.2$</th>
<th>Rank of full GMEC Boundary $\delta=7 \alpha=0.2$</th>
<th>Rank of full GMEC Surface $\delta=7 \alpha=0.2$</th>
<th>Sequence Differences Core $\delta=7 \alpha=0.2$</th>
<th>Sequence Differences Boundary $\delta=7 \alpha=0.2$</th>
<th>Sequence Differences Surface $\delta=7 \alpha=0.2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C75</td>
<td>1 1</td>
<td>1 3</td>
<td>50 1</td>
<td>0 0</td>
<td>0 2</td>
<td>1 0</td>
</tr>
<tr>
<td>1G6X</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1UCS</td>
<td>1 1</td>
<td>8 1</td>
<td>3 16</td>
<td>0 0</td>
<td>3 0</td>
<td>1 1</td>
</tr>
<tr>
<td>2CS7</td>
<td>1 1</td>
<td>11 8</td>
<td>37 4</td>
<td>0 0</td>
<td>1 1</td>
<td>3 2</td>
</tr>
<tr>
<td>3FIL</td>
<td>1 1</td>
<td>2 1</td>
<td>57 1</td>
<td>0 0</td>
<td>1 0</td>
<td>3 0</td>
</tr>
<tr>
<td>1IQZ</td>
<td>1 1</td>
<td>5062 17</td>
<td>2231 26</td>
<td>0 0</td>
<td>3 1</td>
<td>3 2</td>
</tr>
</tbody>
</table>

boundary, and surface designs, and hence energy cutoffs are better suited for general designs involving all three types of residues.
Figure 7.7: Bounds on the full energy difference (as calculated by Lemma 1) vs. the actual full energy difference between the full GMEC and sparse GMEC for distance cutoff $\delta=7$ Å (blue) and energy cutoff $\alpha=0.2$ kcal/mol (red). (a) 62 core protein design problems, (b) 21 boundary protein design problems, (c) 12 surface protein design problems.

7.4 Novel use of Sparse Graphs and Ensemble-based provable algorithms

Fig. 7.7 plots the calculated upper bounds vs. the actual full energy difference between the full and the sparse GMEC for the core, boundary, and surface designs. What is evident from the difference in the scale of the two axes, the actual energy difference between the full and the sparse GMEC (ranges from 0.05 kcal/mol to 1.6 kcal/mol) is an order of magnitude smaller than the computed upper bound, which can be as high as 40 kcal/mol. This resulted in the fact that Sparse A* returned the full GMEC relatively early, much earlier than all conformations within the upper bound of the sparse GMEC were guaranteed to be enumerated. The full GMEC was found within the first 20 conformations of the sparse GMEC for all 21 boundary design problems with energy cutoffs, and for 19 design problems with distance cutoffs. For the two remaining problems, clps protease adaptor protein (PDB ID: 3DNJ) and bacterial ferredoxin protein (PDB ID: 1IQZ), the full GMEC was the 168th and 5062nd conformation returned by Sparse A* respectively with distance cutoff $\delta=7$ Å.
Table 7.4: Rank of the full GMEC in the gap-free list of conformations generated by Sparse A* for 21 boundary protein design problems, with distance cutoffs $\delta=7$ Å and $\delta=8$ Å, and energy cutoffs $\alpha=0.1$ kcal/mol and $\alpha=0.2$ kcal/mol

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>$\delta=7$ Å</th>
<th>$\delta=8$ Å</th>
<th>$\alpha=0.1$ kcal/mol</th>
<th>$\alpha=0.2$ kcal/mol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1G6X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1H27</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1IQZ</td>
<td>5062</td>
<td>1448</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>1OAI</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1OK0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1PSR</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1TUK</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1UCS</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1VBW</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2B97</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2BWF</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2CS7</td>
<td>11</td>
<td>11</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2DSX</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2HIN</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2IC6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2ZXY</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3DNJ</td>
<td>168</td>
<td>15</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3FIL</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3HFO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3G21</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1C75</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Å. The rank of the full GMEC in the gap-free list of conformations enumerated by Sparse A* for all 21 boundary design problems are given in Table 7.4. For the 12 surface design problems, the full GMEC is within the first 30 conformations of the sparse GMEC for energy cutoffs, but for distance cutoffs, this number is on the order of a few hundreds for some of the design problems. The rank of the full GMEC in the gap-free list of conformations enumerated by Sparse A* for all 12 surface design problems are given in Table 7.5.

Overall, because of the small energy difference, the full GMEC was found by enumerating only the first 1000 conformations returned by Sparse A* (with both distance and energy cutoffs) for all but one protein design problem. This shows that
Table 7.5: Rank of the full GMEC in the gap-free list of conformations generated by Sparse A* for 12 surface protein design problems, with distance cutoffs $\delta = 7 \, \text{Å}$ and $\delta = 8 \, \text{Å}$, and energy cutoffs $\alpha = 0.1 \, \text{kcal/mol}$ and $\alpha = 0.2 \, \text{kcal/mol}$

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>$\delta = 7 , \text{Å}$</th>
<th>$\delta = 8 , \text{Å}$</th>
<th>$\alpha = 0.1 , \text{kcal/mol}$</th>
<th>$\alpha = 0.2 , \text{kcal/mol}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C75</td>
<td>50</td>
<td>97</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1G6X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1UCS</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>2CS7</td>
<td>37</td>
<td>64</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2FMA</td>
<td>18</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2HLR</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2J8B</td>
<td>286</td>
<td>286</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2O9S</td>
<td>840</td>
<td>943</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2RH2</td>
<td>46</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3FIL</td>
<td>57</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1IQZ</td>
<td>2231</td>
<td>213</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>1V6P</td>
<td>14</td>
<td>12</td>
<td>5</td>
<td>22</td>
</tr>
</tbody>
</table>

even when limited time and memory prevent Sparse A* from provably enumerating the full GMEC (because of loose energy bounds), in practice the number of conformations that must be enumerated before Sparse A* returns the full GMEC can be small. This provides useful information that can be used to compute the full GMEC using Sparse A* for design problems where A* fails. This is highlighted by the 3 boundary and 1 surface design problem for which Sparse A* with distance cutoff $\delta = 7 \, \text{Å}$ (because this deletes the maximum number of edges) returned the sparse GMEC and which A* ran out of 30 GB of memory before returning the full GMEC (orange points in Fig. 7.2). Sparse A* returned the sparse GMEC along with a gap-free list of conformations for 3 boundary designs (heterogeneous nuclear ribonucleoprotein K (PDB ID: 1ZZK), Beta-elicitin cinnamomin (PDB ID: 2AIB), Dihydrofolate reductase type 2 (PDB ID: 2RH2), and for 1 surface design of scorpion toxin protein (PDB ID: 1AHO)). The number of conformations enumerated by Sparse A* was 47 for 1ZZK, 3029 for 2AIB, 46 for 2RH2, and 10,000 for 1AHO. Given the results that with distance cutoff $\delta = 7 \, \text{Å}$ the full GMEC can be found almost always within the first 30 conformations for boundary designs, and within 1000 conformations for sur-
face designs, the gap-free list computed by Sparse A* for the above 4 protein design problems almost certainly contains the full GMEC.

Because the number of conformations that must be enumerated by Sparse A* to find the full GMEC can be small, the deleterious effects of using sparse residue interaction graphs can be mitigated by computing a gap-free, in-order list of conformations that contains the full GMEC. This provides a novel way for provable, ensemble-based algorithms and sparse residue interaction graphs to efficiently compute the full GMEC for previously intractable design problems.
This chapter describes my work on developing a novel branch-decomposition based algorithm, BWM, for protein design with sparse residue interaction graphs. Section 8.1 describes the previous work done to develop the initial BWM algorithm, Section 8.2 describes my work on improving the time complexity of the BWM algorithm, Section 8.3 provides the empirical comparison of run times of A*, Sparse A*, and BWM algorithm, and Section 8.4 describes the extension of the BWM algorithm for generating a gap-free list of conformations. This work is in collaboration with Jonathan D. Jou and Dr. Ivelin Georgiev, and partly adapted from


As mentioned in Chapter 6, using sparse residue interaction graphs in protein design make the conformations at certain residues not directly dependent on the
conformation of certain other residues. The conformations of such groups of residues can be calculated independently and can be looked up later to return the complete conformation. This divides the design problem into smaller sub-problems that can be solved independently and combined later, making it suitable for a dynamic-programming setup. Taking advantage of this, protein design algorithms based on the concepts of tree-decomposition and tree-width have been developed for sparse residue interaction graphs that compute the sparse GMEC (Canutescu et al., 2003; Leaver-Fay et al., 2004; Xu and Berger, 2006; Krivov et al., 2009; Xu, 2005). However, these algorithms compute only the sparse GMEC and do not return an ensemble of low-energy conformations, which is necessary not only for better results (see Section 6.1.1), but also to take advantage of sparse residue interaction graphs for computing the full GMEC (see Section 7.4).

The Sparse A* algorithm presented in Chapter 7 does return a gap-free list of conformations, but merely modifies the energy function for the A* algorithm and does not take explicit advantage of the substructure introduced by the sparse residue interaction graphs. Moreover, A* does not guarantee any improvement over exhaustive search. When the GMEC is computed, A* conformation enumeration tree consists of one fully assigned conformation and a large set of partially assigned conformations (a conformation where not all residue positions have been assigned rotamers). A* provides no guarantees on the number of and size of these partially assigned conformations that have to be enumerated before it returns the GMEC. In the worst case, A* must explore a significant portion of the search space to guarantee that the first conformation returned is the GMEC, and each successive conformation returned may require significant further exploration of the search space. Therefore, computing an ensemble of low-energy conformations can be prohibitively expensive.

Branch-decomposition is a concept related to tree decomposition and is widely used in the field of optimization problems in computer science (Cook and Seymour,
2003). Ivelin Georgiev, a former graduate student in the Donald lab had developed a branch-decomposition based dynamic programming algorithm, called Branch-Width Minimization (BWM), to generate the GMEC for a sparse residue interaction graph (Georgiev, 2009), an overview of which is provided in the section below.

8.1 Definitions and Previous Work

8.1.1 Branch-Decomposition and Branch-Width

Let \( G(V, E) \) be a graph. Let \( T(P, Q) \) be a tree, such that every edge in \( E \) corresponds to a leaf node in \( T \), and the degree of every internal node of \( T \) is 3. Such a tree \( T \) is defined to be a branch-decomposition of graph \( G \).

Deleting a tree edge \( e \in Q \) splits \( T \) into two subtrees, arbitrary called the left and the right subtrees of tree edge \( e \). Deleting \( e \) from \( T \) also separates the set of graph vertices \( V \) into three non-overlapping, mutually-exhaustive sets. Set \( L(e) \) contains graph vertices in \( V \) that are found only in the leaf nodes of the left subtree of tree edge \( e \), set \( R(e) \) contains graph vertices in \( V \) that are found only in the leaf nodes of the right subtree of tree edge \( e \), and set \( M(e) \) contains graph vertices in \( V \) that are found in the leaf nodes of both the left and the right subtrees of tree edge \( e \).

The width of the branch-decomposition \( T \) is then defined as:

\[
    w = \max_{e \in Q} |M(e)|
\]  

(8.1)

The branch-decomposition with the minimum value of \( w \) is called the optimal branch-decomposition, and its width is called the branch-width.

8.1.2 Initial GMEC-based BWM Algorithm

Let \( G'(V, E) \) be a sparse residue interaction graph generated as described in Section 6.3.1, and \( T(P, Q) \) be the branch-decomposition of \( G' \) with width \( w \). The value of \( w \) in comparison to the number of mutable residues \( n \) gives an indication of the
sparseness of the residue interaction graph, i.e. a lower value of \( w \) indicates a sparser graph. As the set of graph edges \( \mathcal{E} \) represent the pair-wise interaction between the mutable residues in the design problem, the leaf nodes of \( T \) also correspond to these pair-wise interactions. Therefore for any tree edge \( e \in Q \), the sets \( L(e) \) and \( R(e) \) (constructed as described above) represent residue sets that do not interact with each other, and only interact with the residues in the \( M(e) \) set. Hence the optimal conformation for each residue in \( L(e) \) depends only on the conformations of other residues in \( L(e) \) and \( M(e) \), and can be computed independent of the residues in \( R(e) \).

The BWM algorithm takes as input the branch-decomposition \( T \) and returns the sparse GMEC for the sparse graph \( G' \). The algorithm proceeds by rooting the branch-decomposition \( T \), by creating a root node and a root edge \( r \). This is done primarily to provide a parent-child relationship to the nodes of \( T \) (used in the subsequent steps) and to make sure that \( L(r) = V \) contains all the \( n \) mutable residues for the design problem. Then, a post-order traversal of rooted \( T \) tree is performed. For every tree edge \( e \in Q \), a set called the \( \lambda(e) \) is calculated that consists of residues that satisfy the following two conditions: residues must belong to \( L(e) \), and not belong to \( L(e) \) set of any other tree edge in the subtree rooted at \( e \). The following two lemmas prove important properties for any \( \lambda(e) \).

**Lemma 4.** For any tree edge \( e \in Q \), \( |\lambda(e)| \leq w \), where \( w \) is the width of the branch-decomposition \( T \).

**Proof.** Let \( T_e \) be the subtree of \( T \) rooted at edge \( e \), \( c_1 \) and \( c_2 \) be two child edges of \( e \), and \( x \in \lambda(e) \) be a residue position \( x \in V \). By definition of \( \lambda(e) \), \( x \in L(e) \), \( x \notin L(c_1) \), and \( x \notin L(c_2) \). Therefore \( x \) either belongs to \( R(c_1) \) or \( M(c_1) \) (as \( L(\cdot), R(\cdot), \) and \( M(\cdot) \) are mutually exhaustive sets). If \( x \in R(c_1) \), then \( x \) must either be in \( L(c_2) \) (which by definition is not true), or in the subtree that is formed by removing \( T_e \) from \( T \). However, since \( x \in L(e) \), then \( x \) should only be found in a leaf node in \( T_e \). This leads
to a contradiction, so \( x \notin R(c_1) \), and analogously, \( x \notin R(c_2) \). Hence, for any \( x \in \lambda(e) \), \( x \) belongs to both \( M(c_1) \) and \( M(c_2) \), which implies \( \lambda(e) = M(c_1) \cap M(c_2) \). Since, by definition, \( M(\cdot) \leq w \) (Eq. 8.1), \( |\lambda(e)| \leq w \).

Lemma 5. For each residue position \( x \in V \), there is exactly one tree edge \( e \in Q \), such that \( x \in \lambda(e) \).

Proof. Let there be two tree edges \( e \in Q \) and \( f \in Q \) such that \( x \in \lambda(e) \) and \( x \in \lambda(f) \). Let \( T_e \) and \( T_f \) be the two subtrees of \( T \) rooted at \( e \) and \( f \) respectively. Since by definition of \( \lambda(\cdot) \), \( x \in L(e) \), and hence \( x \) can only be found in the leaf nodes of \( T_e \). Therefore for \( x \in L(f) \) (also by definition of \( \lambda(\cdot) \)) to be true would require either \( f \) to be a edge in subtree \( T_e \), or \( e \) to be an edge in subtree \( T_f \). However, this violates the second condition for \( x \) to belong to either \( \lambda(e) \) or \( \lambda(f) \). This leads to a contradiction, so there must be exactly one tree edge \( e \) with the desired property.

According to Lemma 5, if \( x \in \lambda(e) \), then all pair-wise interactions of \( x \) are confined to the subtree \( T_e \). Therefore the optimal conformations of \( x \) can be completely determined at tree edge \( e \), and can be stored to be simply looked up later as the algorithm progresses to compute the sparse GMEC. This division of the overall problem to subproblems is what makes the BWM algorithm provably more efficient than exhaustive search.

To compute the sparse GMEC, the algorithm performs the following steps for every tree edge \( e \) visited during the post-order traversal of the branch-decomposition \( T \), where \( \lambda(e) \neq \emptyset \):

1. Generate all possible \( q^{|M(e)|} \) (\( q \) is the maximum number of rotamers at each residue position) possible rotamer assignments for residues in \( M(e) \).

2. For each rotamer assignment in \( M(e) \), compute the optimal rotamer assignment for residues in \( L(e) \) and the corresponding energy. This is done by generating
all possible $q^{|\lambda(e)|}$ rotamer assignments for residues in $\lambda(e)$, and looking up the optimal assignment for residues in $L(e) - \lambda(e)$ that have been previously computed when the post-order traversal visited the edges in the subtree rooted at $e$.

3. For all possible rotamer assignments in $M(e)$, store the corresponding optimal assignment of residues in $\lambda(e)$ (size $|\lambda(e)|$) and the energy computed in step 2 in a matrix $A_e$, of size $q^{|M(e)|}X(\lfloor |\lambda(e)| + 1 \rfloor)$. This matrix is used for looking up optimal assignments in step 2 when the post-order traversal visits other edges.

When the post-order traversal completes the above steps for the root edge $r$, the optimal assignment and energy have been computed for $L(r)$. Since $L(r)$ contains all mutable residue positions in the protein design problem, this optimal assignment is the sparse GMEC. The time complexity of the BWM algorithm is $O(\beta_t + n^5q^{2w})$, where $\beta_t$ is the time taken to compute a branch decomposition $T$ of graph $G'$, $n$ is the number of residues to be designed, $q$ is the maximum number of rotamers per residue position, and $w$ is the width of the branch-decomposition $T$. The space complexity of the BWM algorithm is $O(\beta_s + nwq^w)$, where $\beta_s$ is the space required to compute the branch-decomposition $T$.

8.2 Improvements to the GMEC-based BWM Algorithm

In the second term in the time complexity function for the BWM algorithm, $O(n^2)$ is the time taken to calculate the energy of any conformation of residues in $M(e) \cup L(e)$, $O(n^2)$ is the time taken by the back-tracking procedure to return the optimal conformation for residues in $L(e) - \lambda(e)$, and $O(q^{2w})$ is the time taken to enumerate all possible conformations in $M(e) \cup \lambda(e)$ when the computation of the $A_e$ matrix is being carried out for any tree edge $e$. The time complexity of BWM can be improved by re-using the already computed and stored energies, retrieving the actual rotamer
assignments for each residue only in the end, and proving tighter bounds on the size of \( M(e) \cup \lambda(e) \). The following sections describe the changes made in each of these categories to reduce the overall time complexity of the BWM algorithm.

8.2.1 Energy calculation at a tree edge

Let any tree edge \( e \) such that \( \lambda(e) \neq \emptyset \) be called a \( \lambda \) edge. This section gives details on how the energy calculation can be improved while computing the \( A_e \) matrix for any \( \lambda \) edge \( e \). The partial energy of the partial conformation \( p \) consisting of contributions by the residues in the set \( M(e) \cup L(e) \) can be defined as:

\[
E_e = \sum_{j \geq i; i,j \in M(e) \cup L(e)} E(p_i, p_j).
\]  

(8.2)

where \( E(p_i, p_j) \) is the pair-wise interaction energy between residues \( i \) and \( j \). Eq 8.2 can be broken down into two components as shown below:

\[
E_e = E_{e1} + E_{e2}
\]

(8.3)

where

\[
E_{e1} = \sum_{j \geq i; i,j \in M(e) \cup \lambda(e)} E(p_i, p_j).
\]

(8.4)

\[
E_{e2} = \sum_{i \in M(e) \cup \lambda(e) \setminus L(e) - \lambda(e)} E(p_i, p_j) + \sum_{j \geq i; j \in L(e) - \lambda(e)} E(p_i, p_j).
\]

(8.5)

Here \( E_{e2} \) represents the energy contributed by the residues in the set \( L(e) - \lambda(e) \).

The objective here is to minimize \( E_e \) (Eq 8.2) for each rotamer assignment in \( M(e) \). The initial algorithm did this by enumerating all possible rotamer assignments in \( \lambda(e) \) for every rotamer assignment in \( M(e) \), and back-tracking on the edges in the subtree \( T_e \) to return the optimal rotamer assignment for residues in \( L(e) - \lambda(e) \) for every rotamer assignment in \( M(e) \cup \lambda(e) \), and finally recomputing the energy \( E_e \). However, there is no need to re-calculate the pair-wise energies contributed by
residues in \( L(e) - \lambda(e) \) \( (E_{e2}) \), since they were already calculated and stored when the edges in the subtree rooted at \( e \) were visited as part of the post-order tree traversal. Hence, instead of returning the optimal rotamer assignments, the back-tracking procedure need only return the optimal energy for residues in \( L(e) - \lambda(e) \) \( (E_{e2}) \). The actual rotamer assignments for residues in \( L(e) - \lambda(e) \) are no longer required at this stage, and can be looked-up towards the end to return the conformation of the sparse GMEC (Section 8.2.3). The new energy calculations that need to be performed are only for residues in \( M(e) \cup \lambda(e) \) \( (E_{e1}) \). The size of both \( M(e) \) and \( \lambda(e) \) is bounded above by width \( w \), the time complexity for computing the value of \( E_e \) can be reduced from \( O(n^2) \) to \( O(w^2) \).

\( E_{e1} \) and \( E_{e2} \) can be added together to get the value of \( E_e \), which can then be used to determine the optimal rotamer assignment for residues in \( \lambda(e) \) for every rotamer assignment in \( M(e) \). This is stored in the \( A_e \) matrix as before, but the corresponding energy stored in the \( A_e \) matrix will now be the optimal energy contributed by the residues in \( L(e) \), instead of the value of \( E_e \) in the initial algorithm. This is to avoid double counting the interaction energies of residues in the \( M(e) \) set with each other when this value is returned by the back-tracking procedure called from other edges in the post-order traversal.

The correctness of new procedure described above is established in the following proposition.

**Proposition 6.** For each rotamer assignment in \( M(e) \), \( E_e \) is minimized.

**Proof.** For each rotamer assignment in \( M(e) \), all possible rotamer assignments in \( \lambda(e) \) are enumerated and the corresponding energy \( E_{e1} \) (Eq. 8.4) is computed. For every value of \( E_{e1} \), the back-tracking procedure returns the optimal/minimum value of energy \( E_{e2} \) (Eq. 8.5), as established in Proposition 9. As \( E_e \) is a sum of \( E_{e1} \) and \( E_{e2} \) (Eq. 8.3), the minimum value for \( E_e \) can be determined for each rotamer assignment.
8.2.2 The back-tracking procedure

This section gives details on the improvements made to the back-tracking procedure as a result of the changes made in Section 8.2.1.

For any \( \lambda \) edge \( e \), let \( F_e \) (as defined earlier in (Georgiev, 2009)) be the set of all \( \lambda \) edges in the subtree \( T_e \) such that the path from edge \( e \) to any edge \( f \in F_e \) does not pass through any other \( \lambda \) edge in \( T_e \). This implies that if two edges \( f, g \in F_e \), then \( g \) cannot be an edge in the subtree \( T_f \) rooted at \( f \), and vice versa. The following lemmas (partly derived earlier in (Georgiev, 2009)) establish some properties of the \( F_e \) set.

**Lemma 7.** For any residue \( x \in L(e) - \lambda(e) \), there is exactly one tree edge \( f \in F_e \), such that \( x \in L(f) \).

*Proof.* As \( x \notin \lambda(e) \) but \( x \in L(e) \), then by Lemma 5, there must exist some edge \( g \) in the subtree \( T_e \) such that \( x \in \lambda(g) \). By definition of \( \lambda(\cdot) \), if \( x \in \lambda(g) \) then \( x \in L(g) \). Consider two cases: \( g \in F_e \), and \( g \notin F_e \). If the first case is true then it is already proved that \( x \in L(g) \). If the second case is true, then by definition of \( F(\cdot) \), \( g \) must be a \( \lambda \) edge in the subtree \( T_f \) for any \( f \in F_e \), and would require for any residue \( x \in L(g) \) to also \( x \in L(f) \). Therefore there is at least one edge \( f \in F_e \) such that \( x \in L(f) \).

Let there be two edges \( f, g \in F_e \) such that \( x \in L(f) \), as well as \( x \in L(g) \). This means that \( L(f) \cap L(g) \neq \emptyset \). Therefore by definition of \( L(\cdot) \), either \( g \) is an edge in the subtree \( T_f \), or \( f \) is an edge in the subtree \( T_g \). But this would violate the condition for both \( f \) and \( g \) to be in \( F_e \). This leads to a contradiction, so there is exactly one edge satisfying the desired property. \( \Box \)

**Lemma 8.** For a given tree edge \( e \), \( M(f) \subseteq M(e) \cup \lambda(e) \), \( \forall f \in F_e \).
Proof. By definition of $M(\cdot)$, $L(\cdot)$, and $F_e$, $M(f) \subseteq M(e) \cup L(e)$. From Lemma 7, if $x \in L(e) - \lambda(e)$, then $x \in L(f)$. Therefore $x \notin M(f)$, which means $(L(e) - \lambda(e)) \cap M(f) = \emptyset$. Therefore $M(f) \subseteq M(e) \cup \lambda(e)$. □

The goal of the back-tracking procedure in the initial BWM algorithm was to return the optimal rotamer assignments for residues in $L(e) - \lambda(e)$ given a rotamer assignment in $M(e) \cup \lambda(e)$, when it was called while computing the $A_e$ matrix for any $\lambda$ edge $e$. It accomplished this by visiting every $\lambda$ edge in the subtree $T_e$ and looking up the optimal rotamer assignments stored in the corresponding $A$ matrices. However, due to the changes made to the procedure made in Section 8.2.1, the actual identities of the rotamers for residues in $L(e) - \lambda(e)$ is no longer required. Hence the back-tracking does not need to descend to the bottom-most level every time to return the optimal rotamer assignment. It needs to return only the optimal energy contributed by the residues in $L(e) - \lambda(e)$. This is achieved by looking up the optimal energies for vertices in $L(f), \forall f \in F_e$ from the matrices $A_f$s, which are already computed and stored as part of the post-order tree traversal. As the matrix $A_f$ stores the optimal energy for vertices in $L(f)$ for every rotamer assignment in $M(f)$, therefore for a given rotamer assignment in $M(f)$, optimal energy for residues in $L(f)$ can be unambiguously looked up.

The correctness of the back-tracking procedure is established in the following proposition.

**Proposition 9.** For a given state assignment in $M(e) \cup \lambda(e)$, the back-tracking determines the optimal energy contributed by vertices in $L(e) - \lambda(e)$.

Proof. According to Lemma 7, if $x \in L(e) - \lambda(e)$, then $x \in L(f)$ for exactly one $f \in F_e$. Therefore it is guaranteed to get the optimal energies for all $x$ by looking up the energies contributed by $L(f)$, stored in $A_f$s. Since $M(f) \subseteq M(e) \cup \lambda(e)$ (Lemma 8), for a given rotamer assignment in $M(e) \cup \lambda(e)$, unique rotamer assignments for all
$M(f)s$ can be determined, which can then be used to look up the optimal energies for residues in the corresponding $L(f)$. This completes the proof.

The following lemma and proposition prove the time complexity of the backtracking procedure.

**Lemma 10.** For any $\lambda$ edge $f$ (with the exception of the root edge $(s, r)$), there is exactly one other $\lambda$ edge $e$ such that $f \in F_e$.

**Proof.** Consider the path from the root edge $r$ to the $\lambda$ edge $f$. If the path does not cross any other $\lambda$, then $f \in F_r$. If the path does cross other $\lambda$ edges, then the smallest subtree $T_e$ can be found containing the edge $f$, such that $e$ is a $\lambda$ edge. Then $f \in F_e$. This shows that there is at least one $\lambda$ for $f$ which satisfies the required properties.

Let there be two $\lambda$ edges $e$ and $g$, such that $f \in F_e$, and $f \in F_g$. From the definition of the set $F(\cdot)$, it implies that $f$ is an edge in the subtree $T_e$ and also $f$ is an edge in the subtree $T_g$. Since $f$ is in both the subtrees, then either $e$ must be in the subtree $T_g$ or $g$ must be in the subtree $T_e$. If $e$ is an edge in $T_g$, then $f \notin F_g$, by definition of $F(\cdot)$. Therefore $e$ is not an edge in $T_g$. Similarly, $g$ is not an edge in $T_e$. This is a contradiction, therefore $f$ must be in the $F(\cdot)$ set of only one other $\lambda$ edge.

**Proposition 11.** Back-tracking procedure runs in constant time amortized.

**Proof.** Since the back-tracking procedure looks up the energy field of the matrix $A_f$ for each $f \in F_e$ and each look-up is constant time, the procedure takes $O(|F_e|)$ time to return the optimal energy for vertices in $L(e) - \lambda(e)$. According to Lemma 10, any $\lambda$ edge can be in the $F(\cdot)$ set of only one other $\lambda$ edge, and since there can be at most $n$ $\lambda$ edges (for each of the $n$ mutable residue positions, Lemma 5), the following has to be true:
\[
\sum_{\lambda(e) \neq \emptyset} |F_e| \leq n \tag{8.6}
\]

Therefore it would take \(O(n)\) time to run the back-tracking procedure once for every \(\lambda\) edge \(e\). Since number of \(\lambda\) edges at most \(n\), the back-tracking takes time \(O(1)\) amortized to run once for each \(\lambda\) edge \(e\).

\[\square\]

8.2.3 Final Retrieval of the sparse GMEC

When the post-order traversal reaches the root edge \(r\) and all computation as described in Section 8.2.1 is complete, the optimal energy contributed by residues in \(L(r)\) is known, along with the optimal assignments for residues in \(\lambda(r)\). Since \(L(r)\) contains all the mutable residues of the design problem, the optimal energy for \(L(r)\) is the energy of the sparse GMEC. However, the actual rotamer assignments for all residues (except those in \(\lambda(r)\)) is still unknown.

The optimal rotamer assignment for each residue \(x \in L(r) - \lambda(r)\) is stored in the \(A_e\) matrix of tree edge \(e\) for which \(x \in \lambda(e)\). These need to be retrieved from the corresponding edges to assemble the sparse GMEC. This is done by calling a recursive procedure described below. For clarity of the recursive procedure, details to retrieve the optimal assignments for residues in \(L(e) - \lambda(e)\) given the optimal assignments for residues in \(M(e) \cup \lambda(e)\) for any tree edge \(e\) are described below.

The procedure visits all \(f \in F_e\) (Section 8.2.2), and for each \(f\), returns the optimal rotamer assignments for vertices in \(L(f)\). This is done by carrying out the following two steps for each \(f\):

1. Using the optimal state assignment for \(M(e) \cup \lambda(e)\), get the optimal state assignment for \(M(f)\) (Lemma 8), and unambiguously look up the optimal state assignments for all vertices in \(\lambda(f)\) corresponding to the state assignment in \(M(f)\) from the stored matrix \(A_f\).
2. Once the optimal state assignment for \( M(f) \cup \lambda(f) \) is known from step 1, recursively call the procedure with \( e = f \), to return the optimal state assignments in \( L(f) - \lambda(f) \).

Once both the steps are carried out \( \forall f \in F_e \), the optimal rotamer assignments for all residues in \( L(S_e) - \lambda(e) \) are known. If \( F_e = \emptyset \), \( L(e) = \lambda(e) \), and this is the terminating condition for the recursive procedure.

When the above retrieving procedure is called for the root edge \( r \), it returns the optimal state assignments for vertices in \( L(r) - \lambda(r) \). Since by definition \( M(r) = \emptyset \) and \( R(r) = \emptyset \), after this procedure completes, the optimal rotamer assignments along with the sparse energy of the sparse GMEC are known.

The following proposition establishes the time complexity for retrieving the optimal rotamer assignments for all residues.

**Proposition 12.** The retrieving procedure runs in \( O(n) \) time, where \( n \) is the number of mutable residues in the design problem.

**Proof.** For each tree edge \( e \) such that \( \lambda(e) \neq \emptyset \), the retrieving procedure accesses the matrix \( A_e \) and returns the optimal state assignments for residues in \( \lambda(e) \). Then the cost of retrieving the optimal state assignments at this edge is \( O(|\lambda(e)|) \). Since according to Lemma 5, a residue can be in the \( \lambda(\cdot) \) set for only one tree edge, the following must be true:

\[
\sum_{\lambda(e) \neq \emptyset} |\lambda(e)| = n
\]  \( (8.7) \)

Therefore the total cost of retrieving the optimal state assignments for all residues in the sparse GMEC is \( O(n) \). \( \square \)
8.2.4 New Time Complexity of BWM Algorithm

This section proves tighter bounds on $M(e) \cup \lambda(e)$ for any $\lambda$ edge $e$, and provides the new time complexity for the GMEC-based BWM algorithm.

Let $c_1$ and $c_2$ be two child edges of a $\lambda$ edge $e$. By definition of $M(\cdot)$ and $L(\cdot)$, $M(e) \subseteq M(c_1) \cup M(c_2)$. By Lemma 4, $\lambda(e) = M(c_1) \cap M(c_2)$. Therefore, the following equation is true:

$$M(e) \cup \lambda(e) = M(c_1) \cup M(c_2) \quad (8.8)$$

Intuitively, the value of $|M(c_1) \cup M(c_2)| \leq 2w$ (as $|M(\cdot)| \leq w$), where $w$ is the branch-width. But this value can only be equal to $2w$ when $M(c_1) \cap M(c_2) = \emptyset$. This is not possible for a $\lambda$ edge since $\lambda(e) = M(c_1) \cap M(c_2)$. Therefore for any $\lambda$ edge $|M(c_1) \cup M(c_2)| < 2w$. This implies $|M(e) \cup \lambda(e)| < 2w$ (Eq. 8.8).

The following Lemma proves an upper bound on the size of $M(e) \cup \lambda(e)$. This was done in collaboration with a graduate student in the Donald lab, Jonathan Jou.

**Lemma 13.** For any $\lambda$ edge $e$, $|M(e) \cup \lambda(e)| \leq \frac{3}{2}w$, where $w$ is the width of the branch-decomposition $T$.

**Proof.** As $|M(\cdot)| \leq w$, therefore,

$$|M(c_1)| + |M(c_2)| + |M(e)| \leq 3w \quad (8.9)$$

As from Lemma 4, $\lambda(e) = M(c_1) \cap M(c_2)$, let $M^*_c$ and $M^*_e$ be defined as:

$$M^*_c = M(c) - \lambda(e) = M(c) \cap M(e) \quad (8.10)$$

$$M^*_e = M(e) - \lambda(e) = M(e) \cap M(e) \quad (8.11)$$

This implies that

$$M(e) = M^*_c \cup M^*_e \quad (8.12)$$
Substituting \(|M(c_1)|\) and \(|M(c_2)|\) from Eq. 8.10 and Eq. 8.11 in Eq. 8.9,

\[
|M^*_c| + |\lambda(e)| + |M^*_e| + |\lambda(e)| + |M(e)| \leq 3w \tag{8.13}
\]

Subtracting \(|M^*_c \cap M^*_e|\) from the left maintains the inequality.

\[
|M^*_c| + |M^*_e| - |M^*_c \cap M^*_e| + |M(e)| + 2|\lambda(e)| \leq 3w \tag{8.14}
\]

Substituting Eq. 8.12 into the above equation,

\[
|M(e)| + |M(e)| + 2|\lambda(e)| \leq 3w \tag{8.15}
\]

\[
2|M(e) + \lambda(e)| \leq 3w \tag{8.16}
\]

\[
|M(e) + \lambda(e)| \leq \frac{3}{2}w \tag{8.17}
\]

The following proposition establishes the new time complexity of the GMEC-based BWM algorithm.

**Proposition 14.** For a given protein design problem with \(n\) mutable residue positions, at most \(q\) rotamers per residue position, and a pairwise energy function, BWM computes the sparse GMEC in time \(O(\beta_t + nw^2q^{2w})\), where \(\beta_t\) give the time cost for finding the branch-decomposition \(T\) for the sparse residue interaction graph \(G'\), and \(w\) is the width for \(T\).

**Proof.** For each \(\lambda\) edge \(e\), all possible rotamer assignments for residues in \(M(e) \cup \lambda(e)\) are enumerated in \(O(q^{2w})\) time (by Lemma 13). For each rotamer assignment in \(M(e) \cup \lambda(e)\), the energy can be calculated in \(O(w^2)\) time (Section 8.2.1), and the back-tracking procedure returns the optimal energy for residues in \(L(e) - \lambda(e)\) in \(O(1)\) amortized time (by Proposition 11). There are at most \(n\ \lambda\) edges (by Lemma 5).
The final retrieval of the optimal rotamer assignments for all \( n \) residues takes \( O(n) \) time (by Proposition 12). Therefore BWM runs in time \( O(\beta_t + n w^2 q^{3w} + n) = O(\beta_t + n w^2 q^{3w}) \).

8.3 Results

To empirically test the performance of the improved BWM algorithm against the Sparse A* algorithm (described in Section 7.2), protein design problems of different sizes involving active site, core, and boundary residues were used as test cases. Dead-End elimination (DEE) was run on each of the protein design problems, followed by generating the sparse residue interaction graph, and then running either Sparse A* or BWM to obtain the sparse GMEC. For each protein design problem, Sparse A* and BWM were run four times with the following distance (\( \delta \)) and energy cutoffs (\( \alpha \)):

1. \( \delta = 8 \text{ Å} \).
2. \( \alpha = 0.2 \text{ kcal/mol} \)
3. \( \delta = 8 \text{ Å}, \alpha = 0.2 \text{ kcal/mol} \)
4. No distance or energy cutoff. This results in a full residue interaction graph, and hence the full GMEC.

Table 8.1 shows the run times of Sparse A* and BWM for the following six representative design problems:

1. 60 non-Proline residues, from the structure of \( \beta_1 \) domain of protein G (PDB ID: 1IGD), were modeled as flexible using rotamers but not allowed to mutate. The number of conformations remaining after DEE pruning was \( 10^4 \).
2. 7 residues in the active site of GrsA-PheA (PDB ID: 1AMU) were modeled as flexible using rotamers and allowed to mutate. The number of conformations remaining after DEE pruning was $10^8$.

3. 18 residues in the core of the structure of plastocyanin (PDB ID: 2PCY), were modeled as flexible using rotamers and allowed to mutate. All 18 residues were allowed to keep their wild-type identity or to mutate to Ala, Val, Leu, Phe, Ile, Tyr, and Trp. The number of conformations remaining after DEE pruning was $10^{11}$.

4. 17 residues in the boundary of protein CAL were modeled as flexible using rotamers and allowed to mutate. All 17 residues were allowed to mutate to all amino acids except Pro, Gly, Cys and Met. The number of conformations remaining after DEE pruning was $10^{13}$.

5. 17 residues in the boundary of protein ubiquitin (PDB ID: 1UBI) were modeled as flexible using rotamers and allowed to mutate. All 17 residues were allowed to keep their wild-type identity and allowed to mutate to all amino acids except Pro, Cys, Met and Gly. The number of conformations remaining after DEE pruning was $10^{15}$.

6. 15 residues in the boundary of $\beta_1$ domain of G protein (PDB ID: 1PGA) were modeled as flexible using rotamers and allowed to mutate. All 15 residues were allowed to keep their wild-type identity and allowed to mutate to all amino acids except Pro, Gly, Met and Cys. The number of conformations remaining after DEE pruning was $10^{15}$.

As can be seen from Table 8.1, BWM performs worse than Sparse A*, or crashes before returning the GMEC for designs with the full residue interaction graph ($\delta = \infty$, $\alpha = 0$). This is not surprising, and can be explained by looking at the time
Table 8.1: Shows the running times (in minutes) of A* and BWM for four combinations of distance and interaction energy cutoffs. The last row shows the width of the branch-decomposition, $w$. γ experiment not performed, † run crashed.

(a) PDB ID: 1IGD

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>0.10</td>
<td>0.142</td>
<td>0.14</td>
<td>0.13</td>
</tr>
<tr>
<td>BWM</td>
<td>2.97</td>
<td>0.2</td>
<td>0.11</td>
<td>0.10</td>
</tr>
<tr>
<td>Width</td>
<td>59</td>
<td>24</td>
<td>15</td>
<td>13</td>
</tr>
</tbody>
</table>

(b) PDB ID: 1AMU

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>0.054</td>
<td>γ</td>
<td>γ</td>
<td>γ</td>
</tr>
<tr>
<td>BWM</td>
<td>10.53</td>
<td>10.51</td>
<td>10.38</td>
<td>10.43</td>
</tr>
<tr>
<td>$w$</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

(c) PDB ID: 2PCY

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>0.21</td>
<td>0.26</td>
<td>0.26</td>
<td>0.26</td>
</tr>
<tr>
<td>BWM</td>
<td>†</td>
<td>9.60</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>$w$</td>
<td>17</td>
<td>9</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

(d) CAL Protein

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>197.23</td>
<td>139.29</td>
<td>218.35</td>
<td>137.27</td>
</tr>
<tr>
<td>BWM</td>
<td>†</td>
<td>0.22</td>
<td>296.31</td>
<td>0.17</td>
</tr>
<tr>
<td>$w$</td>
<td>16</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

(e) PDB ID: 1UBI

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>†</td>
<td>181.47</td>
<td>γ</td>
<td>175.864</td>
</tr>
<tr>
<td>BWM</td>
<td>γ</td>
<td>8.01</td>
<td>γ</td>
<td>8.04</td>
</tr>
<tr>
<td>$w$</td>
<td>γ</td>
<td>5</td>
<td>γ</td>
<td>5</td>
</tr>
</tbody>
</table>

(f) PDB ID: 1PGA

<table>
<thead>
<tr>
<th>$(\delta, \alpha)$</th>
<th>$(\infty, 0)$</th>
<th>$(8, 0)$</th>
<th>$(\infty, 0.2)$</th>
<th>$(8, 0.2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>†</td>
<td>427.31</td>
<td>γ</td>
<td>426.95</td>
</tr>
<tr>
<td>BWM</td>
<td>γ</td>
<td>51.34</td>
<td>γ</td>
<td>51.04</td>
</tr>
<tr>
<td>$w$</td>
<td>γ</td>
<td>7</td>
<td>γ</td>
<td>7</td>
</tr>
</tbody>
</table>

128
complexity expression of BWM, where the exponent for the maximum number of
rotamers per residue position ($q$) is $\frac{3}{2}w$ (Proposition 14). For the full residue interac-
tion graph, $w = n-1$, therefore, running BWM is as bad as enumerating all possible
conformations. However, when BWM is performed with sparse residue interaction
graphs, the run times for BWM are comparable to corresponding run times for Sparse
A* when the number of conformations left after DEE is small (Table 8.1(a)). The
real advantage of BWM over Sparse A* can be seen in Table 8.1(d), (e), and (f),
where the number of conformations left after DEE is large (close to $10^{15}$), and the
distance and energy cutoffs generate significantly sparse residue interaction graphs
($w \ll n$). This indicated that the BWM algorithm is likely to be most useful for
designs where the number of conformations to be enumerated are large and running
Sparse A* fails to produce any results.

8.4 BWM*: Ensemble-based algorithm for sparse residue interaction
graph

BWM algorithm described in Section 8.2 computes the sparse GMEC for a given
sparse residue interaction graph. This section briefly describes the extension to the
algorithm, called BWM*, to enumerate a gap-free list of conformations in increasing
order of sparse energy. BWM* algorithm consists of two phases: the preprocessing
phase and the enumeration phase.

8.4.1 Algorithm: Preprocessing

The preprocessing phase of BWM* has two goals: compute the energy of the sparse
GMEC, and construct a data structure for efficient enumeration of conformations
in increasing order of sparse energy. Similar to the GMEC-based BWM algorithm,
BWM* performs a post-order traversal of the branch-decomposition $T$. For each tree
edge $e$ with $\lambda(e) \neq \emptyset$, the following operations are performed:
1. Exhaustively enumerate all possible rotamer assignments to the residues in $M(e) \cup \lambda(e)$, and for each assignment look up optimal assignments for the residues in $L(e) - \lambda(e)$, which were previously computed when the post-order traversal visited the child edges.

2. Instead of storing the optimal rotamer assignment for residues in $\lambda(e)$ set for each rotamer assignment to the residues in $M(e)$ set, store all assignments to the residues in $\lambda(e)$ in a canonical min heap\(^1\), called a $\lambda$-heap, with the key being the energy of each rotamer assignment.

3. Construct a recursive heap (described below) for each rotamer assignment to the $M$-set using the $\lambda$-heaps from step 2 and the previously constructed child heaps from its two child edges. These were constructed earlier as a consequence of post order traversal.

A recursive heap is a canonical min heap satisfying the heap property, with the following additional properties:

1. For every node, there are zero, one, or two child heaps. These child heaps are also recursive heaps, and their heap nodes have the same properties.

2. The key of a node in this recursive heap is the sum of the its own key and the smallest keys of its two child heaps, i.e. the root keys of its two child heaps.

For every assignment in the $M(e)$-set, the preprocessing phase enumerates all assignments of the $\lambda(e)$-set and looks up all remaining assignments to the $L(e) - \lambda(e)$-set from the child edges. Therefore at the end of this procedure we have the energy of the optimal assignment to the $L(e)$ set for each assignment to the $M(e)$ set in

\(^1\) A min heap is a binary tree with each node storing a key-data pair. A min heap satisfies the heap property, i.e. the key of a node is always smaller than the key of its child nodes. This makes the key in the root edge the minimum key in the heap. Its takes $O(\log n)$ time to insert or delete a node from a min heap, where $n$ is the number of nodes in the min heap.
each recursive heap. Similar to the BWM algorithm, as the $L$-set of the root edge contains all $n$ mutable residues, when this procedure finishes for the root edge of the branch-decomposition $T$, the energy of the sparse GMEC is contained in the root node of the $\lambda$-heap constructed at the root edge.

Once the traversal has returned to the root edge $r$, the optimal solution for every residue has been calculated. The sparse GMEC can then be calculated by finding the lowest energy partial conformation in the $\lambda$-set at the root, and recursively looking up the optimal assignments in its children to reassemble the full conformation which produces this energy. This lookup takes linear time.

### 8.4.2 Algorithm: Enumeration

After preprocessing, the root node of the recursive heap contains the energy of the sparse GMEC and the corresponding rotamer assignments to the residues in its $\lambda$-set. The optimal assignments to the residues in its $(L-\lambda)$-set can be looked up in linear time (as mentioned in Section 8.4.1) by indexing recursively through its child heaps. After the sparse GMEC has been returned, the heap must be updated to return the next best conformation. This procedure has two steps:

1. Call this procedure recursively on its two child heaps, and update the energy of the root node with the new energies at the roots of its two child heaps.

2. After the energy of the root node is updated, it may no longer contain the minimum energy of the heap and must be updated to restore the heap property. This is done by the “bubble-down” operation for a min heap, after which the

---

2 In the bubble-down operation on a min heap, the key of a given node is compared with the key of both its child nodes, and the nodes are swapped if the key of any of its child node is greater. This is done until the given node is a leaf node with no child nodes. This operation restores the heap property that the minimum key is in the root node, and no parent node has a key value greater than that of either of its child nodes. If there are $n$ nodes in the heap, bubble-down takes $O(\log n)$ time.
heap property is restored and the root node contains the minimum energy in the heap.

After these steps complete the resulting root of the heap now contains the energy of the next best conformation. This procedure is called repeatedly to enumerate additional conformations.

**Theorem 15.** Enumerating the next best conformation takes $O(n \log q)$ time, where $q$ is the maximum number of rotamers at each residue position, and $n$ is the number of mutable residues.

The implementation and testing of the BWM* algorithm was carried out in collaboration with Jonathan Jou, and further details can be found in:

This chapter describes my work on and tests the implementing RNA $\chi$ angle flexibility to enable RNA and RNA/Protein design in the design package OSPREY. Section 9.1 provides background on experimental methods for RNA design and computational methods for modeling RNA structural flexibility, Section 9.2 describes the implementation of $\chi$ angle flexibility in OSPREY, and Section 9.3 describes the results of testing the $\chi$ angle flexibility on representative design problems.

9.1 RNA Design Background

As mentioned briefly in Chapter 1, RNA design is a new and an emerging field for creating novel RNA and RNA binding molecules for various applications (Dirks et al., 2004). One of the most common approaches to design novel RNAs is called SELEX (Tuerk and Gold, 1990; Stoltenburg et al., 2007) which stands for Systematic Evolution of Ligands by Exponential Enrichment. This is an in-vitro selection technique based on the Darwinian concept of survival of the fittest to produce a novel RNA which binds tightly to a specific target. The process involves successive rounds
of mutations and selection to evolve and identify the best binder to a given target (which can be a protein or a small molecule), from a large random or semi-random pool of RNA molecules.

The above approach is often known as irrational design, as a detailed comprehension of the structure/function properties of RNA is not necessary. Therefore, even though this technique has been successful in designing RNA aptamers for various targets, there is little understanding of the reason for its success. Rational design, on the other hand, relies on the knowledge of structural properties of RNA to design molecules, and provides a means to test and improve the current knowledge of RNA biology (Westhof et al., 1996; Breaker and Joyce, 1994). Three-dimensional structures of RNA and RNP complexes are being used as scaffolds to design novel antibiotics (Franceschi and Duffy, 2006), to stabilize RNA binding proteins (McColl et al., 1999), and to design peptide mimics that disrupt RNA-protein binding (Davidson et al., 2009; Froeyen and Herdewijn, 2002; Hamy et al., 1997; Davis et al., 2004). Various docking programs are used to calculate theoretical binding affinities of hundreds of small molecule ligands that bind RNA (Hermann and Westhof, 2000). Docking algorithms, along with algorithms for predicting the RNA secondary and tertiary structures (Das et al., 2010), have been used for in-silico design of RNA sequences as starting libraries for SELEX experiments (Chushak and Stone, 2009; Kim et al., 2010). In addition to designing novel molecules, computational approaches also provide a tool to investigate and test the principles of formation of RNP complexes (Chen et al., 2004).

9.1.1 RNA Structural Flexibility and Design

As the success of rational design depends on the knowledge of RNA structure, it is important to develop a more comprehensive understanding of RNA structure and its interactions with other molecules (Fulle and Gohlke, 2010; Gallego and Gabriele
RNA molecules are capable of folding into complex three-dimensional structures, and can undergo conformational changes upon binding to proteins or small molecules. These conformational changes can be small movements in the RNA backbone with a few bases flipping their positions, or large scale changes affecting a number of residues (Hermann, 2002; Leulliot and Varani, 2001; Frankel, 2000). Many unbound RNAs can also sample a large number of conformations, as has been shown by crystal structures and NMR studies (Bardaro et al., 2009). This conformational flexibility has been exploited to lock RNA molecules into conformations that are unfavorable for its normal function (Murchie et al., 2004).

However, attempts to account for RNA conformational flexibility have been limited to docking approaches, which use the following two strategies to approximate RNA conformational flexibility: first is a soft-docking strategy which allows for some overlap between atoms of the target and the ligand; second is docking the ligand to multiple structures of the target molecule. The multiple structures can either be different crystal and NMR structures of the same molecule, or can be generated using molecular dynamics simulations (Fulle and Gohlke, 2010; Moitessier et al., 2006). Molecular dynamics has also been used to study RNA-protein complexes (Reyes and Kollman, 2000), but conformational flexibility and computational mutagenesis have been restricted to proteins. However, to design RNA-protein interfaces, it is important to consider both the proteins and the RNA chain as flexible simultaneously.

### 9.2 RNA \(\chi\) angle flexibility in OSPREY

The OSPREY software package described in Chapter 6 was built for designing proteins and small molecules that bind proteins, modeling continuous flexibility for side chains and backbones (Gainza et al., 2013). OSPREY can be used to design proteins that bind to RNA and DNA molecules, but keeps the nucleic acids static and treats them as part of the template (Section 6.2). Because the RNA molecules cannot move...
or change, OSPREY cannot design RNA molecules. This section describes my work to implement RNA flexibility in OSPREY so that it can be used for designing RNA molecules and RNP interfaces.

In order to design RNA molecules, an individual residue has to be allowed to mutate to any of the four standard RNA bases, and allowed to sample different conformations. As mentioned in Chapters 2 and 3, the dihedral angle across the glycosidic bond (covalent bond connecting the RNA base to the backbone) is the χ angle, and different values of χ are different conformations of the RNA bases relative to the backbone. Unlike protein side chains, the χ angle is the only degree of freedom for RNA bases, but the χ angle is not rotameric in the traditional sense of protein side chains. The main conformations for the χ angle are anti, syn, and high anti (Section 2.1), but just three conformations do not cover the entire range of χ angle values, and there are not always clear barriers between them. In addition, the χ angle values are also pucker specific (Section 3.3.1). Therefore, the χ angle values sampled in the design process should be pucker-specific, and cover the ranges that represent different possible conformations of the RNA bases.

I extended OSPREY v2.0 to allow RNA residues to mutate, and to allow χ angle flexibility during the design process. The χ angle flexibility is modeled by sampling the pucker and sequence specific χ angle ranges (Section 3.3.1) at regular intervals, and treating them as discrete “RNA rotamers” (equivalent to protein side-chain rotamers) in the design process. Since these RNA rotamers are input to the algorithm, the χ angle values can be sampled any way the user pleases.

9.3 Testing the software

9.3.1 Experiments

To test the χ angle flexibility implementation in OSPREY, it was run on a number of representative design problems. As the first test, in all these problems, the residues
are modeled as flexible, but are not allowed to mutate.

1. 24/28 residues in two strands (mostly base-paired) of the helical Rev response element from HIV-1 (PDB ID: 1CSL) were modeled as flexible. This was designed with energy window of $E_w = 0.2$ kcal/mol.

2. 38/63 residues in the structure of the hammerhead ribozyme (PDB ID: 2OEU) were modeled as flexible.

3. 17/24 residues for the two strands (mostly base-paired) of structure of the loop E from the isolated *E. coli* 5S rRNA (PDB ID: 354D) were modeled as flexible. This was designed with energy window of $E_w = 0.2$ kcal/mol.

4. 6/9 residues in the RNA chain and 5 residues in the protein chain of the structure of the HUD protein bound to the single-stranded C-FOS mRNA (PDB ID: 1FXL) were modeled as flexible.

5. 10 residues in the RNA chain and 19 residues in the protein chain in the structure of the c-di-GMP riboswitch (PDB ID: 3IWN) were modeled as flexible. The original $\chi$ angle values were treated as a separate rotamer.

6. 14 residues in the protein chain and 7 residues in the RNA chain in the interface of the structure of c-di-GMP riboswitch (PDB ID: 3MXH) were modeled as flexible.

7. 9 RNA residues and 15 protein residues in the interface of the structure of the glycine riboswitch (PDB ID: 3P49) were modeled as flexible. The original $\chi$ angle values were treated as additional rotamers.

To create the RNA rotamer library, the range of $\chi$ angle values were sampled at 10° intervals. The ranges cover the anti, high-anti and syn regions for purines (A,
and G) and the anti range for pyrimidines (C, and U) (described in Section 3.3.1).

Since these ranges are pucker specific, a pre-processing step is required before the
design process. This pre-processing step identifies the pucker of every residue in the
input PDB file (including the ones that are part of the template) using the backbone
δ angle and the Pperp test (described in Section 2.2.2), and renames the residues
to be either $RN3$ or $RN2$, where $N$ can refer to A, U, G, or C. After this, the 2’
and 3’ pucker residues are considered as different residue types. Table 9.1 lists the
RNA rotamers used as input for the above design problems. For each of the design
problems, DEE was performed, followed by A* search to return the GMEC. The
energy function used was AMBER (Cornell et al., 1995), with terms for electrostatic
and van der Waals pair-wise energy.

Table 9.1: List of RNA $\chi$ angle rotamers for the C3’ and C2’ pucker RNA residues.

<table>
<thead>
<tr>
<th>Residue</th>
<th>Range</th>
<th>No. of Rotamers</th>
<th>$\chi$ angle values</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA3</td>
<td>Anti, High-anti, Syn</td>
<td>17</td>
<td>75, 85, 95, 105, 160, 170, 180, -170, -160, -150, -140, -130, -120, -110, -100, -90, -80</td>
</tr>
<tr>
<td>RG3</td>
<td>Anti, High-anti, Syn</td>
<td>17</td>
<td>55, 65, 160, 170, 180, -170, -160, -150, -140, -130, -120, -110, -100, -90, -80, -70, -60</td>
</tr>
<tr>
<td>RC3</td>
<td>Anti</td>
<td>10</td>
<td>170, 180, -170, -160, -150, -140, -130, -95, -85, -75</td>
</tr>
<tr>
<td>RU3</td>
<td>Anti</td>
<td>11</td>
<td>160, 170, 180, -170, -160, -150, -140, -130, -120, -110, -100</td>
</tr>
<tr>
<td>RA2</td>
<td>Anti, High-anti, Syn</td>
<td>19</td>
<td>35, 45, 55, 65, 75, 85, 95, -165, -155, -145, -135, -125, -115, -105, -95, -85, -75, -65, -55</td>
</tr>
<tr>
<td>RG2</td>
<td>Anti, High-anti, Syn</td>
<td>19</td>
<td>45, 55, 65, 75, 85, 95, -170, -160, -150, -140, -130, -120, -110, -100, -90, -80, -70, -60, -50</td>
</tr>
<tr>
<td>RC2</td>
<td>Anti</td>
<td>11</td>
<td>-165, -155, -145, -135, -125, -115, -105, -95, -85, -75, -65</td>
</tr>
<tr>
<td>RU2</td>
<td>Anti</td>
<td>9</td>
<td>-165, -155, -145, -135, -125, -115, -105, -95, -85</td>
</tr>
</tbody>
</table>
9.3.2  Results

Table 9.2 lists the number of RNA residues designed in each of the design problems and the number of RNA rotamers recovered in the GMEC returned by the A* search. Since the $\chi$ angle ranges were sampled at $10^\circ$ intervals, a rotamer was considered to be recovered if the $\chi$ angle of a residue in the GMEC was within $\pm5^\circ$ of the $\chi$ angle in the original structure. As can be seen from the table, the percentage of the RNA rotamers recovered is much higher for good resolution structures (around 70%) as compared to when designing structures poorer than 2 Å resolution (around 25%). The number of rotamers recovered increases when the $\chi$ angle window for rotamer recovery is set at $\pm10^\circ$. This indicates that in a majority of the cases, the software is able to place the RNA base in the vicinity of the original model, if not extremely close. Note that for the residues where the RNA rotamer was recovered only with the $\pm10^\circ$ window, there was a closer RNA rotamer available in the rotamer library but it was not picked to be part of the GMEC.

Table 9.2: Total number of RNA residues made flexible and the number of rotamers recovered in the GMEC for the 7 sample design problems.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Resolution (Å)</th>
<th>Residues Flexible</th>
<th>% Rotamers Recovered ($\pm5^\circ$)</th>
<th>% Rotamers Recovered ($\pm10^\circ$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNA Only</td>
<td>RNA/Protein Interface</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>354D</td>
<td>1.5</td>
<td>17</td>
<td>70</td>
<td>100</td>
</tr>
<tr>
<td>1CSL</td>
<td>1.6</td>
<td>24</td>
<td>71</td>
<td>91</td>
</tr>
<tr>
<td>2OEU</td>
<td>2.0</td>
<td>38</td>
<td>72</td>
<td>89</td>
</tr>
<tr>
<td>1FXL</td>
<td>1.8</td>
<td>6</td>
<td>4</td>
<td>83</td>
</tr>
<tr>
<td>3MXH</td>
<td>2.3</td>
<td>7</td>
<td>29</td>
<td>100</td>
</tr>
<tr>
<td>3IWN</td>
<td>3.2</td>
<td>10</td>
<td>30</td>
<td>70</td>
</tr>
<tr>
<td>3P49</td>
<td>3.55</td>
<td>9</td>
<td>22</td>
<td>66</td>
</tr>
</tbody>
</table>

One of the reasons why some of the RNA rotamers were not recovered in the designed structure is because the designed rotamer missed some interactions made
Figure 9.1: Original (cyan) and the designed (purple) conformation for the base of residue G71 in the PDB ID: 1CSL. Hydrogen bonds are shown as dotted pillows. The $\chi$ angle for the original conformation is $65.2^\circ$ and for the designed conformation is $75^\circ$.

by the native model. Fig. 9.1 shows one such example, in the high-resolution PDB ID: 1CSL. The original G71 residue (cyan) is in a helical region and forms a nice base pair with two H-bonds. It is in the syn conformation for the $\chi$ angle, and makes an H-bond with the non-bridging oxygen of its phosphate group. The rotamer in the design result (purple) misses one of the two H-bonds in the base pair and the H-bond with the backbone. Previous studies in the Donald lab have shown that it is often necessary to add extra terms for hydrogen bonding to the energy function for protein design, as the van der Waals and electrostatic terms in the AMBER energy function alone to not recapitulate the hydrogen bonds. Adding the explicit term for the hydrogen bonding potential in RNA and RNA protein design is very likely to improve the algorithm’s performance.

Not surprisingly, the residues for which the RNA rotamers were recovered with
the angle window of $\pm 5^\circ$ were mostly in the helical regions. Also what can be seen from Table 9.2 is that for RNA-protein interfaces, the rotamer recovery is very low with angle window of $\pm 5^\circ$. This may be potentially connected to the recovery of the side-chain rotamers in the protein, and correlating the recovery of RNA rotamers with that of protein rotamers will provide useful insights. Moreover, the rotamer recovery for RNA-protein interface improve when the angle window is increased to $\pm 10^\circ$. This shows that the algorithm places the RNA base more or less in the vicinity of the original model. As modeling continuous flexibility for the protein structure in the design process produces better results (Section 6.2.4), extending RNA design to work with continuous flexibility for both RNA and protein chains should also help improve results.
Conclusions and Future Directions

My work in this thesis extends our understanding of RNA 3D structural characteristics, and improves upon the current tools used for RNA structure validation, refinement, and correction. Using these new tools and techniques for RNA structure rebuilding demonstrates that significant improvements can be made to RNA structures, occasionally leading to changes in the biological conclusions derived. On the protein redesign side of things, my work analyzes the effects of the use (quite ubiquitously) of sparse residue interaction graphs on the end results generated by design algorithms, and shows that combined with ensemble-based provable algorithms, they can be used to solve previously intractable design problems. My work on developing a novel dynamic programming algorithm, BWM*, for ensemble-based design with sparse residue interaction graphs that runs faster than the traditional A* search algorithm, is a step further in the same direction. My work on implementing RNA $\chi$ angle flexibility in OSPREY provides the first design software that can redesign both RNA and proteins, and allow them to be flexible simultaneously.

As mentioned in Section 2.3, the current version of ERRASER used extensively in this work recognizes only non-modified RNA residues, and removes all other residue
types (protein residues, waters, ions, modified residues, DNA, ligands) from the PDB file before starting the structure optimization. This is a problem for rebuilding RNA residues that interact with the neglected residue types, such as in large RNP complexes. ERRASER can be misled into moving the new RNA conformation into protein or ligand density, which creates impossible steric clashes during the final step of merging the rebuilt structure with the original PDB file. Steven Lewis, a post-doc in the Richardson lab, is working on the thorough rewrite of ERRASER that will allow it to recognize the currently neglected residue types, and thus take into account their interaction with RNA residues during the rebuilding process. This should greatly improve modeling and correction of both RNA-small molecule and RNP structures.

The newly recognized RNA backbone suite conformers (Chapter 5) will be added to the Suitename program (Section 2.1.3) for automated backbone conformer assignment, and will then be utilized for structure validation and correction in MolProbity and PHENIX. The list of backbone conformers could also be very valuable in initial fitting of RNA models, especially in non-helical regions. Some initial trials have been done, and work is in progress to take advantage of that possibility. The conformer set is similar in nature to protein side-chain rotamers, but perhaps even more useful because they apply to the continuous backbone rather than just to an individual side branch. In implementing such a tool, the philosophy is to work from the features best seen in electron density, a phosphorus and its two flanking glycosidic bonds, but using at least seven parameters, so as to encapsulate the full information contained in those positions. Such a system would be integrated into the automated structure building procedures in PHENIX.

The BWM* ensemble-based algorithm for protein design (Chapter 8) currently treats protein side-chain rotamers as rigid, and is not compatible with the minimization-aware algorithms developed in the Donald lab. By generating a sparse graph for BWM*, the underlying assumption is that if two residues do not have a direct inter-
action with each other, then any change in the first residue can lead to a change in
the second residue only through a chain of pair-wise interacting residues. Since all
possible conformations between residues that interact are enumerated in BWM* and
during minimization rotamers are allowed to minimize within the voxels, the mod-
ified criterion of DEE with minimization (Section 6.2.4) can be applied for sparse
residue interaction graphs without losing any guarantees. However, the use of dis-
tance and interaction energy cutoffs will have to be modified to be provably used
to generate the sparse residue interaction graph after DEE pruning, to account for
the movement of the rotamers within the defined voxels. Jonathan Jou, a graduate
student in the Donald lab, is working on extending BWM* for both side-chain and
backbone continuous flexibility.

The RNA and RNA-protein interface design capability implemented in OSPREY
(Chapter 9) is a good stepping stone for future studies, as more and more researchers
are getting interested in understanding the principles of interaction between RNA
and proteins, and exploiting their potential for therapeutic purposes. The software
currently moves the base $\chi$ angle and keeps the RNA backbone rigid during the de-
sign process. One possible idea to implement backbone flexibility would be to allow
small changes to the glycosidic bond (bond connecting the base and the backbone)
direction, analogous to the backrub motion in proteins. This would need empirical
analysis to find out preferred direction and axis for such a motion, but can be imple-
mented using the code already present in OSPREY for allowing backrub motions in
protein chains. Further extending the work to account for continuous RNA base and
backbone flexibility, and extensively testing it for rotamer and sequence recovery will
help to build upon this nascent tool.
Bibliography


Biography

Swati Jain was born March 29, 1986 in New Delhi, India to Anil Kumar Jain and Chitra Jain. She has one younger sister, Ruchir Jain.

Swati has always had an interest in mathematics, biology, and astronomy. Learning about the functioning of nature, from the smallest to the largest scale, fascinates her. She got an opportunity to attend a workshop during her high school in which an experiment to extract DNA from yeast cells was performed. The fact, that something tangible is the basis of abstract concepts like thoughts and emotions, gripped her and inspired her to learn more about this field.

Swati completed her schooling in May 2004 from Delhi Public School, Noida, India, taking her senior and senior secondary exams with the Central Board of Secondary Education (CBSE) India.

She went to on International Institute of Information Technology (IIIT), Hyderabad, India for her undergraduate studies. Her undergraduate mentor, Dr. Abhijit Mitra, introduced her to the field of RNA structure and using computational tools and algorithms for its study. She graduated with a Bachelor of Technology (BTech) in Computer Science and Engineering, with a honors in Computational Natural Sciences in July 2008.

She was admitted to the Structural Biology and Biophysics graduate program (SBB) at Duke University in August 2008, and later joined the Computational Biology and Bioinformatics (CBB) Program for her PhD thesis. She was extremely
fortunate to find excellent advisors in Jane and Dave Richardson, and Bruce Donald. She has co-authored the following scientific publications in grad school:


Swati plans to graduate from Duke with a PhD in Computational Biology and Bioinformatics and a certificate in Structural Biology and Biophysics in May 2015.