Conformational sampling in protein structure prediction

by

Sabareesh Subramaniam

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computational Biology)

at the

UNIVERSITY OF WISCONSIN–MADISON

2014

Date of final oral examination: 02/19/2014

The dissertation is approved by the following members of the Final Oral Committee:

Alessandro Senes, Assistant Professor, Biochemistry
Qiang Cui, Professor, Chemistry
Julie Mitchell, Associate Professor, Biochemistry and Mathematics
Sriraam Natarajan, Assistant Professor, Informatics and Computing, Indiana University
Jude Shavlik, Professor, Computer Sciences

To Mom, Dad and Motu

ACKNOWLEDGMENTS

I would like to first thank Prof. Alessandro Senes for making this thesis possible. Alessandro's motivation and encouragement have been invaluable throughout this journey. His dedication to science and relentless work ethic are traits I have often admired and attempted to cultivate. In addition to the exciting science we worked on together, I am also grateful for the many hours Alessandro spent on helping me communicate my research clearly and concisely.

My lab mates Rika, Ben, and Loren have been great friends and collaborators over the years. I would like to thank them for making the lab an enjoyable experience.

I would like to thank all members of my thesis committe for their invaluable guidance. Special thanks to Prof. Sriraam Natarajan for the exciting collaboration on our machine-learning project and the truly memorable visit to Wake Forest University.

Dan Kulp, Jason Donald and Brett Hannigan have been great collaborators on the MSL project. Although, our interactions have been few, I have gained valuable protein modeling and software development experience from all of them.

I would also like to acknowledge Will Benton for making available the template on which this thesis is based.

Lastly, I would like to thank my parents, Lakshmi and Subramaniam, and brother, Gireesh, for their constant support and encouragement. Despite financial hardships, they provided for a good education and gave me a chance to pursue my dreams.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CONFORMATIONAL SAMPLING IN PROTEIN STRUCTURE PREDICTION

Sabareesh Subramaniam

Under the supervision of Assistant Professor Alessandro Senes
At the University of Wisconsin-Madison

This thesis describes computational structure prediction methods I developed for the study of membrane proteins.

Protein structure prediction may be considered as two almost independent stages: the modeling of the backbone, followed by the optimization of the side chains for each backbone geometry. Side chain optimization can become the bottleneck stage of structure prediction, and therefore, needs to be as efficient as possible. In the first part of this thesis, I describe novel methods to improve the speed and accuracy of side chain modeling, which I later leverage to predict the structure of membrane protein complexes.

Side chain optimization is a highly combinatorial task complicated by the great degree of side chain conformational freedom. A common approach to model side chain flexibility is to *discretize* the space in a set of representative conformations, called conformer libraries. These libraries need to provide sufficient sampling of the underlying space, while remaining as small as possible, for the sake of computational efficiency. To achieve a good balance between these conflicting needs, I have developed a novel energy-based criterion to create conformer libraries (chapters 2 to 4). Through experiments I demonstrate that these energy-based conformer libraries enable faster and more accurate side chain modeling using a smaller number of conformers.

Membrane proteins often associate with each other to form complexes which are essential for their function. I have developed high-throughput methods, enhanced by the use of energy-based conformer libraries, for predicting the structure of these complexes (chapters 5 and 6). The method described in chapter 5 interprets “low resolution” experimental results,

such as mutagenesis data, to create a structural model of the bacterial division protein FtsB. This model has been used to guide the experimental characterization of the FtsB protein.

Ab initio structural prediction methods are important when experimental results are not available. Chapter 6 describes the geometric analysis of a common transmembrane motif ($GAS_{\mathtt{right}}$) which reveals that the motif is optimized for Cα hydrogen bonding. The analysis led to the creation of "CATM", a method that predicts *ab initio* the structure of $GAS_{\mathtt{right}}$ motifs at near atomic resolution.

Alessandro Senes

ABSTRACT

This thesis describes computational structure prediction methods I developed for the study of membrane proteins.

Protein structure prediction may be considered as two almost independent stages: the modeling of the backbone, followed by the optimization of the side chains for each backbone geometry. Side chain optimization can become the bottleneck stage of structure prediction, and therefore, needs to be as efficient as possible. In the first part of this thesis, I describe novel methods to improve the speed and accuracy of side chain modeling, which I later leverage to predict the structure of membrane protein complexes.

Side chain optimization is a highly combinatorial task complicated by the great degree of side chain conformational freedom. A common approach to model side chain flexibility is to *discretize* the space in a set of representative conformations, called conformer libraries. These libraries need to provide sufficient sampling of the underlying space, while remaining as small as possible, for the sake of computational efficiency. To achieve a good balance between these conflicting needs, I have developed a novel energy-based criterion to create conformer libraries (chapters 2 to 4). Through experiments I demonstrate that these energy-based conformer libraries enable faster and more accurate side chain modeling using a smaller number of conformers.

Membrane proteins often associate with each other to form complexes which are essential for their function. I have developed high-throughput methods, enhanced by the use of energy-based conformer libraries, for predicting the structure of these complexes (chapters 5 and 6). The method described in chapter 5 interprets "low resolution" experimental results, such as mutagenesis data, to create a structural model of the bacterial division protein FtsB. This model has been used to guide the experimental characterization of the FtsB protein.

Ab initio structural prediction methods are important when experimental

results are not available. Chapter 6 describes the geometric analysis of a common transmembrane motif ($GAS_{\texttt{right}}$) which reveals that the motif is optimized for Cα hydrogen bonding. The analysis led to the creation of "CATM", a method that predicts *ab initio* the structure of $GAS_{\texttt{right}}$ motifs at near atomic resolution.

1 CONFORMATIONAL SAMPLING IN PROTEIN STRUCTURE PREDICITION

Summary

The goal of protein structure prediction is to obtain a 3-dimensional model of the target protein describing the location of all, or a set of its atoms, with respect to each other. Typically, protein structure prediction is posed as a search problem over a space of possible geometries. This search space is then presented to an algorithm which is required to determine the geometry with the least energy, as defined by an energy function.

Conformational sampling refers to the process of modeling the target conformational space by compiling a set of representative geometries. Discrete conformational sampling makes the problem accessible to a number of algorithms that can efficiently search discrete spaces. In the case of protein structure prediction, conformational sampling may be performed over the space of possible side chain as well as backbone geometries. This sampling is an important component of all stages of protein design and structure prediction because of the great conformational flexibility of proteins, it is a computationally intensive task. Described in this thesis are efficient sampling techniques for backbone and/or side chain conformational sampling.

This chapter introduces the need for structure prediction and the importance of conformational sampling in a variety of problems. Specific issues and solutions for side chain conformational sampling are discussed in chapters 2 to 4. Backbone conformational sampling when experimental results are available is explained in chapter 5. An analysis of protein conformational space which led to the CATM method for predicting structure directly from protein sequence is discussed in chapter 6. Chapter 7 describes the C++ software library developed in-house, called MSL, which enabled all the molecular modeling described in this thesis.

1.1 Background

The central dogma of molecular biology [Crick (1970)] explains how genetic information encoded in the DNA (Deoxyribonucleic acid) is transformed into functional proteins in living organisms. A simplified, popular restatement of the dogma is *"DNA makes RNA makes protein"*. This discovery directed scientific efforts towards the decoding of chromosomal DNA resulting in the creation of a new field of science called genomics.

Genomics is the field of science that involves DNA sequencing, sequence assembly, annotation and analysis of the genome. Genomics, and in particular DNA sequencing, benefited from the advancements in computer science [Staden (1979)] and the two fields advanced rapidly making the sequencing of entire genomes possible. In 1990, the human genome project was set up as a collaborative worldwide project in an effort to understand human diseases better. The goal of the project was to decode the human DNA and eventually understand the complex biological processes in the human body. This project, completed in 2003, sequenced the entire human DNA and identified approximately 20,000-25,000 protein-coding genes [Consortium (2004)]. These genes undergo transformation resulting in proteins which are a central component of most biological processes.

Proteins are molecules which perform a wide variety of functions in living organisms as catalysts, structural elements, regulators, channels, transporters, receptors [Schlessinger (2000); Fagerberg et al. (2010); Endres et al. (2011)] and so on. Therefore, a thorough understanding of proteins should lead to a better understanding of biological processes. However, in order to understand protein function at a molecular level, it becomes necessary to determine their 3-dimensional structure, in addition to their amino acid sequence. X-ray crystallography [Kendrew et al. (1958)] is the most widely used method to determine protein structure; followed by NMR (Nuclear Magnetic Resonance) [Wuthrich (2001)]. However, the experimental structure determination process is time-consuming and has been far outpaced by

the whole genome sequencing projects [Xiang (2006)].Therefore, a gap exists between the number of protein sequences available and the number of protein structures available; blocking more comprehensive scientific analyses of these proteins. Computational approaches to determining protein structure are being employed in an attempt to bridge this gap.

The sequence-structure gap is especially severe in the case of membrane proteins. Membrane proteins are a class of proteins associated with the cell membrane, which comprise about 30% of all genomes and serve as targets for about 50% of the pharmaceuticals in the market today [Krogh et al. (2001); Fagerberg et al. (2010)]. Owing to their abundance and significance, knowledge of membrane protein structure is essential. However, inspite of technical advancements in crystallography [Carpenter et al. (2008)] and NMR, the sequence-structure gap is wide in the case of membrane proteins because of a variety of technical difficulties. As demonstrated in this thesis, computational structure prediction presents a promising alternative to study these proteins.

Protein structure prediction presents itself in a wide range of scenarios with different levels of difficulty. For example, in some cases experimental data may be available, and this data can be incorporated to guide computational prediction. Sometimes *ab initio* methods that model structures directly from primary sequence are required. Frequently, computational methods to design or re-engineer proteins are used as a means to test our understanding of proteins. This thesis presents the development of computational methods applicable to many of these scenarios.

1.2 Protein structure prediction

Computational protein structure prediction is an important tool to bridge the gap between the available sequence data and structural information. Protein structure has been the topic of much study in the 20th century and

a brief review of the elements that constitute protein structure and guide structure prediction is useful for further discussion.

Protein Structure

Proteins are molecules made up of one or more linear chains of amino acids. Successive amino acids in the protein chain are linked together by an amide or "peptide" bond linking the alpha carbon atoms and therefore proteins are also described as polypeptide chains. They are composed of 20 naturally occuring amino acids, each containing a unique type of side chain. Protein structure may be considered as two parts - the backbone and the side chains. The backbone is a repeating sequence of atoms common to all amino acids while the side chains differ based on the amino acid in each position.

Protein structure is often conceptualized as four levels of increasing complexity: primary, secondary, tertiary and quarternary structure. Primary structure of a protein is the amino acid sequence defined by the gene coding for the protein. The amino acid sequence is variable in size and is believed to encode the structure and function of the protein [Anfinsen (1973)].

Proteins have been observed to fold into regular local sub-structures referred to as the secondary structure. Two main types of secondary structure are observed, the alpha helix and the beta sheet. These structures exhibit a regular geometry and are preferred because they permit the saturation of all hydrogen bond acceptors and donors on the protein backbone. The secondary structures fold into a globular three-dimensional structure called the tertiary structure. This folding is driven by the hydrophobic effect whereby the protein buries much of its hydrophobic surface in its core. The tertiary structure is stabilized by interactions such as hydrogen bonds and disulphide bridges. Multiple folded sub-units are arranged together to form the quarternary structure.

Protein Conformation

Protein structure may be defined in terms of several degrees of freedom defined with respect to the bonded atoms. Bond distances, bond angles and dihedral or torsional angles are sufficient to fully describe a protein's 3-dimensional structure. Bond angles and bond distances have been observed to be more or less invariant whereas the dihedral angles are known to vary systematically. The variance of both the backbone dihedral angles, called ϕ and ψ (Figure 1.1a), as well as the side chain dihedral angles, χ1-4 (Figure 1.1b and c), have been analysed thoroughly and these analyses have formed the basis of current structure prediction and design techniques.

An analysis of backbone dihedral angles was performed by Ramachandran and colleagues [Ramachandran et al. (1963)] which resulted in the ramachandran plot (Figure 1.2). This plot is a representation of the allowed and disallowed values for ϕ and ψ. Theoretically, ϕ and ψ can take values in the range [-180,+180), however, a significant portion of this space was found to be unsuitable because of steric constraints i.e) it is impossible to pack atoms at most combinations of ϕ and ψ with favorable contacts. This analysis also detailed the relationship between secondary structure and the backbone dihedral angles showing that different regions of the ϕ-ψ space were preferred by different local geometries (Figure 1.2). For example, the common secondary structures, alpha helix and beta sheet were observed to have characteristic backbone dihedral angles. The amino acids in a right-handed alpha helix have a characteristic backbone conformation around ϕ=-57° and ψ=-47° with 3.6 amino acids per turn of the helix. The beta sheets may be parallel(ϕ=-119° and ψ=+113°) or antiparallel (ϕ=-139° and ψ=+135°).

The side chain dihedral space has also been studied extensively. It has been discovered that, like the backbone dihedral space, a significant portion of the side chain space is unsuitable. Further, side chain dihedral values have been known to cluster in the dihedral space into regions called rotamers

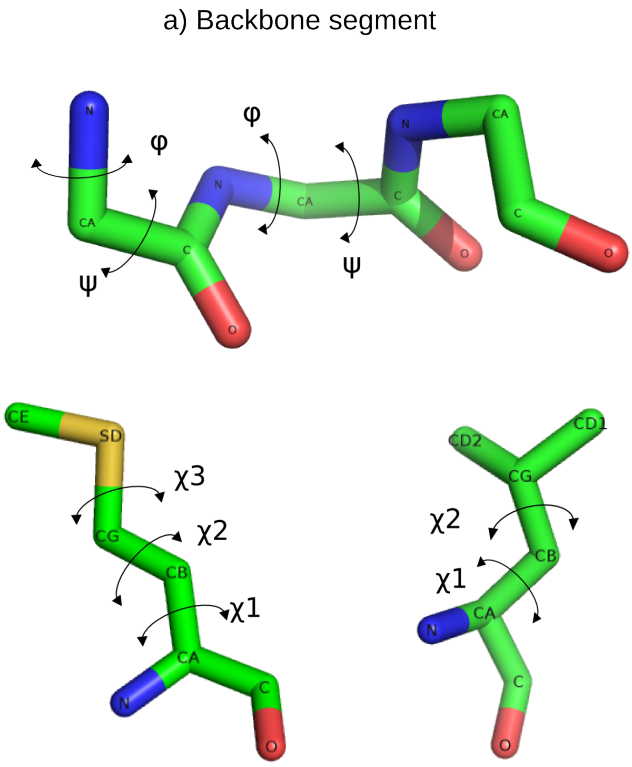


Figure 1.1: *The bond distances and angles in a protein remain relatively fixed compared to the dihedral angles. The conformation of the backbone and side chains is determined primarily by the rotation around the dihedral angles. Panel a) shows the backbone dihedral angles* ϕ *(C'-N-CA-C) and* ψ *(N-CA-C-N') in a tripeptide backbone, b) shows the three side chain dihedral angles* χ*1 (N-CA-CB-CG),* χ*2 (CA-CB-CG-SD) and* χ*3 (CB-CG-SD-CE) on methionine and c) shows* χ*1 (N-CA-CB-CG) and* χ*2 (CA-CB-CG-CD1) on leucine.*

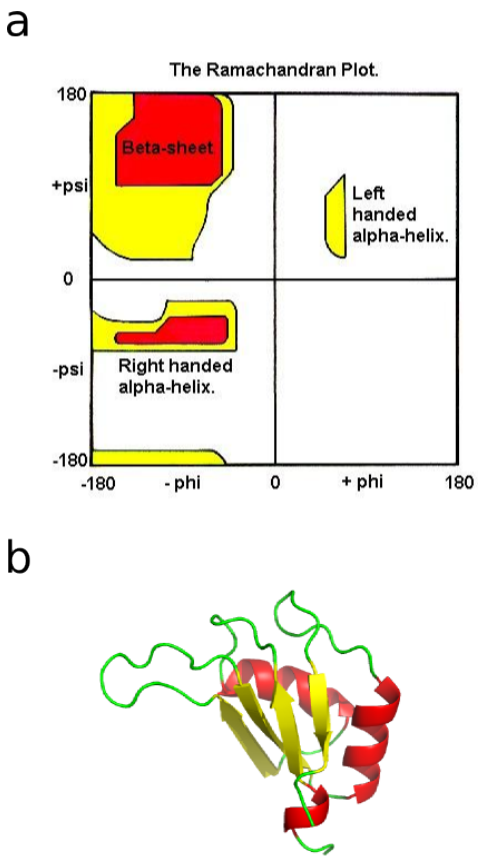


Figure 1.2: *a) The Ramachandran plot is a color-coded plot of the backbone dihedral angles* ϕ *along the x-axis and* ψ *along the y-axis. In dark red are the fully allowed regions with no atomic overlap, in yellow are additionally allowed regions and in white are regions that are not allowed. The labels associate different regions of this space with a secondary structure. b) Shows alpha helices (red) and beta sheets (yellow) from a ribonuclear protein (PDB ID: 1L3K).*

[Zimmerman et al. (1977)]. This observation was used in the creation of rotamer libraries [Dunbrack (2002)] which are used extensively to model side chain flexibility in most protein modeling applications. Further, the interdependance of backbone and side chain conformations has also been analyzed resulting in the use of backbone-dependent rotamer libraries[Dunbrack and Karplus (1994),Chakrabarti and Pal (2001),Shapovalov and Dunbrack (2011)]. Several software packages that make use of these libraries have been used successfully to predict side chain conformations [Krivov et al. (2009), Kulp et al. (2012)].

Structure prediction as energy minimization

Protein folding is a sophisticated process by which proteins attain a specific low energy state following a kinetic pathway in the energy landscape. Proteins tend to fold into specific, stable, globular structures in order to be functional, with the information required to achieve the folded state being specified by the amino acid sequence. Therefore, it is believed that the folded structure of a protein can be predicted from its amino acid sequence. The functional forms of proteins are believed to be stable low-energy conformations, well seperated from other intermediate states in the protein energy landscape. Based on this theory, the structure prediction problem is often posed as an energy minimization problem, where the structure with the minimum energy (as defined by a force field) is sought for the target protein sequence. Since atoms are free to rotate about each bond in the protein, structure prediction presents a minimization problem of very high dimensionality which has made it inaccessible to "brute force" methods.

The protein energy landscape is believed to resemble a deep funnel with a number of local minima [Onuchic et al. (1997)] and is modeled using mathematical functions called force fields. Force fields are systems of mathematical equations, with associated parameters, to describe the energetics of molecular systems. Developing force fields that accurately

capture the behaviour of proteins is complicated and is a field of active research[Beauchamp et al. (2012)]. CHARMM [Brooks et al. (1983)] and AMBER [Cornell et al. (1995),Wang et al. (2004)] are two force fields used extensively in protein modeling, particularly in molecular dynamics, while a wide variety of other force fields are available [Ponder and Case (2003)]. Force fields may be derived based on statistical or quantum mechanics, called physical force fields [Brooks et al. (1983); Cornell et al. (1995); Wang et al. (2004)] or they may consist of potentials extracted from a database of proteins, called knowledge-based force fields [Russ and Ranganathan (2002)]. A combination of physical and knowledge-based force fields have become a common feature of modern structure prediction and design methods.

Despite valuable insights into the folding mechanism, the determinants of protein folding and structure are still not completely understood; resulting in deficiencies in the force fields used to model protein folding. Even state-of-the-art force fields are unable to completely capture the sequence-structure specificity observed in proteins i.e) in some cases, the force fields are not accurate enough to capture the drastic structural changes that accompany slight sequence variations in nature. Inspite of the deficiencies, energy minimization based on a force field is still a widely used method for most computational protein modeling methods today.

1.3 Types of protein modeling applications

In practical applications, several variants of the structure prediction problem have been encountered. The difficulty of protein structure prediction depends on the specific kind of protein being studied and the scope of the study. This section describes and attempts to classify the common scenarios where computational modeling is employed to predict protein structure.

Homology modeling

Homology modeling or template-based modeling refers to the structure prediction of a protein which exhibits a high sequence identity with one or more proteins of known structure. For example, proteins performing similar functions in different organisms, may have similar amino acid sequences. This sequence similarity often extends to structural similarity, where related proteins retain their basic structure. It has been observed that 30% sequence identity is sufficient for the successful application of homology modeling [Xiang (2006), Nayeem et al. (2006), Wallner and Elofsson (2005)]. The advancements in protein structure prediction are assesed by CASP (Critical assessment of Protein Structure Prediction), a yearly conference, where recently determined, unpublished, crystal structures are used to evaluate the state-of-the-art in structure prediction. Results from the latest CASP10 [Kryshtafovych et al. (2013)] show that template-based modeling involving multiple templates have improved drastically and this improvement is attributed to the increase in the number of high quality protein structures available.

Homology modeling became important with the emergence of structural genomics initiatives which are worldwide collaborations to accelerate the production of meaningful structural information [Goldsmith-Fischman and Honig (2003)]. Structural genomics may help attain a structural understanding of entire genomes by experimentally determining the structures of a smaller representative subset of proteins. The experimentally determined structure should serve as templates to model closely related proteins via homology modeling. Computational methods for homology modeling have been developed to this end and are being successfully employed to complement the genomics initiatives. Several software packages are available for homology modeling [Bordoli et al. (2008),Wang et al. (2008)].

The different stages in the homology modeling of a target sequence are 1) identifying one or more homologs of known structure from the Protein

Data Bank(PDB), 2) sequence alignment of the target with the homologs 3) modeling the aligned regions 4) modeling the loops and gaps and 5) refining the model and side chain optimization. Sequence alignment is a well studied problem in the field of bioinformatics and high throughput methods like BLAST [Altschul et al. (1990)] have been deployed and used widely. Other important components of homology modeling are loop modeling and side chain optimization. Side chain optimization in particular requires the sampling of a large space of conformations and can become the most computation intensive stage of the process. Side chain optimization is discussed in detail in chapters 2 to 4.

Molecular Docking

Docking refers to the prediction of molecular conformations when distinct molecules are bound together, if possible, to form a stable complex. Scientific applications may require proteins docked to other proteins, nucleic acids or to small molecules such as drugs in an effort to study their binding conformation and/or affinity [Ritchie (2008)]. Protein-protein docking is important for the biochemical study of pharmaceutical compounds and a survey of challenges in protein-protein docking is available in [Moreira et al. (2010)]. While the design of accurate scoring functions remains the greatest challenge in this area, modeling side chain flexibility is also an important component for successful protein docking studies [Wang et al. (2005)]. The problem of predicting transmembrane dimers, discussed in chapters 5 and 6, are in a sense, examples of protein-protein docking.

Modeling based on experimental data

In many cases, no structural information is available in the form of homolog structures. However, some experimental data that reveals important structural features, such as the oligomerization state or the interacting

interface, may be available. This experimental data may be used to trigger a computational-experimental analysis cycle where results from one stage feed back into the other leading to an improved model of the protein after each stage. Typically, the results from computational modeling may help plan biophysical experiments and/or the experimental results may guide computational modeling.

Computational modeling is especially useful for studying protein interactions when results from experimental mutation or cross-linking studies are available. Biophysical experiments may be employed to perform protein mutations and study their effect on protein interaction. Results from these studies may reveal some important information. For example, disruptive mutations that alter the behaviour of the protein may indicate interfacial positions where sequence variation introduced by the mutation is inappropriate. Similarly, silent mutations which have no impact on protein behaviour may indicate off-interface positions where sequence variation is tolerable. These mutation effects may be used to design geometric constraints which can be used to prune the search space and model the interacting proteins. The presence or absence of hydrogen bonds, disulphide bridges can also be used to prune the search space to simplify computational modeling. This kind of modeling typically involves backbone sampling to extract backbone candidates that satisfy all or most of the experimental constaints followed by side chain modeling. The granularity and quality of backbone as well as side chain conformational sampling are the most important factors in this kind of computational modeling. A successful case of experimental data driven modeling of the transmembrane dimer of the FtsB protein is presented in chapter 5.

Ab initio prediction

Ab initio prediction, along with protein design (described next) present the most fundamental, but exciting challenges in computational modeling.

Ab initio or *de novo* prediction refers to methods that attempt to predict tertiary structure based only on the amino acid sequence. Traditionally, these methods involve prediction of the secondary structures from sequence, followed by conformational sampling and evaluation of tertiary sructure to determine the global minimum energy configuration. However, with increase in the number of available crystal structures, fragment-based methods [Simons et al. (1997)] have proven to be superior to template-free models.

The fragment based method exploits the relationship between local sequence and structure. Even in the absence of sequences that can be described as homologs (> 30 % identity), several known structures with high local sequence identity may be obtained. These high identity fragments may be assembled using simulated annealing to predict tertiary structure with high accuracy. This fragment-based method has been implemented in the popular Rosetta package [Bonneau et al. (2001)].

Protein design

Computational modeling is also widely employed in protein design, also called the inverse folding problem. Protein design is the process of determining the protein sequence that would fold into a desired 3-dimensional structure. The goals of protein design are manifold; it helps scientists test and refine their knowledge of protein folding and more often, it is aimed at engineering new proteins that can perform better or under a wider range of conditions.

Protein design begins with the identification of a target structure, followed by a definition of the residue degrees of freedom. The residue degrees of freedom refers to the permitted amino acids at each position of the backbone along with sidechain flexibility. Given this framework, the goal is to determine the amino acid sequence that folds into the target structure and *does not* fold into any other state [Street and Mayo (1999),Samish et al. (2011)]. This component which renders specificity to the sequence is called negative design.

The development of accurate force fields remains a major challenge for protein design [Samish et al. (2011)]. Conformational sampling to model side chain flexibility, discussed in detail in chapter 2, is also an important factor which determines the quality as well as the computational cost of protein design.

1.4 Conformational sampling in protein modeling

Continuous, non-linear optimization techniques have been applied to protein structure prediction prediction problems described in the previous section. For example, molecular dynamics programs, which are computer simulations that attempt to model the motion of proteins, predict the trajectory of atoms by numerically solving molecular mechanics equations. These methods have been highly successful in addressing localized dynamics or mechanistics questions, but not very successful in predicting folding or binding. Several advanced software packages such as NAMD[Phillips et al. (2005)] and GROMACS [Berendsen et al. (1995)] are available to biologists. However, molecular dynamics is computationally intensive and application to large systems and long time scales is still limited despite advancements in computer hardware. Several methods, including specialized hardware [Shaw et al. (2007)], are underway to extend molecular dynamics to longer trajectories and larger systems. Although methods that operate in continuous space exist, they are not the focus of this research and will not be discussed further.

Techniques that discretize or sample the conformational space are common in protein structure prediction. Typical approaches involve modeling all possible backbone and side chain conformations, using a representative set of conformations, followed by a search to determine the global minimum energy conformation. In these approaches, the underlying conformational space, which is continuous, is discretized to extract a set of representative backbone

and side chain conformations. The discretization of all levels of protein structure, called conformational sampling, is performed to facilitate the application of advanced search algorithms that operate efficiently in discrete spaces. Since this sampling determines, even prior to the subsequent search, the best attainable solution, it is an important component of the structure prediction process. The development of effective conformational sampling methods is therefore essential for fast and accurate structure prediction.

Side chain optimization

An important component of all the above-mentioned protein structure prediction applications is the side chain modeling stage. Like, structure prediction, side chain modeling is also typically performed as an energy optimization process. Generally stated, the goal of side chain optimization is to identify the most favorable configuration of the side chains for a given backbone. It is a fundamental component of most protein structure prediction and design applications. While the specific details may vary, side chain optimization generally involves four key elements: (1) a backbone that provides a structural template; (2) a side chain library that provides conformational freedom to the various positions; (3) a set of physical and/or empirical energy functions of statistical derivation for scoring; and (4) a search strategy to identify the lowest energy state among all possible configurations.

In many applications, side chain modeling is an independent module employing both discretization as well as continuous optimization, and combinations of the two [Vasquez (1995)]. The use of discrete side chain conformations, called rotamers or conformers, have been extremely popular in this domain due to their speed. This thesis presents methods to improve conformer-based side chain modeling, discussed in detail in chapters 2 to 4.

1.5 Overview of the thesis

This thesis describes the computational methods I have developed for the structure prediction and analysis of membrane proteins. These proteins are notoriously difficult to study using experimental methods. Therefore, computational structure prediction is an important alternative for understanding the structure and function of these proteins.

From a technical stand-point, structure prediction may be considered as two almost independent stages: 1) the modeling of the backbone, followed by 2) the optimal placement of the side chains for each backbone geometry. Since multiple cycles of side chain optimization are often required – one for each conformation of the backbone – side chain placement can become the bottleneck of structure prediction methods. Therefore, fast and accurate side chain optimization methods are essential, particularly for accelerating high-throughput structure prediction methods. I focus on this aspect in the first part of the thesis, which is structured in two parts. In chapters 2 to 4, I focus on methods to improve the speed and accuracy of side chain optimization. The improved performance is important for the structure prediction methods of membrane protein complexes, which are the topic of the later chapters (chapters 5 and 6).

Side chain optimization (described earlier in section 1.4), is a search for the minimum energy configuration among all the combinations of conformations allowed to each individual side chain. This combination results in an extremely high-dimensional search space. Searching this space is simplified by the discretization of side chain conformation, using a set of representative conformations called "conformer libraries". These libraries enable the use of several algorithms that can efficiently search discrete spaces and make the combinatorial problem of side chain modeling tractable. While these libraries cannot be too large, for the sake of computational efficiency, they need to provide sufficient sampling of the conformational space to ensure accurate modeling. To achieve the best possible balance between these two

conflicting goals, I have developed a novel criterion to create conformer libraries based on energy – the same criterion used to select a structure in protein prediction.

Chapter 2 describes the creation of the Energy-Based Library (EBL). I start from an extremely fine-grained conformer library. The energy-based method is applied to sort the list of side chain conformations according to their propensity to fit, energetically, into side chain environments. The top conformers selected by this process enable faster and more accurate side chain prediction. This speedup, demonstrated by experiments on known protein structures, is because the EBL requires a fewer number of conformers compared to traditional libraries.

In the next two chapters, I seek further improvement of side chain optimization by considering aspects of individual side chains. In Chapter 3, I describe the creation of a Backbone-dependent Energy-Based Library (B-EBL) by considering the conformation of the local backbone. Backbone geometry is a major determinant of side chain conformation and backbone-dependent conformer libraries have been known to perform better in side chain optimization. B-EBL is created by reordering the EBL, using the same energy criterion, for each distinct region in protein backbone space. This reordering captures the side chain preferences in these regions more accurately and further reduces the number of conformers required for accurate side chain optimization.

In Chapter 4, I recognize that every position on the protein backbone has a different sampling requirement. For example, solvent exposed positions require less sampling than positions in the core of a protein. This observation may be exploited to accelerate side chain optimization if the sampling requirement of each position can be correctly predicted. I employ powerful machine learning algorithms to predict and allocate the required sampling for each position on the target backbone. I adopted a 3-level classification for the positions based on factors such as the backbone geometry, the solvent

accessible surface area, the amino acid type and so on. Positions were classified according to three different levels of sampling requirements (high, average, low). This customization of sampling on a position-dependent basis led to faster and more accurate side chain optimization. This method highlights another important advantage of the EBL, i.e. its *adjustable sampling*. The EBL is created as a sorted list of conformers that can be truncated to any desired size. This characteristic enables tailoring the size of the library to the needs of each individual position. Conversely, traditional conformer libraries allowed very little choice in the number of conformers used for an application and extracting a library of higher or lower granularity was complicated.

The efficient conformer libraries developed in chapters 2 to 4 speed up side chain optimization and thus, enable the development of high-throughput methods to predict the structure of membrane protein complexes. Membrane proteins often associate with each other to form complexes essential for their function. Therefore, to study their function, a structural model of these complexes is required. Sometimes, these structures need to be predicted *ab initio* from their sequence, whereas in some other cases useful information is available in the form of experimental results. In this thesis, I have developed methods for both these situations: a structural model of the bacterial division protein FtsB was created by interpreting experimental results (Chapter 5); and an *ab initio* method (CATM) was created to predict the structure of a common transmembrane motif at near atomic resolution (Chapter 6).

Chapter 5 describes the structural modeling of the bacterial division protein FtsB, guided by mutagenesis data. FtsB was analyzed using biochemical techniques which revealed its self-associating tendency. Site-directed mutation experiments led to a mapping of the interaction interface of the FtsB dimer complex. I developed a computational method that interpreted this data and created an atomistic structural model to help further experimental analysis. The developed model corroborated the experimental results and

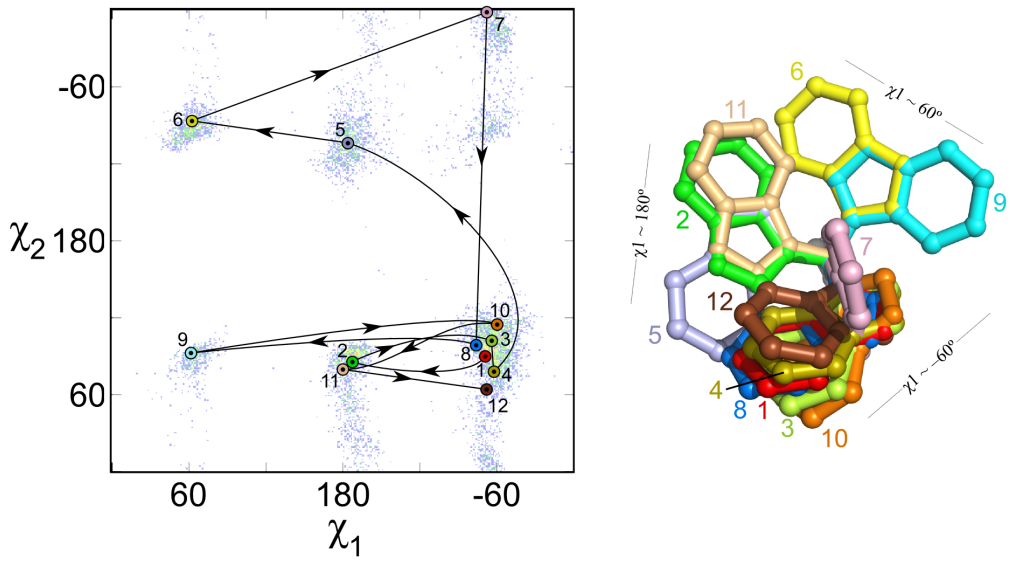
enabled the functional analysis of the FtsB protein complex *in vitro* and *in vivo*.

Chapter 6 describes the geometric analysis of the frequently occurring GAS_{right} motif and the resulting *ab initio* structure prediction method called CATM. GAS_{right} is the most common dimerization motif in membrane proteins. It is characterized by a pair of helices crossing in a right handed fashion and almost invariably, displays GxxxG (or GxxxA/AxxxG) sequence patterns at the helix-helix interface. These glycines permit the helical backbones to come in close contact resulting in the formation of networks of carbon hydrogen bonds between the alpha carbon (Cα) donors on one helix and carbonyl acceptors on the other helix. These hydrogen bonds, presumably, are important for stabilizing the association of these dimers although experimental proof is still lacking. I have addressed this issue by performing an analysis of hydrogen bonding for all possible dimer geometries, revealing that the GAS_{right} motif is, in fact, optimized for Cα hydrogen bonding. Based on this analysis, I developed the high-throughput CATM prediction program, which predicts the structure of GAS_{right} at near atomic resolution.

I have been a major contributor to the development of the software infrastructure on which this thesis is based [Kulp et al. (2012)]. Chapter 7 describes the open-source C++ molecular software library (MSL) used to perform all the computational modeling described in this thesis. MSL is a set of tools that supports a large variety of algorithms for the design, modeling, and analysis of macromolecules. Among the main features supported by the library are methods for applying geometric transformations and alignments, the implementation of a rich set of energy functions, side chain optimization, backbone manipulation, calculation of solvent accessible surface area, and other tools - I have implemented or enhanced several of these features. MSL has a number of unique features, such as the ability to store alternative atomic coordinates (for modeling) and multiple amino acid identities at the

same backbone position (for design). It has a straightforward mechanism for extending its energy functions and can work with any type of molecules. It allows the rapid implementation of simple tasks while fully supporting the creation of complex applications.

2 ENERGY-BASED CONFORMER LIBRARIES (EBL)



based on

Subramaniam S and Senes A "An Energy-Based conformer library for side chain optimization: improved prediction and adjustable sampling", *Proteins* 2012 **80**, 2218-34

Summary

Side chain optimization refers to the process of determining the side chain conformations of the global minimum energy configuration starting from a fixed protein backbone. It is a fundamental component of protein modeling applications such as docking, structural prediction, and design. In these applications side chain flexibility is often provided by rotamer or conformer libraries, which are collections of representative side chain conformations. This chapter demonstrates that the sampling provided by the library can be substantially improved by adding an energetic criterion to its creation. The result of this new procedure is the energy-based library, a conformer library selected according to the propensity of its elements to fit energetically into natural protein environments. The new library performs outstandingly well in side chain optimization, producing structures with significantly lower energies and improved side chain conformation prediction. In addition, because the library was created as an ordered list, its size can be adjusted to any desired level. This feature provides unprecedented versatility in tuning conformational sampling. It allows to precisely balance the number of conformers required by each amino acid type, equalizing their chances to fit into structural environments. It also allows the scaling of sampling to the specific requirement of any given side optimization problem. A rotameric version of the library is also produced with the same method to support applications that require a dihedral-only description of side chain conformation. This chapter is based on [**?**] and the libraries are available online at http://seneslab.org/EBL.

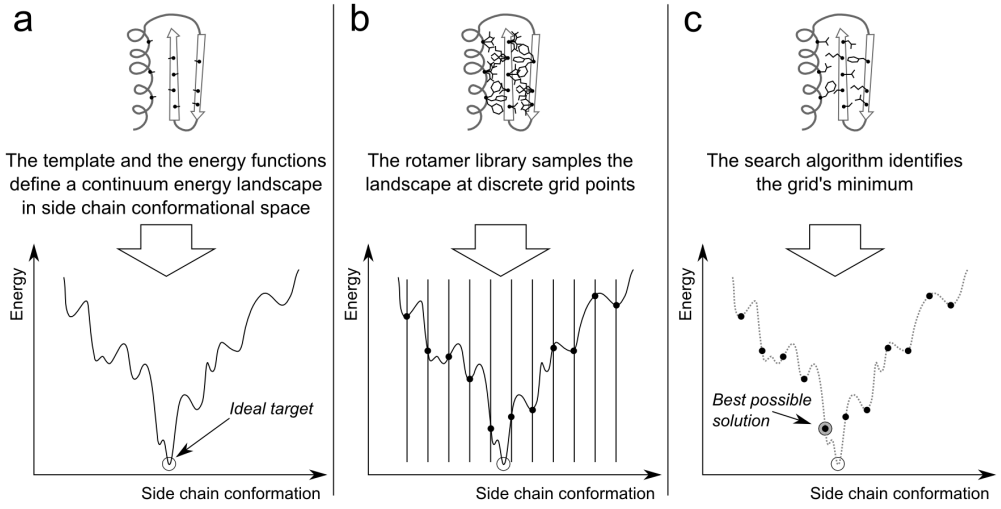


Figure 2.1: *The side chain library predetermines the best possible accuracy of a side chain optimization procedure. (a) The template and the energy functions define a multidimensional landscape (here schematized in 1-D) whose dimensions are degrees of freedom of the side chains. The global minimum of the landscape is the ideal target of the optimization. (b) The introduction of a side chain conformation library produces a grid that discretizes the space. (c) The search algorithm can identify the grid point with lowest energy. Depending on the choice of library this point may lie near or far from the global minimum of the entire landscape.*

2.1 Introduction

Generally stated, the goal of side chain optimization is to identify the most favorable configuration of the side chains for a given backbone. It is a fundamental component of most protein structure prediction and design applications. While the specific details may vary, side chain optimization generally involves four key elements: (Figure 2.1) (1) a backbone that provides a structural template; (2) a side chain library that provides conformational freedom to the various positions; (3) a set of physical and/or empirical energy functions of statistical derivation for scoring; and (4) a search strategy to

identify the lowest energy state among all possible configurations.

Side chain optimization poses a difficult challenge, as the search space grows combinatorially with the number of positions involved and their conformational freedom. The side chain library is essential to transform what is a continuum search space into a discretized problem for which a number of powerful deterministic or stochastic algorithms are available (such as Dead End Elimination [**?**], Branch and Bound [**?**], and Graph Theory [**?**], Monte Carlo [**?**], Self Consistent Mean Field[**?**, **?**]. It is important to remark that the library is key to the quality of the outcome. This is demonstrated in Figure 2.1. The theoretical target of the optimization procedure is the global minimum of the side chain conformational energy landscape, but the landscape is sampled only in a finite number of locations, while the rest remains unknown. The "winner" can approach the global minimum only if the correct side chain conformations were provided by the library. Therefore, the choice of a library predetermines, even before the search is started, the best possible accuracy of the procedure.

The most trivial way to increase accuracy would be increasing the size of the library, and indeed, it has been shown that high-sampling libraries can improve the outcome of side chain optimization [**???**Xiang (2006)]. Higher sampling, however, comes at a significant computational cost as the total number of states can easily reach an intractable number of combinations. Another possible solution to this problem is to adopt a continuous sampling of side chain space. The trade off is reduced efficiency, but methods such as minDEE (minimized Dead End Elimination) [**?**] can be particularly suited for protein design applications in which correct prediction of the landscape global minimum is most critical.

Nevertheless, library-based sampling is still very popular because it is simple to implement, it can be integrated with many algorithms and offers a good balance between speed and accuracy, which is essential, for example, when side chain optimization is repeated multiple times, such as in protein

prediction methods. The questions that we ask here are: how do we improve the library accuracy without affecting its efficiency? Or, for applications in which speed is paramount, how do we improve the library efficiency without affecting its accuracy? The answer to both questions is to identify, for any given size of the library, the set of side chain conformations that will maximize its performance, which is the goal of this work.

2.2 Rotamer Libraries

Currently, the majority of the libraries used for side chain optimization are derivatives of statistical rotamer libraries [**?**Dunbrack (2002)] such as the "Penultimate" library [**?**] and, most commonly, the backbone-dependent (BBD) library of Dunbrack [**??**] which is still actively curated Krivov et al. (2009); Shapovalov and Dunbrack (2011). These statistical libraries are based on the analysis of the distribution of the amino acids' χ angles (the torsional rotations around bonds), which are the main determinants of side chain conformation. The rotamer libraries define the clusters in torsional space, providing their average, dispersion and relative population. Figure 2.2(a) plots the rotamers for an amino acid that contains χ angles exclusively between sp3 carbons (Leu) and one with an sp2 carbon (Asn). The nine rotamers of *Leu* cluster at combinations of the classical staggered conformations, (near -60°, 180°, +60°). The nine theoretical minima, however, are far from being evenly populated because some of the rotamers are disfavored by local conformational strains [**?**]. The tight clustering displayed by *Leu* side chains is not observed when the side chain torsions involve an sp^2 carbon, such as in the case of the $\chi 2$ dimension of *Asn*, and the density is more dispersed.

The adoption of a rotamer library allows to focus the search only on the favorable regions of conformational space. However, the rotameric wells are generally too wide to be covered by a single conformation. Providing

sufficient sampling is indeed a critical issue for side chain optimization because even small atomic clashes can prevent a favorable solution from being identified [Wang et al. (2005); **?**]. A commonly implemented scheme to increase sampling is to expand the main rotamers with a combinatorial addition of ± 1 standard deviations in the $\chi 1$ and $\chi 2$ dimensions resulting into a nine-fold expansion of each rotameric center. Alternatively, expansions can be produced such that the addition of ± 1 standard deviations is operated in the $\chi 1$ or in the $\chi 2$ dimensions, producing the five-fold expansion illustrated in Figure 2.2(a)[Krivov et al. (2009)]. While rational, such expansions are in part arbitrary and do not consider the fact that the relative populations of these regions can range significantly, raising the question of what would be the most effective strategy. For example, the relative density of the nine clusters of *Leu* ranges from as high as 63% of the total in one rotameric region (+60°/180° cluster) down to a mere 0.02% of the density in the least populated region (60°/-60° cluster). A distribution of sampling that somehow reflects this bias would likely be beneficial.

An alternative approach to side chain conformational sampling is the adoption of a conformer library [**??**]. These are collections of side chain conformations extracted from high-resolution structures. They are created from an exhaustive set of side chains that is reduced to a desired number by removing conformers that are too similar to each other using a filter based either on χ-angle similarity [**?**] or on root mean square deviation (R.M.S.D.) [**?**]. The conformer libraries do not involve clustering and expansion and are directly suitable for finegrained sampling, as they can be created in different sizes by tuning the similarity filter. An advantage of conformer libraries is that they retain variation of all degrees of freedom, including, bond distances and angles, in addition to the dihedral angles. In particular, they capture any systematic bond angle variation occurring in sterically strained rotameric regions, which can be large enough to affect the energies [**?**]. It should be noted, however, that the application of a filter based on

geometric similarity flattens the differences between the most populated and the rare regions. For example, while the *Leu* conformer library illustrated in Figure 2.2(b) does not sample the very rare 60°/-60° rotameric region, its relative coverage of the -60°/180° (63%), 180°/60° (30%), 180°/180° (2.6%) and -60°/60° regions (0.7%) is not proportional to their densities.

The rotamer and conformer libraries have been fundamental tools in protein modeling and design. Particularly the seminal backbone-dependent library is at the core of a number of modeling methods which have enabled important achievements in prediction and design [**???**]. Their continued development is important to improve accuracy and reduce run time when applications require high throughput, high sampling, or when side chain optimization is repeated multiple times in concert with backbone motions. The expanded rotamer libraries and the conformer libraries are both based on the natural distribution of side chain conformations in proteins. Both approaches greatly reduce the size of the search space by providing good guidance on where sampling should be allocated, excluding any regions of conformational space that are energetically unfavorable. The question is up to what point the natural distribution of side chain conformation can inform how to best prioritize sampling within the rotameric regions, and what additional information could be used to improve the sampling strategies. An important consideration in this regard is that protein side chains are co-evolved with their environment to complement each other. Consistently, it has been observed that the conformational preferences of the amino acids in the structural database reflect primarily the internal steric constraints and local backbone interactions, and only marginally the effects imposed by the surrounding environment [**?**]. On the other hand, in side chain optimization the backbone is generally fixed, the side chains only have discrete mobility,and the structure is unable to undergo those small movements that would relax any minor clashes. Therefore, the primary factor that determines the probability that a given set of representative

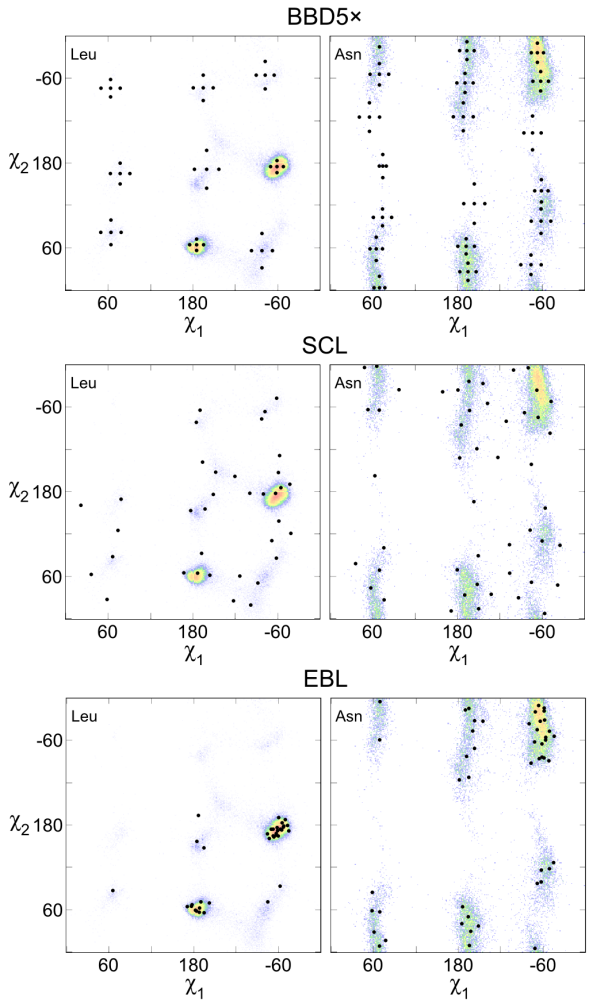


Figure 2.2: *χ1/χ2 plot of an expanded rotamer library, a conformer library and the energy-based conformer library. (a) a 5-fold expansion of the Backbone Independent library in which each rotameric region has been evenly enriched with subrotamers that vary by ±1 S.D. in either χ1 or χ2 dimension. The figure shows the χ1/χ2 plot (black dots) for a side chain with torsions between two sp3 carbons (Leu, 9 x 5 = 45 rotamers) and one characterized by a sp2 carbon in the χ2 dimension (Asn, 18 x 5 = 90rotamers). The dots are overlaid on a color coded density map of the side chain distribution in the structural database. (b) χ1/χ2 distribution for the same two amino acids in the mid-sized (0.5 Å RMSD) conformer library of Shetty et al. Leu: 36 conformers. Asn: 48 conformers. (c) χ1/χ2 plot of the first 36 conformers of Leu and 48 conformers of Asn of the energy-based Library. The numbers are chosen to allow a direct visual comparison with the SCL (b). In the EBL the conformers are not evenly spaced but tend to cluster with a bias that is similar to the conformational distribution observed in the structural database.*

conformers will contain a fitting solution are the interactions of the side chain with the environment in which it is reconstructed. It follows that in order to maximize their chances of fitting, instead of using a pure geometric criterion either in cartesian or torsional space, it would be preferable to space the conformers evenly according to the energetic impact of their motions i.e. the likelihood that a motion would produce significant energy variation.

We hypothesized that the introduction of an energetic criterion into the selection of the conformer library would lead to more effective prioritization of sampling in side chain optimization. The relationship between side chain geometry and the energetic impact of their motions is complex and difficult to derive analytically, as they depend very specifically on their structures and the degrees of freedom altered. As illustrated in Figure 2.3, the energetic impact is related to the number of atoms that are displaced and also to the distance traveled by these atoms, which depends on their distance from the axis of rotation. For example, $\chi 1$ rotations are likely to impact the energies more than $\chi 2$ rotations because they translate more atoms and for a further distance. For the same reason, $\chi 1$ rotations of the bulky *Trp* are more likely to impact the energies than $\chi 1$ rotations of the smaller Leu. Therefore, it is clear that $\chi 1$ should be allocated more sampling than $\chi 2$, and that the bulky *Trp* should be allocated more sampling than Leu. The question is how much more? Here we address the problem with a practical approach based on the analysis of how an extensive library of conformers interacts with a wide variety of natural protein environments. The data is used to sort the conformers by their propensity to fit (energetically) into protein environments. We have compared the resulting library with three libraries from the literature and observed important performance improvements with the new approach, both in energetic terms as well as side chain conformation recovery. The approach also introduces a new beneficial feature: because the library is sorted, the number of conformers can be resized to any desired level of sampling. This feature provides unprecedented flexibility in adjusting,

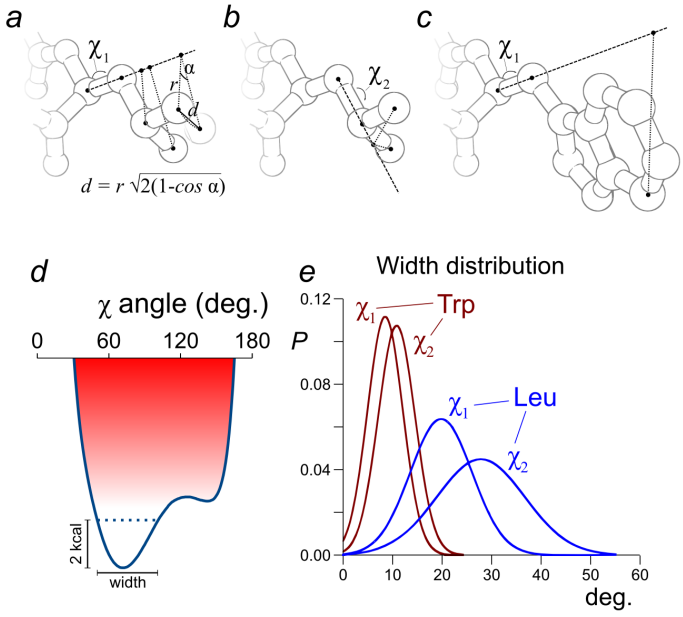


Figure 2.3: *The energetic impact of* χ *angle rotations varies depending on the* χ *angle and the amino acid type. The energetic impact of a rotation (i.e. the relative change in energy) depends on the number of atoms that are moved and the distance traveled. The cartesian distance d traveled by an atom after rotation* α *is proportional to the distance r from the axis of rotation.* χ*1 rotations (a) rotate more atoms over a longer distance than* χ*2 (b). This same rotation applied to larger side chains, such as* Trp *(c), will result in even larger displacements, and likely, greater energy variations. This is demonstrated by the different distribution of "energy wells" when the* χ *rotations are applied in a fixed environment. d) The width of the energy well is defined as the range that the angle can travel without exceeding a threshold of 2 kcal mol* $^{-1}$ *from the minimum energy (calculated using the CHARMM parameter 22) during a single* χ *angle rotation in a fixed environment in a protein crystal structure. e) The average well for Leu's* χ*1 is significantly narrower than* χ*2 (20°* *and 28°* *respectively). The bulky* Trp *shows significantly narrower tolerance to rotation, with 9°* *and 11°* *of average allowed rotation. The figure shows the distribution of energy well width for 1000 buried* Leu *(max SASA 20Å*^{2}*) and* Trp *(max SASA 60Å*^{2}*) side chains in their specific environment in high resolution crystal structures.*

even dynamically, the combinatorial size of the optimization to match the precise needs and limits of a procedure.

2.3 Materials and methods

Structure database preparation

A collection of 2159 high resolution x-ray structures was obtained from the Protein Data Bank (PDB) using the following conditions: resolution < 2.0 Å ; deposition date: later than 01/01/1998; method: X-ray diffraction; molecule type: protein (no DNA, no RNA); no ligands. The proteins were filtered to allow no more than 30% sequence identity between individual chain. Hydrogen atoms were added with the program Reduce [**?**] which also performed any necessary rotation of the hydroxyl groups, flipping the side chain of *Asn, Gln, and His* and determine the protonation state of *His* to optimize hydrogen bonding (-BUILD -ROTEX options). The three protonation states of His are referred here as His-δ (neutral, protonated on ND1), His-ϵ (neutral,protonated on NE2), and His-p (doubly protonated and positively charged). The proteins were curated with an automated procedure that rebuilt missing side chain atoms, removed multiple side chain conformations, and converted any main chain missing amino acids into chain termini. All protein structures were then minimized with CHARMM35 (using the CHARMM 22 potential), with 3 cycles and 50 steps of adopted basis Newton Raphson method using a harmonic potential with a force constant of 100 kcal $\mathrm{mol}^{-1}\mathrm{Å}^{-2}$ [Shapovalov and Dunbrack (2011)]. Minimization was required for two reasons. The bond lengths needed to be homogenized because differences in refinement methods create variability which is within experimental uncertainty but sufficient to produce significant energy penalties. Minimization can also resolve the occasional small clashes that may occur in poorly refined regions of the crystallographic models. The minimization procedure was selected to reduce these unwanted effects

while preserving the natural conformation observed in the crystallographic models. The final RMSD of the crystallographic and minimized models was on average 0.05 Å. The differences in the side chain torsion angles are at most few degrees, and it has been previously demonstrated that preminimization of the structures has no significant influence on side chain prediction [**?**]. A typical example of crystallographic and minimized models is shown in Supporting Information Figure S2.

Preparation of the fine grained (unsorted) conformer library

A set of 1000 proteins was randomly selected from the structural database for the creation of the conformer library and the selection of the environments. All side chains with a B-factor $\geqslant 40$ and those with missing atoms in the original structure were not considered. Any side chain with a Ca to Ca distance below 8 Å from side chains with missing density was excluded in the selection of the environments. For each amino acid type, up to 5000 side chains were randomly selected as environments (except *Cys*: 1614; *Met*: 4296; and *Trp*: 2734). One single set of environments was selected for all protonation states of *His*. Up to 25,000 side chains were set aside for the creation of the initial (unsorted) conformer library. The conformers were selected at random and added to the conformer list if they had an RMSD $\geqslant$0.05 Å from all other previously collected conformers (RMSD filtering). A conformer library was created independently for the three protonation states of *His*-δ, *His*-ϵ, *His-p*. Each conformer library was topped to 5000 conformers (except *Cys*: 1780; *His*-δ: 2906; *His*-ϵ: 4221; *His-p*: 542; and *Val*:3916). *Cys* residues in disulfide bonds were excluded from the analysis. For the creation of the rotameric version (dihedral only) of the energy-based library, the bond lengths and angles of the conformers were standardized to the standard values from CHARMM 22 topology prior to RMSD filtering. The remainder of the procedure followed was the same for both conformer

and rotamer versions of the library.

Creation of the sorted energy-based library

The fine grained conformer library was sorted by the propensity of its elements to fit in the largest number of natural environments, creating the energy-based library. The sorting procedure is schematically explained in Figure 2.4. For each amino acid type, the conformer that satisfied the largest number of environments was selected as the top conformer. All the environments satisfied by the first conformer were marked and no longer considered.The conformer that satisfied the largest number of remaining environments was then selected and the process was repeated. After each selection, however, the threshold was lowered and made more stringent: if an environment that was previously excluded was no longer satisfied at the lower threshold, it was put back into consideration. The process was repeated until all conformers were sorted. The threshold was scaled down linearly from its initial value to reach zero at the end of the sorting process.

Preparation of the benchmark libraries

Three previously published rotamer and conformer libraries of different sizes were selected for comparison. The 5-fold expansion of the 2010 version of the backbone-dependent library [Krivov et al. (2009); Shapovalov and Dunbrack (2011)] (here referred as BBD5x) was built with standard bond lengths and bond angles from CHARMM 22 topology. The rotamer library mean rotamers were expanded by ± 1 standard deviation in $\chi 1$ or $\chi 2$ but not in both dimensions simultaneously. This led to a 5-fold expansion of amino acids with at least two χ angles, and 3-fold expansion of amino acids with a single χ angle. The dihedral relative to the hydrogen atom of hydroxyl groups was sampled at the canonical -60°,180°, and +60° minima, each one expanded by $\pm 30°$ (9 total steps) for *Ser* and *Thr*, and every 45° (8

steps) for *Tyr*. The expansion generated a total of 3,755 conformers (Table A.1). The two benchmark conformer libraries selected were the "medium" size library (0.5 Å RMSD) from Shetty et al [**?**]. (here referred as SCL) which contains 1547 conformers, and a small conformer library from Xiang and Honig [**?**] created from a database of 297 proteins, 100% coverage and 40° tolerance, which contains 1134 conformers (here referred as XCL). For consistency, and in particular to avoid any significant differences in the bonded energies of the conformers, both conformer libraries were subjected to constrained minimization (conformers were built and minimized in a Gly-X-Gly tripeptide).

Single side chain repack tests

Single side chain repack tests were performed on a set of 2000 environments obtained from 700 proteins that were set aside from the initial structural database for testing purposes. The test is similar to the conformer sorting procedure, in which the native side chain found in an environment is remodeled into a conformer and the interaction energies are calculated. Conformers were defined to satisfy the environment by the condition previously explained. An environment was defined to be satisfied by a set of n conformers if at least one of the elements satisfied the environment.

Complete protein repacks

Complete side chain repacks were performed on a subset of 560 of the 700 proteins set aside for testing purposes. All side chains were removed and predicted except *Gly, Ala, and Pro. His* residues were predicted using in the protonation state assigned by Reduce [**?**]. The optimization was performed with the program *repackSideChains* using a sequence of algorithms: first a run of Dead End Elimination (DEE) using Goldstein single criterion [**?**] was used to reduce the combinatorial space. A round of Self Consistent Mean

Field (100 cycles, temperature 300 K) was performed on the conformers that were not eliminated during the DEE phase and the protein was set in the resulting most probable state. Finally a Monte Carlo simulated annealing procedure was run (50,000 cycles, with exponential cooling from 1000 to 0.5 K). The structure with the lowest energy identified by the Monte Carlo run was the final product of the optimization.

Analysis

Conformation prediction of the crystallographic side chain conformation (side chain conformation recovery) was performed by matching the $\chi 1$ and $\chi 2$ of the predicted and crystallographic structure with a tolerance of 40°. The analysis was performed on all side chains and on a subset of buried side chains. A side chain was defined buried if it had a solvent-accessible surface area (SASA) below 25% of the maximum possible SASA for the side chain reconstructed into a *Gly-X-Gly* backbone (with X being the amino acid type under exam). The hydrogen bonding recovery was calculated as follows. First, all side chain-to-side chain and side chain-to-backbone hydrogen bonds were identified in the native structure if they had nonzero energies using the explicit hydrogen bonding function. A hydrogen bond was considered recovered if the interaction between the same donor and acceptor had nonzero hydrogen bonding energy in the predicted structure.

Programs

All calculations (modeling, energy evaluations, conformational analysis, SASA measurements, etc.) were performed with ad hoc programs written using MSL [Kulp et al. (2012)], a C++ object oriented software library for molecular modeling and analysis, which is freely distributed under an open source license at http://msl-libraries.org. The total protein repacks were performed with the program *repackSideChains*, which is distributed with

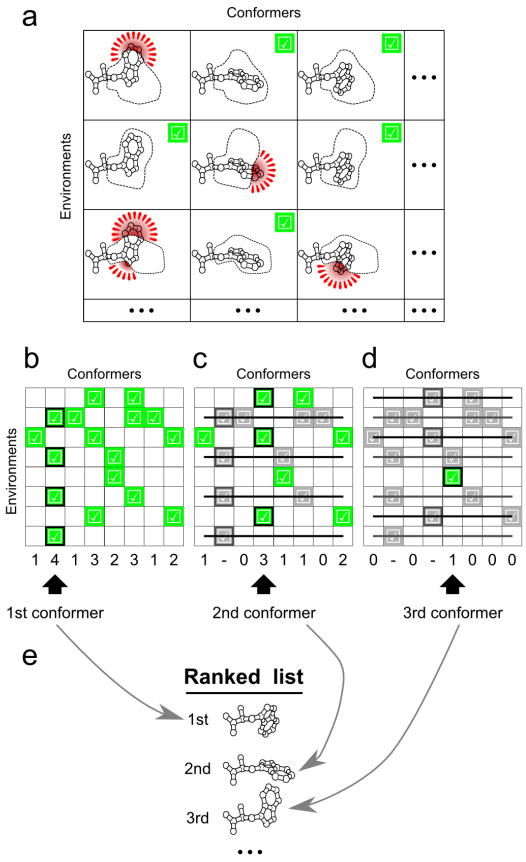


Figure 2.4: *Procedure for the creation of the energy-based conformer library. (a) Each conformer of a fine-grained library of size N is built in each one of M of environments that contain the same side chain type (*Trp *in the figure) in protein crystal structures. The interaction energies of each conformer in each environment are calculated and if the energy is below a certain threshold, the conformer is considered a fit for the environment (illustrated as a green check mark in the figure). (b) The results are stored in a NxM boolean table. The number of environments satisfied by each conformer is determined (number under the table). The conformer that fits the largest number of environments is the first to be selected (black arrow). (c) The environments that were satisfied by the first conformer are no longer considered, and the procedure is repeated to find the conformer that would satisfy the most environments that are still uncovered. (d) The procedure is repeated until completion. (e) The resulting library is compiled as a ranked list, in which every additional element complements the previous. The major advantage of ranking the conformers is that it allows the user to truncate the library at any desired size, which is not possible with a traditional conformer library.*

MSL.

2.4 Results and discussion

The energy-based conformer library

The energy-based library (EBL) is an extremely finegrained conformer library sorted by the propensity of its elements to fit in a wide variety of natural protein environments. The procedure used to derive the library,explained in detail in the Methods section and illustrated in Figure 2.4, is the following:

1. A very finely grained library of N conformers is created for each amino acid type.

2. A large number M of environments that contain the same amino acid type is selected at random from high-resolution crystal structures.

3. The native side chain of each environment is remodeled into each of the conformers, and the interaction energy between the conformer and the environment is measured [Fig.2.4 (a)].

4. The data is collected in an N3M table of energies.

5. Each energy is converted to a boolean value, indicating if the environment is "satisfied" (True, if energy $<$ threshold) by the conformer, or not satisfied (False).

6. The conformer that satisfies the largest number of environments is added to the library [Fig. 2.4(b)].

7. All environments satisfied by the conformer are marked and no longer considered.

8. The threshold is lowered by a small amount. Any previously satisfied environment that would no longer be satisfied at the new more stringent threshold is brought back for consideration.

9. The procedure is repeated from #6 until all conformers are sorted.

This procedure selects the conformers with the highest propensity to energetically fit in environments that contain the amino acid type in natural proteins. The first conformer selected is invariably a conformer near the center of the most populated region of side chain conformational space. The second conformer complements the first by covering another dense region, most often the center of the second most populated cluster. Step 7 is the key step that ensures this complementarity. Without step 7, the second and the other top conformers would most likely be very close structural neighbors of the first pick. By removing from consideration the environments satisfied by all previous conformers, the procedure ensures that each element extends the coverage to new areas of conformational space (the problem would be classified as a classical *Set Cover Problem* in computational complexity theory). The use of a variable threshold that becomes more stringent at every cycle (Step 8) allows to sample further between conformers as the library becomes larger.

The method is based on an energetic criterion for the selection of conformers, but it also incorporates two sources of natural conformational bias. The fine-grained conformer library - albeit being "flattened" by the application of a similarity filter - excludes any energetically unfavorable regions of the conformational space. The most important factor, however, are the environments. They were randomly selected and not filtered, and thus the environments reflect the natural conformational preferences of the amino acids side chains they contained. During the sorting process, the environments essentially "vote" for conformers, and are more likely to chose those that belong to the same rotameric region of the side chain they originally contained. A second important aspect that is likely to affect the

selection process is how tightly packed the environments are around their side chain. The environments of surface exposed positions are more likely to accommodate a variety of conformers, voting more indiscriminately than those that belong to core positions. This aspect not only affects the selection of the conformers, but as discussed later (in the "sampling level" section) it also has important ramifications for balancing sampling between the various amino acid types.

Choice of energy functions

An important initial step in defining the procedure for the creation of the library was to identify a good choice of energy functions. All bonded terms (bond, angle, dihedral, improper terms, from the CHARMM22 parameter set [MacKerell et al. (1998)]) were included to penalize conformers that are internally strained. The first function analyzed was the van der Waals function. A common practice in side chain optimization is to soften the repulsive component of this function, reducing the negative impact of any small clashes that may occur, under the rationale that they would be readily relaxed by small side chain and backbone motions in a flexible protein structure. This is often accomplished by the adoption of ad hoc functions and/or by rescaling the van der Waals radii [**?**]. Here we tested whether the use of reduced radii was beneficial for the creation of the library. The second issue tested was related to the electrostatics, salvation, and hydrogen bonding functions. These three inter-related forces are notoriously difficult to model in side chain optimization and their treatment varies widely between applications [Krivov et al. (2009); **?**); **?**]. The simple inclusion of partial charges may not improve side chain prediction when the effect of solvent are unaccounted for [**??**]. Moreover, the hydrogen bond âŁ" a key factor in predicting the structural organization of protein folds and proteinâŁ"protein interfaces âŁ" has a complex geometry dependency and is not well modeled by an integration of Coulomb and Lennard-Jones interactions [**?**]. To try

to maximize the hydrogen bonding prediction capabilities of the library we chose to test three simple conditions: (1) pure coulombic interactions, (2) an explicit hydrogen bonding function without the electrostatic term, and (3) an equal weight of both terms. We selected the hydrogen bonding function implemented in the SCWRL4 program[Krivov et al. (2009)] because it is based on elements of the CHARMM force field and has multiple angle dependencies.

To test van der Waals radii rescaling, we created three separate conformer libraries using 100% (full), 95% and 90% radii. We tested the libraries in a series of procedures in which a single side chain was placed in fixed protein environment (referred here as "single side chain repacks"). In the procedure we determine what percentage of the environments was satisfied by each truncation of N conformers of the library (for N = 1 to the size of the library). A direct comparison of the performance of the resulting libraries, performed under all three conditions, that is, 100, 95, and 90% radii, revealed minor differences and did not identify a significant advantage in using rescaled radii. In the second test we found that the repacking efficiencies were similar but the hydrogen bonding recovery was higher when an explicit hydrogen bond function was used without electrostatics. These conditions were chosen for the remainder of the work.

The energy-based library

The energy-based library is a sorted conformer library of up to 5000 conformers for each amino acid type, except *Gly, Ala, and Pro* (Table A.2). The three protonation states of His were treated separately because they are chemically distinct (referred here as His-δ: neutral, protonated on ND1; His-ϵ: neutral,protonated on NE2; and His-p: charged). A library of 5000 conformers per amino acid is exceedingly large but the sorted list can be truncated to any desired number of elements. Figure 2.2(c) shows a plot in $\chi 1/\chi 2$ coordinates of the top ranking elements of the EBL library of

Leu and *Asn*. To allow a direct visual comparison with the conformer library of panel *b*, the number of conformers shown is identical. The EBL conformers are less evenly spaced and have a higher propensity to sample the most common regions, which is particularly evident in the case of *Leu*. To illustrate the precise order in which the conformers are ranked and how the algorithm initially prioritizes sampling of the most populated area and gradually extends coverage, Figure 4 shows a "walk" through the first 12 conformers of *Trp*. The first conformer lands in the center of the -60°/90° region (35% of the total density), which is sampled six times within the first 12 conformers. The library then visits the second most populated region (-180°/90°, 14% of the density) and remaining conformers gradually extend sampling, roughly in the order of the relative density of the clusters. The structural superimposition of the first twelve *Trp* conformers in Figure 2.5(b) shows how the conformers complement their coverage of tridimensional space. Even the closely spaced conformers that belong to the -60°/90° region are sufficiently shifted by $\chi 1$ and $\chi 2$ variations (and in parts also by bond angle variations) to cover different portions of tridimensional space.

Selection of benchmark libraries for testing

To test the performance of the energy-based library we selected three previously published libraries: (1) a 5-fold expansion of the backbone-dependent rotamer library of Dunbrack [**?**Shapovalov and Dunbrack (2011)] (BBD5x); (2) a medium-size conformer library from **?** (SCL, for Shetty Conformer Library); and (3) a small conformer library from **?** (XCL). The benchmarks were chosen based on their sizes, to compare the performance of the EBL over a range of sampling levels. The backbone-dependent library is the most popular rotamer library in the literature. The expansion scheme adopted is the one implemented in SCWRL4 [Krivov et al. (2009)], in which "sub-rotamers" are added by expanding either $\chi 1$ or $\chi 2$ by ± 1 S.D. but not both at the same time (illustrated in Fig. 2.2). The version of the BBD

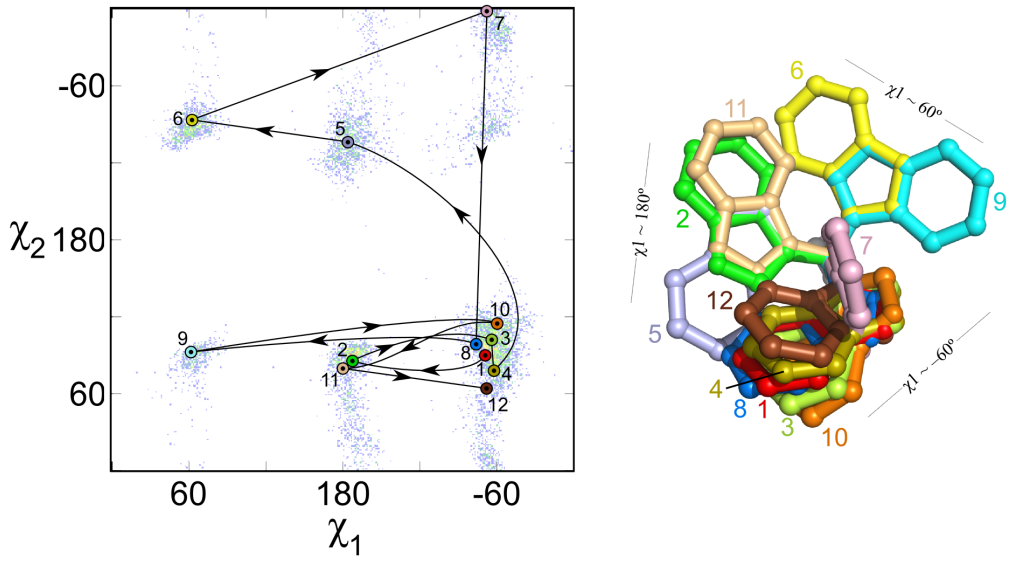


Figure 2.5: *A "walk" in* Trp *space. (a)* χ*1/*χ*2 plot of the first 12 conformers of* Trp *overlaid on a color coded density map of the side chain distribution in the structural database. The most populated region (-60°/90°) is sampled multiple times while the coverage gradually extends to the other regions of density. (b) Structural representation using the same color coding of panel a. The figure demonstrates how the conformers are arranged to cover complementary regions of tridimensional space.*

library adopted is the most recent [Shapovalov and Dunbrack (2011)], as it demonstrated significant performance enhancement compared to the previous version [**?**] in our preliminary tests. The BBD5x contains a total of 3755 conformers, which were built with the bond lengths and bond angles defined in the CHARMM 22 topology file. The SCL selected was the intermediate possibility (0.5 Å R.M.S.D. similarity filter), which contains a total of 1549 conformers. The conformer libraries of Xiang and Honig are at the core of SCAP and Jackal, a suite for modeling and analysis [**?**]. Among the many possible sizes, we selected the library derived from 297 proteins with 100% coverage and 408 bins totaling 1136 conformers, which provided the

opportunity to test the EBL at a relatively low level of sampling.

Performance test using single side chain repacks

To test the performance of each individual amino acid of the new library against the benchmarks we first performed a series of single side chain repacks in fixed protein environment using a set of proteins that was set aside for testing purposes. The results are shown in Figure 2.6. The histograms show the total number of EBL conformers that are necessary to match the performance of the benchmark library. The data was obtained in the following way: for each individual amino acid we calculated the fraction of protein environments that was satisfied by at least one of the conformers of the benchmark library and then we determined the number of EBL conformers that were necessary to satisfy the same fraction of environments. In all cases the difference in performance is extremely significant. All across the three comparisons, the sampling requirements of the EBL are always lower, often by a factor or 10 or more and always by at least a factor of 2. These results demonstrate that the EBL conformers have a high propensity to fit into protein environments that should be able to accommodate the side chain, which was the original premise behind the selection procedure. The next question was whether the improved performance would also be observed in side chain optimization procedures in which multiple side chains are modeled at the same time.

Performance test using total protein side chain predictions

We tested the energy-based library in a series of protein side chain prediction runs in which all positions in a protein (excluding *Gly, Ala, and Pro*) were remodeled using side chain optimization (referred here as total protein repacks). The scatter plots in Figure 2.7(a) compare the final energies of

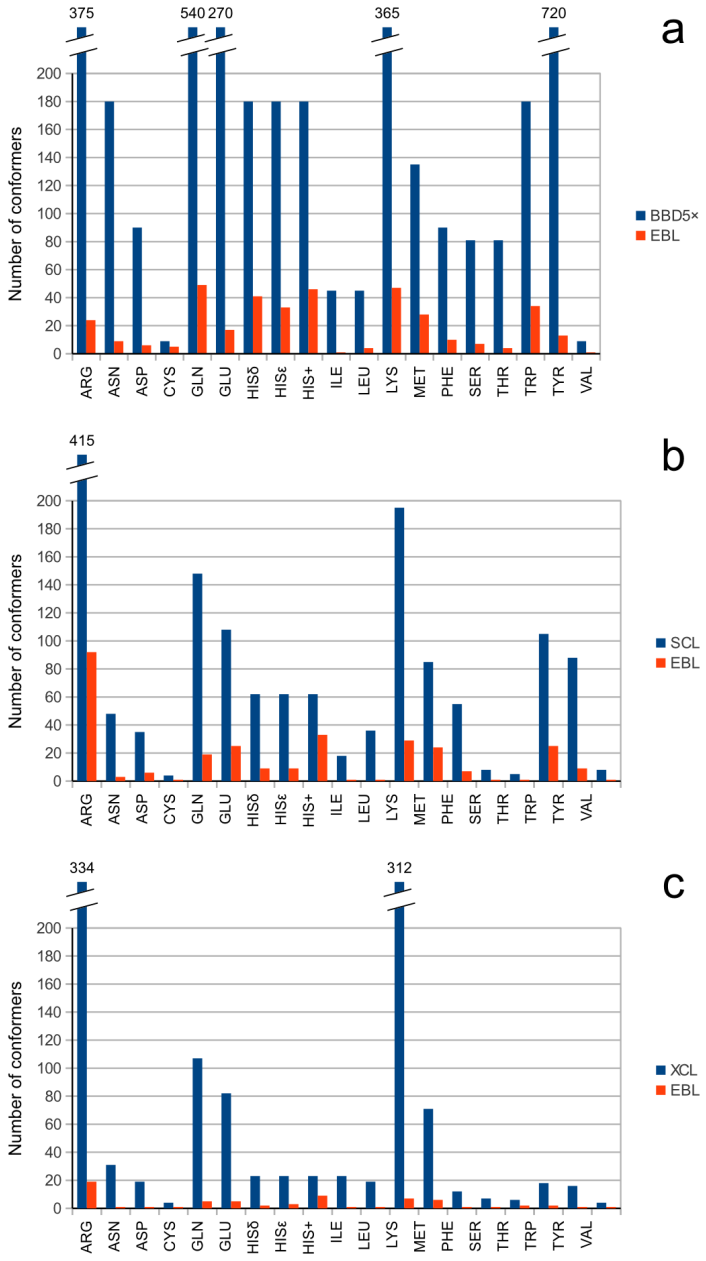


Figure 2.6: *Performance test on single environment repacks. The figure compares the number of conformers that are required for equivalent performance between the energy-based library and three benchmark libraries (BBD5x: 5-fold expansion of the 2010 backbone-dependent library; SCL:a medium conformer library from Shetty et al.; XCL: a medium conformer library from Xiang and Honig). The number of conformers of the benchmarks is a fixed number (blue bar). We determined the fraction of environments that are satisfied by at least one of these conformers. The red bar represents the number of EBL conformers that are required to satisfy the same fraction of environments. For example, the XCL has 334 Arg conformers, which satisfy 55.0% of* Arg *environments. It takes only 19 conformers of the EBL to satisfy at least the same fraction.*

a set of 560 proteins after optimization with the EBL against the three benchmarks. In these repacks the number of conformers for each individual amino acid was exactly matched to the benchmark library. In all comparisons the majority of the points lay above the diagonal (97.3% against the BBD5x, 88.8% against the SCL, and 73.1% against the XCL), demonstrating that the EBL is much more likely to reach lower energy solutions. For ease of comparison, the energies are plotted after subtracting the "crystal energy", that is the energy of the native structure after constrained minimization. The crystal energy is used here solely as a convenient reference under the assumption that in many cases, but certainly not in all, the minimized crystal structure is devoid of strains and represents a good target for an optimization. Panels b of Figure 2.8 represents the same data of panel a in histogram form. This view highlights the distribution of energies obtained with the different libraries with respect to the crystal energy (zero value). In the three sets of calculations the EBL fares well, with a number of solutions below the crystal energy (87.3%, 70.8%, 32.0%) that is significantly higher than the respective benchmarks (BBD5x 13.3%, SCL 10.3%,and XCL 1.0%, respectively). The modes of the energy distributions are shifted by hundreds of kcal mol^{-1} compared to those of the benchmarks. It should also be noted that even at the lowest level of sampling (1136, the same number of conformers of the XCL) the EBL produces a number of proteins below crystal energy that is greater than the larger SCL (1549 conformers) and the BBD5x (3755 conformers). The data demonstrates that the introduction of an energetic criterion in the creation of the conformer library greatly enhanced the energetic performance of the library in side chain optimization.

Sampling levels

In the first performance test, the number of conformers of the energy-based library was matched exactly with the respective benchmark library. However,

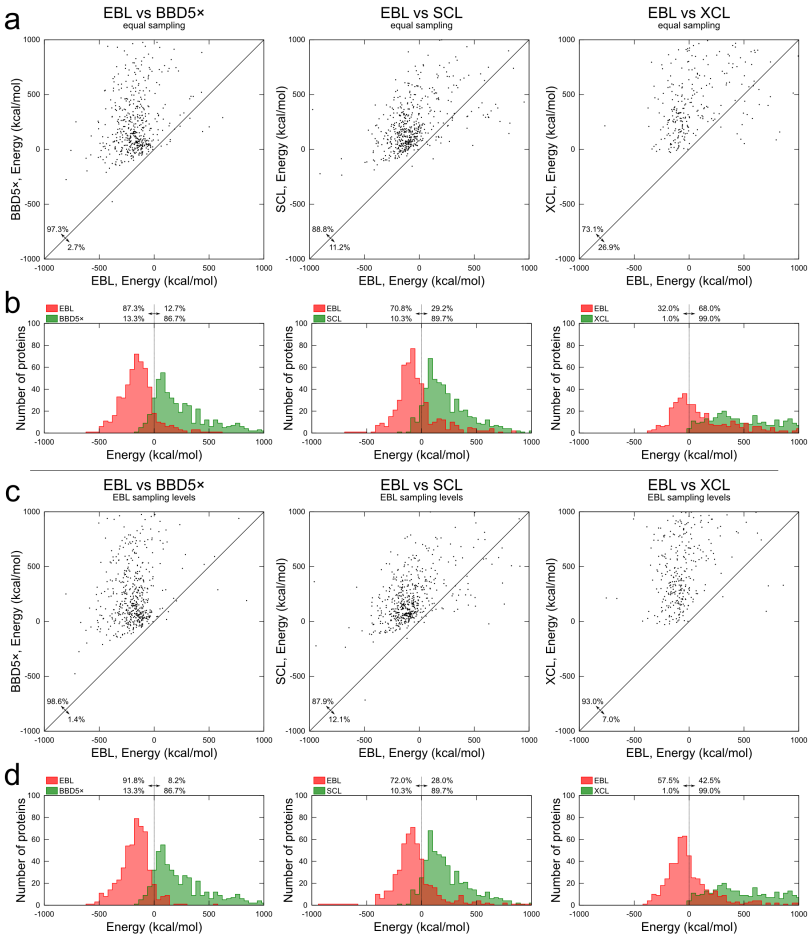


Figure 2.7: *Performance of the energy-based library in total protein repacks. (a) The scatter plots graph the final energy after optimization of all side chains in 560 proteins, for the energy-based library (EBL, x axis) and three representative rotamer and conformer libraries (see Methods). The majority of the points lie above the diagonal indicating that the EBL on average achieves better performance that the benchmarks. For easier comparison energies are plotted after subtracting the energy of the minimized crystal structure. (b) Representation of the same data as histograms. The dashed line separates the proteins that score better than the crystal energy (percentages indicated), a convenient reference under the assumption that in most cases it represents a good target for an optimization. In a and b the calculations were made with an equal number of conformers compared to the benchmark for each amino acid type (equal sampling). The BBD53 has 3,755 conformers, the SCL 1,549 and the XCL 1,136. Panels c and d report the results of the same calculation performed using EBL sampling levels of similar total complexity of that of the benchmarks.*

since the number of conformers can be adjusted to any desired number, it is possible that the optimal distribution of sampling between the various amino acid types could be different. We addressed this question using data from the single side chain repacks against fixed protein environments. We determined the number of conformers that are required to satisfy a certain percentage of protein environments in the test set. This led to the creation of a series of "sampling levels" which, at least in principle, should provide each amino acid type with an equal chance to fit into protein environments. We created 14 levels, from very sparse sampling (282 total conformers in the 60% level) up to very high sampling (6985 conformers in the 99% level). The percentage refers to the number of environments that are energetically satisfied by the set of conformers in single side chain repacks. These levels will be referred as SL60 (Sampling Level 60%) to SL99. The number of conformers in each sampling level is reported in Table A.3. The balance within each level is consistent with the expectation that larger amino acids would require more sampling than smaller amino acids. A second factor that presumably contributes is the propensity of an amino acid type to occur in tightly packed positions (which likely require more sampling) versus solvent exposed positions (which can be satisfied by a larger variety of conformers). A substantially larger number of conformers is given to amino acids with hydroxyl groups compared to other amino acids with similar structure (for example, *Thr* vs *Val*, and *Phe* vs *Tyr*), an indication that a significant amount of sampling is required to satisfy hydrogen bonding.

Complete protein repacks using the EBL sampling levels

The total repack tests of the same set of 560 proteins were repeated using the sampling levels. The results are shown in Figure 2.7(c,d). Since the number of conformers for each individual amino acid is no longer matched to the benchmark libraries, to ensure a fair comparison we matched the

total combinatorial complexity of the search space, that is the product of the number of conformers given to each position. The total combinatorial complexity of each optimization was calculated for the benchmark library, and the same protein was repacked with the largest EBL sampling level that did not exceed the benchmarkâŁ™s total complexity. It should be noted that this criterion always puts the EBL at a disadvantage, at times minimally, at times significantly, ensuring a stringent test. The most frequently selected levels were (in order of frequency) the SL95, SL96, and SL92.5 against the BBD53 library, the SL85, SL87.5, and SL82.5 against the SCL, and the SL80, SL75, and SL82.5 against the XCL. This discussion will refer to this strategy as "sampling levels" and to the previous strategy, in which benchmark and EBL were equally matched, as "equal sampling".

Optimization with the sampling levels produced a significant improvement of the energies in the comparisons against two of the three libraries, the BBD5x and XCL. In the comparison against the BBD5x library, the fraction of proteins below the crystal energy increased from 87.3% to 91.8%. This improvement can be appreciated visually by comparing the frequency of proteins just above zero energy in Figure 2.7(b,d). The most significant improvement was found in the comparison against the smallest of the three libraries, the XCL. In this case the number of proteins below crystal energy almost doubled, going from 32.0% to 57.5%. The introduction of sampling levels was not as beneficial in the test against the SCL (the number of proteins below crystal energy increased from 70.8% to 72.0%). It should be noted, however, that the adopted strategy to compare with the "equal sampling" puts the "sampling levels" strategy at disadvantage and likely contributes for this small improvement. This can be appreciated by comparing the total size of the most commonly used sampling levels. The main sampling levels used against the SCL (SL85, SL87.5, and SL82.5 levels, with 1231, 1464, and 1039 conformers, respectively) are all smaller in size than the size of the "equal sampling" strategy (1549 conformers).

In fairness, it should also be noted that the overall size of the BBD53 can be reduced by excluding the most rare rotamers from the library [**?**Krivov et al. (2009)]. If this filtering is applied to maintain at least 99% of the cumulative density, the total complexity of the BBD5x in total repacks decreases approximately to the same level of the SCL. The application of a 90% filter reduces the BBD5x to approximately the XCL complexity. If we compare the energetic performance of the EBL at the reduced sampling levels of the SCL and XCL [Fig. 2.7(d), center and right panels, red areas] against the full-size BBD5x [Fig. 2.77(d), left panel, green area] we observe, however, that the smaller size EBLs still maintain a significant edge against the full size BBD5x. This advantage is likely in part due to the fact that with the EBL some representation of these rare areas can still be maintained while sampling is gradually reduced, while with a transitional rotamer library entire rotameric regions need to be completely removed.

Recovery of crystal structure conformation

After establishing that the energy-based library performs well in total protein repacks from an energetic stand point, we investigated if the performance translated to improved prediction of side chain conformation. Figure 2.8 shows the recovery of the side chain crystallographic conformations in the 560 total repacks (buried positions only, $\chi1+\chi2$, with a tolerance threshold of 40°). In the conditions tested the EBL recovers on average nearly 80% of all side chain conformations, ranging from about 55% (*Ser*) to 90% (*Phe, Tyr, and Val*). In all three comparisons, the EBL performs better than the relative benchmark, by 18% against the BBD5x library, by 16% against the SCL, and by a substantial 118% margin against the smaller XCL. The use of sampling levels (EBL-lev) resulted in a slight improvement of the recoveries compared with "equal sampling" (EBL-eq). Comparing the performance of the EBL in the three trials, it is remarkable that the total $\chi1+\chi2$ recovery is already high at the lowest sampling levels (78.8% recovery in the test against

the XCL) and does not further grow substantially (79.4% against the SCL and 80.3% against the BBD5x). In comparison, the energies significantly improved at every increase of sampling size (Fig. 2.7). This is an interesting finding that suggests that the lowest sampling levels could be effective in side chain optimization, particularly if used with a softened van der Waals function. The R.M.S.D. analysis of the repacked structures compared to the native structures provides further confirmation that the EBL achieves superior performance in side chain prediction. The R.M.S.D. obtained with the EBL is significantly lower than that observed with the benchmarks (1.55 Å to 1.85 Å , Table 2.1) even at the lowest sampling level (1.38 Å). The data relative to $\chi 1+\chi 2$ recovery of all side chains (independently of burial) is shown in Figure A.4. As expected the average recovery drops significantly compared with the buried positions (66% for all three tests) but the overall trend remains similar.

Figure 2.10 examines the hydrogen bond recovery in the protein repacks. The figure reports the fraction of the hydrogen bonds present in the original structure that were correctly predicted. In the three tests, the energy-based library recovers between 47% and 60% of the crystallographic hydrogen bonds. Unlike the $\chi 1+\chi 2$ recoveries, here we observed an improvement as sampling increases (47.5%, 52.5%, and 60.0% against the XCL, SCL, and BBD5x, respectively). It should be noted that while not all hydrogen bonds are correctly predicted, the total number of hydrogen bonds in the repacks exceeds that of the crystal structures (Figure 2.11), which is likely a consequence of the absence of solvent (implicit or explicit) in the calculations. Once again, the new library's performance demonstrated to be outstanding. Compared with the benchmarks, the total recovery was higher by 112% (BBD5x), 115% (SCL), and 125% (XCL), an improvement that is even more marked than what is observed for the $\chi 1+\chi 2$ recoveries. Although in all three tests the total recovery is very similar for the "equal sampling" and "sampling levels" strategies, at the level of the individual amino acids there

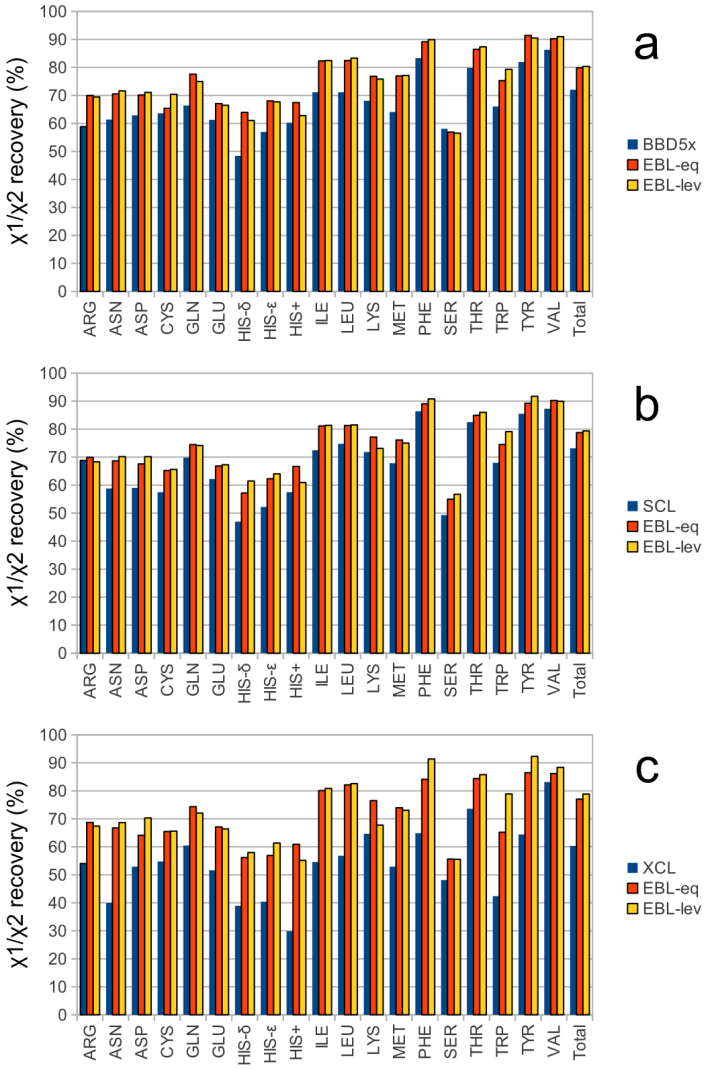


Figure 2.8: *Recovery of the crystallographic side chain conformation in total protein repacks. Recoveries obtained with the EBL are compared to (a) the BBD5x, (b) the SCL, and (c) the XCL. The data is expressed as $\chi 1+\chi 2$ recovery with a tolerance of $\pm 40^{\circ}$ for buried side chains (<25% SASA). The orange bar represents the recovery in repacks made with an equal (eq) number of conformers compared with the benchmark for each amino acid type. The red bar represents the recovery in side chain optimizations made using an EBL sampling level (lev) of similar complexity with respect to the benchmark. With very few exceptions, the EBL performs better than the benchmark, often significantly. In Figure A.5 the data is dissected by $\chi 1$, $\chi 1+\chi 2$,$\chi 1+\chi 2+\chi 3$, $\chi 1+\chi 2+\chi 3+\chi 4$ recoveries. The data relative to all positions, independently of burial, is shown in Figure 2.9.*

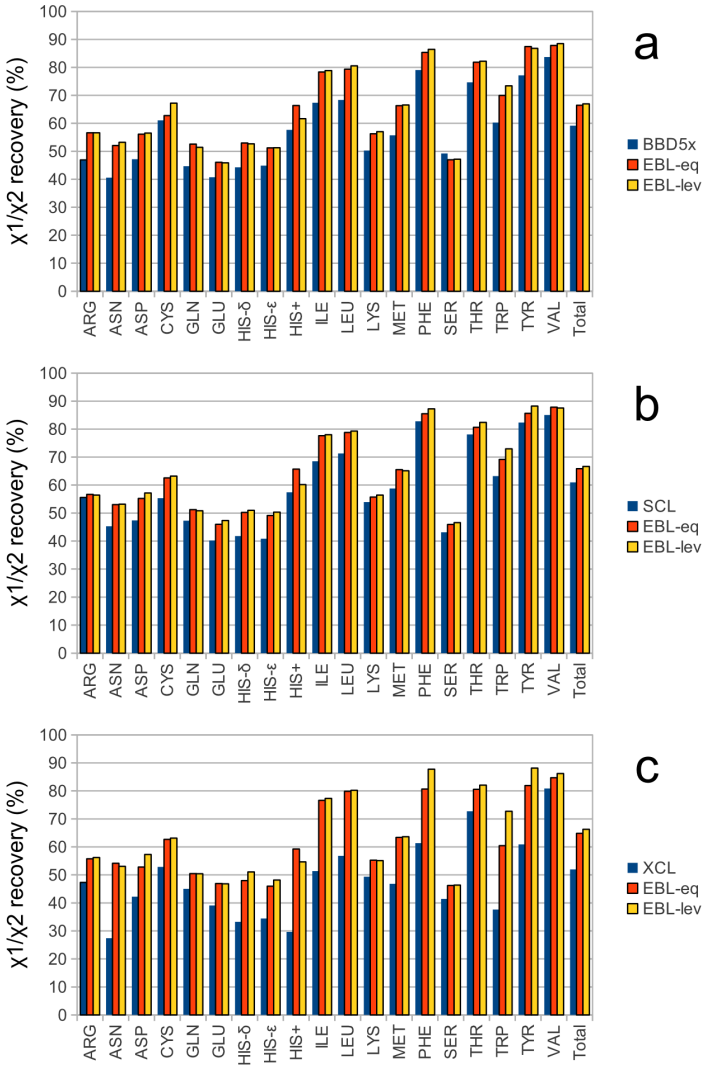


Figure 2.9: *Recovery of the crystallographic side chain conformation in total protein repacks for all side chains independently of solvent exposure. Recoveries obtained with the EBL are compared to a) the BBD5x, b) the SCL and c) the XCL. The data is expressed as $\chi 1+\chi 2$ recovery with a tolerance of $\pm 40°$. The orange bar represents the recovery in repacks made with an equal (eq) number of conformers compared to the benchmark for each amino acid type. The red bar represents the recovery in side chain optimizations made using an EBL sampling level (lev) of similar complexity with respect to the benchmark.*

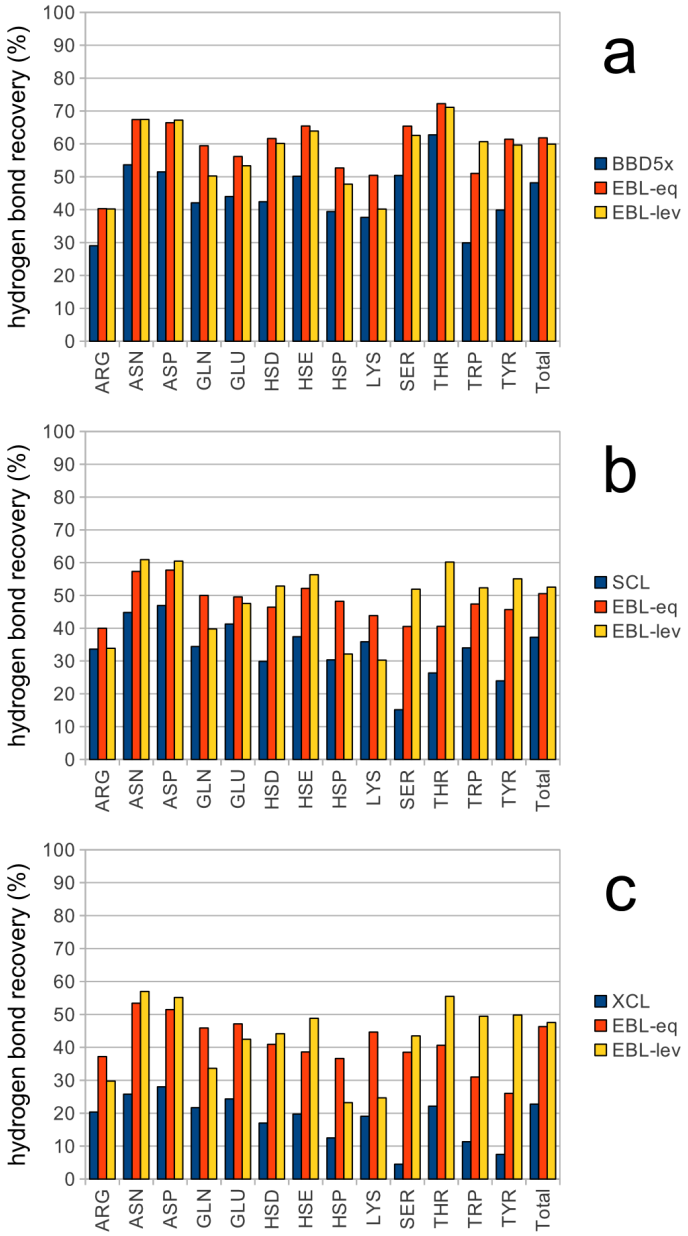


Figure 2.10: *Recovery of the crystallographic hydrogen bonds in total protein repacks. Recoveries obtained with the EBL are compared to (a) the BBD5x, (b) the SCL, and (c) the XCL. The data indicates if a hydrogen bond present in the original crystal structure is recovered after side chain optimization. Any hydrogen bonds that are observed in the repacked structure but are not present in the original structure were not considered. The EBL demonstrated significantly better recoveries in all three cases.*

were noticeable differences.

Experiment	Benchmark	EBL, equal sampling	EBL, sampling levels
BBD5x	1.63 ± 0.36 Å	1.38 ± 0.35 Å	1.35 ± 0.33 Å
SCL	1.55 ± 0.32 Å	1.41 ± 0.36 Å	1.37 ± 0.33 Å
XCL	1.85 ± 0.34 Å	1.48 ± 0.34 Å	1.38 ± 0.33 Å

Table 2.1: *Average Root Mean Square Deviation of Total Repacks Compared with the Native Crystal Structure. The table reports the average RMSDs (Å standard deviation) between the predicted side chains of each protein and the native crystal structure. The set includes only buried side chains (<25% SASA) and all atoms (heavy and hydrogen atoms).*

Overall the three measures - energy, side chain conformation recovery, and hydrogen bond recovery, depict an extremely favorable portrait of the energy-based library. The library displays superior performance in side chain optimization across a range of sampling levels. This efficiency, combined with its unique flexibility in rescaling sampling, means that the library can be a powerful and versatile tool that can be tailored precisely to improve quality and/or decrease run time in side chain optimization procedure.

Analysis of sampling levels

The three sets of total protein repacks demonstrated that increase in sampling produces substantial decrease of the energy. The 14 sampling levels proposed here, from SL60 to SL99, vary in the total number of conformers by a factor of 25 (Table A.3). To gain more precise information on their relative efficiency, we performed a series of total repacks systematically at each individual level on a random subset of 40 proteins. Figure 2.12 summarizes the results of this trial. In the figure the highest level (SL99) is chosen as the reference. The histograms shows the number of proteins that reached an energy within a threshold of 2, 10, or 20 kcal mol^{-1} from the energy obtained with the SL99. The data indicates that to obtain approximately a 50% chance that a

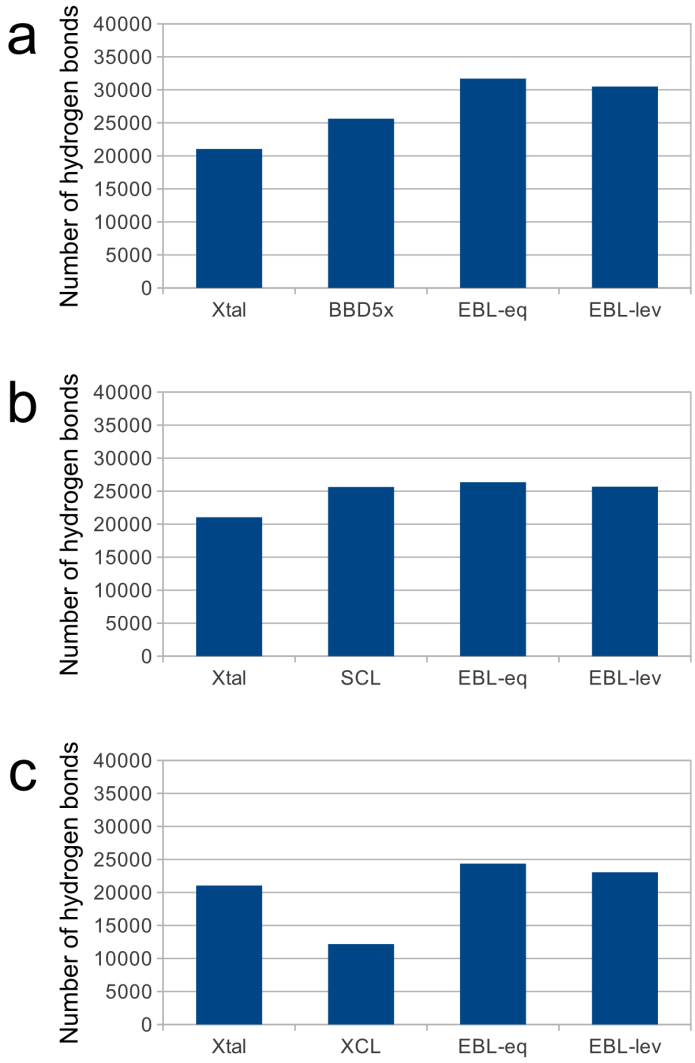


Figure 2.11: *Total number of hydrogen bonds in the native crystal structure and in predicted structures. The histogram reports the total number of side chain-to-side chain and side chain-to-backbone hydrogen bonds in the crystal structure (Xtal), in structures predicted with one of the benchmark libraries (a: BB5x, b: SCL, c: XCL), and in structures predicted using the EBL with either sampling equal to the benchmark library (EBL-eq), or with sampling provided by EBL sampling levels of comparable complexity (EBL-lev).*

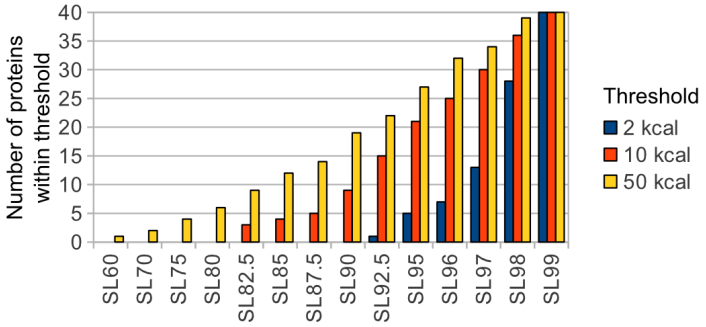


Figure 2.12: *Comparison of the energetic performance of the EBL sampling levels. Forty proteins were repacked at each of the 14 proposed levels (from SL60 to SL99). The figure shows the number of proteins that had an energy below the energy of its SL99 optimization plus a threshold of 2, 10, or 50 kcal/mol. The results demonstrate a gradual increase in performance between the levels.*

protein energy is within 50 kcal mol^{-1} of its SL99 energy, one should adopt at least the SL90 level (1795 conformers). At the SL95 (2910 conformers) about half of the proteins are within 10 kcal mol^{-1} from the best level. To obtain the same proportion below the 2 kcal mol^{-1} threshold the levels required are the SL97 (3938) or the SL98 (4786 conformers). The most important observation, however, is that the levels display a continuum and relatively smooth increase in performance. Although it is likely that an intensive analysis of performance based on total protein repack data would lead to the creation of even more effective levels, the data demonstrates that the proposed levels based on single side chain repacks are a suitable option.

An energy-based rotamer library

The method used for the creation of the energy-based conformer library can also be applied for the generation of a dihedral-only rotamer library. To create this library we standardized the bond lengths and angles of the extremely fine grained library, and followed the same procedure (RMSD

filtering, rebuilding in protein environments, energy-based sorting). Figure 2.4 shows a comparison of the performance of the Energy-Based Rotamer library (EBRL) against the BBD5x library. Although the performance of the EBRL is significantly decreased with respect to the conformer version, the rotamer version compares favorably against the benchmark. The EBRL scores better energies in 89.4% of the proteins compared to the BBD5x library. The percentage below crystal energy drops from 91.8% to 35.8%, but it remains significantly higher than the benchmark. The average side chain recoveries also drop (Supporting Information Fig. S9) but they still compare favorably against the benchmark. The data indicates that the conformers approach is substantially more effective than the use of rotamers. Nevertheless, the energy-based Rotamer library can be a useful alternative for applications that could benefit from the efficient and flexible sampling offered by an energy sorted library but require the use of a dihedral-only rotamer representation.

Performance of the energy-based rotamer library (EBRL) in total protein repacks. (a) The scatter plots and (b) histograms representation of the final energy after side chains optimization of 560 proteins, for the energy-based rotamer library compared to the BBD53 library. While the performance of the rotameric version of the energy-based library is decreased compared to the conformer library (Fig. 2.7), it represents an efficient alternative for applications that required a dihedral-only representation of the library.

2.5 Conclusions

We have presented a new type of conformer library for protein modeling that introduces a number of innovations in side chain sampling. The library is in essence a sorted fine-grained conformer library. The library is so large that it needs to be trimmed down for most practical purposes, although an application that requires extremely precise positioning and is not of

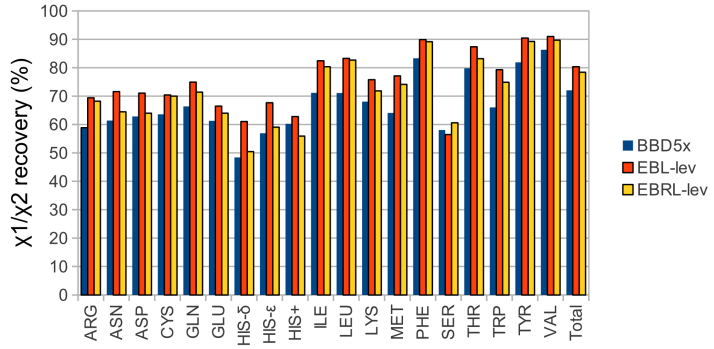


Figure 2.13: *Recovery of the crystallographic side chain conformation in total protein repacks for the Energy-Based Rotamer Library (EBRL). Recoveries obtained with the EBRL are compared to the BBD5x (blue) and the EBL conformer library (red). The data is expressed as χ1+χ2 recovery with a tolerance of ±40°. The EBL and EBRL data represents the recovery in side chain optimizations made using a sampling level (lev) of similar complexity with respect to the benchmark.*

highly combinatorial nature could benefit from its exhaustive sampling. The method for sorting the energy-based library takes into account not only the conformational propensities of the side chains but also the nature of protein environments that host them. The selection of the conformers was made with an energetic criterion, under the hypothesis that using the same metric that selects the "winner" in a side chain optimization procedure would lead to a more efficient distribution of sampling. The results demonstrate that the strategy indeed provides important performance improvements.

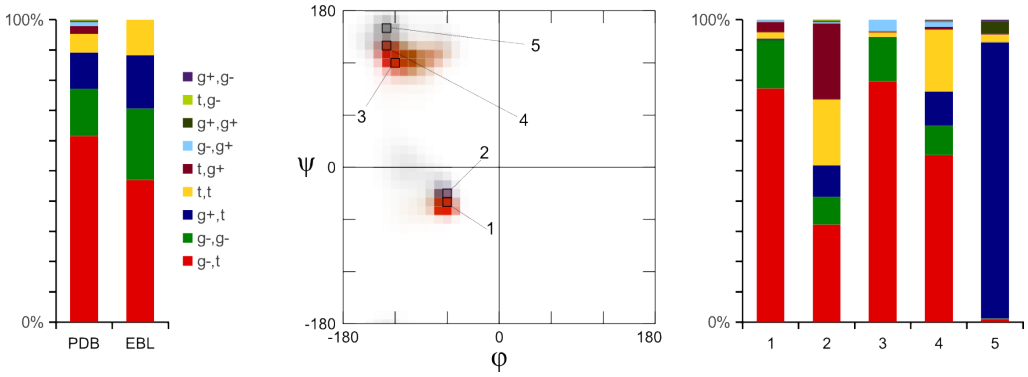
The fact that the library is sorted and can be resized to any desired number represents per se a unique and important new feature. It introduces an unprecedented level of versatility in adjusting conformational sampling to match the specific needs of a modeling procedure. It allows to control the quality of the outcome and to meet any speed or memory requirements. The scalability of the library is also important for balancing the relative amount of sampling given to the different amino acid types and equalize their chances to fit in spaces that should accommodate them. Here we propose a

series of sampling levels that gradually increase the libraryâŁ™s granularity while maintaining the mentioned balance. The versatility of the library can also be an important asset for developing more effective side chain sampling strategies. We have recently shown that that transferring sampling from positions that are likely to be satisfied by a variety of conformers (such as a relatively isolated solvent exposed position) to those that require a conformer from a very narrow and specific range (such as a tightly packed core position) can improve the economy of the calculation and the resulting energies (described in the next chapter [**?**]). The scalability of the EBL enables this and other similar strategies, opening new avenues for further improving performance in side chain optimization.

Compared with the current libraries, the EBL achieves significantly lower energies in side chain optimization. This is certainly a positive finding as it indicates that the EBL is very effective in exploring the energy landscape (Fig. 2.1). For this reason the library could aid the continued development of effective energy functions for protein prediction and reduce the need for artificial softening of the van der Waals function [**??**]. The fact that the EBL was tested with the same energy functions used for its creation may raise a concern that the performance could be in part due to over-training of the libraries to perform well with these specific functions. It is therefore important to note that the native structure recovery parameters testedâŁ"specifically the dihedral prediction, the hydrogen bonding recovery and the RMSD with the native structureâŁ"all improve alongside with the energies, indicating that the library captures well the physical aspects that determine side chain conformation in proteins. More tests will be necessary to understand how performance will be affected when the library is used with different energy functions. Our functions were selected specifically to favor efficient packing, hydrogen bonding and to prevent strains, which are factors that are present in a majority of modeling programs, and thus we are confident that the enhancement will translate well when the library

is used with different functions. While nothing prevents the users from modifying their programs to adopt a set of functions similar to ours, should that be advantageous, we also encourage others to adopt the method to create specific energy-based libraries optimized ad hoc with the energy functions used in their own applications. A tutorial on how to create a library is made available on the EBL web site (http://seneslab.org/EBL) and all software and databases required for building a similar library are freely provided. The energy-based library and the dihedral-only version are distributed in our website. The format of the library is described in Figure A.2. All software used to create the EBL, the modules for reading the library, building conformers from internal coordinates, and for performing side chain optimization are implemented in C++ using the MSL package (http://msl-libraries.org), a suite of molecular modeling tools freely available for download under an open source GPL v.3 license. MSL is described in detail in chapter 7.

3 BACKBONE-DEPENDENT ENERGY-BASED CONFORMER LIBRARIES (B-EBL)



based on

Subramaniam S and Senes A "A backbone-dependent energy-based conformer library improves side chain optimization" (*manuscript in preparation*)

Summary

Protein side chain modeling is an integral component of most protein modeling applications and being computationally intensive, often constitutes the bottleneck stage in these applications. Side chain modeling consists of three widely studied components; an energy function to evaluate candidate structures, a side chain conformer library that presents a discrete side chain conformational space and a search algorithm that explores this combinatorial space. In chapter 2, we developed an energy-based algorithm to extract a side chain conformation library from high resolution protein structures. The resultant backbone-independent energy-based library called the EBL [**?**] outperformed other widely used conformer libraries. This chapter describes a backbone-dependent energy-based conformer library (B-EBL), that combines the energy-based approach with the backbone-dependence of side chain conformations to produce a superior side chain conformation library. This new library is shown to perform better than EBL in terms of the dihedral recovery and the energies achieved on a set of 480 high resolution protein structures. We also show, through single repack experiments, that fewer conformers from B-EBL better represent the side chain conformational space given the local backbone geometry.

3.1 Introduction

Side chain optimization is the process of predicting the 3-dimensional conformation of the side chains given a protein backbone. It is an important and widely studied component of several protein modeling applications such as homology modeling [**?**Nayeem et al. (2006); Wallner and Elofsson (2005); Bordoli et al. (2008); Wang et al. (2008)], structure prediction [Wang et al. (2008); **?**); **?**] , protein design [**?**Street and Mayo (1999); **?**); Samish et al. (2011)], point mutation analysis [**??**], protein -protein and protein-ligand docking [**?**Wang et al. (2005)] and structure refinement [**??**]. In side

chain optimization, the combinatorial space of side chain conformations is searched in order to identify the lowest energy state, which is scored with a variety of physics-based [Brooks et al. (1983); Cornell et al. (1995)] or knowledge-based potential functions [**?**]. Since the conformational space of side chains is extensive but sparsely populated, this space is generally searched by adopting a library of discrete side chain conformations. The library converts a multi-dimensional problem in continuous space into a combinatorial problem in discrete space, facilitating the application of a number of fast, deterministic [**???**] and probabilistic algorithms for determining the global minimum energy conformation [**???**]. In addition, the discretization allows the search algorithms to focus on those areas of side chain conformational space that occur frequently in proteins and discard those that are very rarely encountered.

The discrete libraries used for side chain modeling can be of different derivation, the most common alternatives being rotamer and conformer libraries. The rotamer libraries [**??**Shapovalov and Dunbrack (2011)] are compiled using a statistical analysis of the distribution of the side chain dihedral angles (χ angles) which are the major determinants of side chain conformation. This approach is based on clustering the distribution of side chain conformations observed in the structural database, the Protein Data Bank (PDB). The clusters are reported by indicating their average, standard deviation, and their overall frequency of χ angle combinations. The rotamer libraries generally do not report bond angles and bond lengths, which are only minor determinants of conformation. Therefore, when rotamer libraries are applied to side chain prediction, these variables are generally held fixed at some optimal value.

A second approach is represented by the conformer libraries [**???**] which are compiled by extracting a representative subset of side chain conformations from native structures found in the Protein Data Bank (PDB), chosen so as to represent the most populated regions of conformational space. This

is done by extracting a large number of conformations from the structural database, which are then reduced to a manageable subset by applying a similarity filter, such as angular similarity [**?**] or root mean square deviation (RMSD) [**??**]]. Because the selected conformers are actual side chains found in natural proteins, the conformer libraries retain the native bond lengths and angles variation observed in proteins, which can be beneficial [**?**].

An important issue with both rotamer and conformer libraries is that of controlling the specific granularity of the sampling. Side chain optimization is a combinatorial search that can be computation-intensive and can become a serious bottleneck, particularly if the number of side chains involved is large or if multiple side chain optimizations are required by the application. For example, our group uses side chain optimization in the structure prediction of complexes of transmembrane helices, either ab initio [**?**], or guided by experimentally derived data [**?**]. These methods require extensive exploration of backbone conformational space, and the side chain optimization required at each cycle represents the major bottleneck of the procedure. Therefore, it is important to find the level of sampling level that provides the best compromise between two conflicting requirements: i) minimization of the number of conformations, in order to improve computational efficiency, and ii) maximization of conformational sampling, to achieve the desired level of accuracy [**??**].

To address this challenge, we introduced the energy-based Library (EBL), a conformer library provided as a sorted list that can be truncated precisely at any desired level of granularity [**?**]. While the previously available rotamer and conformer libraries may be produced in different sizes to achieve better performance [**???**], the EBL enables much finer control making it possible to control the size of the library at the level of the individual conformer. As a result of this finer control, the EBL also introduced the concept of sampling levels, a way to control the level of granularity of the library (from low to prioritize speed, to high to prioritize accuracy) while ensuring that

sampling remains balanced between the different amino acid types.

In addition to the customizable granularity, perhaps the most important feature of the EBL is its improved efficiency in side chain optimization. The EBL was constructed under the realization that the adoption of an energetic criterion for the selection of the conformers (i.e., the same metric used in side chain optimization) produces a library with enhanced performance. The EBL was obtained by remodeling a large number of individual positions in known crystal structures with all conformers in a very fine-grained library, and energetically evaluating the interactions between the conformers and the protein environments (Figure 2.4). This large data set was used to sort the conformers by their propensity to fit (energetically) into natural protein environments thereby producing a very efficient backbone-independent library. This new strategy for creating a conformer library resulted in a enhanced product that had improved computational efficiency (EBL requires fewer conformers for the same accuracy) as well as modeling accuracy (EBL obtains lower energies and better dihedral prediction for a similar number of conformers), in comparison to state-of-the-art libraries.

The EBL was initially developed as a backbone independent conformer library because a large dataset of crystal structures was required for its creation. This requirement posed difficulties in obtaining sufficient environments for the method once the available side chain environments were subdivided according to their ϕ/ψ coordinates However, it is well established that side chain conformation is strongly dependent on the local backbone geometry [Chakrabarti and Pal (2001)]. This dependence is not simply based on the secondary structure because small changes in ϕ/ψ angles within the same structural classification (helix, sheet) can produce significant variations in the rotameric distribution [Shapovalov and Dunbrack (2011); Dunbrack and Karplus (1994); **?**]. In addition, local variations of backbone conformation can also influence the average value of side chain χ angles to relax strained interactions between the side chain and the backbone [Dunbrack

(2002)]. Consistently, it has been shown before that a backbone dependent library performs significantly better than a backbone independent library [Dunbrack (2002)].

Here, we present a backbone dependent version of the EBL, which we call B-EBL. We have tested a library that is created in a backbone-dependent fashion for the most populated regions of the ϕ/ψ space, and a single EBL calculated for the sparse regions. we demonstrate that this version achieves significantly better performance than the original EBL. A comparison of performance in side chain prediction operating shows that the B-EBL produces models with lower energy than those produced by the EBL when the two libraries are applied with the same exact number of conformers. Most importantly, we show that a B-EBL of half the size achieves equal or better modeling accuracy than the backbone-independent EBL. The reduction in the number of conformers impacts the computational efficiency drastically because the search space increases combinatorially with the number of conformers for each position, therefore this is an important step forward in performance.

Support for the B-EBL is implemented in the Molecular Software Libraries (MSL) v.1.2, a C++ open source library for molecular modeling, analysis and design.

3.2 Materials and Methods

A dataset of about 2100 high-resolution protein structures with a low sequence identity was obtained from the Protein Data Bank (PDB) [**?**] and minimized using the CHARMM17 program as described in chapter 2. A set of 480 proteins was selected for full repack tests (described below).

Backbone-dependent energy-based conformer library (B-EBL)

The backbone-dihedral space was divided into 10° x 10° partitions, which has been observed to be the most effective subdivision and a B-EBL was created for each partition by reordering the EBL from [**?**]. The subdivision of the ϕ/ψ space for each amino acid was done such that each partition contained a minimum of 100 side chains. All the remaining sparsely populated partitions were combined into one partition. This was necessary since the EBL algorithm requires a good number of sample side chains in order to effectively sample the energy landscape.

The backbone-dependent energy-based library or B-EBL, for each backbone partition, was created by reordering the EBL as described in chapter 2. This process led to the re-sorting of the EBL for each ϕ/ψ partition and created a collection of backbone-dependent energy-based conformer libraries that are better suited for each local backbone geometry. The energy function used was an implementation of the CHARMM22 [Brooks et al. (1983)] energy field along with the hydrogen bonding term from the SCWRL [Krivov et al. (2009)] program. The electrostatic energy term was not used based on analysis described in chapter 2. All molecular modeling and analysis was performed using our in-house C++ molecular modeling software library, the MSL [Kulp et al. (2012)].

Single repacks

Single side chain repack test was performed using the B-EBL and EBL on side chains from the protein dataset. The side chain atoms from native side chains in the dataset were removed to create side chain "environment". Conformers from the B-EBL and EBL were modeled into each of these environments and the interaction energy between the side chain and the rest of the protein was measured for each conformer-environment pair. Conformers were defined to satisfactorily model or predict a test side chain by an energy-based threshold described in chapter 2. The number of conformers, in sorted order, together

required to satisfy the side chains (across all ϕ/ψ partitions) in the dataset was measured using both EBL and B-EBL.

Sampling levels

The flexibility in sampling introduced by the EBL enabled the creation of sampling levels [**?**] which are a means to balance conformational sampling across amino acid types. The number of conformers, for each amino acid, that can together predict a desired fraction of side chains from the dataset constitutes a sampling level. This definition is extended to include each ϕ/ψ partition in the backbone-dependent EBL. Each sampling level specifies the number of conformers to be used for each residue (in each ϕ/ψ partition) such that comparable modeling accuracy is achievable across all residue types. For example, in the (ϕ=-60,ψ=-40) partition, eight Leu conformers constitute the 85% sampling level since they can predict 85% of the Leu side chains in the dataset. Similarly, several sampling levels are defined for all amino acids in all the ϕ/ψ partitions.

Complete repacks or full-protein repacks

Side chain optimization was performed on the set of 480 proteins selected for this purpose. All side chains except Ala, Pro and Gly were removed and predicted using both EBL and B-EBL. His residues were predicted according to the protonation state predicted by Reduce [**?**]. The *repackSideChains* program in MSL [Kulp et al. (2012)] was used for optimization using the EBL [**?**]. For optimization using the B-EBL, an ad hoc program was created which determined the appropriate B-EBL library to be used for each position based on the backbone dihedral angles, ϕ and ψ. The B-EBL was restricted to use the same or lesser number of rotamers than the corresponding EBL.

Side chain optimization was performed using a greedy algorithm that started with an initial conformation for all positions and optimized one

position at a time, while all the other residues were held fixed. It simply scanned all the conformers for each position, and picked the conformation with the lowest energy. This procedure was repeated for all residues until no new conformers were picked for any residue. This procedure was repeated ten times for each protein backbone with different initial rotamer assignments.

Dihedral recovery

To measure the accuracy of modeling, side chain dihedral angles in the modeled structures were compared with the corresponding native structures in the test dataset. The *getChiRecovery* program in MSL [Kulp et al. (2012)] was used to measure the dihedral recovery. A side chain was considered recovered if all its side chain dihedral (chi-χ) angles were within 40° of the values in the native structure.

Two tests were performed, a) B-EBL was constrained to use the same number of conformers as the EBL at different sampling levels and b) B-EBL and EBL used their respective sampling levels of comparable sizes (number of conformers), with the B-EBL always smaller than the corresponding EBL. Side chains in the protein core were determined using a threshold based on the solvent accessible surface area and results on these "core" side chains are reported.

3.3 Results and Discussion

Library creation procedure

Other than the fact that the protein environments were subdivided based on their backbone dihedral angles (ϕ/ψ), which will be detailed later, the procedure followed for the creation and testing of the B-EBL is identical to the original procedure1, as schematically summarized in Figure 2.4. The base was the same very fine conformer library used for the creation of the EBL,

which was sorted independently for each ϕ/ψ partition to produce backbone dependent libraries. As for the EBL, the conformers were reconstructed within fixed protein environments that belong to each ϕ/ψ partition and interaction energies between the side chain and the protein were measured (Figure 2.4a). If an interaction energy was below a certain energy threshold, the environment was deemed to be satisfied by the conformer (green thick marks in Figure 2.4b). The interaction data was used to sort the conformers as follows: the conformer that satisfied the most environments was added to the sorted library, and all the environments that were satisfied by the conformer were removed and no longer considered in the selection of the next conformer. This exclusion ensures that a sequence of conformers that complement each other was selected. However, at every cycle the energy threshold that determined if an environment is satisfied was increased, and thus previously removed environments that resulted no longer satisfied at the new more stringent threshold could be considered again. This mechanism ensured that the environments could return into play and were not rapidly consumed by the procedure. As discussed later, the conformer/environment energy data is also used to create a series of "sampling levels". The sampling levels are a tool for truncating the library at a desired level of granularity while maintaining balanced sampling across amino acid types. The sampling levels ensure that every amino acid type has a similar chance to fit within an environment that should in principle accommodate it.

Partitioning the protein backbone space

The major challenge for the creation of a backbone-dependent energy-based library was the identification of an effective strategy for subdividing the ramachandran (ϕ/ψ) space. The subdivision should result in sufficient data for each partition while at the same time capturing the natural variation of rotamer propensity across the backbone space. The partitioning may be secondary structure-based [**?**], or based on even dihedral angle intervals, for

example, by dividing the ϕ/ψ space into 10°x10° bins [Shapovalov and Dunbrack (2011)]. While the secondary structure-based representation results in large well populated partitions, the explicit 10°x10° scheme results in a small number of partitions that contains a sufficient number of environments for the EBL algorithm. Nevertheless, because the Ramachandran distribution is highly uneven, with relatively small areas of very high density and large areas that are very sparsely populated, a 10°x10° partitioning scheme can still result in a backbone dependent library that covers a large fraction of the entire population.

To avoid over-fitting the training data during the sorting procedure, we estimated that each 10°x10° partition should contain a minimum of 100 environments. Therefore all 10°x10° partitions that contained at least 100 examples for each amino acid were collected for the application of EBL. All the other 10°x10° partitions were clustered into one partition leading to a number of partitions for each residue that ranged from 27 (Val) to 4 (Cys). The EBL algorithm was applied to each partition thus created and a backbone-dependent EBL was created. Single repack tests were performed using the two libraries created. we observed that the explicit 10°x10° partitioning slightly outperformed the secondary-structure based method. This led us to adopt the 10°x10° partitioning strategy.

An interesting feature of the Energy-Based Library is that its composition agrees with the overall natural bias of side chain conformation. This is well illustrated in a side-by-side comparison of the rotamer probability in the PDB [Shapovalov and Dunbrack (2011)] and the composition of the EBL (at the 85% sampling level), shown in Figure 3.1a for Leu and in Figure 3.1d for Ile. This match is not observed in other commonly used libraries [**?**]. This characteristic is a consequence of how the library is constructed. It is also an indication that sampling side chain conformational space proportional to the population of each rotameric region leads to an efficient library. However, that rotamer preferences vary drastically with

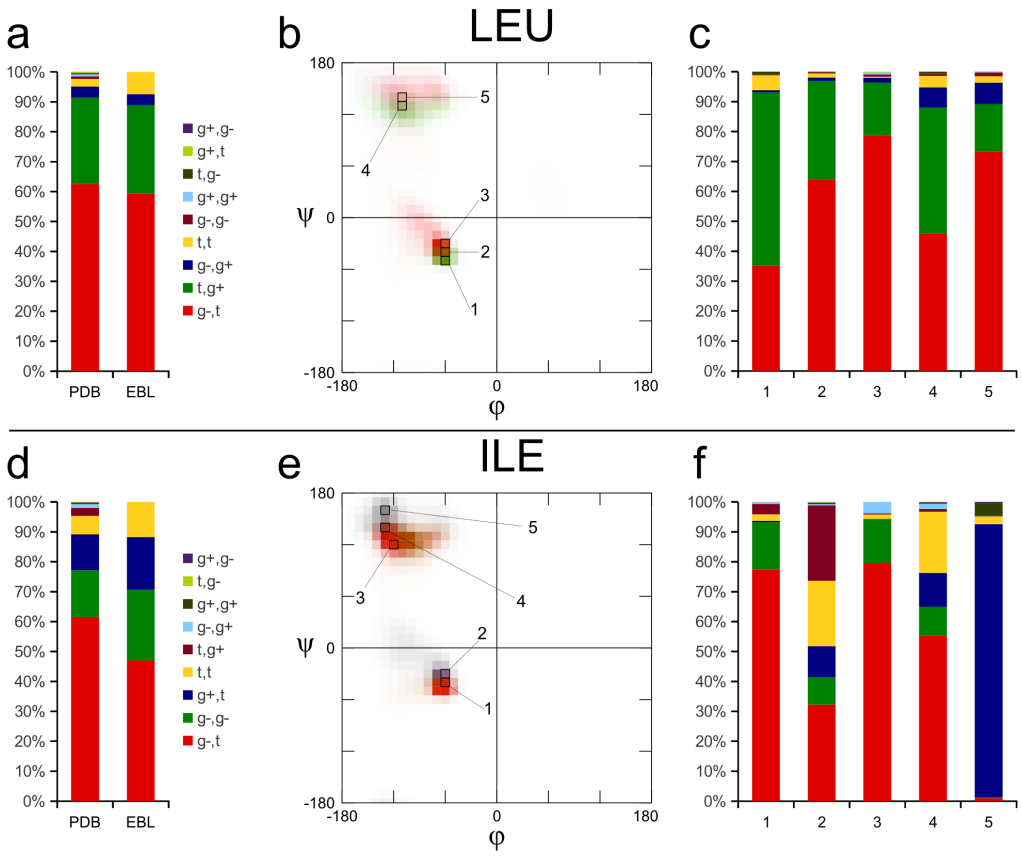


Figure 3.1: *Significant backbone-dependent variations in rotamer distribution. (a) The stacked graph shows the overall rotamer distribution for Leu sidechains in the PDB and the SL85.00 sampling level of EBL (one sampling level shown for simplicity). The similarity in the stacks shows that the overall distribution is well captured by the backbone-independent EBL. (b) Five distinct 10°x10° bins, labeled 1-5, are selected from the backbone dihedral space for Leu and are shown on a color and gradient-coded density plot. The color is a proportional mix of the colors for each of the rotamers and the gradient codes for the relative population in the bin. (c) The rotamer distribution for each of the five regions is shown as a stacked graph. The (g-,t) rotamer is predominant in regions 2,3 & 5, whereas the (t,g+) rotamer dominates in region 1 & 4 . Clearly, there is a significant change in rotamer distribution across the backbone dihedral space. (d,e and f) The corresponding data for the Ile sidechain also illustrates significant variations in rotamer distribution.*

local backbone geometry [Shapovalov and Dunbrack (2011); **?**] and that rotamer libraries that capture this variation are more effective in side-chain sampling [Dunbrack (2002)].

For Leu the most frequent rotamers are [g-,t] (62%) and [t, g+] (30%) (Figure 3.1a). However, the relative frequency of these two rotamers, as well as of the other minor rotamers, varies widely across the ϕ/ψ space. This is illustrated in a color coded representation in Figure 3.1c, and in detail for selected 10°x10° bins in the bar graphs of Figure 3.1c. A similar situation is even more noticeable for Ile (Figure 3.1 d-f). An extreme example is the rotameric probability of the Ile -130°/160° bin, which is marked as number 5 in Figure 3.1e-f. In this bin, the [g+, t] rotamer, which represents just 10% of the total in the overall backbone-independent distribution, becomes dominant over 90%.

An analysis of the most dominant leucine rotamers in each region of the ϕ/ψ space illustrates the variability in rotamer preferences with small changes in backbone geometry. On a set of high quality protein structures from the Protein Data Bank (PDB) the dominant leucine rotamer for each ϕ/ψ partition was determined and is shown in Figure 2.4. The dominant rotamer for the helical region (in Figure 2.4) is the (g-,t) rotamer, shown in red and for the beta sheet region is the (t,g+) rotamer, shown in blue. However, even within each secondary structure region, the rotamer preference (and/or distribution) changes with slight changes to the backbone dihedral angles. This can be observed by the blue patches in the helical region and red patches in the beta sheet region of ϕ/ψ space (Figure 2.4). This analysis showed that a purely secondary structure based subdivision of ϕ/ψ space is probably less informative than an explicit dihedral value-based partition. The analysis also suggests that a carefully compiled backbone-dependant EBL for each ϕ/ψ partition could be dramatically different and probably more effective than the EBL which does not explicitly consider backbone dependence.

B-EBL performs better in single repack tests

The relative efficiency of the library is demonstrated by two experiments. First, the B-EBL is tested in a series of single side chain repacks where a single side chain is predicted with all the other side chains held fixed in their native conformations. The B-EBL is then subjected to the full-protein repacks, in which all side chains are simultaneously optimized. Performance is quantified with two metrics, energy and correct prediction of crystallographic side chains.

A set of side chain "environments" are selected from native protein structures by removing the side chains and are binned into their corresponding ϕ/ψ partition. The conformers from the corresponding B-EBL partition are used to model the side chains within these environments one at a time and the cumulative number of environments "satisfied" by the addition of each conformer is determined using an energy-based threshold. This experiment is referred to as the "single repack test" and the number of conformers that can together predict different fractions (60%, 70%, 75%, 80%, 85%, 90%, 92.5%, 95% and 99%) of the test dataset was measured. Single repack performance is a good indicator of performance in typical side chain optimization procedures as we observed in [**?**].

Results in Figure 3.2 show the single repack performance of B-EBL and EBL for the leucine and isoleucine side chains. The number of conformers used is shown along the x-axis and the fraction of all test side chains that were satisfactorily predicted using these conformers is shown along the y-axis. For both leucine and isoleucine, the B-EBL satisfies more side chains using fewer conformers; to model 90% of all leucine side chains, B-EBL required only its first 29 rotamers whereas EBL required 60 conformers. Similarly, for isoleucine, the B-EBL required only 12 conformers whereas the EBL required 28 conformers. This improvement indicated that the B-EBL may have greater efficiency than EBL in side chain optimization.

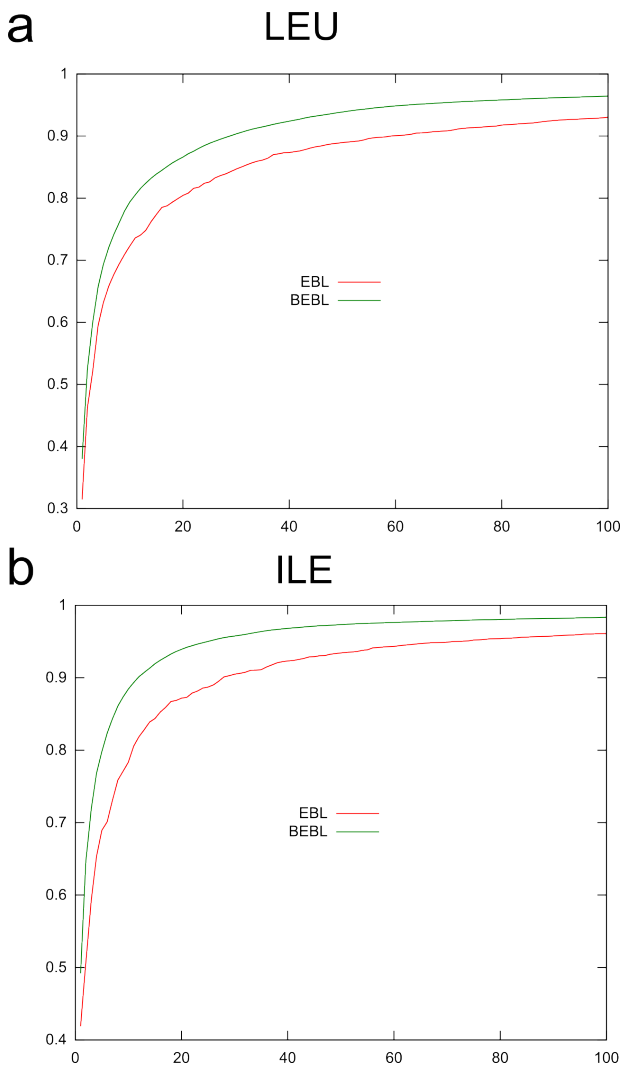


Figure 3.2: *BEBL shows improved performance in single repack tests. Side chains were remodeled in test environments using conformers from the EBL and BEBL. The fraction of environments satisfied by the first N conformers (1 to 100) of the library is reported on the y axis. a) BEBL covers more side chains than the EBL as each conformer is added. 29 BEBL conformers satisfied 90% of Leu side chains whereas 60 EBL conformers were required. b) 90% of Ile side chains were satisfied by 12 BEBL conformers whereas 28 EBL conformers were required. The BEBL requires only half the number of conformers at each position to achieve the same performance as the EBL.*

Customizable granularity of the side chain library using "sampling levels"

Each residue type is unique with respect to its geometry and has different sampling requirements. For example, valine which has a small side chain requires fewer conformers compared to a larger side chain like arginine; therefore, it may be useful to employ a different number of conformations for each amino acid. In the original EBL we used the data from single repack tests to create a set of "sampling levels" of increased granuarity [**?**]. At least in principle, these levels balance the number of conformers across all residue types so that sampling increases homogeneously to provide the different amino acids with the same chance to be correctly predicted in side chain optimization. The number of conformers that satisfy the same fraction of side chains for each amino acid in the single repack test constitutes a sampling level. For example, the EBL sampling level SL85.00 (covers 85% of test sidechains) contains 149 arginine conformers and 7 valine conformers. These sampling levels may be used as guides to balance sampling across amino acids for given computational constraints.

As in EBL, sampling levels were determined for B-EBL, by computing the number of conformers required to achieve single repack coverage. For example, to satisfy 80% of the leucine side chains in the dataset, B-EBL requires its first 11 conformers (across all bins) whereas EBL required 20 conformers. To achieve 95% coverage, the B-EBL requires only 63 conformers while EBL required 188 conformers. Figure 3.3 compares the number of conformers at each sampling level for the EBL and B-EBL and shows that the B-EBL requires fewer conformers for the same single repack performance at any desired sampling level. This result suggests that a far smaller B-EBL might achieve the same performance as a larger EBL in side chain optimization.

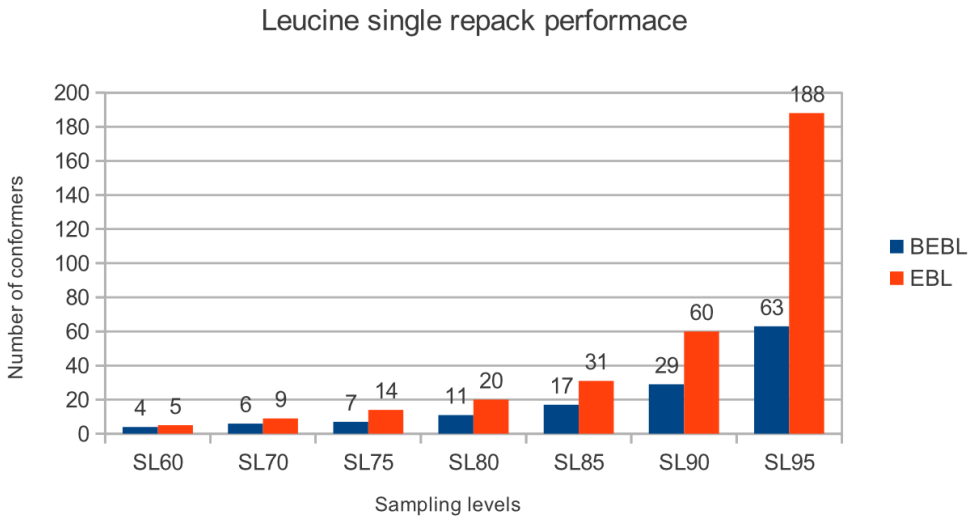


Figure 3.3: *BEBL requires fewer conformers for the same single repack performance. The number of EBL and BEBL conformers required to recover a certain fraction of Leu sidechains was measured (referred to as "sampling levels"). This figure shows that BEBL requires fewer LEU conformers than EBL at any given level.*

EBL algorithm captures backbone-dependence

In this section we describe a simple experiment that analyzes the rotameric distribution of leucine conformations in the B-EBL. The first ten conformers from each partition in the B-EBL were extracted and their rotameric distribution was computed, these distributions were in agreement with the backbone-dependent rotamer library [Shapovalov and Dunbrack (2011)] (Figure 3.1). Leucine side chains contain two dihedral angles referred to as $\chi 1$ and $\chi 2$. In crystal structures, the values of $\chi 1$ and $\chi 2$ are observed to cluster around nine energetic minima corresponding to all the combinations of $\chi 1$ = -60 or gauche-(g-), 180 or trans(t), +60 or gauche+(g+) and $\chi 2$=g-, t, g+, called rotamers. The existence of these rotamers has been attributed to sterical reasons [Chakrabarti and Pal (2001)]. Of these nine rotamers,

two are the most frequent 1) $\chi 1$= t and $\chi 2$ = g+ and 2) $\chi 1$= g- and $\chi 2$ = t. However, distribution of these rotamers is dependent on the backbone dihedral angles ϕ and ψ. For instance, according to [Shapovalov and Dunbrack (2011)], in the partition (ϕ = -60, ψ = -50) the (t,g+) rotamer is more probable than the (g-,t) rotamer whereas in the partition (ϕ =-60, ϕ = -40) the (g-,t) rotamer is more probable than the (t,g+) rotamer. Similarly, there are regions that are extremely biased towards one of the rotamers like the partition (ϕ= -70, ϕ = -20) which is highly biased towards the (g-,t) rotamer. Since the backbone and side chain geometries are so interrelated it becomes necessary to validate that the B-EBL preserves (or captures) these dependences.

The B-EBL contains 20 partitions for leucine, with the last partition being a collection of all the sparsely populated 10°x10° regions in ϕ/ψ space. Figure 3.1 compares the rotameric distribution of the first ten leucine sidechains in all the ϕ/ψ partitions with the probabilities specified by the backbone-dependent rotamer library [Shapovalov and Dunbrack (2011). These results show that the ranking produced by the EBL algorithm in each of the ϕ/ψ partitions arranges conformers in proportion to the underlying rotameric distribution specified in the backbone-dependent rotamer library [**?**]. This agreement between the two distributions illustrates the efficacy of the EBL algorithm to correctly represent the backbone-dependent rotameric distribution. In contrast to the backbone-dependent libraries that present a statistical view of the side chain dihedral space, the B-EBL specifies complete side chain conformations that will be most efficient for a side chain optimization procedure.

B-EBL leads to models with lower energies

Side chain modeling is a search for the lowest energy structure, therefore the energy obtained is an important parameter in estimating the efficiency of a modeling protocol. The “complete repack test” refers to the procedure where

all the side chains in a protein are modeled via a search over the combinatorial space of conformations specified by a side chain library. Complete repacks were performed on 480 proteins from the curated dataset using both the EBL and the B-EBL and the best energies obtained for each protein is compared as shown in Fig 3.4.

The repacks were performed using the same number of conformers from the EBL and the B-EBL; the number of conformers used were the "sampling levels" determined for the EBL. The number of proteins where one library achieves a better energy than the other is shown in Figure 3.4a). The B-EBL achieves lower energies for more proteins than the EBL at all sampling levels. This improved performance is more pronounced in the middle sampling levels and tapers off towards the extremes. This is because at very high sampling levels, both the B-EBL and EBL contain enough conformers to represent the side chain space and the difference in performance is less pronounced. However, the number of conformers at these levels is so high that they may not be suitable for the side chain optimization of large proteins. At very low sampling levels, there are not enough conformers in both EBL and B-EBL to achieve good side chain optimization. Thus, the B-EBL achieves improved performance for those sampling levels that are most likely to be used in a side chain optimization procedure. Panels 3.4b) and c) visualize the results slightly differently for the SL85 sampling level of EBL. Figure 3.4b) shows the normalized energies, obtained by subtracting the energy of the crystal structure from the energy of the repacked structure, as a scatter plot with EBL energies along the y-axis and B-EBL energies along x-axis. In about 75% of the test proteins, the B-EBL achieves lower energies than the B-EBL i.e) 363 out of 480 points lie to the left of the diagonal in Figure 3.4b). Figure 3.4c) is obtained by binning the proteins by their normalized energy and shows that the B-EBL bins are shifted towards lower energies than EBL. These experiments demonstrate that when the same number of conformers are employed B-EBL leads to lower energies than the EBL implying a better

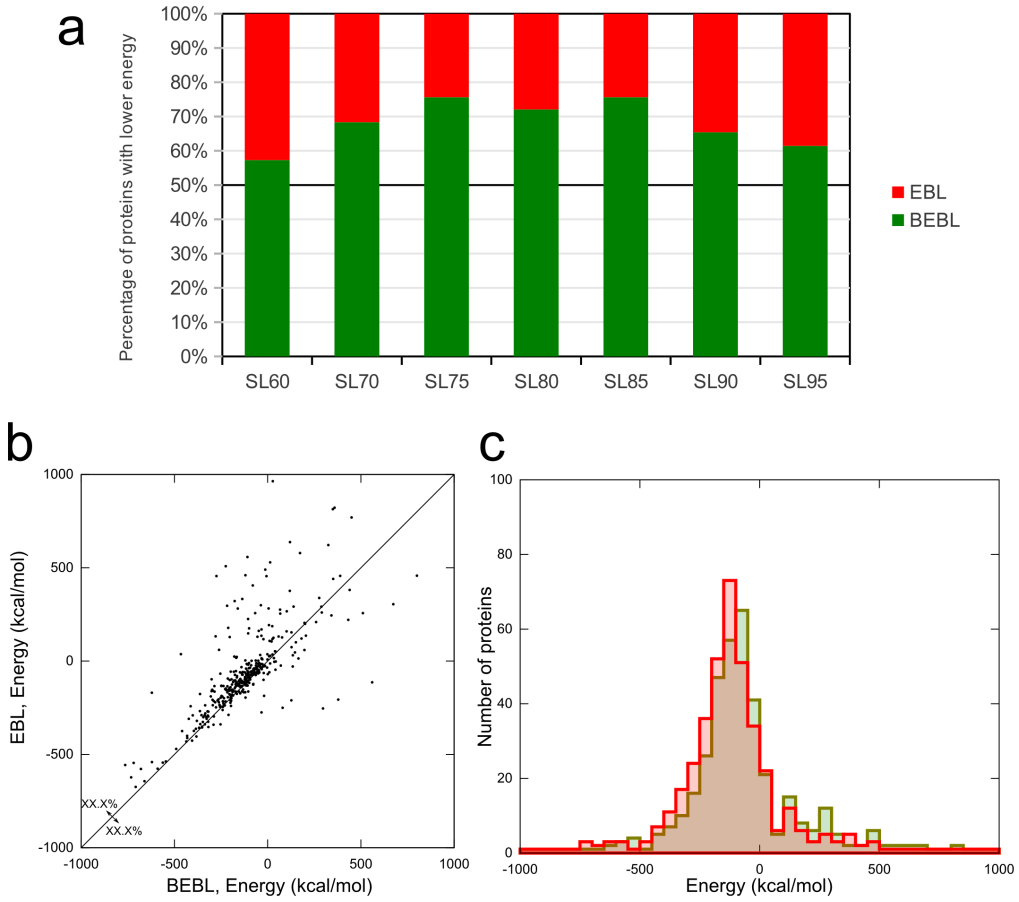


Figure 3.4: *BEBL achieves better energies for the same number of conformers. Sidechain optimization was performed on a set of 480 test proteins using the same number of conformers from the EBL and BEBL. This figure shows the number of proteins where each library achieved better energies than the other. The BEBL performs significantly better than EBL at levels SL70 through SL90. At the largest (SL95) and smallest (SL60) levels, BEBL is still better than EBL but the improvement is less significant. In panel b) is a plot of the energies, after subtracting the energy of the crystal structure, obtained from sidechain optimization at SL85 using BEBL and EBL (along the x- and y-axis respectively). The higher density of sample points to the left of the diagonal shows lower energies obtained by the BEBL. Panel c) shows the data in b) in 10 kcal bins; it shows how the BEBL frequently achieves lower energies than the EBL.*

sampling of conformational space.

B-EBL achieves more accurate structure prediction

The accuracy of protein modeling is typically measured in terms of the number of side chain dihedral angles predicted within 40° of the native structure. In this section, we describe two experiments to compare the dihedral recovery of B-EBL and EBL, both performed at "sampling levels" determined earlier. In the first experiment, complete repacks were performed at the "sampling levels" determined for the EBL and B-EBL. The number of conformers in the B-EBL is far fewer than EBL at any given "sampling level" as described earlier in this section, therefore the size of the search space explored using the two libraries varied significantly. The actual size of the combinatorial space explored for the EBL and the B-EBL was measured for each protein and the average size for each sampling level was determined. Figure 3.7 shows that at any desired sampling level, the B-EBL presents a far smaller search space compared to the EBL. However, despite this apparent disadvantage, the B-EBL achieves similar modeling accuracy as the EBL as shown in Figure 3.5. The average dihedral recovery across all amino acids in the test dataset of 480 proteins is determined using both B-EBL and EBL and a comparison at several sampling levels is shown in Figure 3.5. The fact that the average dihedral recovery for the EBL and B-EBL does not differ by more than two percent demonstrates that the B-EBL achieves similar modeling accuracy as EBL at a lower computational cost i.e) by using fewer conformers.

The second experiment compares the dihedral recovery of EBL and B-EBL of comparable sizes; B-EBL sampling levels were selected such that the search space was as close as possible but never larger than that of the EBL sampling level being compared. Complete repacks were performed at the selected sampling levels and the average dihedral recovery was determined as before. Figure 3.6 compares the average dihedral recovery of EBL and

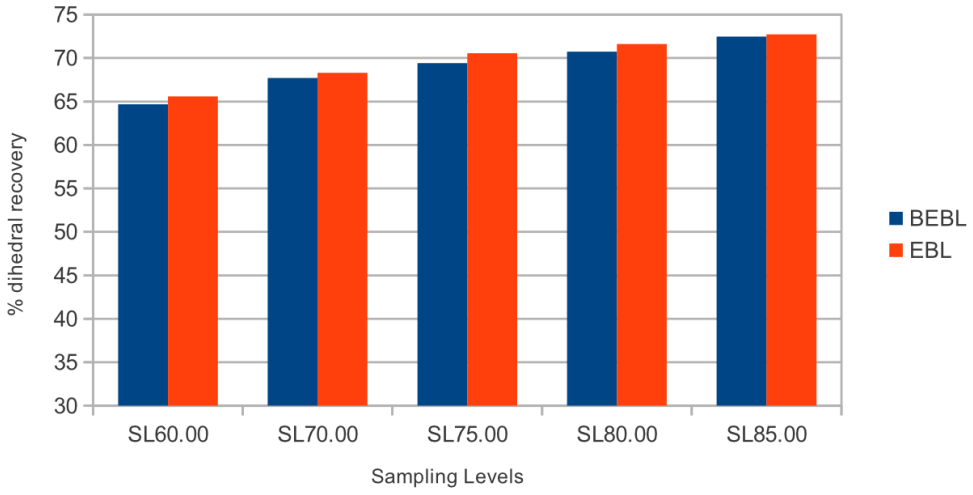


Figure 3.5: *BEBL achieves comparable dihedral recovery with fewer conformers. Sidechain optimization was performed over 480 proteins in the test dataset using both EBL and BEBL at the same levels, and the dihedral recovery was measured over all amino acid types. The BEBL libraries were considerably smaller in size compared to the EBL(see Figure 3.7), in spite of this disadvantage, BEBL dihedral recovery was comparable to the EBL at all the measured "levels".*

B-EBL of comparable sizes (L1 < L2 < L3 < L4). It can be seen that B-EBL performs exceedingly well at low sampling levels and the difference is not as pronounced at the higher sampling levels (with B-EBL always performing better). This result makes a strong case for the use of B-EBL when application requirements mandate a quick repack procedure that does not compromise too much on modeling accuracy.

3.4 Conclusions

We have presented a backbone-dependent energy-based conformer library (B-EBL) that improves protein side chain modeling. Our experiments show that

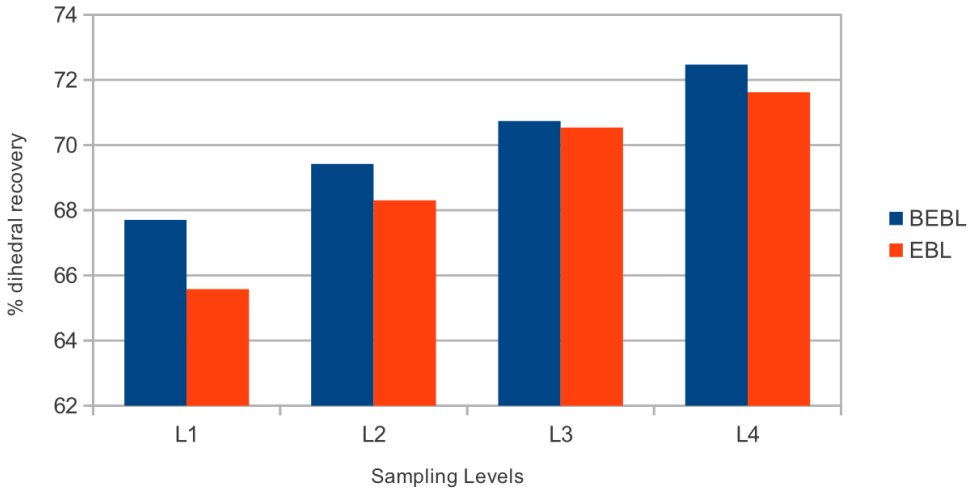


Figure 3.6: *Improved dihedral recovery with BEBL of comparable size. Sidechain optimization was performed on 480 proteins in the test dataset using both EBL and BEBL of approximately the same size (the BEBL was always smaller in size), and the dihedral recovery was measured across all amino acid types. The figure shows how BEBL libraries performed considerably better than the EBL at all the measured "levels".*

the EBL algorithm can be successfully applied in the creation of an energy-based conformer library that exploits the backbone-dependence of side chain conformation. The B-EBL was compared against a backbone-independent energy-based library (EBL) which has been shown to perform better than state-of-the-art conformer libraries in [**?**]. The B-EBL outperforms EBL in terms of both the energies achieved through side chain optimization as well as modeling accuracy measured by dihedral recovery. B-EBL exploits the backbone-dependence of side chain conformation to improve performance while at the same time retaining the flexibility of the EBL. we have experimentally demonstrated that the B-EBL requires fewer conformers to achieve the same performance as the EBL and achieves better performance for the

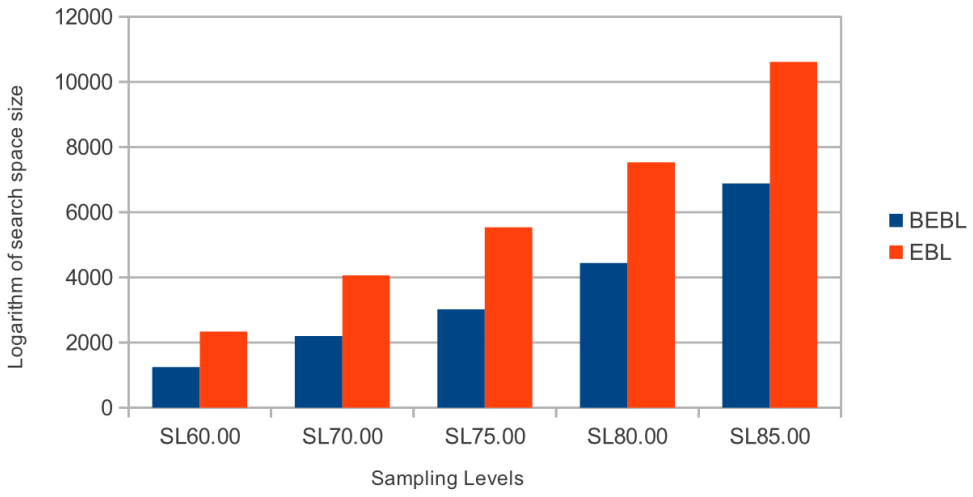
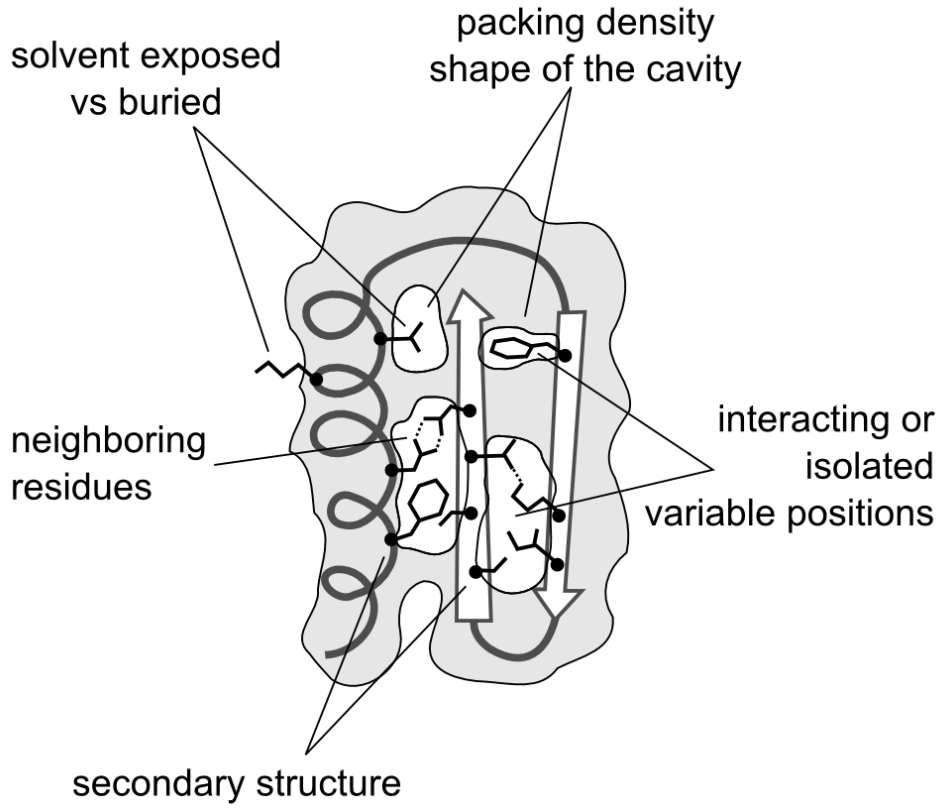


Figure 3.7: *The combinatorial size of the search spaces produced by the EBL and BEBL under comparison are shown in this figure. At all the defined sampling levels, BEBL presents a much smaller search space compared to the EBL. Inspite of this apparent disadvantage, the BEBL achieves better dihedral recovery than the EBL as shown in Figure 3.6.*

same number of conformers. The reduction of computational overhead, via the reduction in number of conformers. will benefit several protein modeling applications where side chain modeling is the bottleneck. On the other hand, improved accuracy for the same library size will help applications achieve better modeling without sacrificing computational efficiency.

4 POSITION-DEPENDENT ENERGY-BASED CONFORMER LIBRARIES



based on

Subramaniam S, Natarajan S, and Senes A "A Machine Learning based Approach to Improve Protein Sidechain Optimization", *ACM Conference on Bioinformatics,Computational Biology and Biomedicine (ACM BCB)* 2011, 478-480

Summary

Side chain optimization is the process of packing the sidechains of a protein onto a fixed backbone structure, such that the energy of the resultant structure is minimized. The continuous space of sidechain conformations is typically handled by discretizing (sampling) into a finite set of representative conformations called a "conformer library". In this chapter we use machine learning methods to allocate conformational sampling on a position-dependent basis. Different positions in a protein backbone have different sampling requirements, for example, solvent exposed positions require less sampling than positions in the core of a protein. Machine learning algorithms are used to identify the sampling requirements of each position in a target protein backbone based on a quantitative representation of its environment. A 3-ary categorization of every position in a target protein backbone is performed using several machine learning algorithms and the classification produced is evaluated in actual side chain optimization performed on high-resolution protein structures. Results demonstrate that in comparison with an unbiased distribution of conformational sampling, this position-dependent sampling helps to distribute sampling more efficiently for sidechain optimization. These techniques help achieve better prediction accuracy using fewer conformers, thereby reducing the computational cost of side chain optimization.

4.1 Position-dependent conformer sampling

In chapter 3, we have detailed strategies to determine how sampling should be prioritized to produce the most effective conformer libraries for sidechain optimization. While this library of top ranking conformers is a great resource, an exhaustive search over a large number of conformers for every position can be prohibitively expensive. It would be appropriate to consider a different search space for each position based on its structural properties. Since the

environment around each position in a protein varies widely, the sampling requirements of each position is also variable. For example, different positions in a protein may be surrounded by amino acids with different degrees of mobility and by immobile backbone atoms. Some positions may be exposed to the outside of the protein and have very loose constraints imposed by the environment. While it is clear that the level of sampling required to fit a sidechain in a different position depends on the tridimensional constraints of the environment, an exact relationship is difficult to identify given the extreme complexity of the problem. The question that we address in this chapter is the following: *How do we identify the set of sidechain conformations that will maximize performance in sidechain optimization?*

The problem is ideally suited for a Machine Learning (ML) approach and in this work, we present a method that analyzes the structural characteristics of each specific sidechain optimization problem and produces customized libraries that best suit the exact needs of each position in the problem. We take the following approach to this problem: in the first-step, we label each position based on the "ease of fit" for a set of proteins whose crystal structures are extracted from the Protein Data Bank (PDB) [**?**]. In the second step, a classification algorithm is trained to predict the label for each position based on environmental features that can be easily obtained for unseen proteins in the target set. The classification allows us to drastically reduce the search space through the sorted conformer library. As far as we are aware, this is the first attempt at using machine learning for directly addressing the preprocessing step of sidechain optimization - i.e., to improve the search space over which the underlying sidechain optimization algorithm would operate.

This chapter is based on [**?**] and describes several key contributions: First, it considers the extremely hard problem of distributing sampling among different positions in a protein. To our knowledge, there is no method that attempts to differentiate between positions of the same amino acid type i.e.,

search a different set of conformers for each position. Second, the proposed two-stage approach effectively exploits the training set energies and learns a classifier that can be generalized to an unseen protein. This proposed approach is directly based on the objective function of the optimization (*energy*) and uses structural features that allow for generalization. Third, evaluation on 44 proteins shows that the ML-based approach can yield a better search space compared to the baseline method with similar memory and runtime constraints.

Conformer Library

The experiments in this chapter are based on the energy-based conformer library we created in our previous work [**?**] : a library that distributes sampling based on the energetic impact of sidechain conformation instead of pure geometry. The energy-based library is a ranked list of conformers for each amino acid type where the first n conformers is probably the best set of 'n' conformers.

4.2 Machine Learning to Distribute Conformer Sampling

Formally, this problem can be posed as follows: Given a set of backbones B, where each $\mathsf{b_i} \in \mathsf{B}$ corresponds to the backbone of protein i and a (possibly infinite) set of conformers C, the goal is to identify a smaller subset of conformers to be evaluated at each position $\mathsf{r_i^j}$ for $\mathsf{b_i}$ such that the energy $\mathsf{e_i}$ of protein i is minimized. We denote sets using capitalized letters and individual items using small letters. The conformer library that we have created is a great resource to fit a conformer in each position. It provides a sorted list of conformers for each amino acid. The size of the conformer library (let us denote this as $\mathsf{C_l}$) is significantly smaller than C. Even with

C_l, the challenge is to bound the search space i.e., determine the number of conformers to be evaluated at every position.

The typical strategy is to use the same number (say n) of conformers at each position. If there are m_i residues for the current backbone b_i, this leads to a search space of n^{m_i}. This could be a sub-optimal solution because:

1. Some positions can be very flexible, a search over a smaller number of conformers ($< n$) would suffice at such positions.

2. Some positions can be very "hard" to fit and would require a search over possibly a very large number of conformers ($> n$).

In the first case, this method would yield an unnecessarily large search space while in the second case due to the restricted search space, the energy of the protein may not be sufficiently minimized. Thus, in most cases, we require a different number of conformers to be evaluated at every position.

Since determining a different number of conformers for every position is an extremely hard problem, we propose to classify the positions r_i^j into a small number of categories (say Y) based on the ease of conformer fit. To this effect, we employ a machine learning approach that we present in detail in this chapter. For this problem, the dataset consists of the proteins $b_i, i = 1..n$ with their crystal structures and corresponding energies $e_i, i = 1..n$ and the sorted list C_l. The goal is to learn to predict the number of conformers that need to be searched in a new protein in order to achieve its minimal energy state.

Labeling

Given this data, the goal is to categorize every position r_i^j in the backbone of each protein based on the fit of the conformers. We adopt the following strategy to obtain a label (y_i^j) for each position r_i^j. We *score* each r_i^j of each b_i in the training set based on the fit of the conformers in C_l. More

precisely, we traverse the sorted list and for each position, find the number of conformers that result in a "good" energy of interaction for the sidechain with the rest of the protein. An important thing to note is that all the other positions $r_i^{j'}, j' \neq j$ are held in their natural state i.e., they all retain their crystal structure). The *score* s_i^j of r_i^j is then the number of conformers that are a good fit for the position.

Every position with a high score can be fit by a large number of conformers, therefore, it is easy to find a conformer in the library that fits this position and so it will be labeled as "easy" . A position with a low score, on the other hand, will be labeled as "hard". More precisely, we label the top and bottom 25 percentile of positions (based on the score) as "easy" and "hard" respectively, and all positions in between are labeled "unclassified" indicating that we cannot label them as "easy" or "hard" with high confidence . Thus the set Y consists of the elements easy, unclassified and hard.

Hence, in the first step, we label the different residues (r_i^j's) in the backbone of the training set proteins based on their score (s_i^j) into different categories (denoted by the set Y). The number of conformers (n_k) required for each of these categories (y_k) is obtained from the *energyTable* described in section 2.3. The "easy" positions are assigned numbers from a lower optimization *level* than the "hard" positions.

Given a new protein backbone, it is computationally expensive to categorize each position of the backbone with the above strategy. This is due to the fact that such a labeling step requires traversing through each position and manually evaluating the fit of every conformer in the library. It is also worth noting that the above strategy requires the sidechain atoms of other positions to be present when a conformer is evaluated at a position. So, in this work, we propose using a machine learning algorithm to predict the category of each position and then using these labels to assign the level of sampling for each position. Such a prediction algorithm would make it

possible to pre-define the number of conformers to be evaluated at each position taking into account the sampling requirements for that position.

Classification

One of the main advantages of machine learning methods is the ability to learn a model from a training set and predict on a target set that consists of the same set of features. In the labeling step, we computed and used the score at each position to label positions in the training set. As noted earlier, obtaining this score on an unseen protein backbone is computationally expensive. Hence, we use a set of structural features that can be obtained easily from the backbone for this step.

We now modify the training set to contain the label y_i^j for each position r_i^j of each backbone b_i and a set of chosen geometric and positional features (F_i^j) for each r_i^j. Recall that the labels are the manual labels created in the previous step. The features include: (a) the *backbone dihedral angles* (ϕ and ψ) for 4 positions before and after the current r_i^j in b_i, 16 in total (b) *local sequence* information (i.e., the residues for 4 positions before and after r_i^j), 9 features (c) the *number of backbone atoms* not in the local sequence and are within $\langle 4, 8, 12, 16 \rangle$ Å of r_i^j, and (d) *solvent accessible surface area* for r_i^j. Hence, we use a total of 30 local environment features. We believe that the local environment is a reasonably good indicator of the label for every r_i^j. This modified data set is then used for the second step where we learn a model to predict the category of each residue (position) for a given backbone.

Now the goal is to learn a function $f(F_i^j) = y_i^j$ that predicts the "hardness" of r_i^j based on its features F_i^j. Please note that these features can be easily computed for an unseen protein as well, thus justifying their use. Hence, the learned model can easily be generalized to an unseen protein backbone. We demonstrate this empirically in the next section. We employ different learning algorithms ranging from Naive Bayes [**?**] to decision trees [**?**] to

ensemble methods such as bagging [?] and boosting [?] in our experiments. The goal of this work is not to particularly recommend a single classification method, but to explore the use of machine learning for this challenging problem.

Algorithm for sidechain optimization

The result of the classification step is a label that predicts the "hardness" of a position. Given the hardness level, we use a preset number of conformers to fit at each position. In this subsection, we present the algorithm that actually searches through the given set of conformers to identify the conformer that results in minimum energy i.e., the actual *sidechain optimization* algorithm.

The program used to perform sidechain optimization implements dead-end elimination [?], self-consistent mean field [?] and metropolis monte carlo methods [?] to search the space of conformations defined by the the conformer library. The algorithm begins by rejecting conformers according to the dead end elimination theorem. Then, it estimates the size of the search space, if this size is smaller than a configurable threshold, it performs an exhaustive search. If the search space is larger, self-consistent mean field(SCMF) and monte carlo methods are used to search for the minimum energy configuration. The monte carlo method is biased using the state obtained by the SCMF method, i.e., the starting state for the monte carlo method is the most probable state obtained by the SCMF method.

Algorithm for distributing sampling The different steps involved in our algorithm are presented visually in Figure 4.1. As can be seen, our first-step is to use the protein database that consists of the backbones b_i and their corresponding energies e_i $\forall i$. Then using the conformer library C_l, we compute the scores s_i^j for each r_i^j and create a new training database with r_i^j and s_i^j $\forall i, j$. Then, we assign a label y_i^j for each r_i^j that is a measure of its "hardness". In the next step, we create a new database that consists

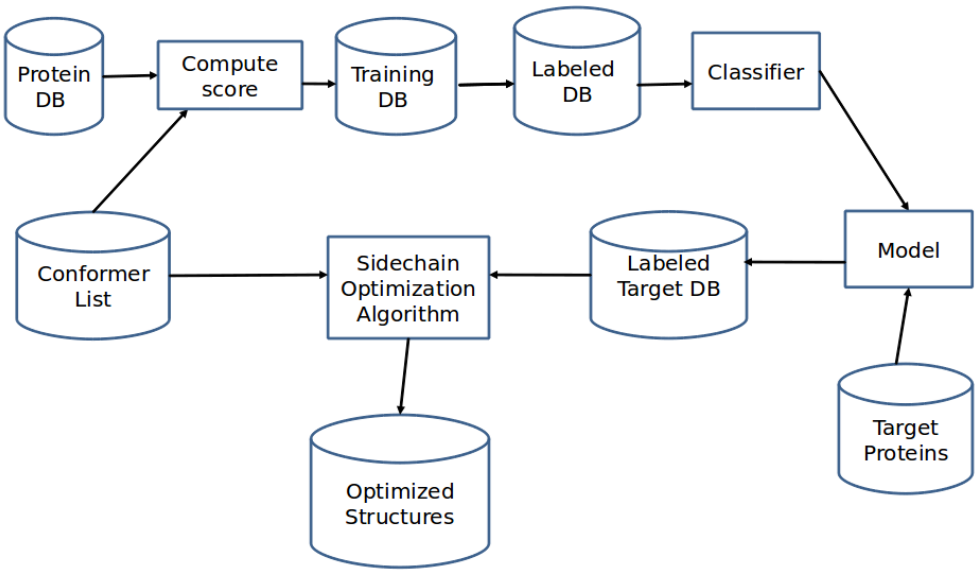


Figure 4.1: *Schematic representation of the biased sampling strategy. Each environment in a training database is labeled with one of three labels {Easy,Medium,Hard} depending on its sampling requirements. A machine learning algorithm is employed to learn a model that can effectively classify a previously unseen environment. Given a target backbone, this algorithm classifies each position and allocates conformational sampling according to the label.*

of the labels y_i^j and the features F_i^j for each r_i^j. We then learn a classifier M that learns the mapping $f(F_i^j) = y_i^j \ \forall i, j$. This model M is then used on the unseen protein database to label each residue of each protein based on the hardness. Then the sidechain optimization algorithm presented in Section 4.2 is used to identify the conformer that leads to the minimum energy for the current protein.

The algorithm has several advantages: First, we address the previously unaddressed problem of reducing the search space while achieving minimum energies for the residues using a machine learning approach. This is a very important contribution as we are the first to explore the possibility of assigning different conformer sampling to the different positions on proteins

using their structural features. Second, it makes use of the conformer library more effectively than choosing a random number of conformers to test on each residue. This could reduce the search space over the conformer set drastically or help explore lower energies than before at a similar computational cost. Finally, our algorithm is independent of the underlying machine learning techniques. Any existing algorithm can be used for the supervised learning task and the sidechain optimization step, to improve performance.

4.3 Empirical Evaluation

In this section, we present the results of our experiments on a dataset that consists of approximately 720 proteins. We selected a small subset as the test set containing around 44 proteins and compared several different supervised learning algorithms. The crystal structures of all proteins in the database were derived from the Protein Data Bank(PDB). The proteins were modeled using MSL [Kulp et al. (2012)], which is an open source C++ library for analysis, manipulation, modeling and design of macromolecules. MSL implements all the energy terms in the CHARMM [Brooks et al. (1983)] potential along with the hydrogen bonding term implemented in the SCWRL [Krivov et al. (2009)] program. We include the following energy terms a) *bonded* energy terms - *bond*, *angle*, *dihedral* and *improper* b) *non-bonded* energy terms - *Van der Waals' interaction*, *hydrogen bonding*. We select all the positions in these 720 proteins except PRO, ALA and GLY resulting in a total of around 140000 positions. Each position is then scored as described in subsection 4.3. Classification is then performed on this dataset for each amino acid separately. To achieve this, we use the WEKA [**?**] package.

The goal of this empirical evaluation is to answer the following questions:

1. Does the labeling scheme really help achieve the goals of reduced search spaces and/or improved energies?

2. How do the classification methods compare against a baseline method that does not differentiate between the different positions (but uses a search space of similar size)?

3. Do the structural features used really have predictive value ?

4. How do different classifiers compare against each other?

The overall goal is to answer the following question: How can we explore lower energies while minimizing the computational cost by a biased distribution of conformer sampling? Since, as far as we are aware, there do not seem to be any existing approaches to exactly answer this question, we attempt to provide some insights by analyzing our strategy and answering questions 1 through 4 experimentally.

Labeling Experiments

To answer Q1, we performed the following experiments. The first experiment is to demonstrate that positions that were labeled as “easy" indeed require low sampling. To verify this, we repack the test set proteins using the actual labels. We optimize the test proteins with an unbiased sampling ie., assigning the same number of conformers to all positions with the same amino acid, irrespective of their labels, lets call this scheme 1. We then optimize the same protein with a different sampling for the easy positions alone i.e., all non-easy positions are assigned the same sampling as scheme 1 and all easy positions are assigned a lower *level* of sampling, lets call this call scheme 2. The energies obtained for each protein using the two schemes are then analyzed.

Figure 4.2 shows the energies obtained using schemes 1 and 2 described above, for all of the test set proteins. If our labeling scheme was precise, we would expect all the energies to lie on the diagonal. In fact almost all the points in Figure 4.2 lie very close to the diagonal indicating very high

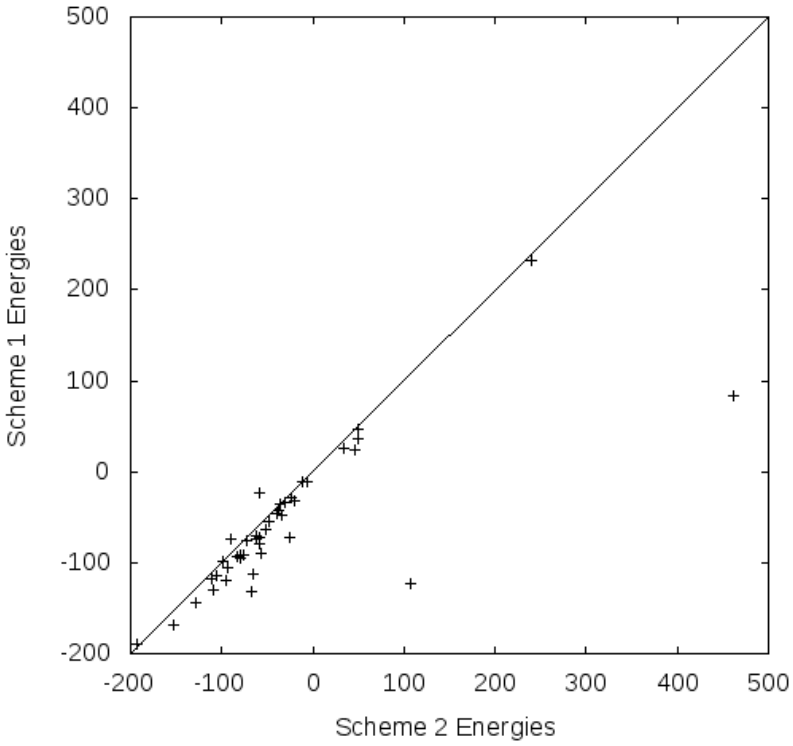


Figure 4.2: *The energies obtained using Scheme 1 (unbiased sampling) and Scheme 2 (low sampling at easy positions) are plotted after subtracting the energy of the corresponding crystal structure. In a majority of cases only a small increase in energy is observed at lower sampling - i.e., most points lie close to the diagonal.*

precision of the labeling process. Considering the fact that the search space of scheme 2 is a strict subset(much smaller) of scheme 1, it is interesting to see that in 3 proteins, scheme 2 produced slightly better energies than scheme 1. This can be explained by the approximate/probabilistic nature of the underlying sidechain optimization algorithms (SCMF and MC). In only 2 (5%) proteins, scheme 1 clearly outperforms scheme 2. This shows that in most cases we are able to achieve good energies at a much lesser computational cost by reducing the sampling for positions that are labeled as *easy*.

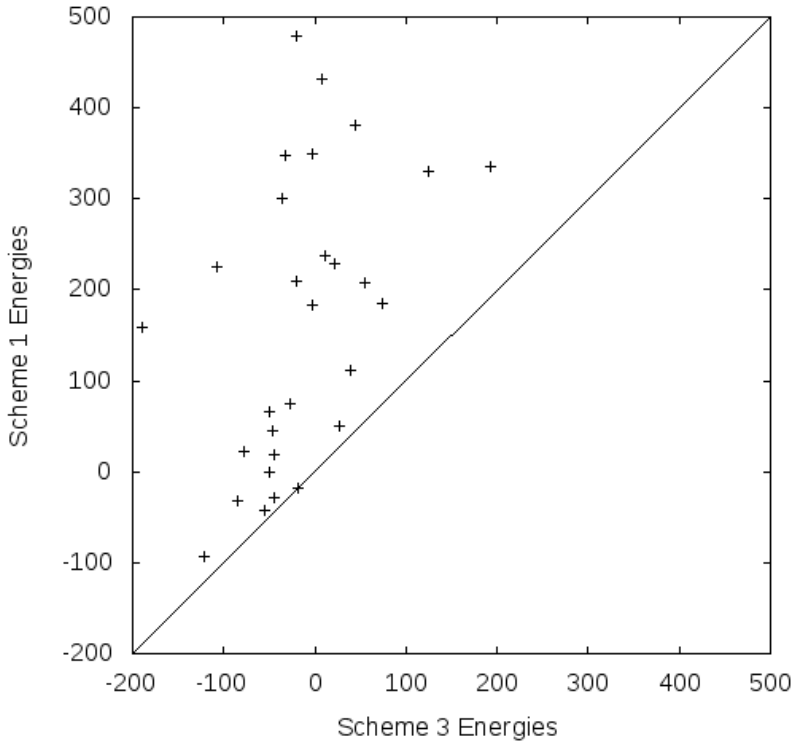


Figure 4.3: *The energies obtained using Scheme 1 (unbiased sampling) and Scheme 3 (high sampling at hard positions) after subtracting the crystal energy are plotted. In all the cases, Scheme 3 achieves lower energies.*

The above experiment measured the precision of the labeling process on the easy positions. Next, we analyze the same on hard positions. We carry out a similar experiment as the one presented above but with the hard positions, and let us denote this as scheme 3. In this case the search space of Scheme 3 is much bigger than that of Scheme 1 and we expect better energies at a higher computational cost. In Figure 4.3 we see that all the points lie to the left of the diagonal, indicating lower energies in scheme 3, as expected.

Ideally, the advantages of both the methods must be obtained, i.e., it should be possible to achieve lower energy configurations at reasonable

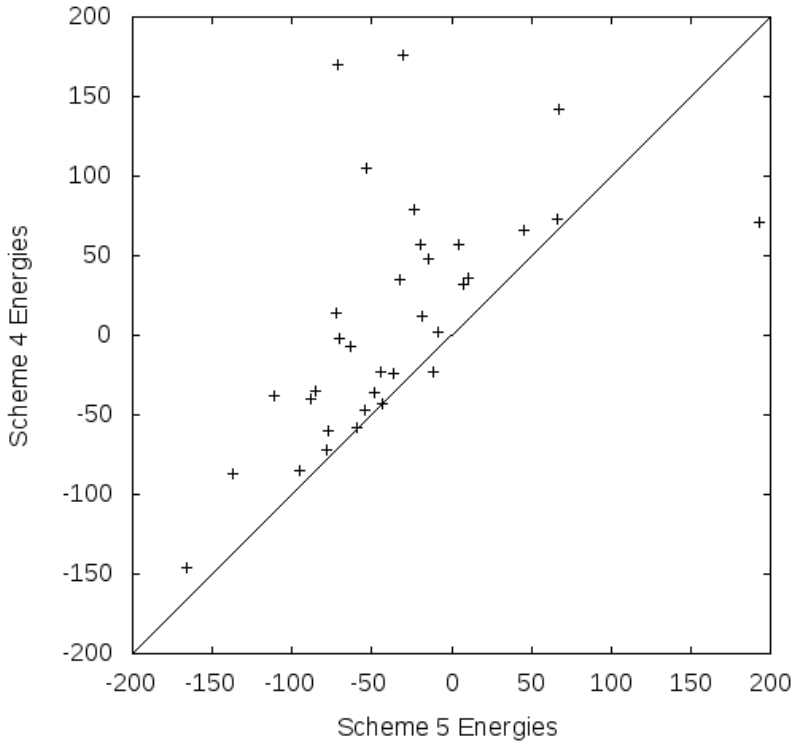


Figure 4.4: *The energies obtained using Scheme 4 (unbiased sampling) and Scheme 5 (lower sampling at easy positions and higher sampling at hard positions) after subtracting the crystal energy are plotted. Scheme 5 achieves lower energies in 95% of the cases.*

computational costs. To understand whether this is possible, we perform the following experiment. We assign a *low* level of sampling to the easy positions and *high* level of sampling to the hard positions. We then choose a `level` of sampling for the unclassified positions such that the difference in the search spaces for the biased sampling and the unbiased sampling scheme is minimal [1], lets call the unbiased sampling scheme 4 and the biased sampling scheme 5. In Figure 4.4 we see that scheme 5 achieves better energies in almost all

[1] In 31 proteins, the difference in log size of the two search spaces was under 2. And in 27 of these proteins scheme 5 had significantly better energies.

the cases, thus justifying the labeling process. This clearly demonstrates that a biased sampling that explicitly distinguishes between the easy and hard cases achieves better energies while being computationally reasonable.

Thus, our results answer Q1 affirmatively. However, as mentioned earlier, it will be computationally expensive to perform the labeling process every time a new sidechain optimization problem is presented. Therefore, if we can train a classifier to predict these labels as accurately as possible we can use this biased sampling strategy on new problems efficiently. The following section presents our experiments with the classifiers available in WEKA.

Classification Experiments

Recall that the features used in the classification step are different from the one (score) used in the labeling step. This is due to the fact that computing the score for each position in a backbone can be computationally expensive. Hence, in the following set of experiments, we used the 30 environmental features presented in section 4.2 and the labels generated in the previous step for training. During evaluation, we use the same set of features to predict on the 44 test set proteins.

We use the following classification algorithms from the WEKA package: a) *Bagging* [**?**] b) *simpleCART* [**?**] - decision tree learner c) *LogitBoost* [**?**] - Boosting of decision stumps and d) *Naive Bayes* [**?**]. In effect, we learn a model using each of these methods to predict the label for each position in each protein. Once the model is learned, we predict the labels at each position of the test-set proteins.

Once the label for each position is identified, the number of conformers is chosen from the `levels` described in chapter 2. The number of conformers chosen for the hard and easy positions of each amino acid are presented in Table 4.1. These numbers correspond to the 87.5%-*level* and 98%-*level* as described in chapter 2. The number of conformers for the "unclassified" positions is computed for each protein as described for scheme 4 in Section 4.3,

Amino Acid	Hard	Easy
ARG	224	20
ASN	20	4
ASP	22	4
CYS	169	8
GLN	48	7
GLU	53	7
HSD	57	9
HSE	44	8
HSP	132	37
ILE	15	5
LEU	27	3
LYS	54	6
MET	76	20
PHE	78	16
SER	3	2
THR	6	2
TRP	284	58
TYR	176	40
VAL	5	3

Table 4.1: *Number of conformers chosen for the hard and easy positions.*

let us call this scheme the baseline for all future experiments. Then, sidechain optimization was performed on the test set proteins by the program described in Section 4.2. Finally, we analyzed the energies achieved by sidechain optimization on the test set proteins.

Figure 4.5 presents the results of the comparison of the models against the baseline method which uses an unbiased sampling for all positions. For each of the classification methods, We compare the energies produced on the classified and baseline models. We compute the fraction of proteins in which the energy for one method is significantly (> 2 $\mathtt{kcal.mol}^{-1}$) lower than the other. The figure presents the results for all the classifier methods listed above. As can be seen, the use of the classification methods yields a superior

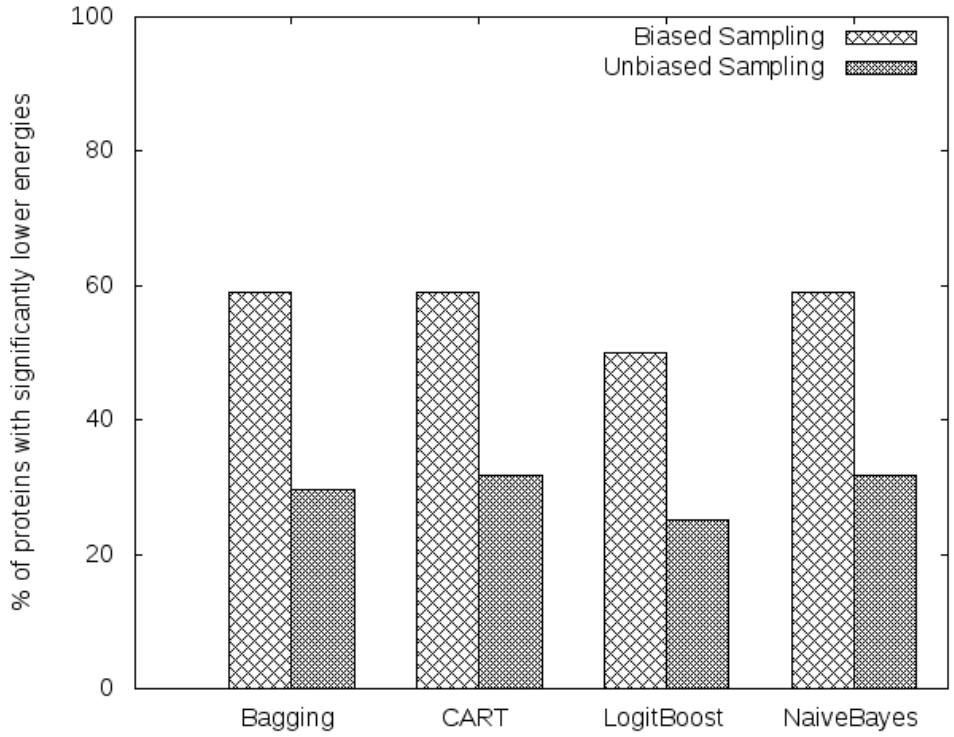


Figure 4.5: *Performance of different classification methods against the baseline. The classifiers achieve significantly lower energies than the baseline in around 60% of the proteins, whereas the baseline achieves lower energies in around 30% of the proteins only. In the remaining proteins, the difference in energies was not significant (< 2* $\mathtt{kcal.mol}^{-1}$*).*

performance for a larger fraction of proteins compared to the baseline which uses a search space of similar size but without explicitly distinguishing each position. The performance seems to be comparable for Bagging, CART and NaiveBayes with LogitBoost performing slightly less efficiently. Hence, we can answer Q2 positively i.e., all the classifiers perform better than a baseline that is unbiased.

The results show that using the labels produced by the classifiers improves performance in most cases. We want to verify that this performance

improvement is not easy to achieve and that the classifiers are indeed making statistically informed decisions. In order to do this, we test the performance of a random process that labels 25% of positions as easy, another 25% as hard and the remaining as "unclassified", uniformly at random (this is the proportion of easy, hard and unclassified positions in the dataset as explained in Section 4.2). So we compare the performance of 4 instances of such a random process against the baseline. Figure 4.6 shows the results for these random classifiers. Random processes which do not consider the structural features perform considerably worse compared to the unbiased baseline method. Therefore, we may conclude that the classifiers based on structural features do indeed learn useful information that helps discriminate between the different kinds of positions and we conclude that Q3 in Section 4.3 has been answered affirmatively.

In order to compare the performance of the classifiers against each other (and answer Q4), we rank the different classification methods for each protein based on the resulting energies. The method that has the minimum energy is ranked first while the one with maximum energy is ranked last for that protein. We compute the inverse reciprocal rank (IRR) for all the methods for all the proteins. If the rank of the current method M_j for a protein i is r_i, IRR can be calculated as:

$$IRR(M_j) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{r_i} \tag{4.1}$$

where n is the number of proteins in the test set (= 44). Hence an IRR of 1 would imply that the method has been ranked as first for all the proteins, while an IRR of 0.5 may mean it was ranked second consistently etc. Figure 4.7 presents the IRR for all the methods.

Bagging seems to be the most efficient classifier with the highest IRR. CART and Naive Bayes have a comparable IRR followed by logitBoost which has the lowest IRR. However, the experiments were performed with default

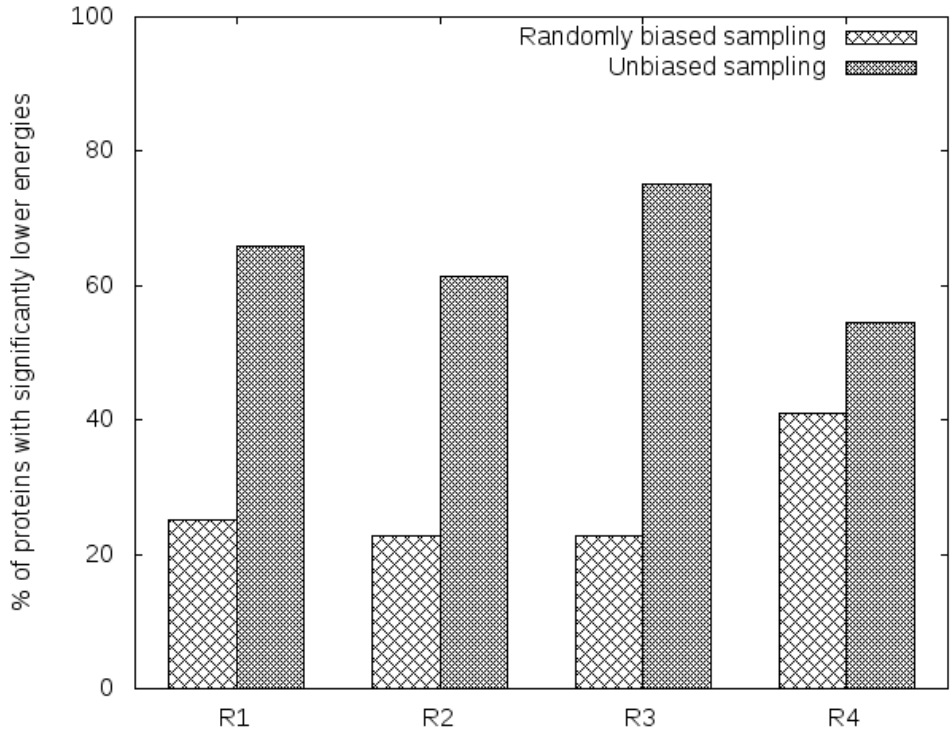


Figure 4.6: *Performance of Random labeling against the baseline. In all of the experiments, random labeling produces higher energies than the baseline method in about 60-70% of the proteins. Clearly random labeling performs significantly poorer than the baseline method which is in turn poorer than the biased sampling method.*

parameters for all the classification methods, it may be the case that tuning the parameters could dramatically change the IRR data presented above. Since our goal is not to suggest any particular method, but to evaluate the effectiveness of a machine learning method for this task in general, we do not tune the parameters to maximize the performance of the classifiers.

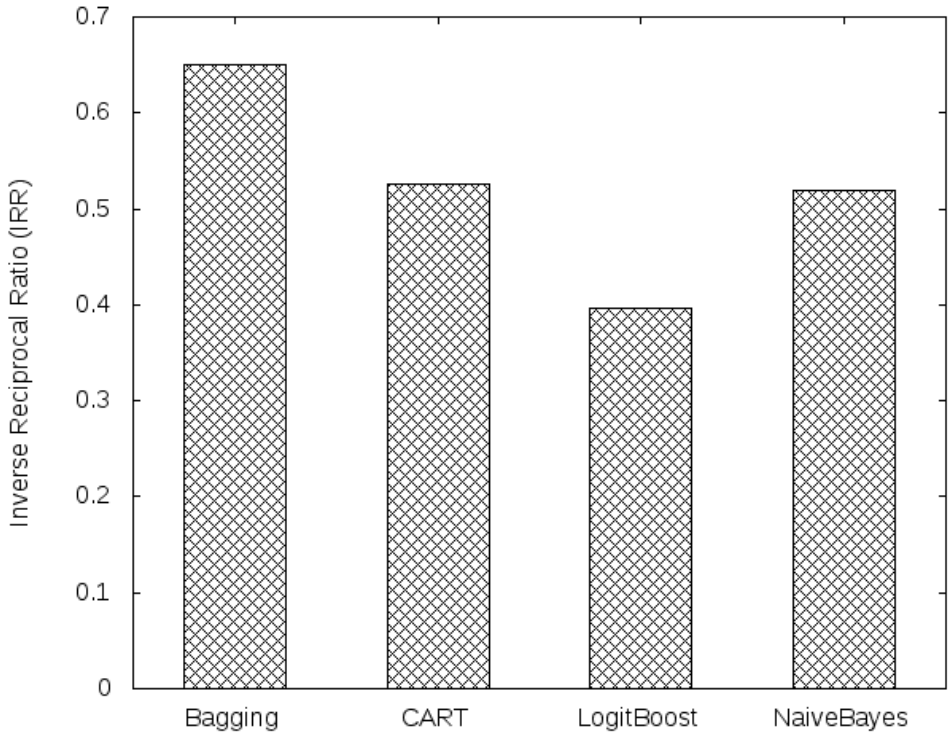


Figure 4.7: *Comparison of different classification methods against each other using the IRR metric. Bagging performs the best, followed by CART and NaiveBayes and LogitBoost has the worst performance.*

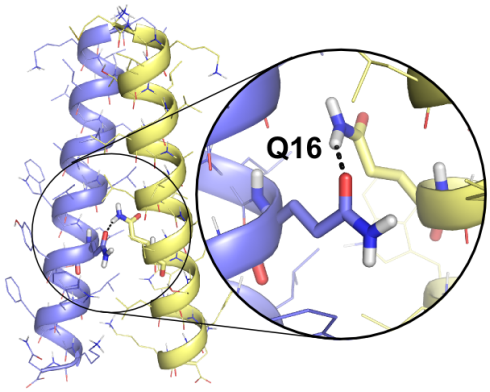
4.4 Conclusion

In this chapter, we addressed the challenging problem of distributing sampling among different positions in a protein. To achieve this, we developed a two-stage approach that used an underlying conformer library. In the first-step, we labeled the positions based on the number of conformers in the library that can fit in the position. In the second step, we used a different set of features (i.e., structural features) to learn a classifier that is able to predict the hardness of the different positions. We evaluated the two-stage approach on 44 proteins from the protein data bank. Our evaluation allowed

us to answer affirmatively a variety of questions ranging from the validity of the labeling strategy to the effectiveness of the environmental features to the performance of various classification algorithms. Our results strongly demonstrate that the use of machine learning indeed results in superior performance compared to a method that does not explicitly differentiate the positions.

As far as we are aware, this is the first attempt at the use of Machine Learning for the problem of distributing sampling across positions. It should be pointed out that we are exploring the use of Machine Learning for this task and do not necessarily recommend a particular method. We did not perform much parameter tuning in our experiments and used the default settings in WEKA. Moreover, our current approach (being a first-step in this direction) considers only a 3-ary classification. It must be possible to define the ease-of-fit at a finer granularity. Hence, it would be interesting to explore the use of more classification levels and understand its impact on the overall energy. Still, our results clearly show that we are able to achieve significantly lower energies while exploring a search space of similar size and/or achieve good energies with a smaller search space, thereby reducing the computational cost of sidechain optimization. While the cost of sidechain optimization may not be a limiting factor in many modeling procedures, improved performance is likely to open the doors to very computationally intensive applications. For example, improved performance becomes critical when sidechain optimization is performed iteratively during numerous rounds of backbone sampling, or when solvent molecules are modeled explicitly during side chain optimization.

5 PREDICTION DRIVEN BY EXPERIMENTAL DATA: STRUCTURAL ANALYSIS OF FTSB



based on

LaPointe LM, Taylor KC, **Subramaniam S**, Khadria A, Rayment I and Senes A "Structural organization of FtsB, a transmembrane protein of the bacterial divisome", *Biochemistry* 2013 **52**, 2574-85

My contribution to this work is the computational modeling of the interacting transmembrane (TM) domains and the flexible linker region between the TM and the periplasmic coiled coil.

Summary

This work is part of the first structural analysis of an integral membrane protein of the bacterial divisome, a set of proteins that bring about cell division. FtsB is a single-pass membrane protein with a periplasmic coiled coil. Its heterologous association with its partner FtsL represents an essential event for the recruitment of the late components to the division site. Using a combination of mutagenesis, computational modeling, and X-ray crystallography, it was determined that FtsB self-associates, and further investigation revealed its structural organization. It is determined that the transmembrane domain of FtsB homo-oligomerized through an evolutionarily conserved interaction interface where a polar residue (Gln 16) plays a critical role through the formation of an interhelical hydrogen bond. The crystal structure of the periplasmic domain, solved as a fusion with Gp7, shows that 30 juxta-membrane amino acids of FtsB form a canonical coiled coil. The presence of conserved Gly residue in the linker region suggests that flexibility between the transmembrane and coiled coil domains is functionally important. We hypothesize that the transmembrane helices of FtsB form a stable dimeric core for its association with FtsL into a higher-order oligomer and that FtsL is required to stabilize the periplasmic omain of FtsB, leading to the formation of a complex that is competent for binding to FtsQ, and to their consequent recruitment to the divisome. The study provides an experimentally validated structural model and identifies point mutations that disrupt association, thereby establishing important groundwork for the functional characterization of FtsB in vivo. This chapter is based on [**?**] and discusses my contribution in this work for the molecular modeling of the FtsB dimer.

5.1 Introduction

Cell division is one of the most fundamental processes in the life of bacteria. In gram-negative bacteria, division requires a complex and coordinated remodeling of the three-layer cell envelope, and therefore mechanisms must exist to sort the duplicated chromosome, to provide constrictive force, to synthesize the septal cell wall, and, finally, to induce membrane fusion. These events are enabled by a multiprotein complex called the divisome. The assembly of the divisome begins with the formation of a ring-like structure at the site of division (the Z-ring), where the polymeric FtsZ likely provides constrictive force and forms a scaffold for the recruitment of the complex [**???**]. In *Escherichia coli* the recruitment of the essential proteins follows a strikingly linear hierarchy, illustrated in Figure 1a [**?**]. The cytoplasmic side of the ring is formed by the early components: FtsA, a membrane-associated actin family member that forms protofilaments [**??**]; ZipA, a single-pass membrane protein that contributes to FtsZ tethering along with FtsA [**??**]; and FtsK, a DNA translocase that is essential for unlinking chromosome dimers after homologous recombination [**?**]. In contrast, the late proteins perform functions related to the reconstruction of the cell-wall: FtsW is a transporter of cell wall precursors across the membrane[**??**]; FtsI is important for the cross-linking of the cell wall during division;[**?**] and FtsN is necessary for the recruitment of nonessential septal components, the murein hydrolase AmiC, [**??**] and the Tol-Pal complex required for proper invagination during constriction [**?**].

The early and late components of the divisome are linked by a trio of single pass transmembrane (TM) proteins: FtsQ, FtsB, and FtsL. The ability of FtsB and FtsL to recruit the late divisome elements suggests that they have a structural role as a scaffold in the assembly of the divisome [**?**], and it has been proposed that they are involved in Z-ring stabilization [**?**]. FtsB and FtsL may also be involved in a regulatory checkpoint of division because the depletion of FtsB from E. coli cells results in the disappearance of FtsL

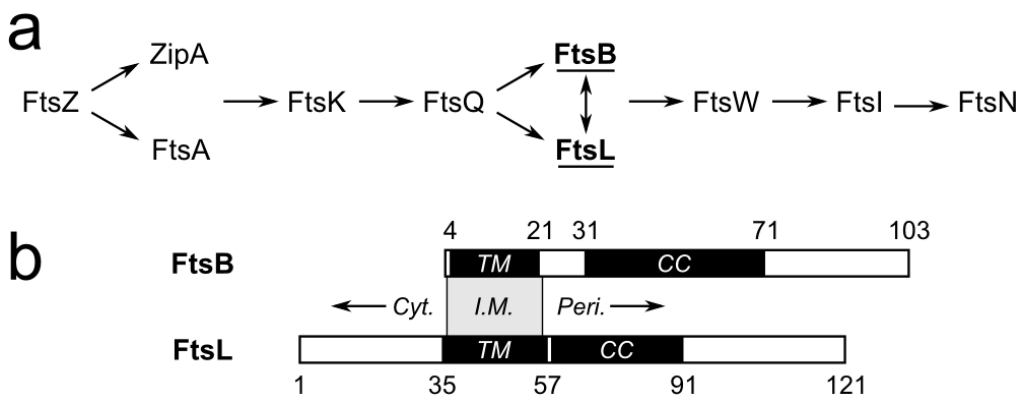


Figure 5.1: *The recruitment hierarchy of the divisome and the predicted topology of FtsB and FtsL. (a) In E. coli the recruitment of the divisome to the division site follows a strict hierarchical dependency [**?**]. A functional FtsZ is required for the recruitment of FtsA and ZipA, which in turn are required for the recruitment of FtsK, and so on. FtsB and FtsL are codependent for their recruitment and both depend on FtsQ. (b) Putative domain topology of FtsB and FtsL, as annotated in UniProt. Their interaction is presumed to be mediated by their single transmembrane domain (TM) and a juxta-membrane coiled coil region (CC). The start and end positions of the predicted TM and CC domains are indicated. Cyt. = cytoplasm. I.M. = inner membrane. Peri. = periplasm.*

[**?**]. The cellular instability of FtsL was also observed in Bacillus subtilis [**???**], where FtsLB is rapidly degraded by the intramembrane protease RasP unless it is stabilized by its interaction with the FtsB homologue (DivIC) [**?**]. These observations led to an unconfirmed hypothesis that active proteolysis of FtsL may be a regulatory factor in the timing of bacterial cell division [**?**].

While the precise function of FtsB and FtsL is not well understood, substantial evidence indicates that they physically interact with each other. As highlighted in Figure 1a, FtsB and FtsL are mutually dependent for their recruitment at the division site, and both proteins depend on the localization of FtsQ [**??**]. A similar picture has been reported in B. subtilis, where the localization of the homologues of FtsLB and DivIC depends on the FtsQ homologue (DivIB) at the temperature at which DivIB is essential[**??**].

There is strong evidence that FtsB and FtsL form a stable subcomplex in vivo. A complex comprising FtsB, FtsL, and FtsQ was isolated from E. coli by coimmunoprecipitation [**?**]. The physical interaction of the E. coli and B. subtilis proteins was also confirmed by two-hybrid analysis [**???**]. Further evidence of a stable interaction between FtsB and FtsL was obtained with a series of artificial septal targeting experiments [**??????**] that demonstrated that FtsL and FtsB interact with each other and can recruit the downstream proteins even when FtsQ has been depleted from the cell [**?**]. Moreover, the B. subtilis homologues FtsLB and DivIC form a complex when coexpressed in E. coli despite the fact that they are unlikely to interact with the significantly divergent E. coli division proteins [**?**].

The domain organization of FtsB and FtsL (Figure 1b) suggests that they may interact through an extended helical structure encompassing the membrane and periplasmic regions. Both proteins contain a TM domain and a juxta-membrane coiled coil, in addition to a small (FtsL) or minimimal (FtsB) cytoplasimic N-terminal tail. The TM and coiled coil regions of FtsB are necessary and sufficient for its interaction with FtsL [**?**] and similarly, the TM and coiled coil regions of FtsL are both essential for its interaction with FtsB [**?**]. A low resolution model of the soluble domains of the Streptococcus pneumoniae homologues of FtsB, FtsL, and FtsQ was proposed by Masson et al.,[**?**] based on a combination of NMR, small angle neutron, and X-ray scattering, and surface plasmon resonance. In this study the TM domains were truncated and replaced by a soluble coiled coil pair [**??**]. More recently, a bioinformatic analysis of the soluble domains of FtsB, FtsL, and FtsQ suggested two alternative models with 1:1:1 or 2:2:2 oligomeric stoichiometries [**?**]. However, no structural information was available regarding the organization of the important TM region.

In an effort toward understanding the structural organization and precise oligomeric state of the FtsB-FtsL complex, the self-association propensities of both TM and soluble regions of the two individual proteins were investigated.

We hypothesized if the FtsB-FtsL complex is larger than a dimer, one or both proteins could potentially self-associate and be studied in isolation. Indeed, it was experimentally determined that FtsB homo-oligomerizes using the TOXCAT assay [**?**]. Here we present a structural analysis obtained with a combination of extensive mutagenesis, computational modeling, and X-ray crystallography. The results provide a theoretical scaffold for the biophysical characterization of the FtsB-FtsL heterologous complex and offer several structure-based hypotheses that can be tested in the context of cell division by functional studies in vivo. This chapter describes the computational modeling (details of the mutagenesis and crystallography are presented in [**?**]).

The following hypotheses and experimental results were available as inputs to the computational modeling process:

1. The TOXCAT assay [**?**] confirmed that the transmembrane domain of FtsB self-associates [**?**].

2. Mutagenesis data for each position in the transmembrane region and its impact on dimerization was available.

3. Sequence analysis of related protein sequences (across variaous bacteriae) led to the hypothesis that a critical polar amino acid (Gln 16) mediates the self-association of the FtsB trasmembrane domain.

The FtsB transmembrane dimer was modeled computationally using the above inputs as described in Section 5.3. The generated TM model was then stitched to the crystal structure of the coiled coil region as described in Section 5.3 to complete the FtsB model.

5.2 Results and Discussion

TOXCAT reveals self-associating FtsB TM domain

The self-association of the TM domains of FtsB and FtsL from E. coli was analyzed using TOXCAT, a widely used biological assay for TM association [**?**]. The assay is based on a chimeric construct in which the TM domain of interest is fused to the ToxR transcriptional activator domain from Vibrio cholera (Figure 5.2a). Oligomerization, driven by the TM helices, results in the expression of the reporter gene chloramphenicol acetyltransferase (CAT). The expression level of CAT (measured enzymatically) is compared to that of a stable dimer, glycophorin A (GpA), as a standard. While TOXCAT is applicable to membrane proteins of any origin (GpA for example is a human protein), it is significant that in the case of FtsB and FtsL the analysis is performed in their native E. coli membrane. This fact also raised an initial concern that the TOXCAT constructs could potentially interfere with the cell division process. Fortunately, this concern was unfounded as the cells grew and divided normally. The results of the TOXCAT analysis of FtsB and FtsL are shown in Figure 5.2b. The CAT activity of both constructs is above background, although the association of FtsL appears to be weak (19% of the GpA signal). The activity of FtsB is approximately half of the GpA signal (48%), indicating that the homo-oligomerization of its TM domain is likely stable.

FtsB mutagenesis data supports self-associating helices

To investigate what amino acids are important for the self-association of FtsB and FtsL, we systematically mutated each position and monitored the effects on association. The expectation is that the changes at interfacial positions would perturb oligomerization more than the changes at positions

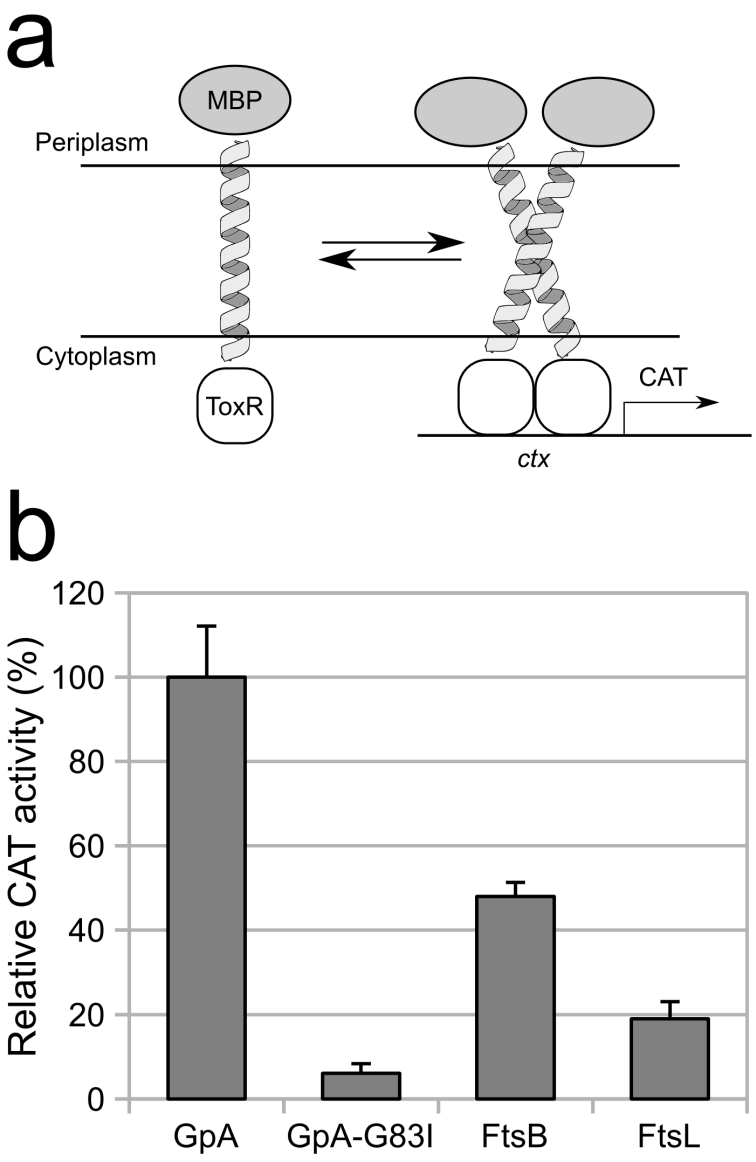


Figure 5.2: *FtsB self-associates in TOXCAT. (a) TOXCAT is an in vivo assay based on a construct in which the transmembrane domain under investigation is fused to the ToxR transcriptional activator of V. cholerae. Transmembrane association results in the expression of a reporter gene in* E. coli *cells, which can be quantified. (b) TOXCAT assay of FtsB and FtsL transmembrane domains. FtsB shows half of the activity of the strong transmembrane dimer of glycophorin A (GpA). The activity of FtsL is above baseline but low, indicating a weak propensity to homo-oligomerize. The monomeric G83I mutant of GpA is used as a negative control. Data are reported as average and standard deviation over four replicate experiments.*

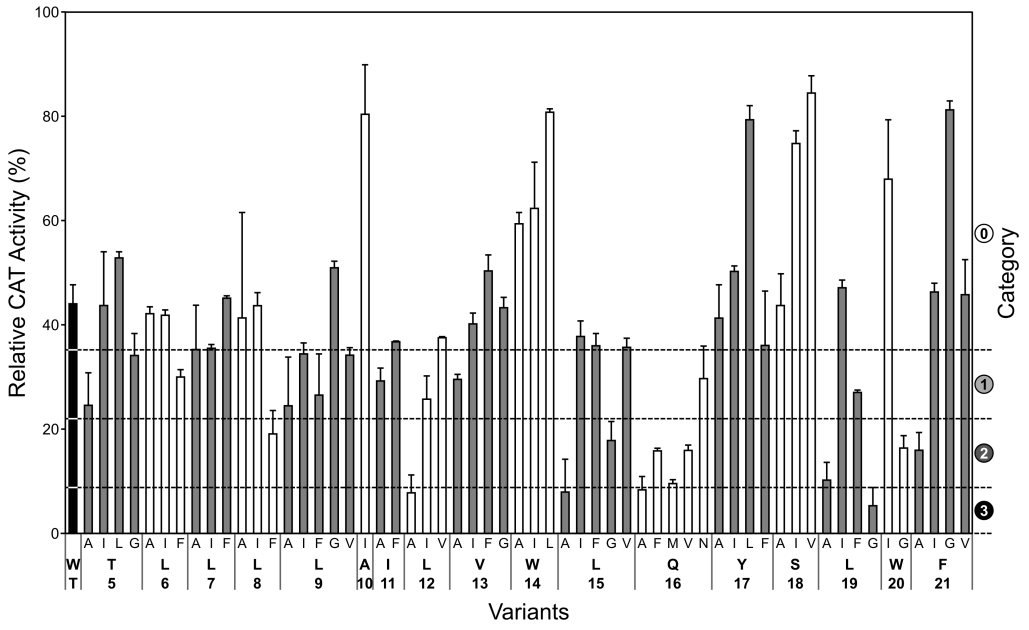


Figure 5.3: *Mutagenesis of the transmembrane helix of FtsB. The figure shows the 57 point mutants of the TM domain of FtsB (residues 5-21) analyzed in TOXCAT. The CAT activity (left axis) is normalized to that of the GpA construct, as in Figure 5.2. The activity of the wild type FtsB construct is in black. The mutations at each position are visually grouped by color. Each mutation has been categorized relative to the wild type FtsB TOXCAT activity (back bar) as "WT-like" (0: >80% of WT), "Mild" (1: 50-80%), "Severe" (2: 20-50%), or "Disruptive" (3: 0-20%), as indicated on the right axis and by the dashed lines. The TOXCAT data for all 57 variants is summarized using the above category scheme in Figure 5.4.*

that are lipid exposed, as commonly observed (for example [**???**]. We applied an initial scanning mutation strategy using both Ala (small) and Ile (large) substitutions and then expanded the mutagenesis to include a larger variety of hydrophobic amino acids.

Figure 5.3 shows the TOXCAT data for the 57 single amino acid variants tested for FtsB. All variants displayed similar levels of TOXCAT construct expression, as verified by Western blot analysis (data not shown). While a

majority of the variants have a CAT activity level comparable to the wild type constructs, there are a number of mutations that display drastically reduced activity. Conversely, several constructs with significantly increased activity were also observed. To obtain an estimate of the overall sensitivity of each position, we applied a simple classification scheme for the variantsâŁ™ phenotypes using four categories (dashed lines in Figure 5.3), labeled as "WT-like" (>80% of wild type CAT activity), "Mild" (50-80%), "Severe" (20â^'50%), and "Disruptive" (0â^'20%). We then averaged the scores to obtain a position specific disruption index. Position-based averaging reduces some of the natural variability of the biological assay, and the method has been reliable in identifying the most sensitive positions at the helix-helix binding interface[**??**]. The classification data is schematically represented in Figure 5.4a.

When the average disruption is projected on a helical wheel diagram (Figure 5.4b), it becomes evident that the sensitive mutations cluster on one helical face defined by positions T5, L6, L8, L9, L12, L15, Q16, L19 and W20. When the average disruption is fit to a sine function to analyze its periodicity (Figure 5.4c), we obtained a value of 3.5 amino acids per turn, which suggests that the helices of the FtsB oligomer interact with a left-handed crossing angle. Interestingly, the variants with enhanced CAT activity (A10, W14, Y17, W20, and F21) are primarily located on the opposite face relative to the disruptive positions. Therefore, it seems unlikely that these variants enhance stability by direct participation to the interaction interface.

A similar mutagenesis analysis for FtsL was performed. While a number of mutations that appear to be disruptive were identified, the disruption pattern does not clearly map to a helical interface as in the case of FtsB, indicating that that the weak self-association of FtsL observed in TOXCAT is unlikely to be specific in nature.

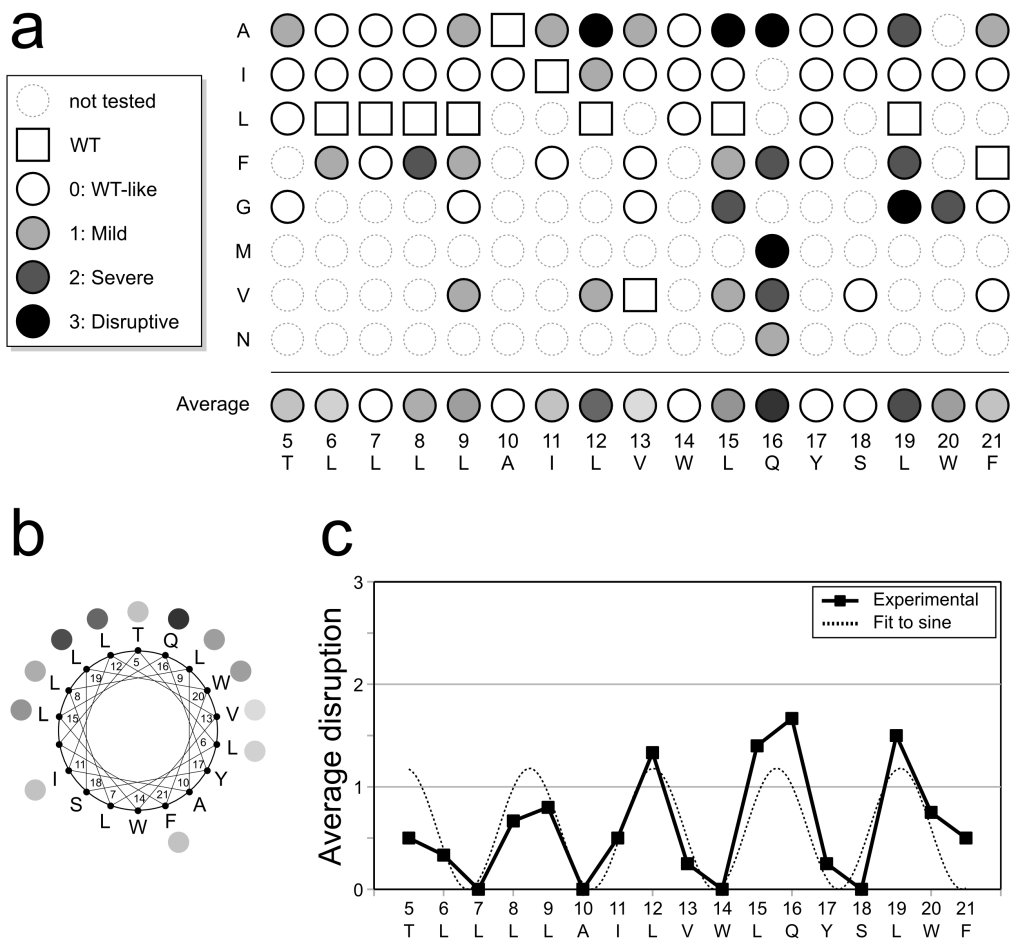


Figure 5.4: *Position specific "average disruption" identifies a helical interface and an essential polar residue. (a) The scheme summarizes the effect of all mutations of FtsB-TM measured in TOXCAT. The data have been categorized as explained in Figure 5.2. An average disruption score is displayed at the bottom of the scheme. While* Gln *16 is the most sensitive position, the introduction of an* Asn *side chain restores association almost entirely, indicating that a hydrogen bond is important for the association. (b) Diagram mapping the average disruption score to a helical wheel. The disruption pattern clusters on one helical face defined by positions T5, L6, L8, L9, L12, L15, Q16, L19 and W20. (c) Fit of the average disruption index to a sine function. The estimated periodicity is approximately 3.5 amino acid per turn, which corresponds to a helical interaction with a left-handed crossing angle (dotted line).*

Interhelical hydrogen bonding of Gln 16 mediates FtsB self-association

Among the positions of the TM domain of FtsB that are sensitive to mutation, Gln 16 is of particular interest. Polar amino acids, such as Gln, Asn, Glu, Asp, Lys, Arg, and His, are not frequent in TM domains, which are primarily composed by hydrophobic residues [**??**]. When present, however, polar residues can stabilize the association of TM helices through the formation of hydrogen bonds, which are enhanced in an apolar environment [**??**]. While the energetic contribution of hydrogen bonding to membrane protein folding appears to be on average rather modest (around 1 kcal mol^{-1}), [**??**] polar amino acids can be important for the association of model peptides [**??**] and of biological systems [**???**]. When present, polar amino acids are also likely to play an important structural or functional role, and it has been observed that phenotypic alterations and diseases are likely to result from mutations that reverse the polarity of an amino acid in membrane proteins.[**??**].

When Gln 16 is substituted by hydrophobic amino acids (Ala, Phe, and Val), the oligomerization of FtsB appears to be severely reduced (Figures 3 and 4). Even when the Gln was replaced by a nonpolar amino acid with similar size and flexibility (Met) the CAT activity decreased to 22% of WT. Conversely, when position 16 is substituted by Asn, which has the same amide terminal moiety of Gln, the variant retains most of the activity (67%). This result confirms that hydrogen bond formation is likely to play a major role in stabilizing the TM oligomer. Two side chains that could potentially hydrogen bond with Gln 16 across the interface are Tyr 17 and Ser 18. However, the removal of their hydroxyl groups (Y17F and S18A variants) did not appear to reduce oligomerization. This observation suggests that Gln 16 is likely to donate to a carbonyl oxygen atom from the backbone or to form a hydrogen bond with itself (from the opposing helix), a hypothesis that we structurally investigated using computational modeling, as presented in the section 5.3.

Gln 16 and the interfacial amino acids of the FtsB TM domain of FtsB are evolutionarily conserved

To investigate if the interfacial amino acids, and Gln 16 in particular, are evolutionarily important, we performed a multisequence alignment of related FtsB sequences obtained using BLAST [Altschul et al. (1990)] and computed a consensus. A condensed version of the alignment is shown in Figure 5.5. The TM region of FtsB appears to be relatively well conserved across a broad group of gamma and beta proteobacteria. Most importantly, the pattern of conservation (shaded columns) corresponds remarkably well to the positions that have the highest sensitivity to mutagenesis (indicated by a dot). The average amino acid identity conservation of the sensitive positions is 68%, compared to 42% of the other positions. The most conserved positions is Trp 20, which is found in over 95% of the sequences. The key Gln 16 is also almost invariable (91% of the sequences). Interestingly, its most frequent substitution is His (4%), another polar amino acid. These observation supports the hypothesis that the structural organization of the TM domain of FtsB is evolutionarily conserved and therefore must be of biological importance for cell division.

The X-ray crystal structure of the periplasmic region of FtsB reveals a canonical coiled coil

After determining the organization of the TM domain of FtsB, we were interested in establishing if the periplasmic coiled coil region is also compatible with the formation of a homodimer. We approached this question by using X-ray crystallography. Unlike TM helices, which are stabilized by the hydrophobic environment, the soluble coiled coils tend to be unstable in isolation [**?**]. For this reason we adopted a fusion strategy, replacing the TM region with a soluble globular protein (bacteriophange ϕ29 Gp7) which nucleates the helix and stabilizes the coiled coil. This strategy has been

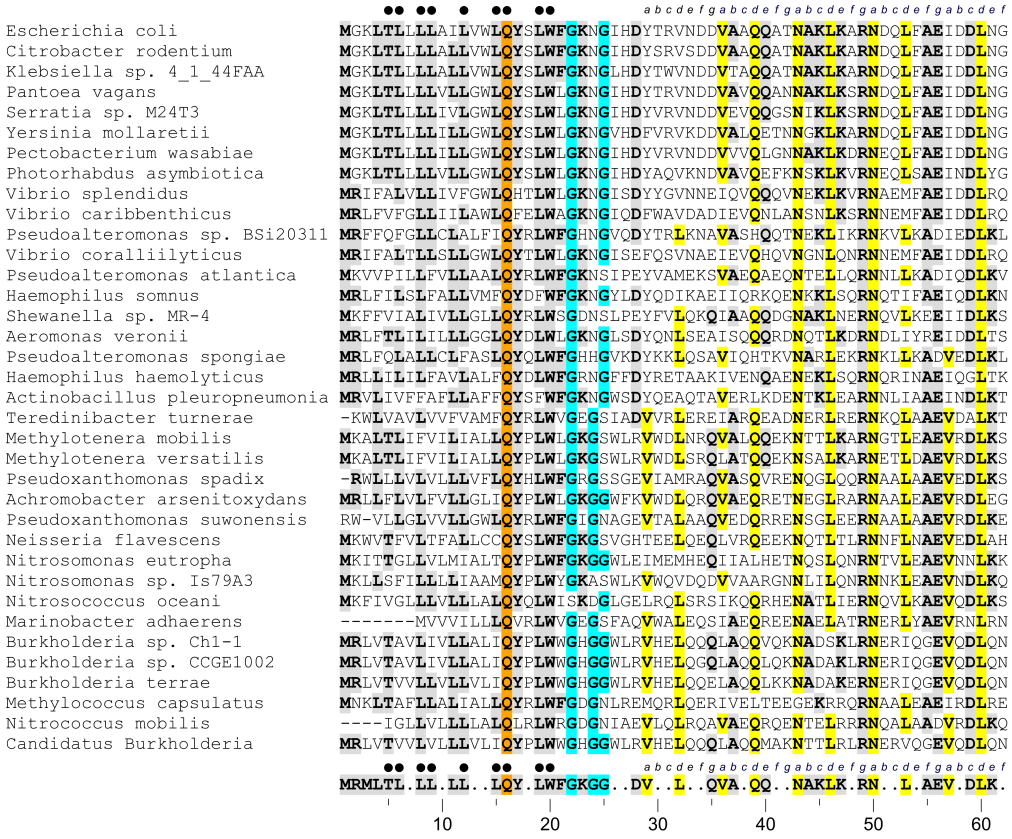


Figure 5.5: *Sequence alignment of FtsB indicates that the interfacial positions are evolutionary conserved. Partial representation of a sequence alignment of FtsB. FtsB is relatively well conserved among a diverse group of beta and gamma proteobacteria. The amino acids that are present in at least 30% of sequence at each position are shown in bold and shaded. These amino acids are also highlighted in the consensus sequence at the bottom of the alignment. The positions that are involved at the FtsB dimer interface (Figure 5.8) are marked with a full circle (•). A remarkable match between conservation and the interfacial positions is evident. In particular, positions Q16 (highlighted in orange), L20 and W21 are almost invariable. The heptad repeat designation (positions a to g) of the coiled coil region is also given, and the conserved amino acids at the interfacial a and d positions are highlighted in yellow. In cyan are highlighted three conserved Gly amino acids (positions 22, 24, and 25) that are likely to confer flexibility to the linker region between the transmembrane domain and the coiled coil region.*

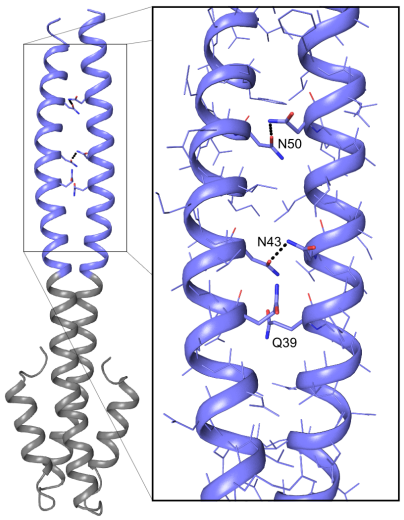


Figure 5.6: *X-ray crystal structure of a Gp7-FtsB$_{CC}$ fusion protein. Ribbon representation of one of the two dimeric molecules in the asymmetric unit. This molecule forms a straight canonical coiled coil. The second molecule in the asymmetric unit exhibits a slight bend, possibly as a result of crystal packing. The N-terminal Gp7 unit which replaces the transmembrane domain is highlighted in gray, and the FtsB sequence is in blue. The inset highlights a number of polar amino acids that are present at the interface in "d" (Q39) and "a" (N43 and N50) positions.*

demonstrated to greatly improve the solubility and crystallization propensity of coiled coil domains [**??**].

A Gp7 fusion construct encompassing amino acids 28-63 of FtsB (Gp7-FtsB$_{CC}$) crystallized readily, and its structure was solved at a 2.3 Å, with two dimeric molecules in the asymmetric unit. The structure of the Gp7-FtsB$_{CC}$ dimer is shown in Figure 5.6, where the Gp7 moiety is highlighted in gray and the FtsB component in blue. This region of FtsB adopts a canonical coiled coil conformation. As expected, the two Asn residues that are present at "a" heptad positions (Asn 43 and 50) form a hydrogen bond across the interface with their corresponding residues on the other chain (Figure 5.6b). The coil is straight for one of the dimers (chains A-B), but it

exhibits a slight kink in the second (near Val 36 on chain C-D). The overall RMSD between the two dimers is 1.74 Å, but it decreases to 0.41 Å and 0.81 Å when the pre- and post-kink segments are aligned separately; therefore, the kink is presumably due to the effect of crystal packing.

The structure demonstrates that, like the TM domain, the coiled coil region of FtsB can also assume a homodimeric form. The structure also determines that the coiled coil region of FtsB can extend at least to position 60. It is not clear whether the coiled coil would extend further, at least in the absence of FtsL. Gonzales and Beckwith determined that a C-terminal truncation of FtsB starting position 55 is still sufficient to interact with FtsL [**?**]. Circular dichroism (CD) analysis of a longer constructs that encompassed seven heptad repeats (positions 28-77) revealed that it is poorly helical. A Gp7-FtsB$_{\mathrm{CC}}$ construct that contains the entire soluble region of FtsB is also only moderately helical.

Computational model of a FtsB left-handed homodimer

Molecular modeling can interpret the wealth of information contained in large-scale mutagenesis and synthesize it into an often highly accurate structural hypothesis [**???**]. The modeling of the TM domain of FtsB was performed with a search protocol implemented with the molecular software library developed in this laboratory (MSL)[Kulp et al. (2012)]. The program generates helices in standard conformation and systematically varies their relative orientation to explore conformational space. In this calculation, we imposed the formation of a symmetrical oligomer. Consistently with the experimental data, we also required that Gln 16 forms an interhelical hydrogen bond in the structure. The calculation produced two well packed dimeric low-energy solutions (Figure 5.8). In one solution (Model 1,panel a), Gln 16 is hydrogen bonded nonsymmetrically with Gln 16 on the opposite side. In Model 2 (panel b), the side chain is hydrogen bonded

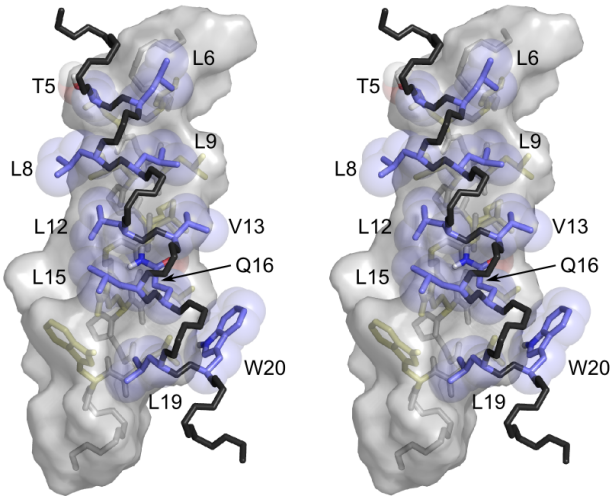


Figure 5.7: *Computational model of FtsB-TM (Model 1). Stereo representation of the side chain interactions across the interface. The interacting positions on the opposite chains are shown in sticks to highlight the arrangement and packing of the side chains. The van der Waals sphere of the side chains of the monomer in the foreground (blue) and the surface of the monomer in the background are also displayed with transparency. L15 and L19 interact against a ridge formed by W20. L12 and V13 pack against each other across the interface, and so do the L8/L9 and the T5/L6 pairs.*

symmetrically to the carbonyl oxygen of Val 13. The two models are closely related (1.5 Å RMSD), having a similar left-handed crossing angle and interhelical distance, and differing by a relative rotation of approximately 60° applied around the helical axis. To identify which solution was most compatible with the experimental data, we applied in silico the same set of mutations that were experimentally tested and computed an analogous average disruption. The theoretical and experimental disruption patterns are compared in Figure 5.8 c,d. Model 2 is in reasonable agreement with the data overall, but its periodicity appears to be slightly off-phase with respect to the experimental data, and the match becomes poor toward the C-terminal end of the helix (panel d). Model 1 (panel c) is in excellent

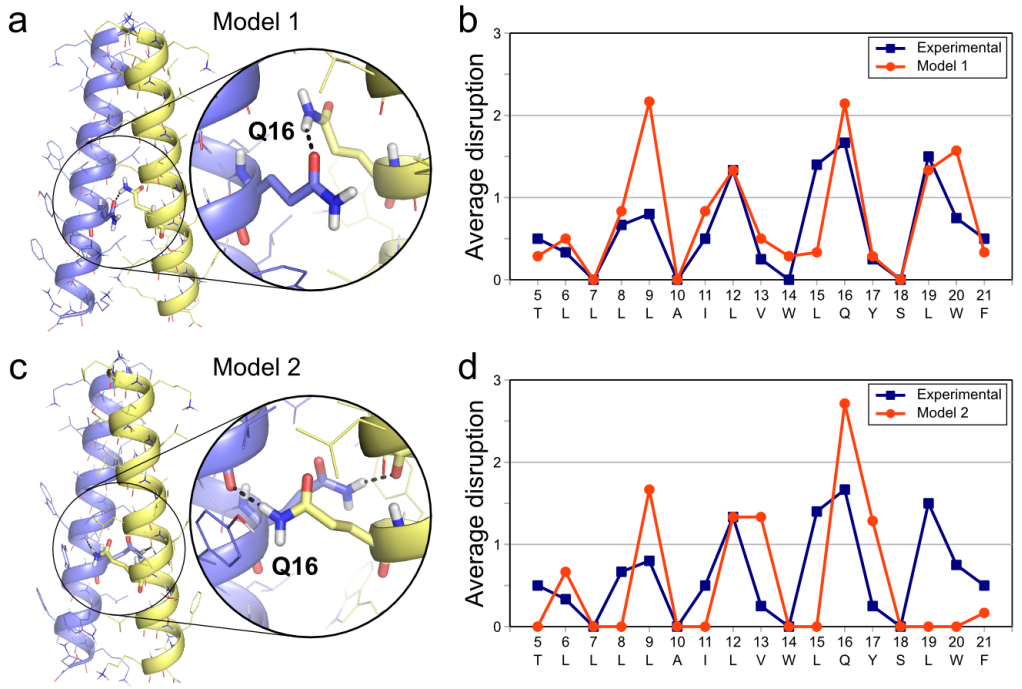


Figure 5.8: *Molecular model of the FtsB transmembrane dimer. Modeling identified two well packed low-energy structures in which* Gln *16 forms an interhelical hydrogen bond (panels a and c). Models 1 and 2 are closely related (Cα RMSD of 1.5* Å*), with a left-handed crossing angle (25° and 20°,respectively) and an interhelical distance of 10.1* Å*. In Model 1 the side chains of* Gln *16 interact with a nonsymmetrical hydrogen bond. The helices of Model 2 are rotated axially by about 60° with respect to Model 1. The side chains of* Gln *16 interact symmetically with the carbonyl oxygen of* Val *13. Panels b and d compare the average disruption index for the computational mutagenesis applied to the models to the average disruption observed experimentally. Model 1 shows an excellent agreement with the experimental data, which is better than Model 2, particularly in the C-terminal side of the transmembrane domain.*

agreement with the experimental data, and therefore we propose it as the most likely structural interpretation. Model 1 is illustrated in more detail in Figure 5.7 where the specific orientation of the side chains at the dimer interface is shown and the contacts are described. A PDB file of the two models can be found as downloaded from http://seneslab.org/FtsBdimer.

Flexibility may be important between the TM and coiled coil regions of FtsB

There is a gap of six amino acids (positions 22-27) between the computational model of the TM domain and the x-ray structural model of the periplasmic coiled coil, raising the question of how these two regions are connected. The simplest hypothesis would be that the two domains form a seamless helical structure that transverses the TM region and extends into the periplasm. Our geometric analysis, however, revealed that the two models cannot be connected by a simple fusion of their helices. While the crossing angle and interhelical distance of the two domains match each other, the orientation of the helices around their main axes is not compatible. The interface of the TM region is rotated around the helical axis by approximately 100° with respect to the interface that would result from a natural extension of the coiled coil.

The analysis of the sequence alignment (Figure 5.5) also supports the hypothesis that a helical break is likely present in the linker region between the domains. The alignment reveals that two Gly residues at positions 22 and 25 are highly conserved (highlighted cyan). Gly 22 is present in 93% of the sequences. Gly 25 is less ubiquitous (63% of the sequences), but a third Gly is frequently present at position 24 (47%). Interestingly, the other amino acids that are prevalent at positions 24 and 25 are Ser and Asn, two residues that have a relatively high propensity for random-coil regions [**?**]. The data suggest that the linker requires either flexibility or adopts a backbone conformation that would be inaccessible to non-Gly amino acids,

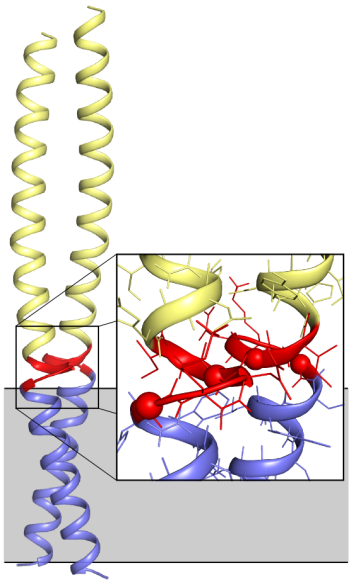


Figure 5.9: *A theoretical model of a FtsB dimer that encompasses the transmembrane and coiled coil domains. The crystal structure of the coiled coil region of FtsB (yellow) and the computational Model 1 of the transmembrane domain (blue) were stitched together using a fragment based approach (see Materials and Methods). The resulting theoretical model includes a hinge between the coiled coil and the transmembrane helix where the helix unfolds (red). This hinge corresponds to conserved a Gly rich region in the sequence alignment (G21 and G25, spheres), suggesting that a flexible connection may be functionally important.*

or perhaps both. According to this view, we mined the structural database for protein fragments that contained two Gly residues with the correct spacing (GxxG) to find candidate linkers for the two models. We extracted all xGxxGx fragments existing in high-resolution structures from the PDB, where x is any amino acid. We also imposed a constraint that two additional residues at each side of the fragment must assume a helical conformation (thus the pattern becomes hhxGxxGxhh, where h is any amino acid in helical conformation). These helical amino acids were geometrically aligned with the ends of the TM and coiled coil structure. With this procedure, we

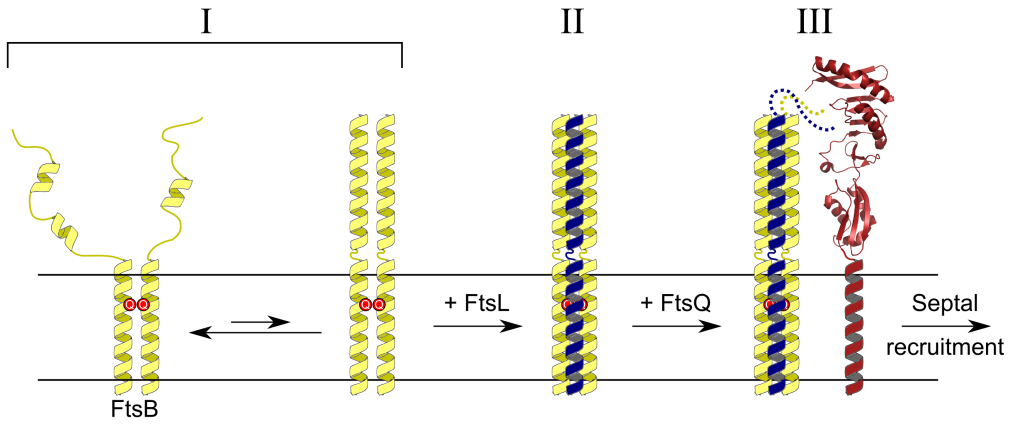


Figure 5.10: *A functional hypothesis for the formation of the FtsB/FtsL complex and its recruitment to the divisome. The transmembrane domain of FtsB self-associates in E. coli membranes, driven by an interhelical hydrogen bond (Gln 16, represented by a red circle), but the coiled coil region is likely to be marginally stable or unstable (I). This finding raises the hypothesis that the interaction with FtsL is required to stabilize the periplasmic domain. It is likely that FtsL (blue) laterally associates with a pre-existing FtsB dimer (II). Alternatively, FtsL may compete with the self-association of FtsB to form an FtsB/FtsL heterodimer (not represented). Once the periplasmic domain is folded, the C-terminal tails of the FtsB/FtsL complex (dotted lines) would bind to FtsQ (red), and the proteins would subsequently be recruited to the division septum (III).*

were able to identify the low-energy solution that connects the two models illustrated in Figure 5.9.

Is FtsL required to stabilize the periplasmic domain of FtsB?

While the model of the linker region in Figure 5.9 is hypothetical, it raises the question of whether the gly-rich segment could effectively nucleate and stabilize the juxtamembrane coiled coil. The question is even more compelling when it is considered that the Gp7-FtsB$_{CC}$ construct has low thermal stability. Although the fusion protein crystallizes readily and is

helical at low temperature, it reversibly unfolds quite rapidly, and it appears to be completely unfolded at 40 °C. Longer constructs, including one that extends to the entire soluble region of FtsB, showed lower helicity and even lower stability. The relatively low stability of the coiled coil, however, is not surprising when it is considered that the structure includes a large number of polar amino acids (Q35, N43, N50) at the buried "a" and "d" positions, which are generally occupied by hydrophobic amino acids [**??**]. These sequence features appear conserved in the sequence alignment (Figure 5.5). Therefore, it is possible that association with FtsL may be required for the stabilization of the periplasmic region of FtsB. The fact that the periplasmic domain of FtsB may be partially unfolded could also account for some of the cellular instability of FtsB, which is rapidly degraded in the absence of FtsL [**?**].

On the basis of our analysis, we hypothesize that a FtsB transmembrane homodimer forms an initial core that laterally recruits FtsL into a higher-order oligomer (Figure 5.10), likely a tetramer as proposed also by a recent bioinformatic analysis of the soluble domains [**?**]. Given the presence of several Thr residues in the TM helix of FtsL, an interesting possibility is that its lateral association could augment the membrane embedded polar network by forming additional hydrogen bonds with the donor or acceptor groups that are left unsatisfied on Gln 16 (Figure 5.8a). The formation of the heterologous complex and the folding of the periplasmic domains may be a determinant for making FtsB competent for binding to the periplasmic domain of FtsQ [**?**], which is required for their septal localization and the recruitment of the late proteins.

5.3 Methods

Computational modeling of the ftsB transmembrane dimer

The transmembrane oligomer of FtsB was modeled with programs written in house and distributed with the MSL molecular modeling libraries v.1.1 [Kulp et al. (2012)] available at http://msl-libraries.org. The *predictHelixOligomer* program creates standard helices and performs a global rigid search altering the interhelical separation, the crossing angle, the crossing point, and the axial orientation of the helices. To impose the formation of an interhelical hydrogen bond involving Gln 16, the conformational space was prescreened prior to the analysis to exclude the region of space that was incompatible with the program *filterOligomerByConstraint*. This was performed on helices in which all amino acids were converted to Ala except Gln 16, Tyr 17, and Ser 18. The backbone was kept rigid during the procedure, while the side chains were optimized using a greedy trials method implemented in MSL [Kulp et al. (2012); **?**]. Side chain mobility was modeled using the energy-based conformer library applied at the 90% level [**?**]. The models were evaluated using a van der Waals function with CHARMM 22 parameters and the SCWRL hydrogen bond function implemented in MSL. The models were sorted by their energies. All low-energy models were visually inspected to verify that they did not include poorly packed solutions containing cavities. The computational mutagenesis was performed on all low-energy models by applying the same mutation studied experimentally in the context of a fixed backbone, followed by side chain optimization. The relative energy of each mutant was calculated as

$$\delta\ E_{mut} = (E_{mut,dimer} - E_{mut,monomer}) - (E_{WT,dimer} - E_{WT,monomer})$$

where $E_{WT,dimer}$ and $E_{mut,dimer}$ are the energies of the wild type and mutant sequence in the dimeric state, and $E_{WT,monomer}$ and $E_{mut,monomer}$ are the energies of the wild type and mutant sequence in a side chain

optimized monomeric state with the same sequence. The effect of each mutation was classified in four categories analogously to the experimental mutagenesis using the following criterion:

- category 0, "WT-like", $\delta\ E_{mut} < 2$ kcal mol^{-1};
- category 1, "Mild", $2 \leqslant \delta\ E_{mut} < 4$;
- category 2, "Severe", $4 \leqslant \delta\ E_{mut} < 8$;
- category 3, "Disruptive",$\delta\ E_{mut} \geqslant 8$.

The numerical category values were averaged to calculate the position dependent average disruption value reported in Figure 5.8.

Creation of TM + coiled coil model

The computational model of the TM domain and the coiled coil region were connected together using fragments from the PDB database. To do this, protein fragments of the pattern hhxGxxGxhh (where x is any amino acid, and h is any amino acid in a helical conformation) were extracted from high resolution X-ray structures deposited in the PDB database with a resolution of 2 Å or better. The MSL program *connectWithFragments* takes these fragments and aligns the helical end residues with the corresponding residues in the coiled coil domain and then the modeled TM domain. Only the N, C, CA, and O atoms were considered for the alignment and the fragments with the lowest R.M.S.D. were selected. The side chains on the fragment were replaced with the one corresponding to the FtsB sequence and their conformation was optimized using a greedy trials method.

FtsB sequence alignment and consensus sequence

The alignment was obtained by entering the sequence of E. coli FtsB as the query in BLAST (http://blast.ncbi.nlm.nih.gov) using the blastp algorithm

with default settings. The resulting 464 sequences were aligned with the multiple alignment facility in BLAST (COBALT). The prevalent amino acid at each position in the sequence was used to determine a consensus sequence if it was present in at least 30% of the sequences.

Accession numbers

Coordinates and structure factors of Gp7-FtsB have been deposited in the Protein Data Bank with PDB ID code 4IFF.

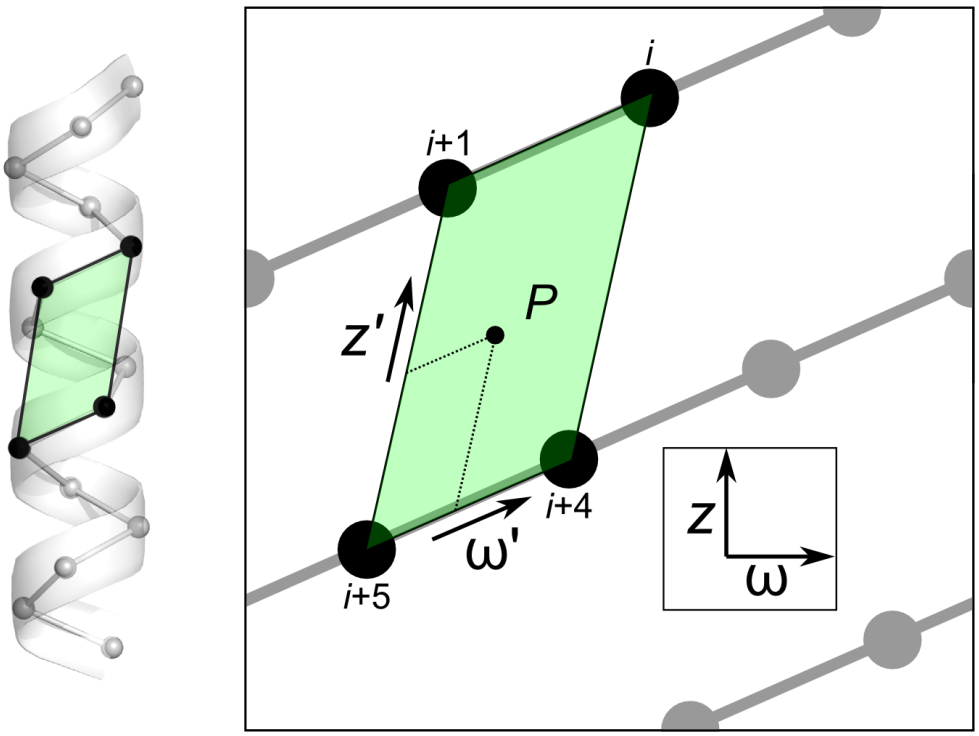
5.4 Conclusions

In this chapter, the first structural analysis of an integral membrane protein of the bacterial divisome was presented. We demonstrated that the TM helix of FtsB self-associates in E. coli membranes. The interaction is mediated by an interhelical hydrogen bond formed by a critical polar residue embedded in the middle of the hydrophobic region. We also report the structure of the juxta-membrane domain of FtsB which forms a canonical coiled coil. The two domains are connected by a linker that is likely flexible. While the present study does not experimentally establish the precise oligomeric state of FtsB directly, the mutagenesis, modeling, and crystallographic data are consistent with the formation of a homodimer.

By defining the protein-protein interaction interface of FtsB and providing an experimentally validated structural model, the present work suggests the hypothesis that FtsB and FtsL assemble into a higher-order oligomer and sets the stage for the biophysical analysis of their heterologous complex. This study also establishes important groundwork for biological studies in vivo that will address whether the self-association of FtsB is essential for division and, specifically, whether the structural features identified here - the TM interaction interface, the Gly-rich linker and the stability of the coiled coil - are important for the localization of FtsB, for its association

with FtsL, and for the recruitment of the other downstream proteins to the divisome.

6 HIGH-THROUGHPUT *ab initio* PREDICTION OF TRANSMEMBRANE DIMERS



based on

Mueller BK*, **Subramaniam S*** and Senes A "The frequent GAS_{right} transmembrane motif is optimized for Cα hydrogen bonding" *Proceedings of the National Academy of Sciences (PNAS)* 2014

*This work was completed with equal contributions from myself and Benjamin K Mueller.

Summary

Carbon hydrogen bonds between Cα-H donors and carbonyl acceptors are frequently observed between transmembrane helices. Networks of these interactions often occur at helix-helix interfaces mediated by *GxxxG* and similar sequence patterns. Cα-H hydrogen bonds have been hypothesized to be important in membrane protein folding and association, but evidence that they are major determinants of helix association is still lacking. This chapter details a comprehensive geometric analysis of homo-dimeric helices which reveals the existence of a single region in conformational space with high propensity for Cα-H...O=C hydrogen bond formation. This region corresponds to the most frequent motif for parallel dimers, the $\mathrm{GAS}_{\mathtt{right}}$, whose best known example is glycophorin A. The finding suggests a causal link between the high frequency of occurrence of $\mathrm{GAS}_{\mathtt{right}}$ and its propensity for carbon hydrogen bond formation. Investigation of the sequence-dependency of the motif determined that *Gly* residues are required at specific positions where only Gly can act as a donor with its "side chain" Hα. Gly also reduces the steric barrier for non-Gly amino acids at other positions to act as Cα donors, promoting the formation of cooperative hydrogen bonding networks. These findings offer a structural rationale for the occurrence of the *GxxxG* patterns at the $\mathrm{GAS}_{\mathtt{right}}$ interface. The analysis identified the conformational space and the sequence requirement of Cα-H...O=C mediated motifs; we took advantage of these results to develop a structural prediction method. The resulting program, CATM, predicts *ab initio* the known high-resolution structures of homo-dimeric $\mathrm{GAS}_{\mathtt{right}}$ motifs at near atomic level.

6.1 Introduction

The transmembrane helices of single-span membrane proteins are commonly engaged in oligomeric interactions that are essential for structure and func-

tion. These interactions often occur in the form of recurrent structural motifs. Here we present an analysis of one of the most important motifs (GAS_{right}), showing that its geometry is optimized to form carbon hydrogen bonds at the helix-helix interface. The analysis reveals the structural basis for its characteristic *GxxxG* sequence signature. We built upon the analysis, creating a method that predicts known GAS_{right} structures at near atomic precision. The work has implications for understanding membrane protein association, and for the prediction of unknown interacting GAS_{right} dimers among the thousands of single-span proteins in the proteomes of humans and other higher organisms.

The transmembrane (TM) domains of membrane proteins that span the bilayer with a single helix are commonly engaged in oligomeric interactions that are essential for the structure and function of these proteins [**?**]. The interaction between these TM helices are often mediated by recurrent structural motifs, which are characterized by specific geometries and display sequence signatures in the form of specific amino acid patterns [**?**]. In this work we present a geometric analysis of one of the most important structural motifs, and implement a method for its structural prediction. The primary feature of this motif is the presence of inter-helical carbon hydrogen bonds that occur across the helix-helix interface between Cα-H donors and backbone carbonyl oxygen acceptors (Cα-H...O=C bonds) [**?**]. The sequence "signature" is the occurrence of glycine and other small amino acids (*Ala, Ser*) at the helix-helix interaction interface, generally spaced at i, i+4 to form patterns such as *GxxxG, AxxxG, GxxxA*, etc. [**?**]. These small amino acids are important to reduce the steric barrier for bringing the backbones of the opposing helices in close proximity, allowing the Cα and carbonyl oxygen (two backbone atoms) to come in contact and form hydrogen bonds [**?**].

While Cα-H...O=C hydrogen bonds can be observed in right- and left-handed TM helical pairs and in both parallel and anti-parallel orientations,

they are most frequently associated with parallel right-handed pairs with a crossing angle around -40° [**?**]. This structural motif has been named GAS_{right} by Walters and DeGrado, from its sequence signature (*Gly, Ala, Ser*) and its crossing angle [**?**]. GAS_{right} is the most frequent motif for pairs of parallel helices and it appears to be extremely frequent in 2-fold symmetrical homo-dimers of single-pass proteins. Indeed, out of approximately a dozen high-resolution TM homo-dimers solved to date as many as five are representatives of the GAS_{right} motif [**?**]. However, whether the Cα hydrogen bonds indeed represent a major stabilizing force in GAS_{right} motifs is yet to be demonstrated.

The carbon-hydrogen bonds (Cα...C=O)

Carbon hydrogen bonds are commonly observed in proteins and nucleic acids, where they can contribute to protein structure, recognition or catalysis [**?**]. While carbons are generally weak donors, the Cα atom of all amino acids is activated by the electron withdrawing amide groups on both sides, and quantum mechanics calculations suggest that the energy of Cα-H hydrogen bonds may be as much as one third to half of that of canonical donors in vacuum [**?**, **?**]. Carbon hydrogen bonds have been proposed to be particularly important in membrane proteins, the membrane being a low dielectric environment that, in principle, should enhance their strength [**?**]. However, obtaining an experimental measurement of their contribution remains difficult. To date, two groups have addressed this question experimentally, with differing results. Arbely and Arkin calculated a favorable contribution of -0.88 kcal mol^{-1} for the carbon hydrogen bond formed by Gly 79 in glycophorin A, using isotope-edited IR spectroscopy [**?**]. Conversely, Bowie and coworkers found that a Cα-H...O bond to the side chain hydroxyl group of Thr 24 was only marginally stabilizing or even slightly destabilizing in a folding study of bacteriorhodopsin variants [**?**]. Mottamal and Lazaridis were able to reconcile this discrepancy by analyzing the different hydrogen bonding

geometries of the two systems [**?**]. Further, quantum mechanical calculations performed on geometries from protein crystal structures also suggested that indeed the orientation of the groups can determine whether an interaction may be strongly favorable or unfavorable [**?**].

More studies are certainly needed to fully understand the energetic contribution of Cα hydrogen bonds in membrane protein folding and interaction. However, their common occurrence as structural elements in membrane proteins postulates that they play an important role [**??**]. To further investigate this issue, we present an analysis of the propensity for Cα hydrogen bond formation as a function of helical geometry in symmetric homo-dimers. Remarkably, the analysis reveals the existence of a single high-propensity conformation that corresponds to the common $\mathrm{GAS}_{\mathtt{right}}$ motif. By defining a suitable frame of reference for the geometries, we were able to investigate the specific sequence requirements of each position at the helix-helix interface. The results rationalize the occurrence of *GxxxG* patterns in $\mathrm{GAS}_{\mathtt{right}}$, and provide a physical explanation for the typical right-handed geometry of the motif based on steric interactions and optimization of hydrogen bonding. Overall, the analysis suggests a strong causal link between the high frequency of occurrence of $\mathrm{GAS}_{\mathtt{right}}$ and its propensity for Cα hydrogen bond formation.

The analysis defines a map of the conformational space that allows the formation of networks of carbon hydrogen bonds between helical dimers. It also identified strict sequence dependencies at specific positions of each individual geometry. Based on this information, we have also created a rapid structural prediction method for the identification of Cα-H...O=C mediated homo-dimers, which we call CATM (Cα TransMembrane). We show that CATM can predict the known high-resolution structures of homo-dimeric $\mathrm{GAS}_{\mathtt{right}}$ motifs at near atomic level. Interestingly and perhaps surprisingly, we found that a minimalistic set of energy functions composed of a hydrogen bonding and a van der Waals function, is sufficient for achieving a highly

accurate level of prediction.

6.2 Results and Discussion

Geometric definition based on the unit cell of the helical lattice

The first step for our geometric analysis was to identify a practical frame of reference to express the relative orientation of the helices, as illustrated in Figure 6.1a. Two parameters are straightforward: the inter-helical distance, d, and the crossing angle, θ. The other two parameters, the axial rotation, ω, and the position of the crossing point along the helical axis, Z, require a reference, such as a specific Cα. We found that it is most intuitive to define the geometry relative to a reference unit cell in the helical lattice (the parallelogram connecting four Cα atoms on the helical face illustrated in Figure 6.1b, and, as a planar projection, in Figure 6.1c). For completeness, we explored conformational space so that the position of the point of closest approach P (i.e. the crossing point) samples the entirety of the unit cell. This is done by expressing Z and ω relative to the helical screw, producing two transformed unit vectors, Z' and ω', that run parallel to the principal components of the unit cell (Figure 6.1c and 6.2). For convenience, we defined a naming convention for the positions that is relative to the reference unit cell. The positions at the four corners were designated as N1, N2, C1 and C2, where "N" and "C" indicate the N- and C-terminal sides of the parallelogram. These four atoms are relatively spaced at i, i+1, i+4 and i+5. The above reference frame and convention greatly helps the analysis and the discussion of the results.

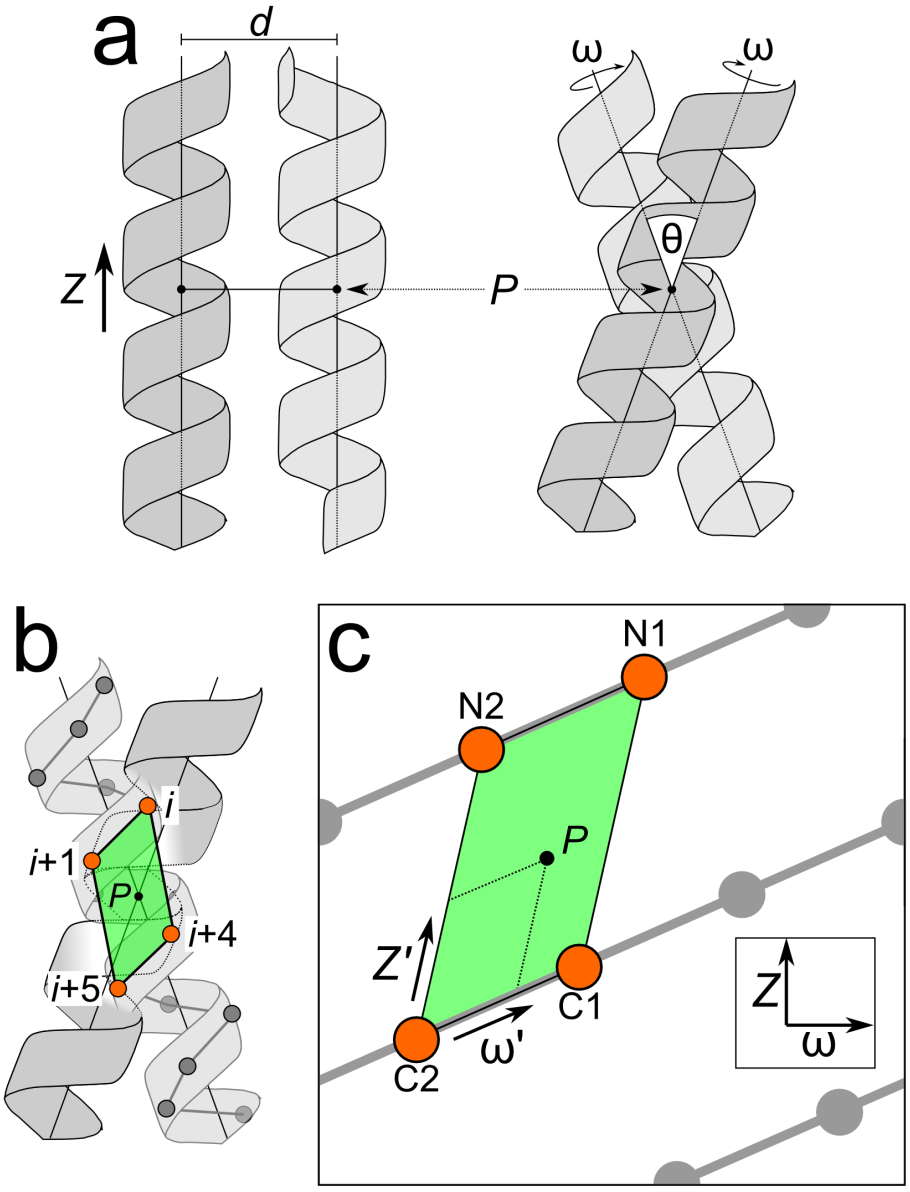


Figure 6.1: *Carbon hydrogen bond formation has preferential regions in inter-helical space. a) Definition of 4 parameters that define the geometry of a symmetrical dimer: the inter-helical distance d; the crossing angle θ; the rotation of the helix around its axis ω; and the vertical position Z of the point of closest approach between the two helical axes (the crossing point P). b) The coordinates can be redefined by expressing them as a function of the unit cell (green) on the helical lattice that contains the point of closest approach P. The four interfacial positions that surround the the point of closest approach are designated as N1 (relative position i), N2 (i+1), C1(i+4) and C2 (i+5). The principal axes are the rotation along the helical screw (ω') and the vector between C2 and C2 (Z'). The mathematical relationship between (ω, Z) and (ω', Z') is provided in Figure 6.2.*

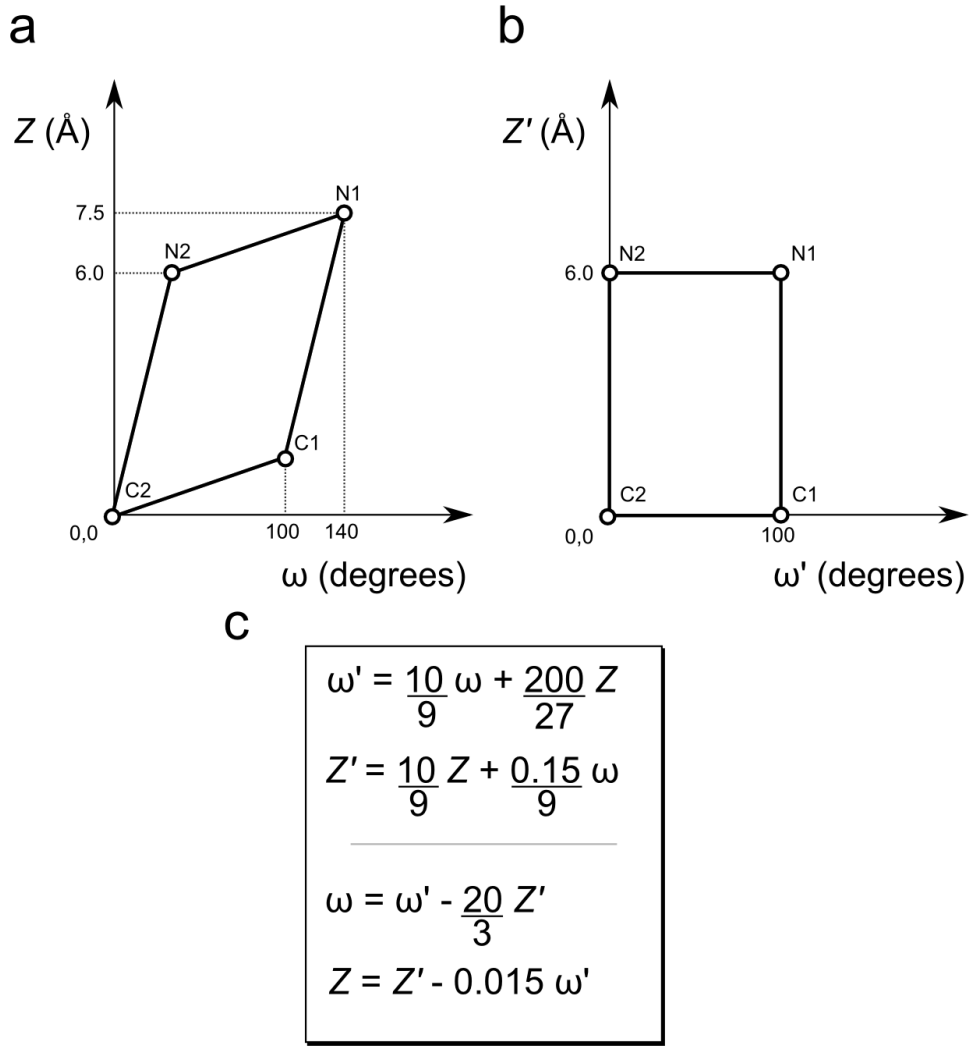


Figure 6.2: *Mathematical definition of Z' and ω' coordinates. a) Unit cell of the helical lattice as with C2 at the origin, as shown in Figure 6.1. In the [ω, Z] set of coordinates C2 is at [0°, 0Å]; C1 is at [100°, 1.5Å]; N2 is at [40°, 6Å]; and N1 at [140°, 7.5Å]. b) The unit vectors ω' and Z' go in the direction of the principal components of the unit cell (C2-C1 and C2-N2, respectively). In the [ω', Z'] set of coordinates C2 is at [0°, 0Å]; C1 is at [100°, 0Å]; N2 is at [0°, 6Å]; and N1 at [100°, 6Å]. c) Mathematical equations for the transformation from one to the other set of coordinates.*

Carbon hydrogen bond analysis reveals a bias for right-handed structures

To investigate the precise geometric requirements for the formation of inter-helical carbon hydrogen bonds, we performed a systematic evaluation of all homo-dimer geometries beginning with poly-Gly. Gly is the only amino acid that doubles the opportunity for hydrogen bond formation by the virtue of having two alpha hydrogens oriented approximately perpendicular to each other (109°) as well as being the residue that permits the two helices to come into the closest proximity. Therefore, poly-Gly is the "best case" sequence for forming carbon hydrogen bond networks, from a geometric stand point.

The hydrogen bonding propensity for each individual geometry was estimated with a hydrogen bonding function borrowed from SCWRL4 [Krivov et al. (2009)] and re-parameterized to include Cα donors. The results are presented as color-coded heat maps in Figure 6.3a. Each graph shows total hydrogen bond energy as a function of axial rotation (ω', on the x-axis) and crossing angle (θ, y-axis) for a different slice in Z'. For simplicity the inter-helical distance d is not explicitly graphed; instead, for each [ω', θ, Z'] point we plot only the energy (E_{min}) at the optimal distance (d_{min}). A larger number of Z' stacks, as well as the corresponding d_{min} values for each point are plotted in Figure 6.4.

A single major high-propensity region is observed in the lower half of the plot, for right-handed crossing angles in the -30° to -50° range. This minimum is situated mid-way between the Cα carbon atoms (C2 and C1) in the ω' dimension, between 40° to 60°. The region persists, with some variation, across the entire range of Z'. Interestingly, the minimum corresponds to the important GAS_{right} structural motif [**?**], a right-handed dimer characterized by presence of *GxxxG*-like patterns at the helix-helix interface [**?**]. Structural examples of GAS_{right} homo-dimers are glycophorin A [**?**] and BNIP3 [**?**], and the motif is also common within the fold of polytopic membrane proteins [**??**].

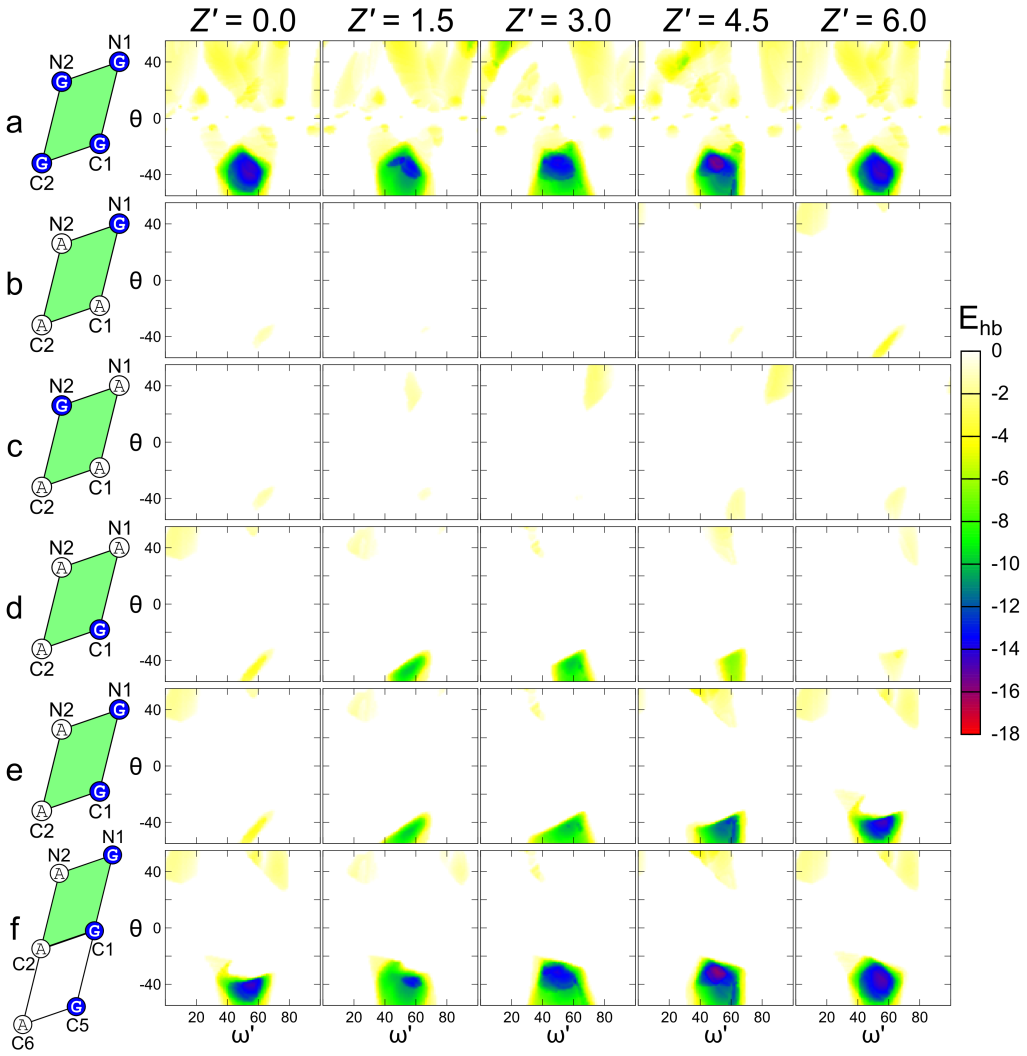


Figure 6.3: *Position C1 must be a Gly for carbon hydrogen bond formation. A map of the carbon hydrogen bonding energy (color bar) as a function of inter-helical geometry (ω': x-axis, θ: y-axis; Z': panels). a) Analysis of poly-Gly: a single broad minimum is observed centered around a region with a right handed crossing angle θ of approximately -30° to -50°. The minimum persists with variation along the entire Z' stack. b, c and d) Poly-Ala sequences with a single Gly at specific positions as indicated on the left-hand side of the figure. The propensity to form hydrogen bonds is almost completely removed compared to poly-Gly unless the amino acid at position C1 is a Gly. (d) e) Introduction of a* GxxxG *motif at the positions N1 and C1 restores some of the low energy regions for higher Z' values. f) When a third Gly is added at C5 the propensity becomes very similar to poly-Gly. In each panel the lowest energy (vdw + hbond) across all inter-helical distances (E_{min} at d_{min}) is plotted for each point.*

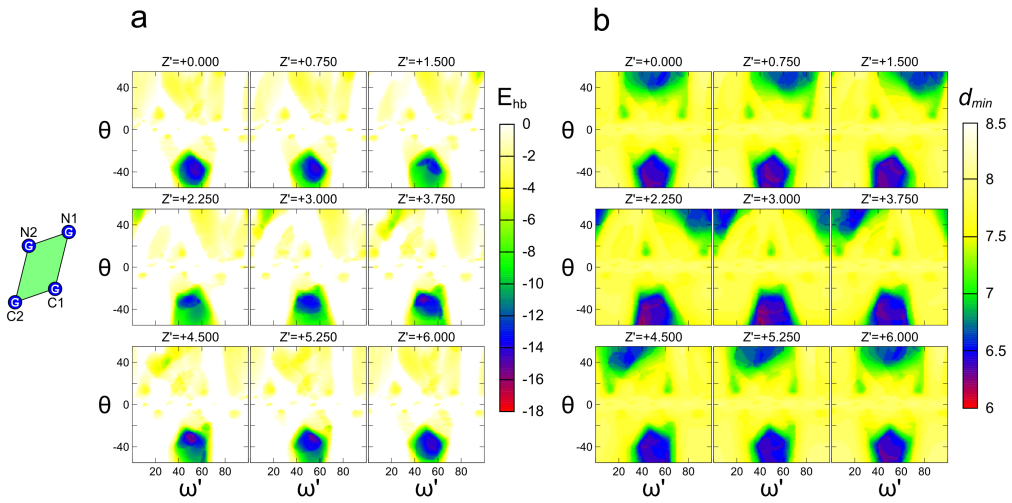


Figure 6.4: *Hydrogen bonding energies and dmin values of poly-Gly. a) Extended version of the hydrogen bonding energy maps for poly-Gly as a function of inter-helical geometry (ω': x-axis, θ: y-axis; Z': panels) for poly-Gly, as in Figure 6.3a. Only the minimum energy $E_{\mathtt{min}}$ across the d dimension is reported. b) Plot of the corresponding dmin distances at which the minimum energy was recorded. While the high-propensity region for Cα hydrogen bonding for right-handed structures display short $d_{\mathtt{min}}$ distances (<7.5Å), the plot demonstrates that other short distance conformations exist that do not lead to strong Cα hydrogen bond network formation.*

6.3 $\mathrm{GAS}_{\mathtt{right}}$ homo-dimeric motifs require a Gly at position C1

To investigate the sequence requirements for carbon hydrogen bonding and to understand the role of *GxxxG* like patterns in $\mathrm{GAS}_{\mathtt{right}}$ motifs, we expanded the geometric analysis to poly-Ala helices in which one or more Gly were inserted in the sequence at specific positions. The poly-Ala sequence has minimal propensity to form hydrogen bonds (Figure 6.5a) but when a single Gly is placed at C1, a significant restoration of the energies is observed for Z' values between 1.5 to 4.5 Å, that is, for dimers that have the point of

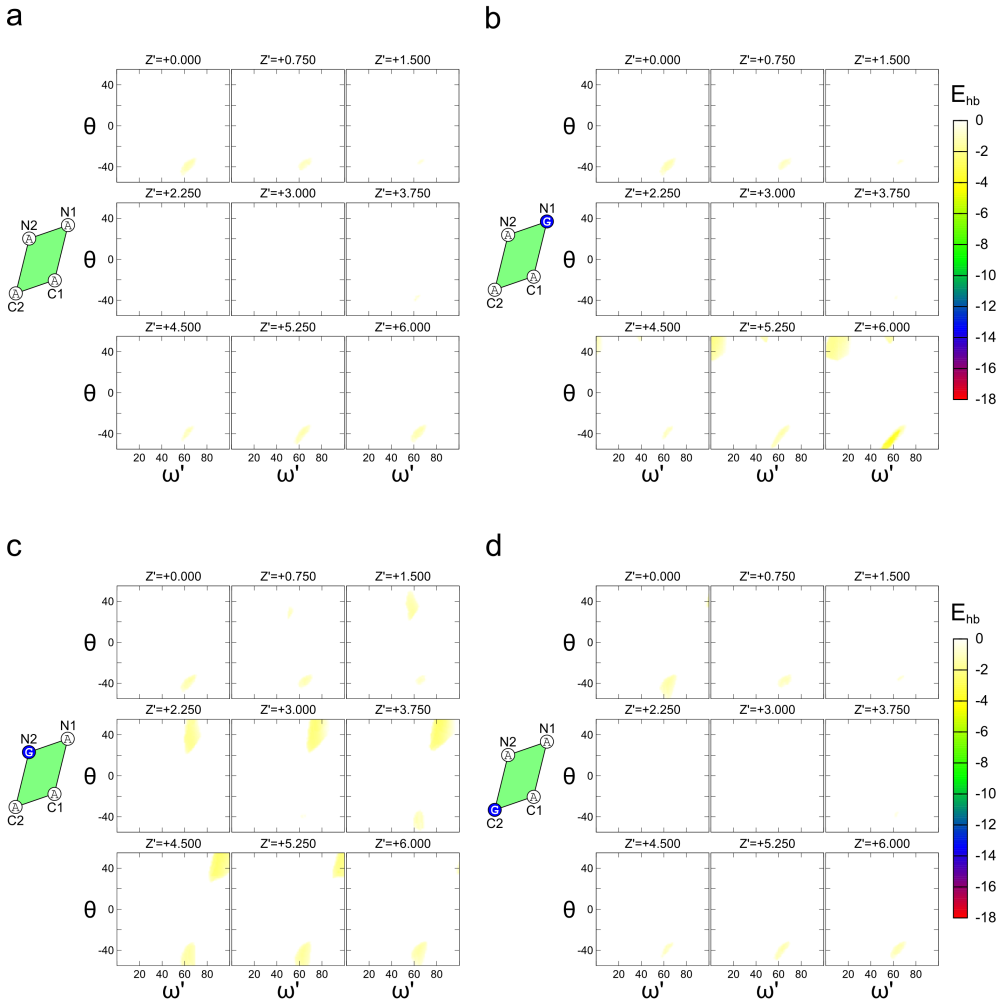


Figure 6.5: *Gly at N1, N2 and C2 in a poly-Ala background does not restore hydrogen bond propensity. Extended version of the hydrogen bonding energy maps for poly-Ala sequences with a single Gly at positions other than C1. a) Poly-Ala with no Gly. b) Gly at N1 as in Figure 6.3b. c) Gly at N2 as in Figure 6.3c. d) Gly at C2 (not shown in Figure 6.3).*

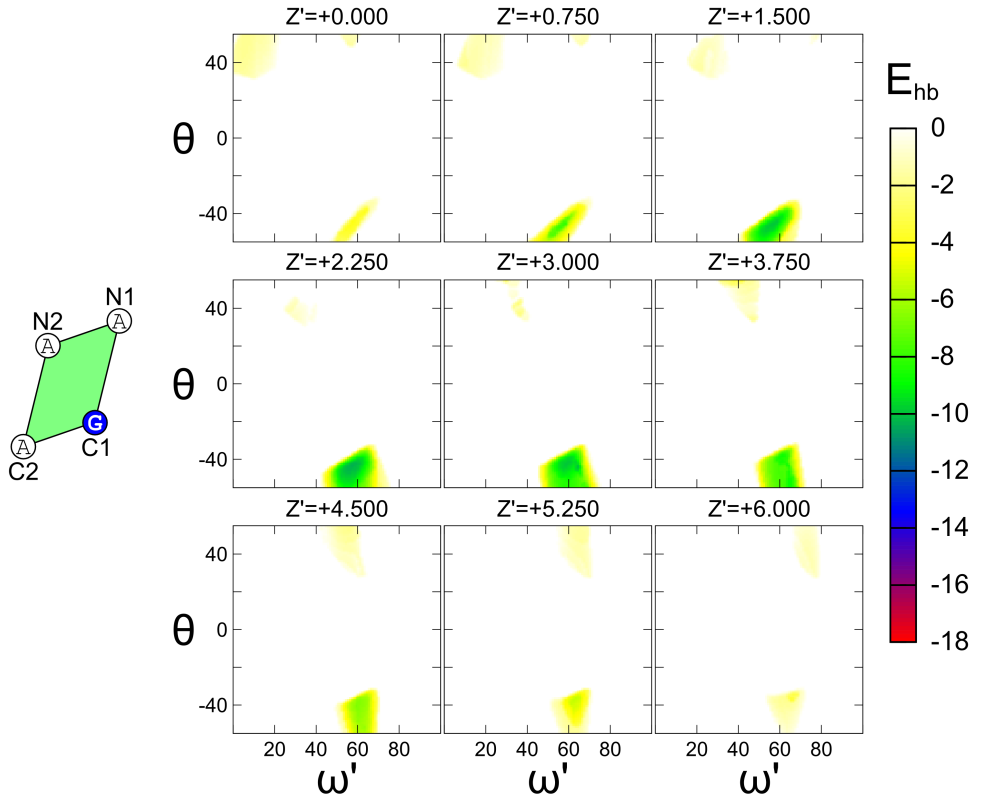


Figure 6.6: *Gly at C1 partially restores hydrogen bond propensity. Extended version of the hydrogen bonding energy maps for poly-Ala with a single Gly at positions C1 as in Figure 6.3d.*

closest approach in the middle section of the parallelogram (Figs. 6.3d and in more detail in Figure 6.6). Above and below these Z' values the backbones are separated by the Cβ methyl groups of either positions N1 or C5 (the amino acid at i±4 with respect to C1).

When a single Gly is placed at any position other than C1 (N1, N2 or C2) the hydrogen bonding energy landscapes present only very shallow minima (Figs. 6.3b, 6.3c, and in further detail Figure 6.5b, 6.5c and 6.5d). This is because the Cβ of C1-Ala invariably comes in contact with the opposing helix, preventing the two helices from being in sufficient proximity. Therefore

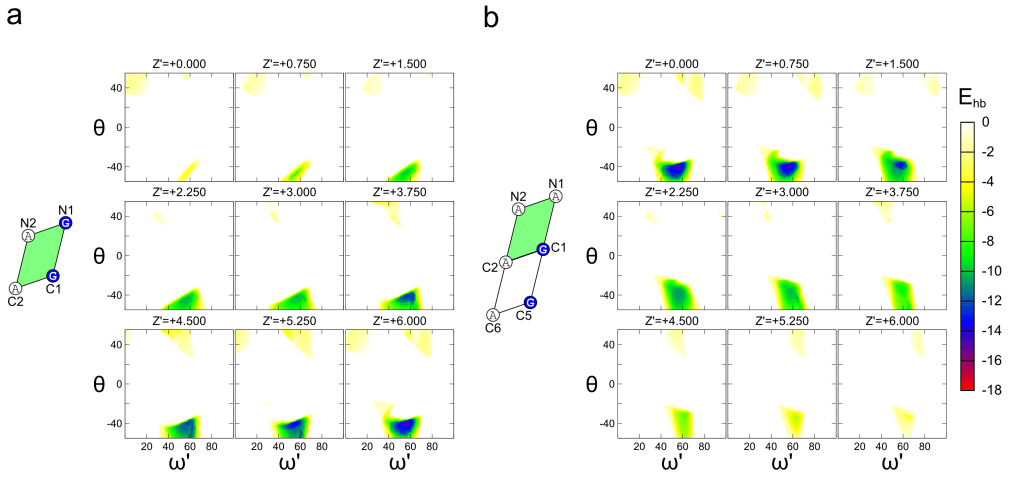


Figure 6.7: *A second Gly at N1 or C5 enhances hydrogen bonding in the presence of Gly at C1. Extended version of the hydrogen bonding energy maps for poly-Ala with two Gly residues on the right-hand side of the unit cell. a) Gly at N1 and C1 as in Figure 6.3e. b) Gly at C1 and C6 (not shown in Figure 6.3)*

we conclude that C1 is the position with the most stringent requirement for Gly.

6.4 GxxxG motifs are important on the right-hand side of the unit cell

If a second Gly is added at i-4 (N1) or i+4 (C5) with respect to C1 to form a *GxxxG* motif on the right-hand side of the unit cell, the hydrogen bonding propensity increases very significantly. If two Gly are placed at N1 and C1, significant restoration of the propensities is present for Z' values that bring the crossing point closest to N1 (Figures 6.10e and 6.7). If two Gly are placed at C1 and C5, the increase is observed for low Z' values that have a crossing point closest to C1 (Figure 6.7). Finally, when N1, C1 and C5 are all Gly to form a Gly zipper motif /emphGxxxGxxxG [**?**], the energy

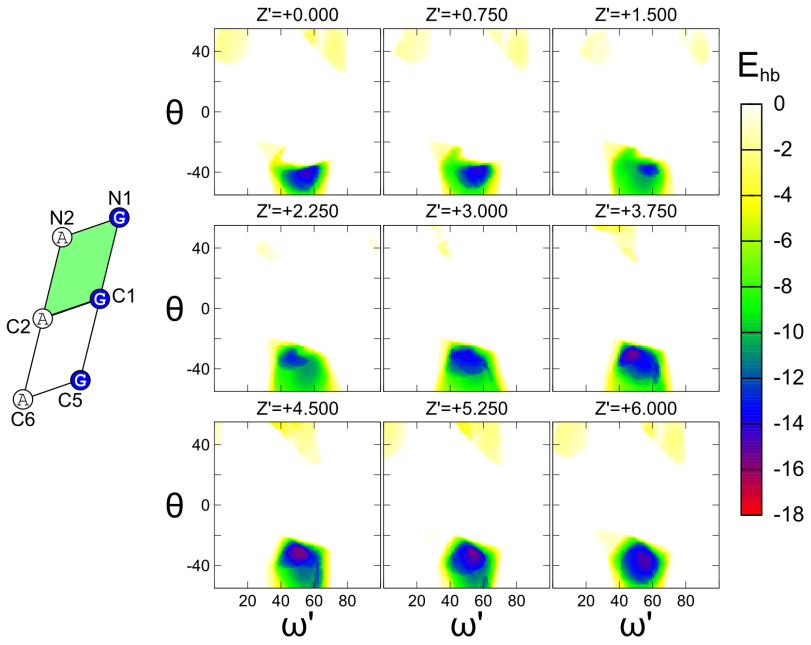


Figure 6.8: *Three Gly on the right-hand side of the unit cell restores almost all hydrogen bonding propensity. Extended version of the hydrogen bonding energy maps for poly-Ala with Gly residues at N1, C1 and C5, as in Figure 6.3f. The main minimum is very similar to that observed for poly-Gly (Figure 6.4a)*

landscape looks very similar to the poly-Gly results (Figs. 6.12f and 6.8). Again, addition of Gly residues on any of the left side positions (N2, C2 or C6, while keeping C1 as Gly) has a negligible effect on the hydrogen bonding energies (Figure 6.9).

The marked distinction between the positions on the right side of the unit cell (N1, C1, C5) and those on the left side (N2, C2, C6) arises from the different orientation of the Cβ and Hα atoms with respect to the interface. This is schematically illustrated in Figure 6.10. The Cβ atom of C2 points away from the interface whereas the Cβ of C1 is oriented directly toward the opposing helix. For this reason, larger amino acids can be accommodated at C2, but Gly is required in C1 to allow the two backbones to come into

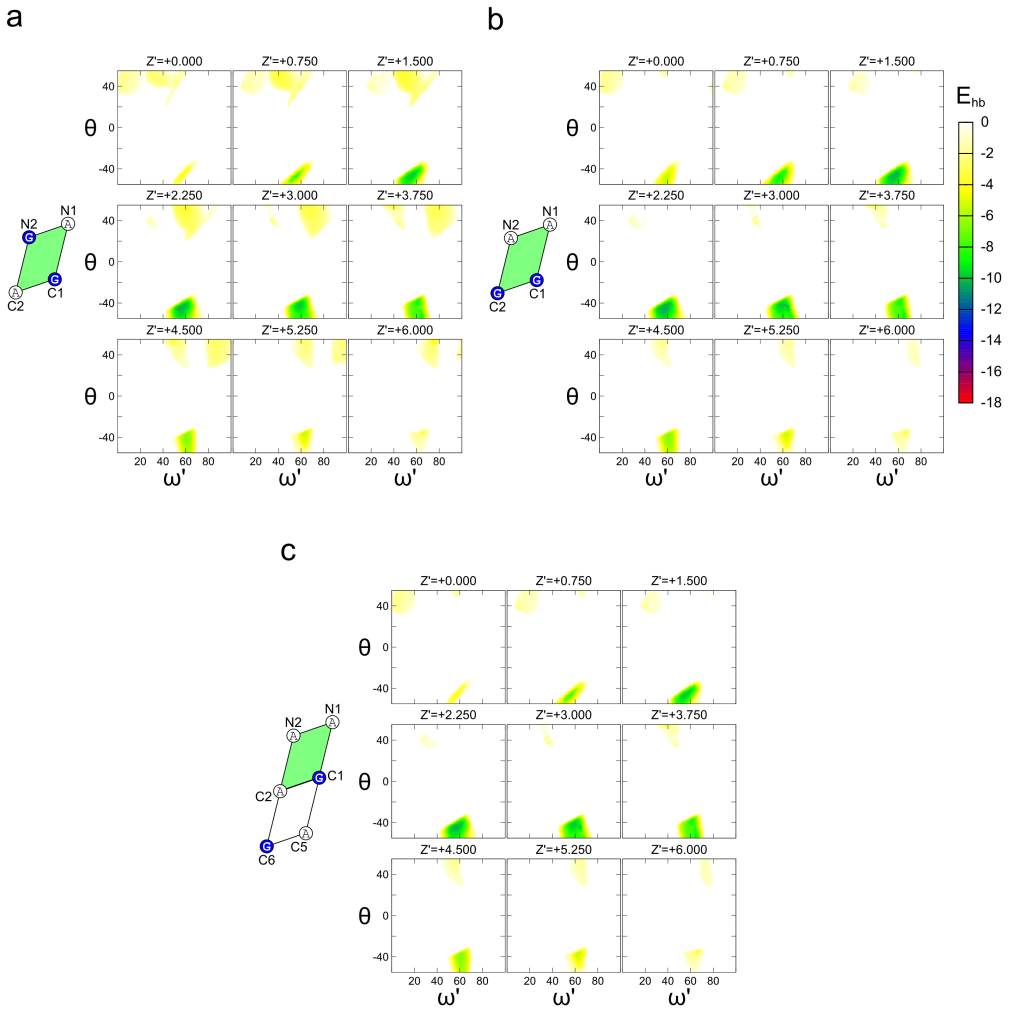


Figure 6.9: *A second Gly at N1, N2 or C6 does not restore hydrogen bond propensity. Hydrogen bonding energy maps for poly-Ala with two Gly residues a) Gly at N2/C1. b) Gly at C1/C2. c) Gly at C1/C6*

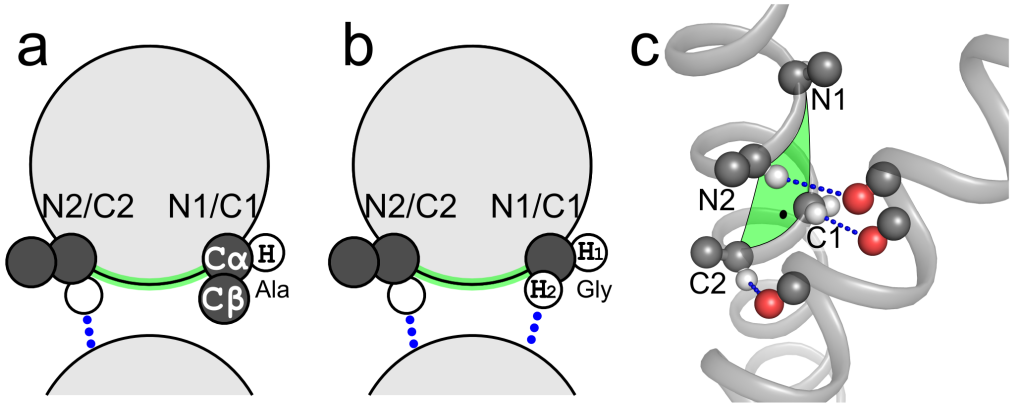


Figure 6.10: *Structural distinction between interfacial positions. a) The amino acids on the left side of the unit cell (N2 and C2) orient their α-hydrogen toward the interface while their the Cβ points laterally, and thus these position can accommodate larger amino acid types. The situation is reversed for positions N1 and C1: the α-hydrogen is oriented laterally and the side chain points directly toward the opposing helix. Larger amino acids in this position may not be accommodated. b) Gly is the only amino acid type that can form a hydrogen bond using the "side chain" hydrogen when present at positions N1 or C1. c) Structural example: in this case the crossing point is close to C1, and there is sufficient space to allow Ala at N1.*

close proximity. A similar argument applies to N1/N2 and C5/C6 as well.

GAS_{right} motifs are optimized for Cα hydrogen bond network formation

Gly performs a second important function as a donor when present at the right-hand side positions. As illustrated in Figure 6.10a, any amino acid can donate at C2 because the Hα atom is pointed toward the interface. However, that same hydrogen is oriented laterally and away from the interface at C1. As schematically illustrated in Figure 6.10b, only Gly can donate from the right side positions (C1, N1, C5) because its "side chain" hydrogen is in the correct orientation. The same point is illustrated in structural terms in

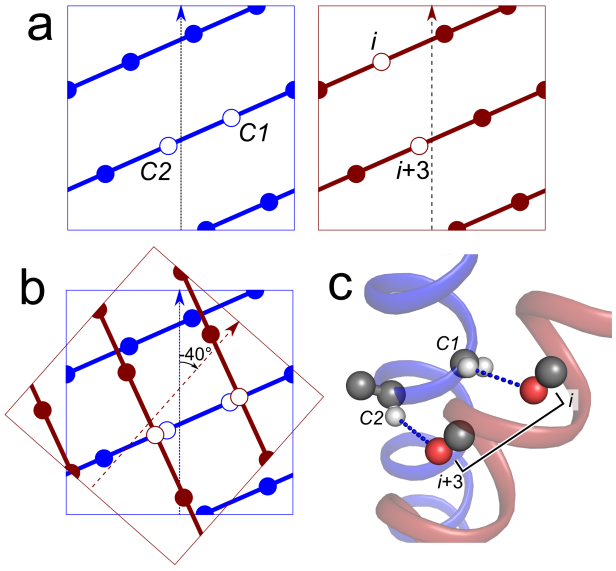


Figure 6.11: *In a* $GAS_{\mathtt{right}}$ *motif the C1 and C2 donors are aligned with carbonyl acceptors at i, i+3 on the opposing helix. a) Helical lattices highlighting the C1 and C2 donor positions (left, blue) and carbonyl acceptors at i, i+3 on the opposing helix (right, dark red). b) A superimposition of the two lattices followed by a -40° rotation aligns the donors and acceptors. c) Structural representation of the same alignment.*

Figure 6.10c.

It follows that both amino acids at C1 and C2 can simultaneously donate to the opposing helix only if C1 is a Gly. However, this requires a correct alignment with acceptors on the opposing helix. As illustrated in Figure 6.11 using a superimposition of helical lattice projections, the crossing angle of $GAS_{\mathtt{right}}$ motifs is optimal for the this purpose. A -40° crossing angle aligns the two donors at C1 and C2 with two carbonyl oxygen atoms spaced at i and i+3 on the opposing helix. This is also shown in structural terms in Figure 6.11c.

Overall, the analysis presents a compelling picture: the $GAS_{\mathtt{right}}$ coincides with the major hot-spot for carbon hydrogen bonding. From a

steric stand point, the geometry appears ideal to allow backbone contacts as long as C1 and either N1 or C5 (or both) are Gly residues. The Gly residues at these same positions are also able to cooperatively extend the hydrogen bonding network by the virtue of having their second hydrogen oriented toward the interface. Finally, the -40° crossing angle is ideal for the simultaneous involvement of C1 and C2 (and, similarly, N1/N2 or C5/C6) in hydrogen bonding interactions. In our opinion, this finding suggests a strong causal link between the high frequency of the GAS_{right} motif in the structural database and its propensity to form networks of carbon hydrogen bonds, supporting the hypothesis that these interactions are important contributors to helix-helix association.

A high-throughput structural prediction method for GAS_{right} motif

The analysis presented above shows that only a small fraction of homo-dimer conformational space allows for the formation of Cα-H...O=C hydrogen bond networks. It also indicates that positions at the interface may have stringent sequence requirements for Gly or a limited set of amino acids. On these premises, we hypothesized that it would be possible to create a rapid method to recognize sequence signatures compatible with the formation of GAS_{right} motifs.

To develop and implement the method, which we named *CATM*, we systematically subdivided the homo-dimer conformational space that allows formation of Cα-H...O=C bonds into a comprehensive "grid" of representative dimer conformations. We then established the specific sequence requirements of each conformation (sequence rules). In this implementation, we did not limit the space to the right-handed region, but allowed any dimer that displayed formation of at least two pairs of symmetrical hydrogen bonds.

When the primary sequence of a TM domain of interest is provided to

CATM, the sequence is built in full atoms over each representative dimer that is compatible with the sequence rules. The two helices are placed at the inter-helical distance in which the two backbones still form a network of carbon hydrogen bonds (d_{out}, which is pre-calculated for each dimer). The helices are then moved closer followed by optimization of the side chains, until the energy reaches a minimum. At that point, the geometry of the dimer is locally optimized with a brief *monte carlo* procedure consisting of cycles in which all four inter-helical parameters changed randomly (d, Z, ω, θ).

At the end of each docking, the energy of the dimer is subtracted from the energy of the helices separated at a distance to obtain an interaction energy. Only the solutions with a negative interaction energy are preserved. Finally, all closely related solutions are clustered by similarity (RMSD $<$ 2Å), and the lowest energy structure is reported as a representative model of its cluster. CATM is explained in full detail in the Methods, and is freely available for download with MSL, a C++ open source macromolecular modeling software library, at http://msl-libraries.org [Kulp et al. (2012)].

A minimalistic set of energy functions predicts known structures with near atomic accuracy

We tested CATM against five known homo-dimeric GAS_{right} structures: glycophorin A [**?**], BNIP3 [**??**], and three members of the Tyrosine Receptor Kinase family, EphA1 [**?**], ErbB1 (EGFR) [**?**] and ErbB4 [**?**]. We began testing using a simple combination of hydrogen bonding (E_{hbond}) and van der Waals (E_{vdw}) to score the structural models. Perhaps surprisingly, we found that this minimalistic set of energy functions predicts the structures at near atomic precision, and in all but one case, the native structure corresponds to the lowest energy model. The finding validates our hypothesis that Cα hydrogen bonds can be an important guiding element for structure recognition, because they offer multiple anchor points between backbones, and because

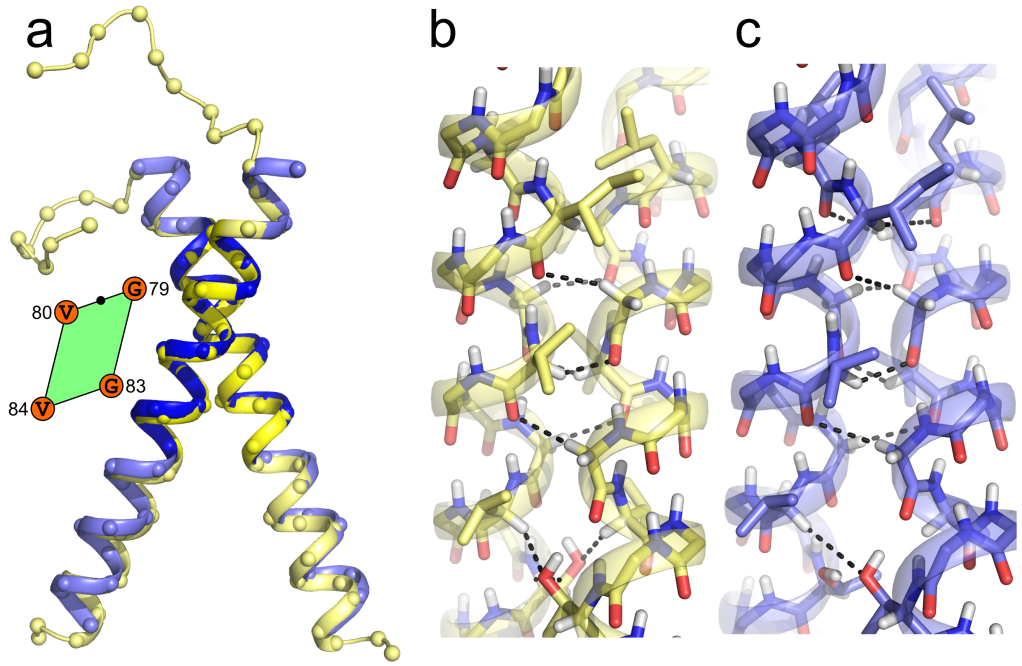
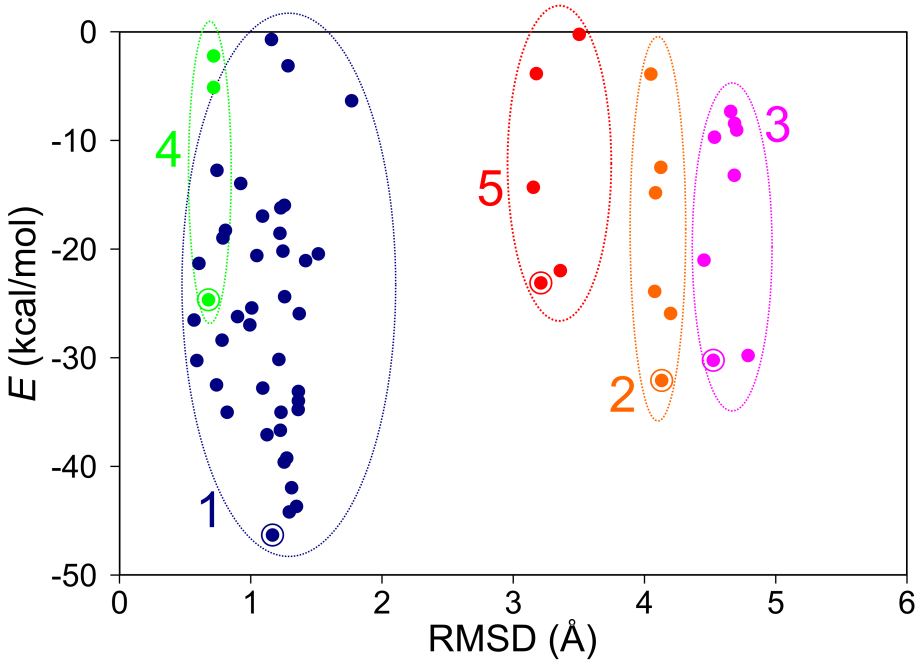


Figure 6.12: *CATM prediction of the TM domain of Glycophorin A. a) Backbone superimposition of the NMR structure (yellow) and the predicted model (blue). The Cα RMSD in the region that encompasses the interface is indicated and highlighted in darker blue and yellow in the ribbon. Panels b and c show the full-atom comparison between the experimental structure and the prediction. The CATM model is close to atomic level, with a similar network of carbon hydrogen bonds. The NMR structure and CATM model differ in the orientation of Thr 87, which hydrogen bonds to its own backbone, while CATM predicts the formation of an inter-helical canonical hydrogen bond.*

they are strongly dependent on good packing, given that the interactions can be easily disallowed by steric clashes [**?**](23). All predicted models discussed below can be downloaded from http://seneslab.org/CATM/structures.

CATM returned 63 solutions for Glycophorin A, the first TM dimer solved by solution NMR [**?**](15) and a major biophysical model systems



	Sequence and interface	d (Å)	θ (°)	Z' (Å)	ω' (°)	E
Model 1	EAEITLIIFGVMAGVIGTILLISYGIRRL	6.3Å	-49.1°	5.9	55.9	-46.3
Model 2	EAEITLIIFGVMAGVIGTILLISYGIRRL	7.5Å	+26.0°	1.7	25.9	-32.0
Model 3	EAEITLIIFGVMAGVIGTILLISYGIRRL	6.7Å	+54.0°	0.4	20.2	-30.2
Model 4	EAEITLIIFGVMAGVIGTILLISYGIRRL	7.4Å	-45.8°	3.0	41.3	-24.6
Model 5	EAEITLIIFGVMAGVIGTILLISYGIRRL	7.1Å	-54.0°	4.2	44.7	-23.1
	70 75 80 85 90 95					

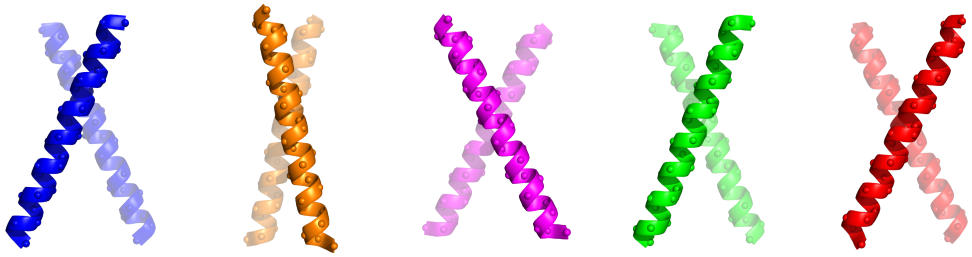
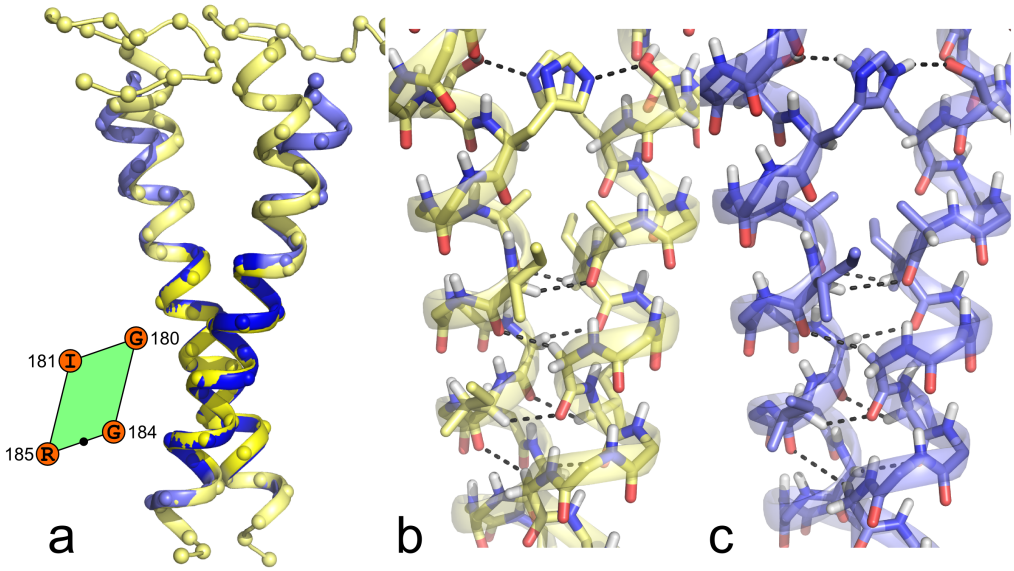


Figure 6.13: *RMSD from the NMR structure vs CATM energy for glycophorin A. CATM produces 63 structures for the transmembrane sequence of GpA, clustered into 5 representative models.. The five clusters are color coded, and the lowest energy model highlighted by a circle. Model 1 and Model 4 are closely related neighboring clusters, both right-handed dimers with a geometry similar to the experimental structure. As for all comparison, the RMSD were calculated in the range of amino acids that encompasses the dimer interfacial region (from L75 to T87) as in Figure 6.12*

for membrane protein association [**?**](24). The 63 solutions were clustered into 5 distinct models. The relationship between RMSD and energy for all 63 structures is plotted in Figure 6.13. The lowest energy model predicted by CATM (Model 1) is a very close match of the NMR structure (Figure 6.12). Measured over the entire TM helix (residues 73-95), the Cα RMSD is 1.31 $\pm$ 0.24 Å (average and standard deviation measured against the 20 NMR models). Measured over the segment that encompasses the interaction interface, discarding the contribution of the divergent ends, the RMSD reduces to 1.1 $\pm$ 0.21 Å (residues 75-87, marked in darker blue in Figure 6.12a). A side by side comparison of the predicted model and the experimental structure shows the matching hydrogen bonding network and the conformation of the interfacial side chains (Figure 6.12, panels b and c). A difference between the two structures is the conformation of Thr 87 which accepts an Cα hydrogen from Val 84 on the opposing helix in the NMR structure, while in the lowest energy CATM model the hydroxyl group of Thr 87 is involved in an inter-helical canonical hydrogen bond, which is consistent with a solid state NMR structure of the dimer [**?**]. Figure 6.12 also shows the position of the point of closest approach in the unit cell at the interface of the dimer. It should be noted that glycophorin A complies with the "Gly at C1" rule identified in our analysis, as all other structures analyzed in the following paragraphs. In fact, C1 is the only position that is invariably Gly across all the examples.

The second structural prediction is BNIP3, a very stable TM dimer [**?**] characterized by a very short inter-helical distance (6.5 Å). The interface consists of $A_{176}xxxG_{180}xxxG_{184}$ a glycine zipper motif . As shown in Figure 6.14a, the model is extremely similar to the NMR structure [**??**]. The RMSD of the helical region of the entire TM domain is 1.10 $\pm$ 0.36 Å and only 0.56 $\pm$ 0.17 Å when it is computed only for the region that participates to the helix-helix interaction. The model replicates the network of carbon hydrogen bonds observed in BNIP3 and all interfacial side chains are predicted in the



Protein	PDB	Residues	Cα RMSD (Å)
BNIP3	2j5d	172-184	0.56 ± 0.17

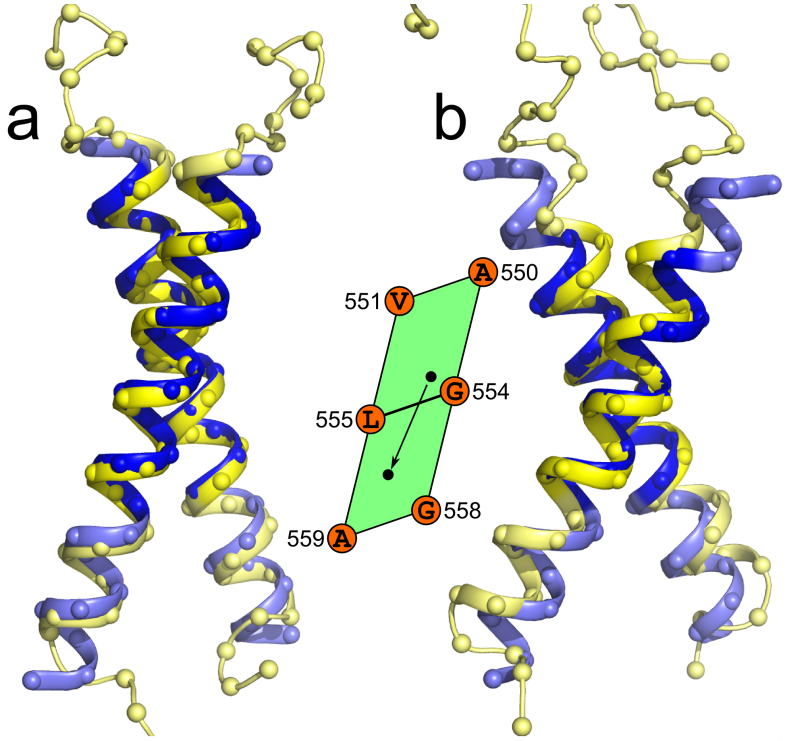
NMR structure
CATM prediction

Figure 6.14: *Structural prediction of BNIP3. CATM produces a single model for BNIP3 that is extremely similar to the NMR structure. The Cα RMSD of the helical region of the entire TM domain is 1.10 ± 0.36 Å, which falls to 0.56 ± 0.17 Å when only the region in contact (darker blue and yellow) is considered. The side by side prediction (panels b and c) shows close similarity in the network of carbon hydrogen bonds and correct prediction of the orientation of all interfacial side chains. The model also accurately captures the canonical hydrogen bond between Ser 172 and His 173.*

correct rotamer, as evident in the side-by-side comparison of panels b and c of Figure 6.14. In addition, CATM accurately captures the inter-helical hydrogen bond between the side chain of His 173 (donor) and Ser 172 (acceptor), an important feature that contributes to the dimer's stability (27).

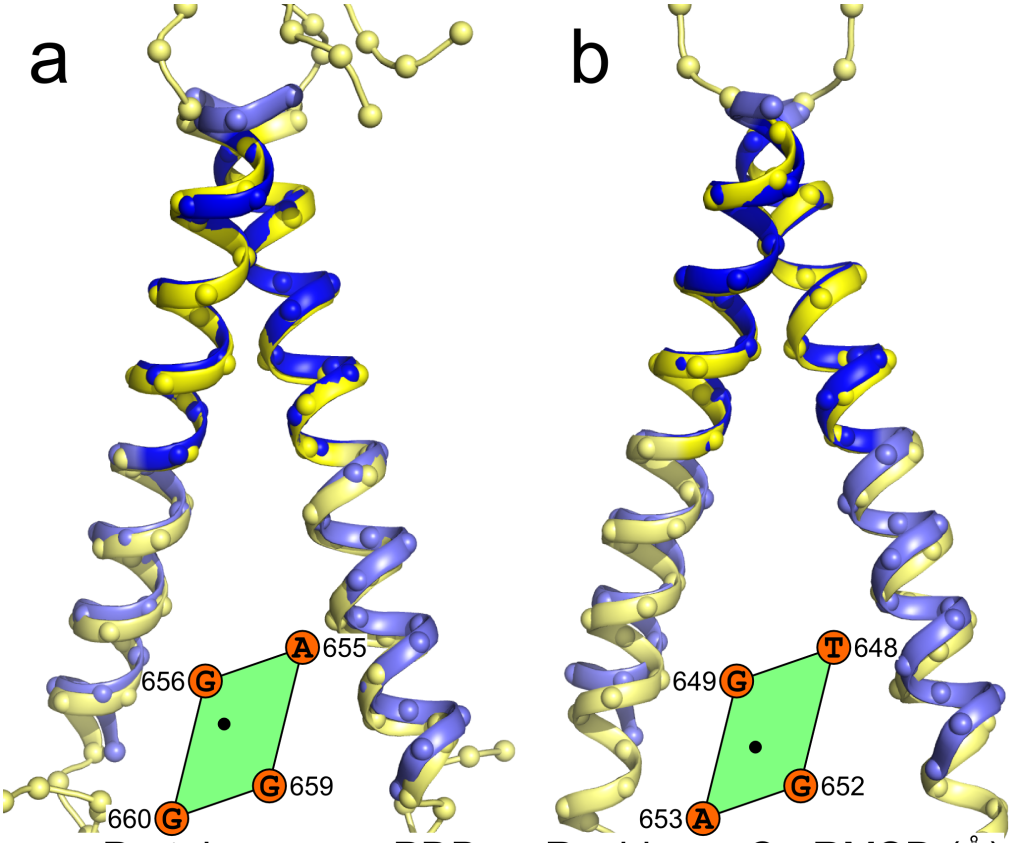
The third comparison is EphA1, which was solved by solution NMR in bicelles at two different pH conditions [**?**]. The dimer displays a conformational change induced by change in protonation state of a membrane embedded Glu residue (E547). CATM captures both conformations with good accuracy (Figure 6.15). The low pH structure is predicted by Model 1 with a Cα RMSD of 1.26 Å. The higher pH structure is predicted by Model 4 with an RMSD of 1.48 Å. The structures are related by a shift of the crossing point of about 3 Å toward the C-terminus that brings the crossing point from the top half to the bottom half of the Glycine zipper motif (*A550xxxG554xxxG558*), as schematically shown in Figure 6.15c. Interestingly, the authors also report the presence of a minor component of some cross-peaks in the higher pH conditions, suggesting a second species (about 10%) was present in the sample [**?**]. While a structural model could not be calculated and was not reported for this minor species, the authors suggest that this competing state associates through the C-terminal *GxxxG*-like motif (*A560xxxG564*), and identify the amino acids involved at the interface as Leu 557, Ala 560, Gly 564 and Val 567. This description is consistent with the interface of Model 2 produced by CATM.

The final two test cases are both members of the epidermal growth factor receptor family [Schlessinger (2000)]. As shown in Figure 6.16a, the NMR structure of ErbB4 [**?**] is predicted well by CATM, with an RMSD of 0.81 Å across the interacting region. However, our prediction of ErbB1 (EGFR) is not in agreement with the experimental structure, the only case among the five structures tested. The experimental structure interacts through the N-terminal *TxxxG* motif [**?**], and this structure is predicted by CATM's



	pH	PDB	Residues	Model	Cα RMSD (Å)
a	4.3	2k1k	547-562	1	1.26 ± 0.01
b	6.3	2k1l	550-562	3	1.48 ± 0.01

Figure 6.15: *CATM predicts multiple states of the EphA1 Tyrosine Receptor Kinase. a) the structure of the TM domain EphA1 determined at a low pH is well predicted by CATM Model 1. b) the structure obtained at higher pH is matched by Model 4. The conformational shift between low and high pH is highlighted schematically in the unit cell representation. The interface remains centered on the Gly-zipper motif (*AxxxGxxxG*) but the crossing point shifts (arrow) toward the C-terminus in the adjacent unit cell. There is also an increase of the crossing angle. EphA1 has multiple* GxxxG*-like motifs and produces four models. Model 2 interacts through a C-terminal* AxxxG *motif. Model 3 is closely related to Model 1.*



	Protein	PDB	Residues	Cα RMSD (Å)
a	ErbB4	2lcx	652-664	0.81 ± 0.05
b	ErbB1	2m20	645-667	0.78 ± 0.05

Figure 6.16: *Prediction of ErbB4 and ErbB1. a) ErbB4 is predicted by the top CATM model, while b) ErbB1 (EGFR), is predicted by the third model. Among the five structured tested, ErbB1 is the only structure that is not predicted by the lowest energy model.*

Model 3 with a Cα RMSD 0.77Å (Figure 6.16b). Instead Model 1 is a well packed dimer that interacts through C-terminal side *AxxxG* motif, of the TM helix and is a likely candidate for a postulated inactive state of the receptor [**??**]. As in the case of EphA1, this finding highlights the potential of offering alternative structural models that may reflect distinct functional states of the TM dimers.

6.5 Conclusions

We have presented an analysis of carbon hydrogen bonding as a function of helix orientation in TM homo-dimers. The analysis demonstrates that there is a single region of conformational space for homo-dimers with a high propensity for formation of hydrogen bond networks. Remarkably, this area corresponds to the $\mathrm{GAS}_{\mathtt{right}}$ motif, lending strong support to the hypothesis that optimization of carbon hydrogen bonding is a major driving factor in its assembly. The analysis also provides a rational structural interpretation of the occurrence of *GxxxG* motifs in $\mathrm{GAS}_{\mathtt{right}}$ homo-dimers, indicating that the Gly residues are essential on a specific side of the helix interface for steric reasons and to act as hydrogen bonding donors.

Based on the analysis, we have created a rapid method for the structural prediction of $\mathrm{GAS}_{\mathtt{right}}$ homo-dimers. We have shown that with a surprisingly simple set of energy functions ($\mathrm{E}_{\mathtt{hbond}} + \mathsf{E}_{\mathtt{vdw}}$), CATM predicts the known structures of $\mathrm{GAS}_{\mathtt{right}}$ homo-dimers with near atomic precision. Future work is necessary to refine, verify and expand the scoring functions. For example, a membrane model such as a depth-depended potential [**?**] or an implicit solvent [**?**], is likely to improve the predictions and any correlation between the computational score and the thermodynamic stability. Nevertheless, CATM appears to capture the essence of $\mathrm{GAS}_{\mathtt{right}}$ motifs already in the current form, and therefore the method is already applicable to the rapid prediction of unknown structures.

6.6 Methods

Software

All calculations were implemented and performed using the MSL molecular modeling libraries v.1.1 [Kulp et al. (2012)](18), an open source C++ library that is freely available at http://msl-libraries.org.

Creation of inter-helical geometries

Two helices, 31 residues in length, were created in idealized conformation, oriented with their axes aligned with the z-axis and the Cα atom at position 16 placed on the x-axis. Position 16 is the position designated as C2 in Figure 6.1c. To create a dimer, the following transformations were performed in order: a rotation around the z-axis (determining the axial rotation ω), a translation along the z-axis (determining the position of the crossing point Z in the z-dimension), a rotation around the x-axis (determining the crossing angle θ), and a translation along the x-axis (determining the inter-helical distance d). One of the two helices was finally rotated around the z-axis by 180° to produce 2-fold symmetry.

The geometric analysis was performed so that the point of closest approach P would explore the entire unit cell defined by N1,N2,C1,C2 as in Figure 6.1c. The transformations were performed using a redefined set of geometric parameters [d, θ, ω', Z'], where ω', Z' are unit vectors that go in the direction of the principal components of the unit cell of the helical lattice using the mathematical relationships defined in Figure 6.2. The conformational space was explored at discrete intervals with the following step sizes: d: 0.1 Å; ω': 1°; Z': 0.1 Å; θ: 1°. The crossing angle θ was constrained to be in the -55° to +55° range.

Energy functions and definitions

Energies were determined using the CHARMM 22 van der Waals function [MacKerell et al. (1998)] and the hydrogen bonding function of SCWRL4 [Krivov et al. (2009)], as implemented in MSL C++ libraries [Kulp et al. (2012)]. Cα hydrogen bonds have been included as part of the energy functions of ROSETTA Membrane [**?**]. We derived a similar adaptation for the SCWRL4 function by adding the following parameters for CÎ± donors: B=60.278; $D_0 = 2.3$Å; $\sigma_d = 1.202$Å; $\alpha_{max} = 74.0°$; $\beta_{max} = 98.0°$. These parameters reduce the hydrogen bonding energy to approximately half that of canonical bonds, and adjust the optimal distance and the angular dependencies.

In the text below the energy of a model is computed as the difference between the the dimer energy minus the energy of the separated monomers (referred to as interaction energy), with the side chains optimized independently in the two states. All side chain optimization procedures were performed using the Energy-Based Conformer Library applied at the 95% level [**?**] with a greedy trials algorithm [**?**] as implemented in MSL.

Determination of Cα-H...O energy landscapes

The energy landscapes were determined for all [θ, ω', Z'] coordinates. Two helices were initially placed at d = 10 Å. The energies were evaluated and the helices were moved closer to each other in 0.1 Å steps until a lowest energy (E_{min}) conformation was identified at a distance d_{min}. Figure 6.3 plots E_{min} as a function of [θ, ω', Z']. A plot of the corresponding d_{min} values is provided for poly-Gly in Figure 6.4.

Development of CATM

CATM is a structure prediction program that performs a systematic search in the subset of homo-dimer conformational space that allows formation of inter-

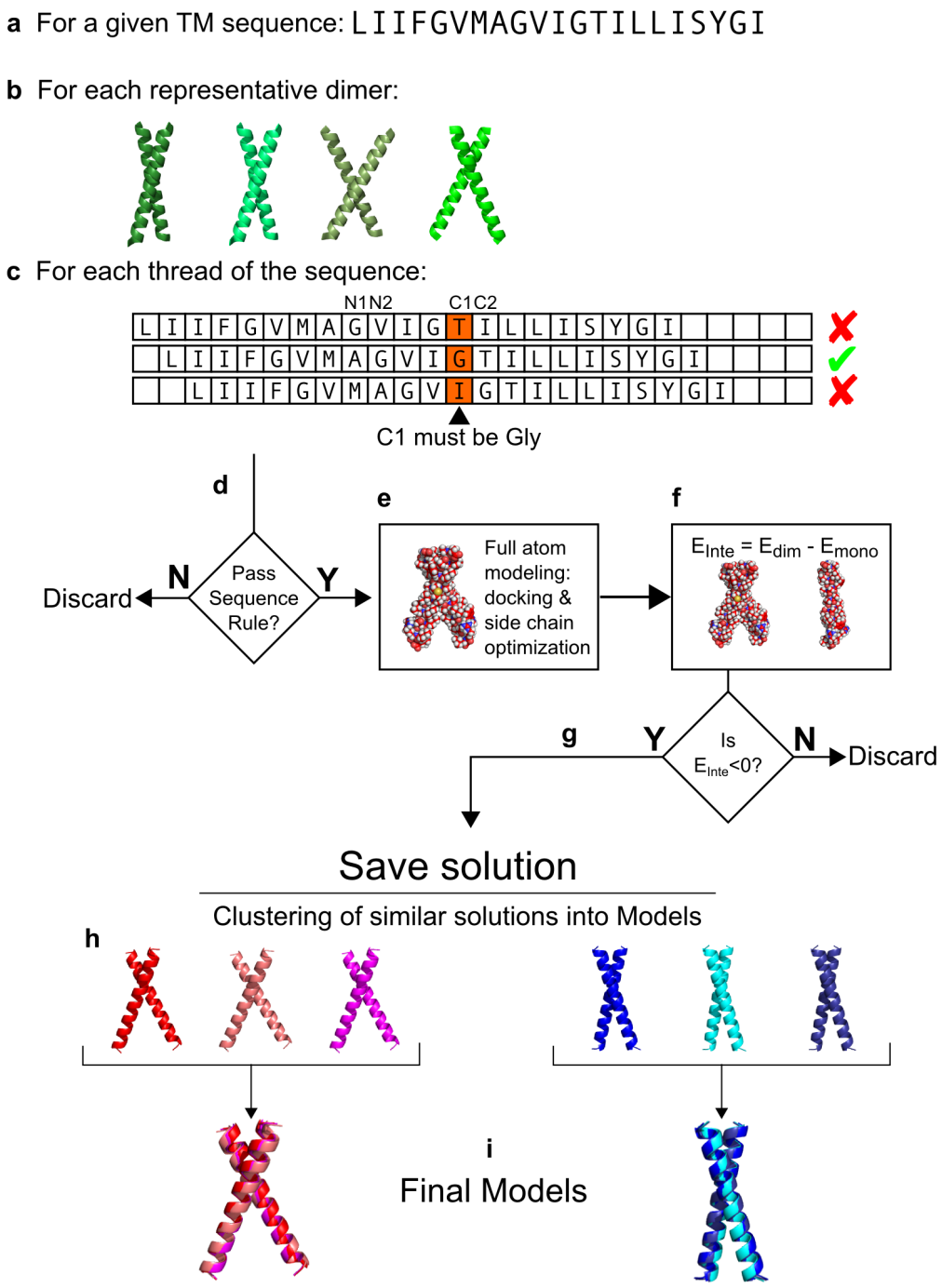


Figure 6.17: *Schematic illustration of CATM. Given a sequence (a), the sequence is threaded onto each of the 463 representative geometries (b) in all possible registries on the α-helices (c). For each thread the sequence rules are checked (d, in this example, we are only checking for a required Gly at C1). If the rules are met, the sequence is built in all atoms and the structure is optimized (e), and an interaction energy is calculated (f). If the interaction energy is negative, the solution is accepted (g). The solutions are then clustered (h) to produce a series of final models, ranked by energy (i)*

helical Cα-H...O hydrogen bonds. The creation of CATM consisted of the definition of the search space and the derivation of a set of sequence exclusion rules. The execution phase of CATM (the actual structure prediction for a given sequence) is schematically illustrated in Figure 6.17.

Definition of the search space

The definition of the search space was based on the geometric analysis of poly-Gly. We selected all conformations in [θ, ω', Z'] space that display at least four inter-helical Cα-H...O hydrogen bonds (two symmetrical pairs). This search yielded a set of approximately 90,000 structures which were then filtered by similarity using a 2.0 Å RMSD criterion to create a representative set of 463 geometries. For each representative geometry we recorded the maximum inter-helical distance in which four hydrogen bonds still exist (d_{out}).

Definition of the sequence rules

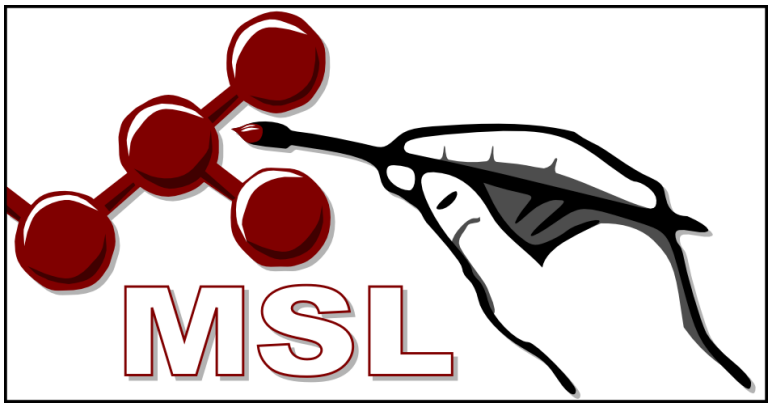
Each representative geometry G was constructed as poly-Gly and was set at d_{out}. Every amino acid type X was built at every position j in every G and its conformation was optimized. If the interaction energy was unfavorable by more than 10 kcal mol^{-1}, a sequence rule was recorded stating that the X is not allowed at j in G. These rules allow for the exclusion of non-productive sequences from the expensive all atom modeling phase.

The CATM program

The input sequence is threaded into a set of different registers at each of the 463 representative geometries (Figure 6.17). For each register, CATM checks if the sequence rules are met. If the rules are met, the sequence is built on the backbone in all atoms, and the helices are placed at d_{out}. The inter-helical distance is reduced in steps of 0.1 Å, and at each step the

side chains are optimized and the interaction energy is evaluated until a minimum energy is found. To further optimize the dimer, the geometry is then subjected to 10 *monte carlo* backbone perturbation cycles in which all inter-helical parameters (d, θ, ω, Z) are locally varied. If the final interaction energy is negative, the solution is accepted. The solutions are then clustered using an RMSD criterion (2 Å) to produce a series of distinct models, with all individual solutions provided as an NMR-style PDB file.

7 MOLECULAR SOFTWARE LIBRARY (MSL): AN OPEN-SOURCE TOOL FOR MOLECULAR MODELING



based on

Kulp DW, **Subramaniam S**, Donald JE, Hannigan BT, Mueller BK, Grigoryan G and Senes A "Structural informatics, modeling and design with an open-source Molecular Software Library (MSL)", *Journal of Computational Chemistry* 2012 **33**(20), 1645-61

*This software package is a joint effort of the listed authors. I have been involved with MSL from its early stages (Sep. 2008) and have made significant contributions to the development of functions related to energetics, conformational sampling, analysis and more. My specific contributions are detailed at the end of the chapter.

Summary

This chapter presents the Molecular Software Library (MSL), a C++ library for molecular modeling based on [Kulp et al. (2012)]. MSL is a set of tools that supports a large variety of algorithms for the design, modeling, and analysis of macromolecules. Among the main features supported by the library are methods for applying geometric transformations and alignments, the implementation of a rich set of energy functions, side chain optimization, backbone manipulation, calculation of solvent accessible surface area, and other tools. MSL has a number of unique features, such as the ability of storing alternative atomic coordinates (for modeling) and multiple amino acid identities at the same backbone position (for design). It has a straightforward mechanism for extending its energy functions and can work with any type of molecules. Although the code base is large, MSL was created with ease of developing in mind. It allows the rapid implementation of simple tasks while fully supporting the creation of complex applications. Some of the features of the software are demonstrated here with examples that show how to program complex and essential molecular modeling tasks with few lines of code. MSL is an ongoing and evolving project, with new features and improvements being introduced regularly, but it is mature and suitable for production and has been used in numerous protein modeling and design projects [**???**]. MSL is open-source software, freely downloadable at http://msl-libraries.org. We propose it as a common platform for the development of new molecular algorithms and to promote the distribution, sharing, and reutilization of computational methods.

7.1 Introduction

Over the past decades, computational biology has been contributing more and more frequently to the understanding of macromolecular structure and the mechanisms of biological function. Although the number of high-

resolution protein structures in the Protein Data Bank (PDB) is steadily growing, the experimental methods currently available for structural determination do not nearly approach the level of throughput that would be necessary to characterize the universe of known protein sequences [**?**]. This generates high interest in reliable and affordable protein modeling methods, as means for investigating the function of proteins and predicting their interactions and specificity. Computational methods can take advantage of today's large structural database and essentially expand it. Homology-based methods have now reached excellent levels of performance in predicting the structure of many proteins when a closely-related protein has been experimentally determined [**??**]. Comparative structural analysis can be used to identify common themes and key interactions in sets of related proteins. The structural database can also be disassembled into fragments, and these fragments form the basis for ab initio structural prediction methods and provide templates for filling in the missing elements in experimental structural models [**?**]. Molecular modeling can today work directly in combination with experimental structural methods such as NMR to help build accurate structural models from incomplete or reduced dataset[**?**]. Modeling is also becoming a fundamental tool for assisting experimental design, rational mutagenesis, and protein engineering. It also provides an invaluable framework for interpreting experimental data. Such approaches have greatly helped to improve our knowledge of proteins that are intrinsically difficult to study with the traditional structural methods, such as, for example, the integral membrane proteins [**???**]. Finally, molecular modeling methods today allow the creation of proteins de novo. Protein design has become an important tool for investigating the fundamental principles that govern stability, specificity, and function in proteins and can be applied to the creation of new reagents and probes.[**????**]

With the continued increase in power and decrease in cost of high-throughput computing, computational biology is likely to continue to grow

and become even more integrated with the experimental disciplines. To fully support this trend and promote the use of the existing methods and the creation of new powerful algorithms, it is important to spread the availability of molecular modeling libraries, providing the community with tools that are fully featured, powerful, easy to use and, ideally, free to distribute and modify. Here, we present MSL (the Molecular Software Library), an open-source C++ library that fulfills these criteria and supports the creation of efficient methods for structural analysis, prediction, and design of macromolecules. MSL is not a single program but a set of objects that facilitate the rapid development of code for molecular modeling. The object-oriented library is targeted toward researchers who need to develop simple or sophisticated modeling and analysis programs. The main objectives of MSL are to allow the implementation of simple tasks with maximum ease (e.g., measuring a distance or translating a molecule) while fully supporting the creation of complex and computationally intensive applications (such as protein modeling and design). This objective has been achieved by developing efficient code. The design of intuitive APIs (Application Programming Interfaces) and the maximization of the modularity of the objects has allowed to keep the code base simple and easily expandable, agnostic to the type of molecule, and thus suitable to work with proteins, nucleic acids, or any other small or large molecules. For these reasons, MSL is ideal for supporting the implementation of a large variety of structural analysis, modeling, and design algorithms that appear in the growing computational biology literature. The adoption of a common platform for the implementation of computational methods would greatly benefit the scientific community as a whole. It would help to avoid fragmentation and promote the distribution of the methods and their integration. The open-source model allows the continued development of the code, higher scrutiny and quality control, and deeper understanding of the methods. This model has been successfully adopted by other scientific projects (e.g., the bioinformatics toolsets BioPerl [**?**] and BioPython [**?**]).

The development of the MSL libraries has been active over the past 4 years and a growing list of algorithms has already been implemented, with more features and enhancements to come. The platform has been successfully applied to numerous areas of biological computing, including modeling [**??**] and de novo design of membrane proteins [**?**] modeling large conformational changes in viral fusion proteins,[**?**] designing a switchable kemp eliminase enzyme,[**?**] studying distributions of salt bridging interactions,[**?**] the development of an empirical membrane insertion potential [**?**], the development of new conformer libraries,[**?**] and other ongoing projects. In this article, we highlight a number of unique and powerful capabilities of MSL, using several key worked examples to provide the reader with a basic understanding of the MSL object structure. For example, we illustrate how to access molecular objects, apply geometric transformations, model a protein, make mutations, apply a rotamer library, calculate energies, and do side chain optimization. Further, we illustrate a side chain conformation prediction program that is distributed with the library and present its performance statistics against a large set of proteins structures. A comprehensive set of tutorials and helpful documentation are currently being assembled on the MSL website (http://www.msl-libraries.org) where MSL is also freely available for download.

Molecular representation: flat-array versus hierarchical structure

At the very core of any molecular modeling software is the representation of the molecule. A simple level of representation may be sufficient for a number of tasks. For example, a program that translates a molecule only requires access to the atoms' coordinates. In such case, a "flat-array" of individual atoms can be rapidly created and is memory efficient. This representation would allow a quick iteration over atoms to apply the transformation. Other tasks may benefit from a more complex representation. For example, a

program for computing backbone dihedral (ϕ/ψ) angles of a position needs to access the C and N atoms of the preceding and following amino acids. The identification of the relevant atoms becomes more rapid if the macromolecule is stored as a "hierarchical" representation, in which the atoms are subdivided by residue, and the residues are ordered into a representation of the chain. Because of these conflicting needs, MSL implements both flat-array and structured hierarchical approaches and lets the programmer decide what is most efficient and appropriate for a given task.

The flat-array representation - called *AtomContainer* - is schematically explained in Figure 7.1. The *AtomContainer* acts as an array of *Atom* objects that can be iterated over using an integer index. Each *Atom* holds all of its relevant information (such as atom name, element, atom type, coordinates, and bonded information). Inside the *Atom*, the coordinates are held by a *CartesianPoint*, which handles all the geometric functions. The *AtomContainer* has functions for inserting and removing atoms, checking their existence by a string "id" (chain id + residue number + atom name, i.e., "A,37,CA"), and has functions for reading and writing PDB coordinate files [**?**].

The hierarchical representation in MSL has seven nested levels, as illustrated in Figure 7.2. The *System* is the top-level object that contains the entire macromolecular complex and is divided into *Chain* objects. Chains are divided into *Position* objects. Within the *Position*, there is one (and sometimes more) *Residue* object, corresponding to the specific amino acid types in a protein, for example *Leu* or *Val*. The distinction between a *Position* and a *Residue* enables easy implementation of mutation and protein design algorithms, where the position along the protein chain remains constant, but the amino acid types within the *Position* are allowed to change. The Residue can be divided into any number of *AtomGroup* objects, which contains the *Atom* objects. This subdivision allows for electrostatic groups or other subdivisions of atoms, such as backbone or side chain atoms.

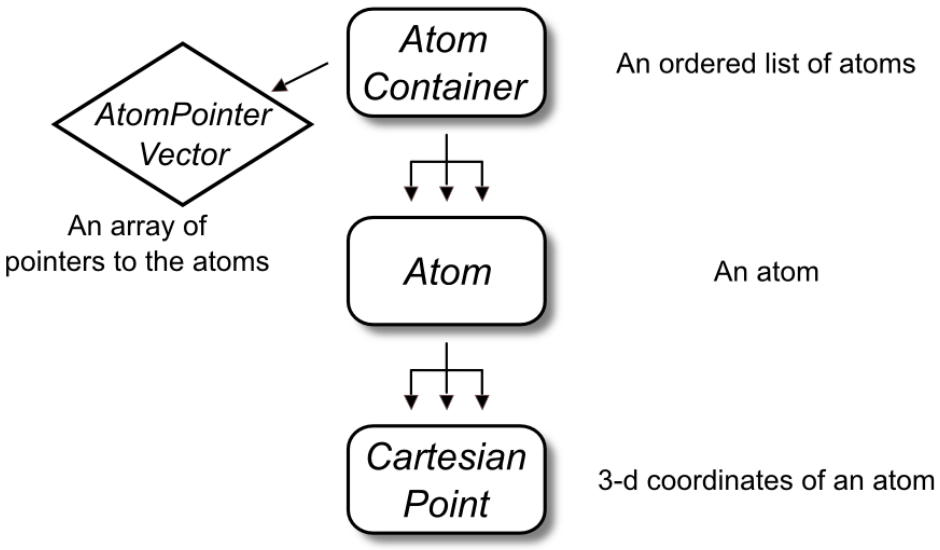


Figure 7.1: *The "flat-array" molecular container: the* AtomContainer*. The* AtomContainer *is the lightweight molecular container included in MSL. Internally, it contains an array of* Atom *pointers (as an* AtomPointerVector*), and it is ideal for tasks that require iteration among atoms. Each* Atom *contains one or more coordinates in the form of* CartesianPoint*s.*

Printing out molecular objects

MSL is created with ease of programming in mind. An example of this philosophy is MSL facilitates the printing of information contained within molecular objects, which is also extremely convenient for debugging. The following example shows how to print atoms and higher molecular containers through the « operator.

```
1 #include "AtomContainer.h"
2 #include "System.h"
3
4 using namespace MSL; // use the necessary namespaces
5 using namespace std;
6
```

```
 7 int main() {
 8   AtomContainer molAtoms; // the flat-array container
 9   molAtoms.readPdb("input.pdb");
10
11   Atom & a = molAtoms[0]; // get an atom by reference
12   cout << "Printing an Atom" << endl;
13   cout << a << endl;
14
15   cout << "Printing an AtomContainer" << endl;
16   cout << molAtoms << endl;
17
18   System sys; // the hierarchical container
19   sys.readPdb("input.pdb");
20
21   cout << "Printing a System" << endl;
22   cout << sys << endl;
23 }
```

The *Atom* prints its atom name, residue name, residue number, chain id, and the coordinates. As explained later, atoms can store more than one set of coordinates, called alternative conformations in MSL. The current conformation and total number of conformations is printed in parenthesis. The *AtomContainer* prints a list of all its atoms. The *System* prints its sequence, where each chain identifier starts the line followed by the three letter amino acid codes of its sequence. The residue numbers are included in curly brackets for the first and last residue, or when the order breaks in the primary sequence numbering.

Printing an Atom
N ALA 1 A [2.143 1.328 0.000] (conf 1/ 1) +

Printing an AtomContainer
N ALA 1 A [2.143 1.328 0.000] (conf 1/ 1) +
CA ALA 1 A [1.539 0.000 0.000] (conf 1/ 1) +
CB ALA 1 A [2.095 -0.791 1.207] (conf 1/ 1) +

C ALA 1 A [0.000 0.000 0.000] (conf 1/ 1) +
..

Printing a System
A: {1}ALA ILE VAL TYR SER LYS ARG LEU {9}ALA

Iterating through chains, positions, and atoms

All containers, even those in MSL's hierarchical representation, can operate on atoms as ordered lists. The *AtomContainer, System, Chain, Position, and Residue* all contain a list of their atoms (stored internally as an *AtomPointerVector* object, which is an array class derived through inheritance from the Standard Template Library [**?**] *vector* class). The individual atoms can be accessed using the square bracket operator ([ ]). The next example shows how to iterate and print all atoms in a System.

```
1  #include "System.h"
2
3  int main() {
4    System sys;
5    sys.readPdb("input.pdb");
6
7    for (uint i=0; i<sys.atomSize(); i++) {
8      cout << sys[i] << endl; // print the i-th atom
9    }
10 }
```

The hierarchical architecture of the System also allows iterate through positions and chains using the appropriate get function.

```
1  ..
2    for (uint i=0; i<sys.positionSize(); i++){
3      cout << sys.getPosition(i) << endl;
4      // print the i-th position
5    }
```

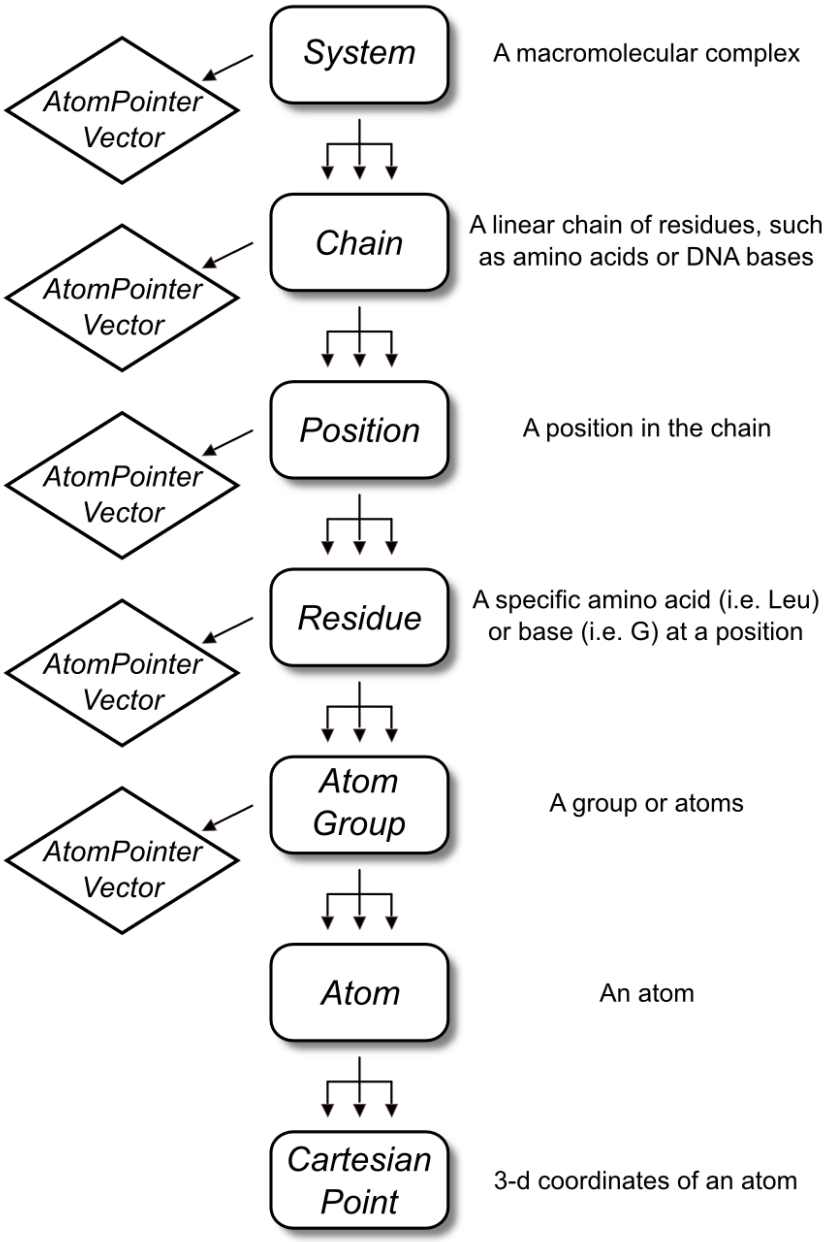


Figure 7.2: *The "hierarchical" molecular container: the* System *and its subdivisions. MSL has several levels of molecular representation, from the* System *to the* Atom*, described in the figure. Note the distinction between a* Position *(designated with a number) and the* Residue *(a specific amino acid type, such as "Leu" and "Ile"). A* Position *can have multiple residues (only one being active at any given time), which is useful for introducing mutations and protein engineering. The* Atom *objects are generated within the* AtomGroup*, but every container builds an array of pointers to the atoms (an* AtomPointerVector*) that belong to their branch. These atom pointers can be requested with a* getAtomPointers() *call and passed to external objects for processing.*

```
6    for (uint i=0; i<sys.chainSize(); i++){
7      cout << sys.getChain(i) << endl;
8      // print the i-th chain
9    }
10 }
```

Accessing atoms by id and measuring distance and angles

A powerful alternative mechanism to access an atom is through a comma-separated string identifier formed by the chain id, residue number, and atom name (i.e., "A,37,CA"). This can be done intuitively using a square bracket operator (["A,37,CA"]). The following example demonstrates how to access atoms with both the numeric index and string id operators. It also shows how to calculate geometric relationships between atoms (using the *Atom*'s functions distance, angle, and dihedral).

```
1  #include "AtomContainer.h"
2
3  int main() {
4    AtomContainer molAtoms;
5    molAtoms.readPdb("input.pdb");
6
7    // Using the operator[string _id]
8    double distance = molAtoms["A,37,CD1"].
9     distance("B,45,ND1");
10
11   // Using the operator[int _index]
12   double angle = molAtoms[7].
13    angle(molAtoms[8],molAtoms[9]);
14
15   // measure the phi angle at position A 23
16   double phi = molAtoms["A,22,C"].
17    dihedral(molAtoms["A,23,N"],
18     molAtoms["A,23,CA"],molAtoms["A,23,C"]);
```

```
19   return 0;
20 }
```

For brevity and simplicity, the examples illustrated here often omit recommended error checking code. In the above example, it would be safe to check for the existence of the atoms with the *atomSize* and *atomExists* functions before applying the measurements:

```
1  if (molAtoms.atomSize() >= 10) {
2    double dihe = molAtoms[7].dihedral(molAtoms[8],molAtoms
        [8],
3     molAtoms[9]);
4  }
5
6  if (molAtoms.atomExists("A,37,CD1") &&
7        molAtoms.atomExists("A,37,ND1")) {
8    double d = molAtoms("A,37,CD1").
9     distance(molAtoms("A,37,ND1"));
10 }
```

Communication between objects with the *AtomPointerVector*

The molecular objects store all their atoms internally as an array of atom pointers, the previously mentioned *AtomPointerVector*. The memory is allocated (and deleted) by the molecular object that created the atoms. All atom pointers of a molecular object can be obtained with the *getAtomPointers()* function.

```
1  #include "AtomContainer.h"
2
3  int main() {
4    AtomContainer molAtoms;
5    molAtoms.readPdb("input.pdb");
6
7    // get the internal array of atom pointers
```

```
8    AtomPointerVector pAtoms = molAtoms.getAtomPointers();
9
10   for (uint i=0; i<pAtoms.size(); i++) {
11     cout << *(pAtoms[i]) << endl; // print the atom
12   }
13   return 0;
14 }
```

The *AtomPointerVector* serves a fundamental purpose in MSL as the intermediary of the communication between objects that perform operation on atoms. The next section exemplifies this work-flow.

Rigid body transformations of a protein structure

The *Transforms* object is the primary tool used in MSL to operate geometric transformations. It communicates with the *AtomContainer* through an *AtomPointerVector*. As shown in the example, just five lines of code are sufficient for reading a PDB coordinate file, applying a translation and writing the new coordinates to a second PDB file. The reading and writing of the coordinate files is accomplished by the *readPdb* and *writePdb* functions of the *AtomContainer*.

```
1  #include "AtomContainer.h"
2  #include "Transforms.h"
3
4  int main() {
5    AtomContainer molAtoms;
6    molAtoms.readPdb("input.pdb");
7
8    Transforms tr;
9    tr.translate(molAtoms.getAtomPointers(),
10    CartesianPoint(3.7, 4.5, -2.1));
11
12   molAtoms.writePdb("translated.pdb");
13   return 0;
14 }
```

Atom Selections

The *AtomPointerVector* is also a mediator in another important function: the selection of subsets of atoms. The *AtomSelection* object takes an *AtomPointerVector* and a selection string (i.e., "name CA") to create subsets of atoms based on *Boolean* logic. The resulting selection is returned as another *AtomPointerVector*. The syntax adopted is similar to that of *PyMOL*, a widely used molecular visualization program [**?**]. In the following example, a selection is used to rotate only the atoms belonging to chain A. The communication between *AtomContainer*, *AtomSelection*, and *Transforms* through the *AtomPointerVector* is made explicit.

```
1  #include "AtomContainer.h"
2  #include "Transforms.h"
3  #include "AtomSelection.h"
4
5  int main() {
6    AtomContainer molAtoms;
7    molAtoms.readPdb("input.pdb");
8
9    AtomPointerVector pAtoms = molAtoms.getAtomPointers();
10
11   // initialize AtomSelection
12   AtomSelection sel(pAtoms);
13   AtomPointerVector pSelAtoms = sel.select("chain A");
14
15   // set Z as axis of rotation
16   CartesianPoint Zaxis(0.0, 0.0, 1.0);
17
18   Transforms tr;
19   tr.rotate(pSelAtoms, 90.0, Zaxis);// 90 degree
20
21   molAtoms.writePdb("rotated.pdb");
22   return 0;
23 }
```

The example above shows a simple selection string but the logic can be complex. For example, “name CA+C+N+O and chain B and resi 1-100” will select the backbone atoms of the first 100 residues of chain B. A label can be added at the beginning of the selection string (“bb_chB, name CA+C+O+N and chain B”). The label itself can then be used as part of the logic in a subsequent selection, as seen in lines 20 and 25 of the following example.

```
1  #include "AtomContainer.h"
2  #include "AtomSelection.h"
3
4  int main() {
5    AtomContainer molAtoms;
6    molAtoms.readPdb("input.pdb");
7
8    // create a selection passing all atom pointers
9    AtomSelection sel(molAtoms.getAtomPointers());
10
11   // select all CA atoms in "allCAs" and print size
12   AtomPointerVector pSelAtoms =
13    sel.select("allCAs, name CA");
14   cout << "The selection allCAs contains " <<
15    pSelAtoms.size() << " atoms" << endl;
16
17   // selections can be operated with complex logic
18   // all backbone atoms of chain B
19   AtomPointerVector pSelAtoms2 = sel.select("bb_chB,
20    name CA+C+O+N and chain B");
21
22   // a selection name can be used as part of the logic
23   // selecting backbone atoms of residue 9 on chain B
24   AtomPointerVector pSelAtoms3 = sel.select("res9B_bb,
25    bb_chB and resi 9");
26 }
```

7.2 Molecular Modeling

Altering the conformation of the molecule

MSL offers a number of methods for remodeling a protein. The coordinates of an atom can be set with the *setCoor* function.

```
1   Atom a;
2   a.setCoor(3.564, -2.143, 6.543);
```

The conformation of a protein can also be changed by rotating around bonds, changing the bond angles, and varying the bond distances. In other words, conformations can be set using a system of "internal" coordinates (bonds, angles, and dihedrals). The *Transforms* object offers functions that can be used to model a protein (*setBondDistance, setBondAngle*, and *setDihedral*). The next example shows how to alter the conformation of the backbone (ϕ/ψ angles).

```
1  #include "AtomContainer.h"
2  #include "AtomSelection.h"
3
4  int main() {
5    AtomContainer molAtoms;
6    molAtoms.readPdb("input.pdb");
7
8    // before changing the conformation we need to know
9    // what atoms are bonded to each other
10   AtomBondBuilder abb;
11   abb.buildConnections(molAtoms.getAtomPointer());
12
13   // lets change the phi/psi of residue A 37
14   Transforms tr;
15   tr.setDihedral(molAtoms("A,36,C"),molAtoms("A,37,N"),
16    molAtoms("A,37,CA"),molAtoms("A,37,C"),-62.0);
17   tr.setDihedral(molAtoms("A,37,N"),molAtoms("A,37,CA"),
18    molAtoms("A,37,C"),molAtoms("A,38,N"),-41.9);
19 }
```

Because in most cases, the intent is to move two parts of the protein relative to each other, and not simply one atom, it is necessary to have the atom connectivity information. This was done in lines 10-11 by passing the atoms to *AtomBondBuilder*, an object that creates the bond information based on the atomic distances. The connectivity information is used to update the coordinates of the atoms that are downstream of the dihedral angle (any atom between the last dihedral atom and the end of the chain). This means that a *setDihedral* invocation takes all the coordinates of the atoms downstream and multiplies them by the appropriate transformation matrix. The strategy illustrated above is straightforward to implement for small changes (i.e., edit a side chain dihedral angle). For larger conformational changes, the procedure is inefficient because most of the coordinates would be recalculated multiple times. A more economic alternative is to edit a table that stores all internal coordinates and use it to rebuild the molecule in the new conformation one atom at the time - a concept borrowed from the molecular force field and dynamics package CHARMM [Brooks et al. (1983)]. MSL implements an object for internal coordinate editing called the *ConformationEditor*.

```
1  #include "System.h"
2  #include "PDBTopologyBuilder.h"
3  #include "ConformationEditor.h"
4
5  int main() {
6
7    // build the IC table using PDBTopologyBuilder
8    System sys;
9    PDBTopologyBuilder PTB(sys,"pdb_topology.inp");
10   PTB.buildSystemFromPDB("input.pdb");
11
12   // Read the angle definitions such as phi, psi,
13   // and conformations such as "a-helix"
14   ConformationEditor CE(sys);
15   CE.readDefinitionFile("PDB_defi.inp");
```

```
16
17    // Edit LEU A 37 to have chi1=62.3 and chi2=175.4
18    CE.editIC("A,37", "N,CA,CB,CG", 62.3);
19    CE.editIC("A,37", "chi2", 175.4);
20
21    // set the backbone of A 37 in beta conformation
22    CE.editIC("A,37", "phi", -99.8);
23    CE.editIC("A,37", "psi", 122.2);
24
25    // even set entire stretches in helical conformation
26    // a-helix defines phi, psi, bond angles
27    CE.editIC("A,20-A,30", "a-helix");
28
29    // the changes are applied at once
30    CE.applyConformation();
31
32    sys.writePdb("edited.pdb");
33  }
```

Storing multiple conformations and switch between them

An extremely useful feature of MSL is the ability of storing multiple coordinates for each atom. This is done internally within the *Atom* by representing the coordinates as an array of *CartesianPoint* objects. Only one of the coordinates is active at any given time. This information is stored by a pointer, and the active coordinates can be readily switched by readdressing it. This feature allows for storing different conformations of parts or the entirety of a macromolecule. The following example demonstrates how to switch between sets of coordinates at the level of an *Atom*.

```
1  #include "AtomContainer.h"
2
3  int main() {
4    AtomContainer molAtoms;
```

```
5    molAtoms.readPdb("input.pdb");
6
7    // add two alt conformation to the first atom, A,1,N
8    molAtoms[0].addAltConformation(4.214,-6.573, 2.123);
9    molAtoms[0].addAltConformation(4.743,3.123, -1.986);
10
11   cout << molAtoms[0].getNumberOfAltConformations()
12     << " alternate conformations exist" << endl;
13   cout << "The active conformation's index is "
14     << molAtoms[0].getActiveConformation() << endl;
15   cout << molAtoms[0] << endl; // print the atom
16   molAtoms[0].setActiveConformation(2);
17   cout << "The active conformation is now " <<
18     molAtoms[0].getActiveConformation() << endl;
19   cout << molAtoms[0] << endl;
20   return 0;
21 }
```

Output (note the change of conformation number with the brackets):

The atom has 3 conformations
The active conformation's index is 0
N ALA 1 A [3.756 -6.987 2.456] (conf 1/ 3) +
The active conformation's index is now 2
N ALA 1 A [4.743 3.123 -1.986] (conf 3/ 3) +

Using a rotamer library

The multiple coordinates provide a mechanism for storing alternate conformations of side chains (or rotamers). The rotamers can be loaded on the molecule using the *SystemRotamerLoader* object, which reads a rotamer library file (line 13). The *setActiveRotamer* function of the System (line 19) can switch between rotamers by changing the active coordinates of all side chain atoms at once.

```
1  #include "System.h"
2  #include "PDBTopologyBuilder.h"
3  #include "SystemRotamerLoader.h"
4
5  int main() {
6
7    // create an empty System
8    System sys;
9    sys.readPdb("input.pdb");
10
11   // SystemRotamerLoader loads 10 rotamers on LEU A 37
12   // which are stored as alternative atom conformations
13   SystemRotamerLoader rotLoader(sys,"rotlib.txt");
14   rotLoader.loadRotamers("A,37", "LEU", 10);
15
16   // LEU at A 37 is set in all possible rotamers
17   for(int i=0; i<sys.getTotalNumberOfRotamers("A,37");
18        i++) {
19     sys.setActiveRotamer("A,37", i);
20     // do something ..
21   }
22 }
```

The rotamer library is stored in a text file with the format of the energy-based conformer library [**?**], which is distributed with MSL. Support for other formats could be easily implemented. The following example shows the format of a rotamer library file, which includes the residue name (RESI), the mobile atoms (MOBI), the definition of the degrees of freedom (DEFI), and the first three rotamers of Dunbrack's backbone independent library[**?**] for Leu (CONF).

1 RESI LEU
2 MOBI CB CG CD1 CD2
3 DEFI N CA CB CG
4 DEFI CA CB CG CD1
5 CONF 58.7 80.7

6 CONF 71.8 164.6
7 CONF 58.2 -73.6
8 ..

The file format can also include variable bond angles and bond lengths (DEFI records with two and three atoms, respectively), which is necessary for the support of a conformer library [**???**].

Temporarily storing coordinates using named buffers

In addition to the alternative coordinates mechanism, MSL supports a second distinct mechanism for storing multiple coordinates, which is essentially a "clipboard" that enables a programmer to save the coordinate, even sets of multiple alternative coordinates, in association with a string label. The label can be used later to restore the saved coordinates, replacing the current coordinates. This is useful, for example, for saving an initial state to return to after a number of moves or to restore a state if a move happen to be rejected. The next example shows how different sets of coordinates can be saved and reapplied.

```
1  #include "AtomContainer.h"
2  #include "Transforms.h"
3
4  int main() {
5    AtomContainer molAtoms;
6    molAtoms.readPdb("input.pdb");
7    // save the original coordinates to a buffer
8    molAtoms.saveCoor("original");
9
10   // move the atoms somewhere else and
11   // save the new coordinates to another buffer
12   Transforms tr;
13   tr.translate(molAtoms.getAtomPointers(),
14   CartesianPoint(3.7, 4.5, -2.1));
```

```
15    molAtoms.saveCoor("translated");
16
17    // restore the desired coordinates
18    molAtoms.applySavedCoor("original");
19    molAtoms.applySavedCoor("translated");
20
21    // remove all saved coordinates
22    molAtoms.clearSavedCoor();
23    return 0;
24 }
```

Making mutations: alternative amino acid types at the same position

MSL supports protein engineering applications, and thus allows easy substitutions of amino acid types at a position. Analogously to how an *Atom* can store and switch between alternative coordinates, a *Position* can store and switch between multiple *Residue* objects, each corresponding to a different amino acid type (see Figure 7.3). Each amino acid type can have multiple rotamers (as shown above), therefore a *System* can simultaneously contain the entire universe of side chain conformations and sequence combinations that is the base of a protein design problem. In the following simple example, we show how to switch amino acid identity after reading a PDB file. The example below uses the *PDBTopologyBuilder* to obtain the new amino acid type from a topology file (in this case, *Lys*). *Lys* and *Phe* coexist at position 37, only one of them being active at any given time, and line 26 shows how to switch back to the original amino acid type.

```
1  #include "System.h"
2  #include "PDBTopologyBuilder.h"
3  #include "SystemRotamerLoader.h"
4
5  int main() {
6
```

```
7    // read a PDB with the PDBTopologyBuilder
8    System sys;
9    sys.readPdb("input.pdb");
10
11   // Add LYS at A 37 using PDBTopologyBuilder
12   // read a topology file
13   PDBTopologyBuilder PTB(sys, "top_pdb.inp");
14   PTB.addIdentity("A,37", "LYS"); // add the LYS
15   sys.setIdentity("A,37", "LYS"); // make LYS active
16
17   // The LYS was in a default orientation.
18   // Let's load the first rotamer from a rotamer
19   // library (no promise it won't clash)
20   SystemRotamerLoader rotLoader(sys, "rotlib.txt");
21   rotLoader.loadRotamers("A,37", "LYS", 1);
22
23   sys.writePdb("mutated_to_LYS.pdb");
24
25   // revert to the original PHE
26   sys.setIdentity("A,37", "PHE");
27   sys.writePdb("original.pdb");
28 }
```

7.3 Energy Calculations

Energy functions

MSL supports a number of energy functions. The code base is designed to provide flexibility in calculating energies and to be easily expanded to include new functions. The energetics in MSL are calculated by an object called the *EnergySet*. As illustrated in Figure 7.4, the *EnergySet* contains an internal hash (stl::map) of all possible energy terms (such as covalent bond energy or van der Waals energy). Each hash element contains an array (*stl::vector*) of pointers to *Interaction* objects. Each Interaction represents

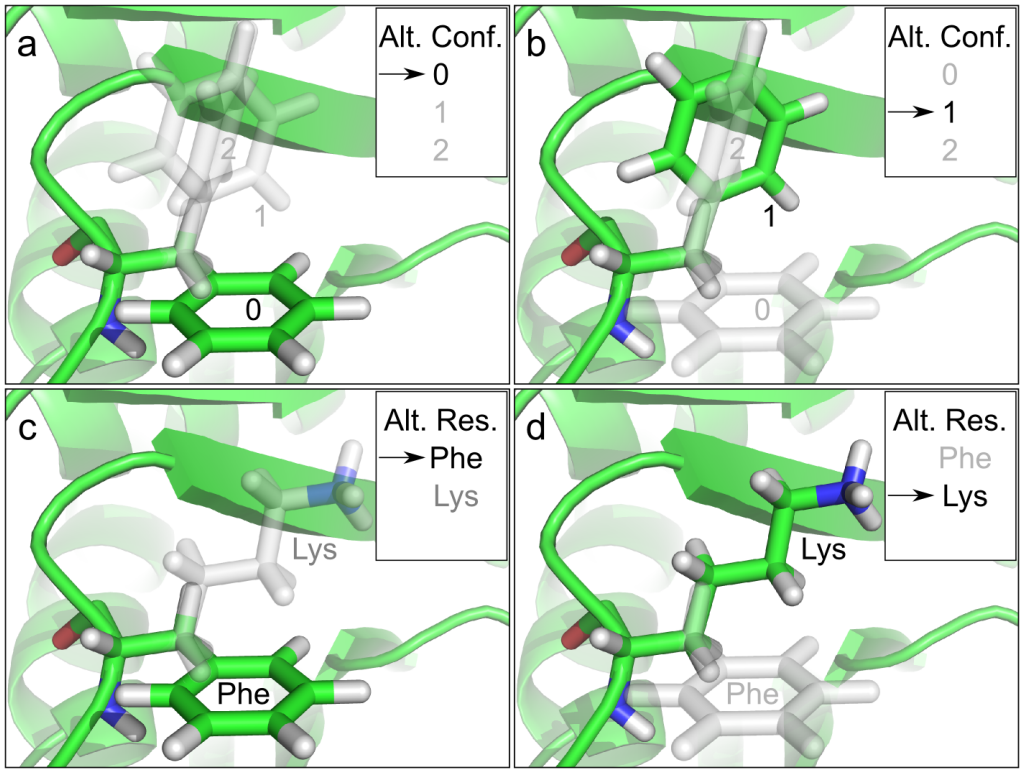


Figure 7.3: *Multiple alternative coordinates and multiple alternative identities. A unique and distinctive feature of MSL is the ability of storing multiple alternative coordinates in an* Atom *and multiple alternative amino acid identities in a* Position*. Panels (a) and (b) illustrate a case in which a* Phe *side chain has three alternative conformations, one of which active (green) and two inactive (gray). The internal redirection of a pointer switches the active conformation of the side chain's atoms from 0 to 1, changing rotamer. Panels (c) and (d) show a case in which a* Position *contains two alternative residues or amino acid identities. The redirection of a pointer switches the active amino acid identity from* Phe *to* Lys*. These two features, multiple coordinates and multiple identities, can be combined, and a* Position *can load multiple amino acid types in multiple conformations, a feature that greatly eases the development of protein design code.*

for example a bond or a van der Waals interaction between two specific atoms. The *Interaction* contains all that is necessary to calculate the energy: the pointers to the atoms involved, the parameters (i.e., for bond energy a spring constant and an equilibrium distance), and a mathematical function to calculate the energy. To calculate the total energy of a *System*, all interactions of each type are summed. It is also possible to calculate the interaction energies of specific subsets of atoms by using selections.

In the next example, we demonstrate the support for the CHARMM basic force field (vdw, coulomb, bond, angle, Urey-Bradley, dihedral, and improper terms). To compute energetics with the CHARMM force field, the *System* must be created using the *CharmmSystemBuilder*. The *CharmmSystemBuilder* reads the information necessary to build the molecule and populate the *EnergySet* from standard CHARMM topology and parameter files (line 10-11). In the example, the coordinates are read from a PDB file (note: the residue and atom names must be in CHARMM format, which is similar to the PDB convention but differs in the naming scheme of some atoms).

```
1  #include "System.h"
2  #include "CharmmSystemBuilder.h"
3  #include "AtomSelection.h"
4
5  int main() {
6
7    System sys;
8    // build the system
9    // with standard CHARMM 22 topology and parameters
10   CharmmSystemBuilder CSB(sys, "top_all22_prot.inp",
11    "par_all22_prot.inp");
12   // note, the PDB must follow CHARMM atom names
13   CSB.buildSystemFromPDB("input.pdb");
14
15   // verify that all atoms have been assigned
16   // coordinates using an AtomSelection
```

```
17    AtomSelection sel(sys.getAtomPointers());
18    // selects all atoms without coordinates
19    sel.select("noCoordinates, HASCOOR 0");
20    if (sel.selectionSize( "noCoordinates") != 0) {
21      cerr >> "Missing some coordinates! Exit" << endl;
22      exit(1); // in case of error, quit
23    }
24
25    // calculate the energies and print a summary
26    sys.calcEnergy();
27    cout << sys.getEnergySummary();
28  }
```

MSL can print a summary (line 27) that details the total energy of each terms and the number of interactions.

Interaction Type	Energy	Interactions
CHARMM_ANGL	15.788323	236
CHARMM_BOND	9.362135	131
CHARMM_DIHE	25.590364	331
CHARMM_ELEC	-55.028815	8279
CHARMM_IMPR	0.009295	21
CHARMM_U-BR	1.840063	120
CHARMM_VDW	-16.147911	8279
Total	-18.586546	17397

Table 7.1: *Energy summary of a protein.*

Subsets of energy terms can be turned off if desired. The following two lines would limit the calculations to the vdW term.

```
1  // set all inactive
2  sys.getEnergySet()->setAllTermsInactive();
3
4  // turn on VDW
5  sys.getEnergySet()->setTermActive("CHARMM_VDW");
```

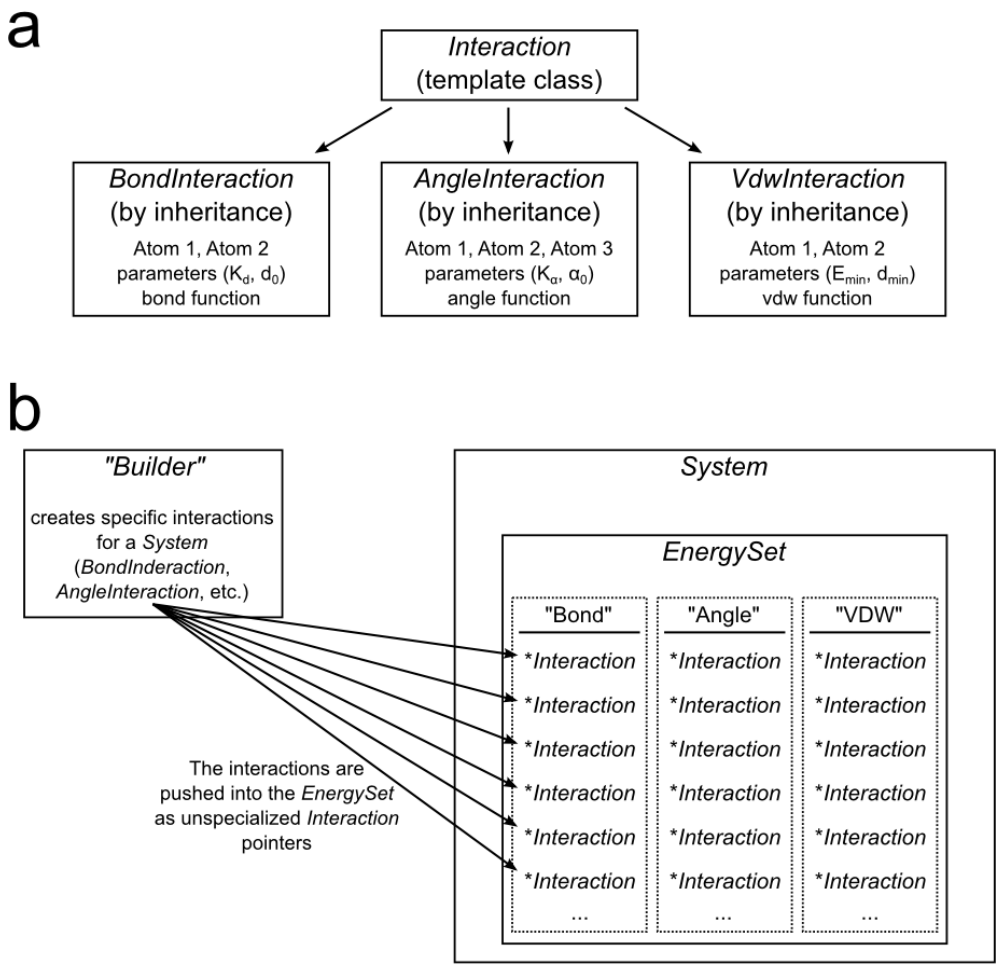


Figure 7.4: *Energetics in MSL:* Interaction *objects and the* EnergySet*. MSL is designed to allow easy addition of new energy functions (or terms). a) Energy terms inherit a generic* Interaction *class. The specialized interaction class (bond, angle, and VDW) contains pointers to the relevant atoms, all needed parameters and the mathematical formula. b) The energy calculations in MSL are performed by the* EnergySet*, which resides inside the* System*. The* EnergySet *stores the interactions in a bidimensional container. The first dimension is a hash (stl::map) in which a string is associated with each specific energy term (i.e., "Bond", "Angle", etc.). Inside the hash is an array (std::vector) of all the interactions pertinent to a specific term. To obtain the total energy of the* System*, the* EnergySet *iterates the two-dimensional structure, summing up the energy of each individual interaction. The* EnergySet *is filled with interactions using a* Builder*.*

The next example shows how to mutate a protein and then find the minimum energy among a set of 10 possible rotamers at that position. Instead of the total energy, in this case, we use selections to calculate the interaction energy between *"Lys,A,37"* and the rest of the protein. The two selection labels (created at lines 23-24) are passed to the *calcEnergy* function to calculate the interaction energy of the subsets of atoms (line 31).

```
1  #include  "System.h"
2  #include "CharmmSystemBuilder.h"
3  #include "SystemRotamerLoader.h"
4  #include "AtomSelection.h"
5
6  int main() {
7
8    System sys;
9    CharmmSystemBuilder CSB(sys, "top_all22_prot.inp",
10    "par_all22_prot.inp");
11   CSB.buildSystemFromPDB("input.pdb");
12
13   // add LYS at position A 37 and make it active
14   CSB.addIdentity("A,37", "LYS");
15   sys.setActiveIdentity("A,37", "LYS");
16
17   // Load 10 rotamers on LYS
18   SystemRotamerLoader rotLoader(sys,"rotlib.txt");
19   rotLoader.loadRotamers("A,37", "LYS", 10);
20
21   // create two selections to calculate energies
22   AtomSelection sel(sys.getAtomPointers());
23   sel.select("LYS_A_37, chain A and resi 37");
24   sel.select("allProt, all");
25
26   // find the best rotamer of LYS A 37
27   uint minRot = 0; double minE = 0;
28   for (uint i=0; i<10; i++) {
29     // set in the i-th rotamer
```

```
30      sys.setActiveRotamer("A,37", i);
31      double E = sys.calcEnergy("LYS_A_37","allProt");
32      if (i == 0 || E < minE) {
33        minRot = i; minE = E;
34      }
35    }
36    cout << "The lowest energy state is rotamer
37     index \# " << minRot << endl;
38  }
```

MSL implements the CHARMM force field, including the required 1-4 electrostatic rescaling (*e14fac*), fixed and distance-dependent dielectric constants, and distance cutoffs, with a switching function to bring the energies smoothly to zero. The energies calculated in MSL reproduce those obtained with CHARMM [Brooks et al. (1983)], as tested. In addition, MSL implements LazaridisâŁ™ EFF1 implicit solvation models [**?**] (the membrane solvation model IMM1[**?** is currently under development), a hydrogen bond term derived from SCWRL4 [Krivov et al. (2009)] the EZ membrane insertion potential, [20] knowledge-based potentials, such as DFIRE [**?**] and a single-body "baseline" term (a value associated with a single atom in a residue, useful in protein design). Weights can also be added to rescale the energy of each individual terms, if needed.

Adding new energy functions to MSL

MSL is geared toward the development of new methods, and it supports the creation and integration of new energy functions. To create a new energy function a programmer needs to code a new type of *Interaction*, which contains all that is needed, the pointers to the relevant atoms and the necessary parameters, to calculate an energy. The specialized interactions are derived using inheritance from a virtual *Interaction* class. The specialized interaction objects are added to the *EnergySet* as generic *Interaction* pointers, and thus the *EnergySet* is blind to the specific nature of the interaction and

does not need to be modified every time a new type of energy is added. To add a new term to the *EnergySet*, an external object called a "builder" (such as the *CharmmSystemBuilder* or the *HydrogenBondBuilder*) is required. The builder is the object that is responsible for the creation of all the individual interactions that are pertinent for a given *System*. This particular strategy supports the introduction of any new type of interaction without having to modify the core of MSL energetics (the *System* and the *EnergySet*).

7.4 Algorithms and Tools

Side chain optimization

MSL supports a number of algorithms for the optimization of side chain conformation that can be applied to protein modeling, docking, or protein design tasks. The *SideChainOptimizatonManager* is the object in charge of this specific task. The *SideChainOptimizatonManager* receives a *System* that already contains positions that have either multiple rotamers and/or multiple identities (known as variable positions). The object separates the interactions of the *EnergySet* into "fixed" (involving atoms that are in invariable positions), "self" (involving atoms from a single variable position), and "pairwise" (involving atoms from two variable positions). From these, it can reconstruct the total energy of any state. In the example below, the system contains three variable positions and the energy of the state defined by rotamers 3, 7, and 0 is calculated.

```
1 #include "System.h"
2 #include "CharmmSystemBuilder.h"
3 #include "SideChainOptimizationManager.h"
4
5 int main() {
6   System sys;
7   CharmmSystemBuilder CSB(sys, "top_all22_prot.inp",
8    "par_all22_prot.inp");
```

```
 9    CSB.buildSystemFromPDB("input.pdb");
10
11    // Add 10 rotamers to 3 positions
12    SystemRotamerLoader rotLoaded(sys,"rotlib.txt");
13    rotLoader.loadRotamers("A,21", "ILE", 10);
14    rotLoader.loadRotamers("A,23", "LEU", 10);
15    rotLoader.loadRotamers("A,43", "ASN", 10);
16
17    // pass the system as a pointer
18    SideChainOptimizationManager SCOM(&sys);
19    // this function pre-calculates all interactions
20    SCOM.calculateEnergies();
21
22    // get the energy of a state
23    // Eg: A21: 4th rotamer, A23 8th rot., etc
24    vector<uint> state(3, 0);
25    state[0] = 3; state[1] = 7; state[2] = 0;
26    double E = SCOM.getStateEnergy(state);
27
28    // print a summary of the state
29    cout << SCOM.getSummary(state) << endl;
30 }
```

The state is the index of the desired rotamer at each position. If there are multiple identities at one *Position*, the state would range to include the sum of all the rotamers for each identity. For example, if a *Position* has two identities (*Leu* and *Ile*) with 10 rotamers each, the state could be any number from 0 to 19 where 0-9 corresponds to the 10 *Leu* rotamers and 10-19 corresponds to the 10 *Ile* rotamers.

The *SideChainOptimizationManager* supports a number of side chain optimization algorithms that search for the global energy minimum in side chain conformational space. The current implementation includes dead end elimination (*DEE*)[**?**] (Goldstein single and pair), simulated annealing *Monte Carlo (MC)*, MC over self-consistent mean field (*SCMF*),[**?**] *Quench*,[**?**] and a linear programming formulation [**?**] (note, at the time of writing, *Quench*

and *LinearProgramming* are present as a stand-alone implementation, but they are currently being folded into the *SideChainOptimizationManager*). The algorithms can be run individually or in sequence. The next example shows how to run *DEE* followed by *SCMF/MC* search.

```
1  int main() {
2    //..
3    // Create System and add rotamers and
4    // alternate identities as in
5    // lines 1-15 of the previous example
6
7    SideChainOptimizationManager SCOM(sys);
8    SCOM.calculateEnergies();
9    // run Dead End Elimination
10   SCOM.setRunDEE(true);
11   // run SCMF/MC on the remaining rotamers
12   SCOM.setRunSCMFBiasedMC(true);
13   SCOM.runOptimizer();
14
15   // get the result
16   vector<uint> bestState = SCOM.getMCfinalState();
17
18   //print the energy summary
19   cout << SCOM.getSummary(bestState);
20
21   // set the system in the final state
22   sys.setActiveRotamers(bestState);
23   sys.writePdb("best.pdb");
24 }
```

Some of the above algorithms require precomputation of all pairwise energies between all rotamers at the variable positions (e.g., *DEE*), whereas others are amenable to computation of the energies as they are needed (e.g., *MC*). The *SideChainOptimizationManager* supports both options.

Energy Minimization

MSL can improve the energy of a structure by relaxing it to the nearest local minimum, a procedure called energy minimization. MSL takes advantages of the multidimensional minimization procedures included in the GNU Scientific Library (GSL)[**?**]. For those energy terms that have been implemented with their Cartesian partial derivatives (such as all CHARMM force field terms), MSL can minimize using faster algorithms such as *Steepest Descent* and the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)*, a quasi-Newton method. When gradient information is not available, minimization can be performed using a *Simplex Minimizer*. The *GSLMinimizer* can perform constrained as well as unconstrained energy minimization. Performing minimization in MSL is extremely simple:

```
1  #include "GSLMinimizer.h"
2  #include "CharmmSystemBuilder.h"
3
4  int main(){
5    // Read input.pdb and build a system
6    System sys;
7    CharmmSystemBuilder CSB(sys, "top_all22_prot.inp",
8     "par_all22_prot.inp");
9    CSB.buildSystemFromPDB("input.pdb"));
10
11
12   GSLMinimizer min(sys); // Initialize the minimizer
13
14   // OPTIONS: One can change the default algorithm
15   // min.setMinimizeAlgorithm(GSLMinimizer::BFGS);
16
17   // can also fix some atoms with a selection string
18   // min.setFixedAtoms("name N+C+CA+O+HN");
19
20   // Can optionally perform constrained minimization
21   // atoms with 10 kcal/(mol*$\AA^{2}$) force constant
```

```
22    // min.setContrainForce(10.0);
23    // only the backbone
24    // min.setContrainForce(10.0,"name N+C+CA+O+HN");
25
26    // Print energy summary before the minimization
27    sys.printEnergySummary();
28    // Perform the minimization
29    min.minimize();
30    // Print the energy summary after
31    sys.printEnergySummary();
32
33    sys.writePdb("output.pdb");
34  }
```

Sequence Regular Expressions

A common feature in software scripting languages is regular expressions, which can describe complex string patterns. A very simple example of using regular expressions to match multiple strings is the regular expression string "[hc]at," which matches both "hat" and "cat". Regular expressions have been used in many bioinformatic algorithms, for instance to match complicated protein sequence motifs [**?**]. A useful analysis task is to find pieces of protein structure that correspond to an interesting and/or functional sequence motif. For example, a common folding motif in membrane proteins is three amino acids of any type bracketed by two glycines (the *GxxxG* motif[**?**]). It may be interesting to find all five amino acid fragments in a database membrane protein structures that fit the *GxxxG* motif. The following example shows how MSL can accomplish this task by using MSL objects built using BOOST functionalities.

First a membrane protein structure file is read in. A single chain is extracted from the *System* (line 4). A regular expression object and search string are then created (line 10). Next, the *GxxxG* pattern of "G.3G" is

searched against the *Chain* object (line 15). A list of matching residue ranges is returned in "matchingResidueIndicies". Lastly, each match is printed out.

```
1   System sys;
2   sys.readPdb("MembraneProtein.pdb");
3
4   Chain &chA = sys.getChain("A");
5
6   // Regular Expression Object
7   RegEx re;
8
9   // Look for GxxxG
10  string regex = "G.{3}G";
11
12  // a sequence search to return the min and
13  // max indices within the Chain object
14  vector<pair<int,int> > matchingResidueIndices =
15   re.getResidueRanges(ch,regex);
16
17  // Loop over each match.
18  for (uint m = 0;m < matchingResidueIndices.size();m++){
19    // Loop over each residue for this match
20    int match = 1;
21    for (uint r = matchingResidueIndices[m].first;
22      r = matchingResidueIndices[m].second;r++){
23
24    // Get the residue
25    Residue &res = ch.getResidue(r);
26
27    // .. print out matched residues ..
28    cout << "MATCH("<< match <<"): RESIDUE:
29     "<<res.toString()<<endl;
30  }
31 }
```

Modeling Backbone Motion

Integrating backbone motion into protein design algorithms has become a major push in the field. In MSL, we have implemented three algorithms for modeling backbone motion between fixed Ca positions: cyclic coordinate descent (CCD)[**?**], Backrub,[**?**] and PDB fragment insertion[**?**] (Figure 7.5). These algorithms can also be used to insert new pieces of protein structure between two fixed Ca positions. These algorithms are Ca based, but all atoms versions can be implemented. The CCD algorithm sets the backbone conformation of a single residue to a random value, breaking the polymer chain. A set of dihedral rotations around the preceding Ca-Ca virtual bonds are discovered that both close the broken chain and produce a new conformation for the peptide. The *Backrub* algorithm works in steps that take three consecutive amino acids and rotates around their Ca-Ca virtual bonds to produce new backbone conformations. The PDB fragment method searches a structural database for stretches of amino acids that fit the geometry of the first and last two residues, but the residues in between are unique conformations. The next examples demonstrate these algorithms.

```
1  #include "CCD.h"
2  âŁ¦
3  // Read C-alpha only pdb file into a System object
4  System sys;
5  sys.readPdb("caOnly.pdb");
6
7  CCD sampleCCD; // CCD algorithm object
8
9  // Do local sampling inside CCD object
10 // 10 models, max 10 degrees
11 sampleCCD.localSample(sys.getAtomPointers(),10,10);
12
13 // System with alternative conformations
14 System newSys(sampleCCD.getAtomPointers());
15
```

```
16 // Write out all the models NMR-style
17 newSys.writePdb("ccdEnsemble.pdb",true);
```

Next, we show how one can use the Backrub algorithm:

```
 1 // Read pdb file into a System object
 2 System sys;
 3 sys.readPdb("example.pdb");
 4
 5 // A BackRub object
 6 BackRub br;
 7
 8 // Do local sampling inside BackRub object
 9 // Start, end residues and # of samples
10 br.localSample(sys.getChain(0),1,7,10);
11
12 System newSys(br.getAtomPointers());
13
14 newSys.writePdb("brEnsemble.pdb",true);
```

Next, we show how one can use the PDB fragment insertion algorithm. Although the previous two examples use transformation operations to move the backbone atoms, this algorithm searches across a database of structures to find a suitable fragment that closes the gap between two positions (called "stem" residues). For a demonstration on how to create a database of structures, we refer to the tutorial section on the MSL website.

```
 1   // Read pdb file into a System object
 2   System sys;
 3   sys.readPdb("example.pdb");
 4
 5   // Stems are kept fixed
 6   // search for segment in-between
 7   vector<string> stems;
 8   stems.push_back("A,1");
 9   stems.push_back("A,2");
10   stems.push_back("A,7");
11   stems.push_back("A,8");
```

```
12
13   // the structure database
14   PDBFragments fragDB("./tables/fragdb100.mac.db");
15
16   // Load the fragment database
17   fragDB.loadFragmentDatabase();
18
19   // Do local sampling inside PDBFragment object
20   int numMatchingFrags =
21    fragDB.searchForMatchingFragments(sys,stems);
22
23   if (numMatchingFrags > 0){
24     System newSys(fragDB.getAtomPointers());
25     ewSys.writePdb("pdbEnsemble.pdb",true);
26   }
```

7.5 Other Useful Modeling Tools and Procedures

Filling in missing backbone coordinates (backbone building from quadrilaterals)

In the following example, we illustrate a geometric algorithm implemented in MSL. The backbone building from quadrilaterals (*BBQ*) algorithm, developed by Gront et al.,[?] allows for the insertion of all backbone atoms into a structure when a Cα only trace is available, as in the *CCD* and *PDB* fragment insertion methods.

```
1  #include "BBQTable.h"
2  #include "System.h"
3
4  int main() {
5    System sys;
6    // Read a pdb file that only includes C-alpha atoms.
```

```
7    sys.readPdb("caOnly.pdb");
8    BBQTable bbq("bbq_table.dat");
9
10   // fill the missing backbone atoms for each chain
11   for(int chainNum = 0; chainNum < sys.chainSize();
12    ++chainNum) {
13     bbq.fillInMissingBBAtoms(sys.getChain(chainNum));
14   }
15
16   // Write out a pdb with all of the backbone atoms.
17   // Note: Due to the way the BBQ algorithm works, no
18   // backbone atoms will be generated for the first
19   // and last residues in a chain.
20   sys.writePdb("output.pdb");
21 }
```

Molecular alignments

A second example of a geometric algorithm is molecular alignment. MSL can be used to align two molecules and compute a RMSD. Alignments are based on quaternion math, supported by the transforms object. The following example demonstrates the alignment of two homologous proteins based on their CA atoms.

```
1  #include "AtomContainer.h"
2  #include "Transforms.h"
3  #include "AtomSelection.h"
4
5  int main() {
6    AtomContainer mol1;
7    mol1.readPdb("input1.pdb"); // read the first molecule
8    AtomContainer mol2;
9    mol2.readPdb("input2.pdb"); // read the second molecule
10
11   AtomSelection sel1(mol1.getAtomPointers());
12   // get the CAs of molecule 1
```

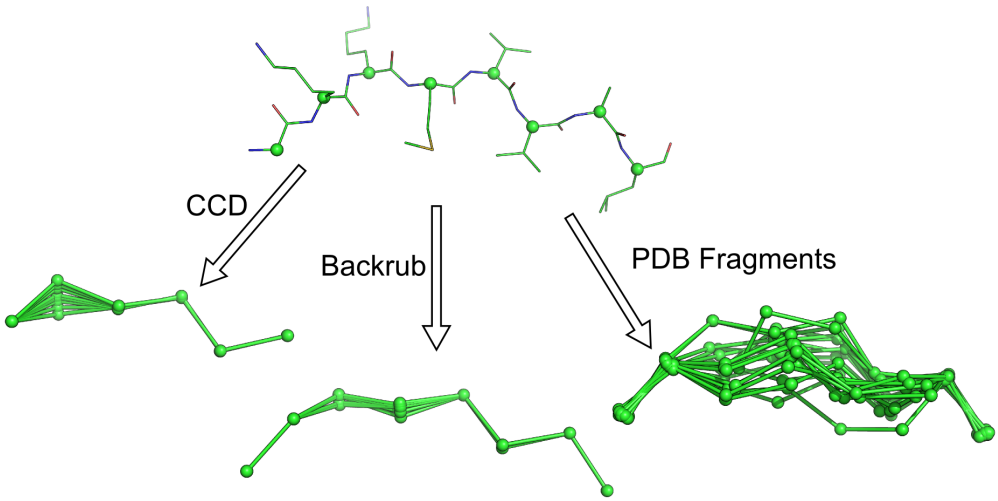


Figure 7.5: *Backbone motions implemented in MSL. The internal C-alpha atoms of an eight-residue peptide were sampled using three different algorithms implemented in MSL. The* CCD *algorithm breaks the peptide chain, then discovers a set of rotations that can close the loop (this algorithm holds the first and last C-alpha atom fixed and are not shown in the figure). The Backrub algorithm was developed to recapitulate the backbone movements found in high-resolution crystal structures and uses rotations around virtual C-alpha-C-alpha bonds. The PDB fragment method searches across a structural database and finds all fragments with the same geometry as found between the first two and last two residues of the original eight-residue peptide.*

```
13    AtomPointerVector CA1 = sel1.select("name CA");
14
15    AtomSelection sel2(mol2.getAtomPointers());
16    // get the CAs of molecule 2
17    AtomPointerVector CA2 = sel2.select("name CA");
18
19    if (CA1.size() != CA2.size()) {
20      cerr << "ERROR: # CA should be identical!" << endl;
21      exit(1);
22    }
23    cout << "Pre-alignment RMSD: " <<
```

```
24     CA1.rmsd(CA2) << endl;
25
26    Transforms tr;
27    // move molecule 2 based on the CA1/CA2 alignment
28    tr.rmsdAlingment(CA2, CA1, mol2.getAtomPointers());
29
30    cout << "Post-alignment RMSD: " <<
31     CA1.rmsd(CA2) << endl;
32
33    mol2.writePdb("input2_aligned.pdb");
34    return 0;
35  }
```

Solvent accessible surface area

The calculation of a solvent accessible surface area (SASA) is an important molecular feature that is used for analysis and modeling purposes. The *SasaCalculator* can use default element-based radii or atom-specific radii if provided (such as the CHARMM atomic radii, e.g., when the molecule is setup with the *CharmmSystemBuilder*).

```
1  #include "AtomContainer.h"
2  #include "SasaCalculator.h"
3
4  int main() {
5  AtomContainer molAtoms;
6  molAtoms.readPdb("input.pdb");
7
8  SasaCalculator SC(molAtoms.getAtomPointers());
9  SC.calcSasa();
10
11 // print a table of SASA by atom
12 SC.printSasaTable();
13
14 // print a table of SASA by residues
15 SC.printResidueSasaTable();
```

```
16  return 0;
17  }
```

7.6 Applications Distributed with MSL

Side Chain Structure Prediction and Backbone Motions

MSL is primarily a library of tools developed for allowing the implementation of new molecular modeling methods.

However, a number of programs are also distributed in the source repository and more will likely be contributed in the future. In the following section, we briefly demonstrate the performance of two of such programs, because of their general utility and because their source could be used to see many of the features previously described "in action" and as a template to create new applications. The program *repackSideChains* is a simple side chain conformation prediction program. It takes a PDB file, strips out all existing side chains (if they are present), and predicts their conformation using side chain optimization. Under the hood, the program uses a series of side chain optimization algorithms previously described. Run with default options, it starts by performing *DEE* [**?**] followed by a round of *SCMF* [**?**] on the rotamers that were not eliminated, and finally a *MC* search starting from the most probable *SCFM* rotamers (the choice of algorithms is configurable by command line arguments). We applied the program to 560 proteins backbones obtained from the structural database. The side chains were placed using a set of energy functions that included CHARMM22 bonded terms and van der Waals function, and the hydrogen bond function from *SCWRL4* [Krivov et al. (2009)], using the energy-based library[**?**] at the 85% level (1231 conformers). The program recovered the crystallographic side chain conformation of nearly 80% of all buried side chain (max 25%

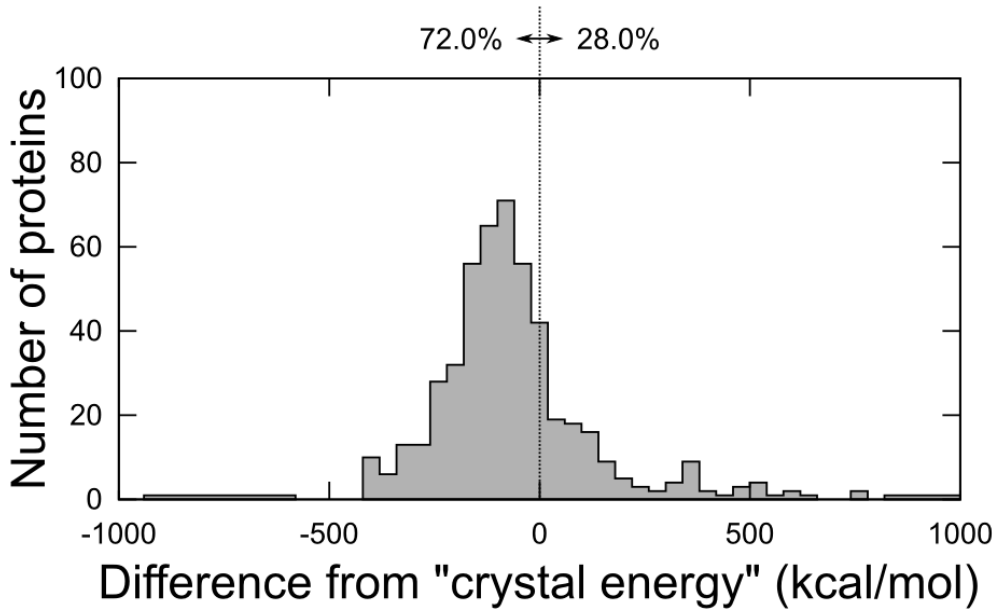


Figure 7.6: *Performance of the energy-based library in total protein repacks. Final energy after optimization of all side chains in 560 proteins, for the energy-based library. For easier comparison, energies are plotted after subtracting the energy of the minimized crystal structure ("crystal energy"). The dashed line separates the proteins that score better than the crystal energy (percentages indicated), a convenient reference under the assumption that in most cases it represents a good target for an optimization.*

SASA, $\chi 1 + \chi 2$ recovery, with a tolerance threshold of 40°), ranging from about 55% (*Ser*) to 90% (*Phe, Tyr*, and *Val*). The total hydrogen bond recovery in the same set of calculations is 60% (all side chains).

Figure 7.6 shows the distribution of the final energy of the repacked proteins compared with the energy of the minimized crystal structures, which is a reasonable reference. The program produces structures that are lower than the energy of the minimized crystal structure in 72% of the cases. The average time for performing side chain minimization was around one minute for a 100 amino acid protein, and 5-8 minutes for a 300 amino acid

protein. It should be noted that the program could also be adjusted to use different combination of energy function or rotamer/conformer libraries. The different terms of the energy functions can also be relatively weighted as desired.

The side chain prediction application *repackSideChains* offers an opportunity to compare the performance of some of MSL's capabilities against other modeling software. Side chain conformation predictions were performed in parallel on a set of 34 medium size proteins (up to 250 amino acids) with *repackSideChains* and three commonly used side chain prediction programs, *Rosetta*,[**?**] *SCWRL*,[Krivov et al. (2009)] and *Jackal*,[**?**] and the resulting χ angle recoveries and average execution times are shown in Figure 7.7. The levels of recovery are similar among the four programs, with *Rosetta* having an edge above the other programs. In term of execution time, *SCWRL* is a clear winner, while the time of the three other programs is comparable. It should be remarked here that MSL's *repackSideChains* is a relatively simple program that has not been extensively optimized to maximize side chain recovery. The program is provided as a utility and as an example for creating programs that incorporate similar functionalities. Nevertheless, its performance is in line with the average in terms of speed and is close in terms of recovery to the other benchmarks.

The availability of a variety of modeling algorithms in MSL enables the solution of complex problems. Here, we demonstrate the utility of one of the flexible backbone algorithms presented above (the *Backrub* algorithm[**?**]). We selected one of the structures in which core amino acids were not predicted correctly by *repackSideChains* (Fig. 7.8a, PDB code 1YN3). The static backbone structure has been implicated as a primary source of error in side chain repacking, and thus prediction can be ameliorated by exploration of near-native models [**?**]. We applied the program *backrubPdb* to generate an ensemble of near-native protein structures of 1YN3. Each of these near-native models was subjected to side chain optimization through *repackSideChains*,

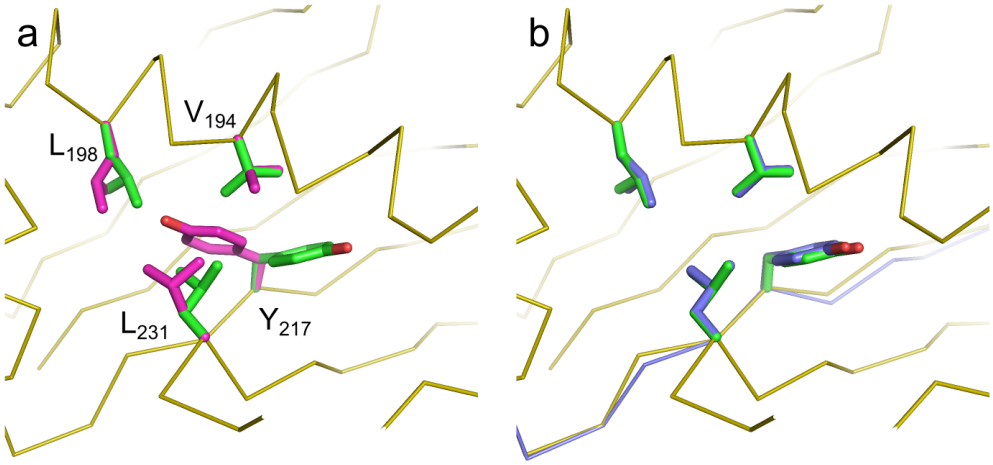


Figure 7.7: *Enhanced performance of rotamer recovery using flexible backbone modeling. In panel (a),the original backbone is shown in orange ribbons. The side chain conformations in the crystal structure of 1YN3 are shown in green. Side chain prediction with the repackSideChains program produced the conformations of four core residues displayed in magenta. In the model, the* $\chi 1$ *of Y217 assumes a g- conformation instead of the g+ conformation that is observed in the crystal structure. Concurrently, there is also a rearrangement of other three nearby positions to non-native rotamers. After the backbone has been locally relaxed with the Backrub algorithm [panel (b), in blue], the lowest energy model recovers the native conformation.*

and the results were analyzed. A slight (<0.5 Å) backbone shift resulted in a structure that was lower in energy than the fixed-backbone model and had correctly placed side chains, as illustrated in Figure 7.8b. The generation of an ensemble of backbones takes only few seconds. The *repackSideChains* and *backrubPdb* are separate standalone programs; however, it would be straight forward to include both backbone flexibility and side chain repacking capabilities into a single program. Tutorials on how to run the two programs are available on the MSL web site.

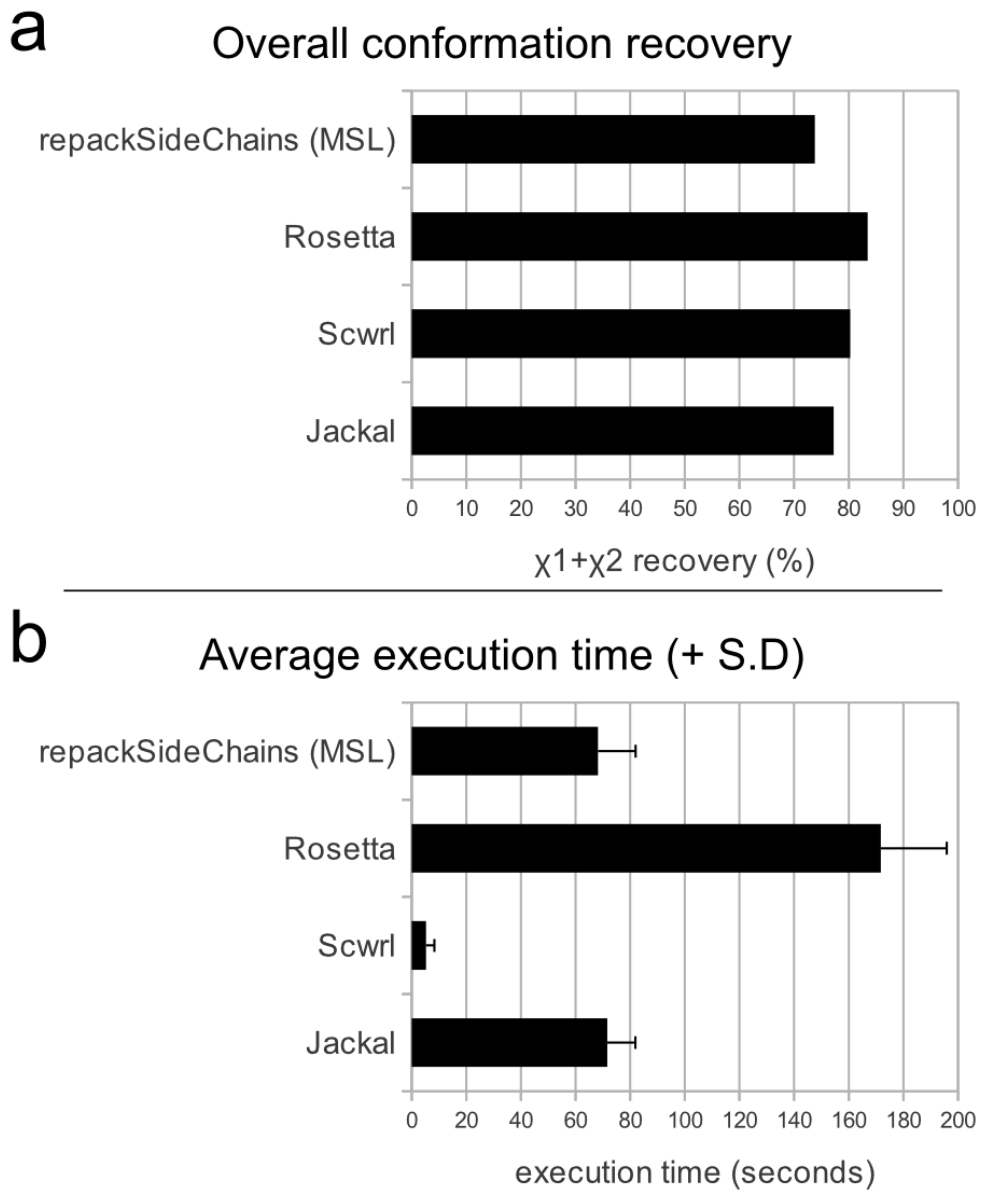


Figure 7.8: *Comparison of the performance of MSL's* repackSideChains *with other side chain prediction programs. a) Side chain recovery performance. The figure plots the overall $\chi 1 + \chi 2$ recovery of all side chains in a set of 34 proteins of size up to 250 amino acids. Only the buried side chains were considered (max 25% SASA). A side chain was considered "recovered" correctly if both $\chi 1$ and $\chi 2$ were predicted with a tolerance threshold of $\pm 20°$. b) Execution time. The histogram shows the average execution time of the 33 side chain prediction runs with the four programs. The error bar represents the standard deviation.* Rosetta *is the program with the best overall recovery in the test, whereas SCWRL is the fastest one. The performance of MSL program* repackSideChains *is in line with the other programs with respect to speed and close to the benchmarks in terms of recovery. It should be noted that* repackSideChains *is distributed as a utility and example program and it has not been extensively refined for maximum performance.*

Version Control

MSL is currently in an advanced beta state and rapidly evolving. The library is used for production work, but new features are being implemented on a regular basis. The API of most core objects is stable, although it can be occasionally revised. The codebase is kept under version control on the opensource repository SourceForge (http://mslib.svn.sourceforge.net). New versions are tagged with four-level number identifiers. At the time of writing, the current version is 1.1.2.9. The first number is the version number, currently one as the software is considered in beta. The second number is incremented with every update that significantly affects the API. The third version number is for significant changes that do not affect the API or do so only in a minor way (such as the addition of a new object). The last number is for small changes and bug fixes. All old versions are available for download from the "tags" subdirectory on the repository. By tagging MSL versions, users can put exact source code versions in publications allowing for reproduction of the result. The entire development history of MSL since the source was opened in 2009 is commented in the file src/release.h. The other function of the release.h file is to define a global variable "MSLVERSION", which is set to the current version number. This variable enables the programmer to encode a mechanism for tracking what specific MSL version was used to compile a program. In the following example, when the -v argument is provided, the programs returns the MSL version.

```
1 #include "release.h"
2
3 int main(int argc, char *argv[]) {
4   if (argc > 1 and argv[1] == "-v") {
5   // the program was called with the -v option:
6   // print the MSL version number
7   cout << "MSL version " << MSLVERSION << endl;
8   }
9   // rest of the code here
```

```
10    return 0;
11 }
```

7.7 Conclusions

MSL is a large, fully featured code base that includes over 130 objects and more than 100,000 lines of code. We have discussed a number of simple examples that demonstrate how to perform complex operations with just a few lines of code. MSL supports some unique features, such as multiple atom coordinates and multiple residue identities, a number of energy functions that are readily expandable, and other tools and algorithms that will enable rapid implementation of a large variety of molecular modeling procedures. Other MSL features that have not been presented here include coiled-coil generation, symmetric protein design, synthetic fusions of two proteins, both *PyMOL* integration and *PyMOL* script generation, integration with the statistical package R[**?**] for producing high quality plots, and use of its statistical procedures. MSL is less specialized and more comprehensive than other open-source packages that have been designed with a specific task in mind (e.g., the *EGAD* package[**?**]). Because it is modular, expandable, and largely agnostic to file formats, it can be applied to any variety of analysis and modeling problems and macromolecular types, including nucleic acids, sugars, or small molecules.

In our opinion, the most important feature of the software library is not any of the numerous methods that are currently implemented, but the fact that it merges all these capabilities together in a single platform. Most of the methods in MSL are already individually present in other programs. However, because they are integrated into a single package, they can be easily adopted by others, improved on, and mixed to create new functionalities. Therefore, any new method contributed to the MSL code base will be immediately available not only to end users but also to the

entire community of developers to build on it. We call for other interested developers to join the open-source project.

Personal Contributions

MSL has been an integral part of my research and this section is an attempt to briefly describe my specific contributions to MSL.

Energetics in MSL is implemented using the potential functions from the CHARMM22 [Brooks et al. (1983)] force field and the hydrogen bonding potential from the SCWRL4 program [Krivov et al. (2009)]. I contributed the initial implementation of these energy functions as a hierarchy of *Interaction*s depicted in Figure 7.4. The *CharmmSystemBuilder* and *HydrogenBondBuilder* which create the list of *Interaction*s for a given molecule were also my contributions.

Side chain modeling in MSL is facilitated by the *SideChainOptimizationManager* and associated classes. I have made significant enhancements to this class including the addition of the greedy *rotamer trials* algorithm. I have also contributed to the *RotamerLibrary*, and *SystemRotamerLoader* which are used to manipulate side chain libraries in order to facilitate the creation of the EBL. I have also contributed a number of applications such as the *createEnergyTable*, *createEBL* which enable the creation of the customizable energy-based libraries. My *repackSideChains* and *getChiRecovery* programs, also distributed with MSL enable the use and evaluation of side chain optimization protocols.

Molecular surface area and other utilities have also been my contributions to the MSL. Molecular surface area calculations are important in protein design and structure prediction and can be calculated using the

SasaCalculator in MSL. I have also added the *FormatConverter* and which enables the interconversion of molecular data across multiple formats. I have also contributed several other enhancements and bug fixes during the course of my research which have become integral components of MSL.

A SUPPORTING INFORMATION FOR EBL

A.1 Calculation of the conformer/environment interactions and creation of the energy tables

The energy data used to derive the energy-based library in Chapter 2 was collected as follows. For each amino acid type, the native side chain of each of the M environments was remodeled as each one of the N conformers and their interaction energy was calculated, producing a matrix of NxM energies. The side chain reconstruction was performed from internal coordinates using the distance, the angle and the dihedral angle relationships relative to three preceding atoms (see Figures A.1 and A.2).

```
1  CartesianPoint CartesianGeometry::buildRadians(const
       CartesianPoint & _distAtom, const CartesianPoint &
       _angleAtom, const CartesianPoint & _dihedralAtom,
2  const double & _distance, const double & _angle, const
       double & _dihedral) {
3    /*************************************************
4    * This function sets the coordinates of a cartesian
5    * point A based on:
6    *  - the position of atoms B C D
7    *  - the distance of A from B
8    *  - the angle A-B-C
9    *  - the dihedral A-B-C-D
10   *
11   *                     A
12   *                      \
13   *                       B--C
14   *                            \
15   *                             D
16   *
```

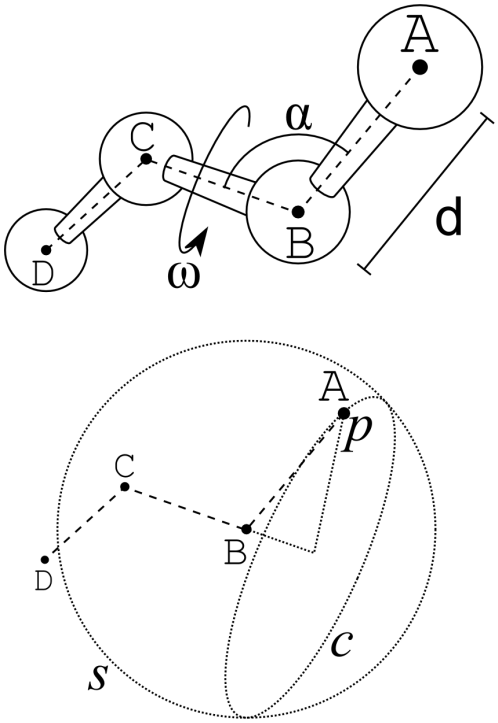


Figure A.1: *Building from internal coordinates. C++ code of the building function buildRadians from the object CartesianGeometry.cpp in MSL [Kulp et al. (2012)]. This code is used to build the conformers from the internal coordinates defined in the library file (Figure A.2). The example in the picture conceptually illustrates how the coordinates of atom A are derived relatively to the position of atoms B, C and D. The AB distance d puts the atom A on a sphere s centered around B. The ABC angle* α *restricts the atoms to a circle c. Finally, the ABCD dihedral* ω *sets atom A to point p. The MSL code is freely available for download at http://msl-libraries.org.*

```
17    * Angles are in RADIANS
18    *
19    * Arguments:
20    *   returned CartesianPoint = atom A
21    *   _distAtom               = atom B
22    *   _angleAtom              = atom C
23    *   _dihedralAtom           = atom D
24    *   _distance               = A-B distance
25    *   _angle                  = A-B-C angle
26    *   _dihedral               = A-B-C-D dihedral
27    *
28    * NOTE: no check points are coded but the distance and the
29    * angle should not be zero and the atoms should not
30    * be overlapping or B-C-D be a 180 angle
31    *
32    ************************************************/
33    // unit vector from _distAtom to _angleAtom (B - C)
34    CartesianPoint uCB = (_distAtom - _angleAtom).getUnit();
35    // distance from _angleAtom to _dihedralAtom (C - D)
36    CartesianPoint dDC = _angleAtom - _dihedralAtom;
37    double angle2 = M_PI - _angle;
38    double dihe2 = M_PI + _dihedral;
39    double rsin = _distance * sin(angle2);
40    double rcos = _distance * cos(angle2);
41    double rsinsin = rsin * sin(dihe2);
42    double rsincos = rsin * cos(dihe2);
43    /***************************************
44    * The following creates the resulting position by
45    * adding three orthogonal components:
46    *
47    * - Set a component in the B-C direction ...
48    *         (uCB * rcos) + ...
49    *
50    * - ... add a component on the B-C-D plane (note * denotes
          dot product used between two vectors) ...
51    *         ... + (dDC - (uCB * (dDC * uCB))).getUnit() *
          rsincos + ...
```

```
52    *
53    * - ... add a component orgogonal to the B-C-D plane
54    *         ... + (uCB.cross(dDC)).getUnit() * rsinsin + ...
55    *
56    * - ... finally, translate the point by the position of
          atom B
57    *         ... + _distAtom
58    *
59    ****************************************/
60    return (uCB * rcos)  +  ((dDC - (uCB * (dDC * uCB))).
          getUnit() * rsincos)  +  ((uCB.cross(dDC)).getUnit() *
          rsinsin)  +  _distAtom;
61  }
```

The interaction energies included the internal interactions of the side chain (including the bonded terms) and the interactions of the side chain with all other atoms. The energies were calculated according to the CHARMM 22 force field [MacKerell et al. (1998)] (bond, angle, urey-bradley, dihedral, improper, van der Waals, and Coulomb electrostatics with an R-dependent dielectric), plus an additional hydrogen bond term as described in the program SCWRL4 [Krivov et al. (2009)]. The nonbonded interactions were calculated with a distance dependent cutoff of 10 Å , using a switching function (cut-on 9 Å , cut-off 10 Å). The calculations were repeated with the van der Waals radii rescaled to 95% and 90% of their parameter 22 size. The table of energies were computed in three different conditions: (1) with full electrostatics and no hydrogen bonding term, (2) with a full hydrogen bonding term and no electrostatics and, (3) with electrostatics plus a hydrogen bonding term, both rescaled to 50%. The NxM energy matrix was converted into an NxM boolean matrix in which a true value indicated that an elementâĽ™s energy was below a given threshold, and thus the environment was satisfied. Because the best energy achievable in each environment varied substantially, an environment dependent threshold was adopted. The threshold was calculated in the following way: first, the

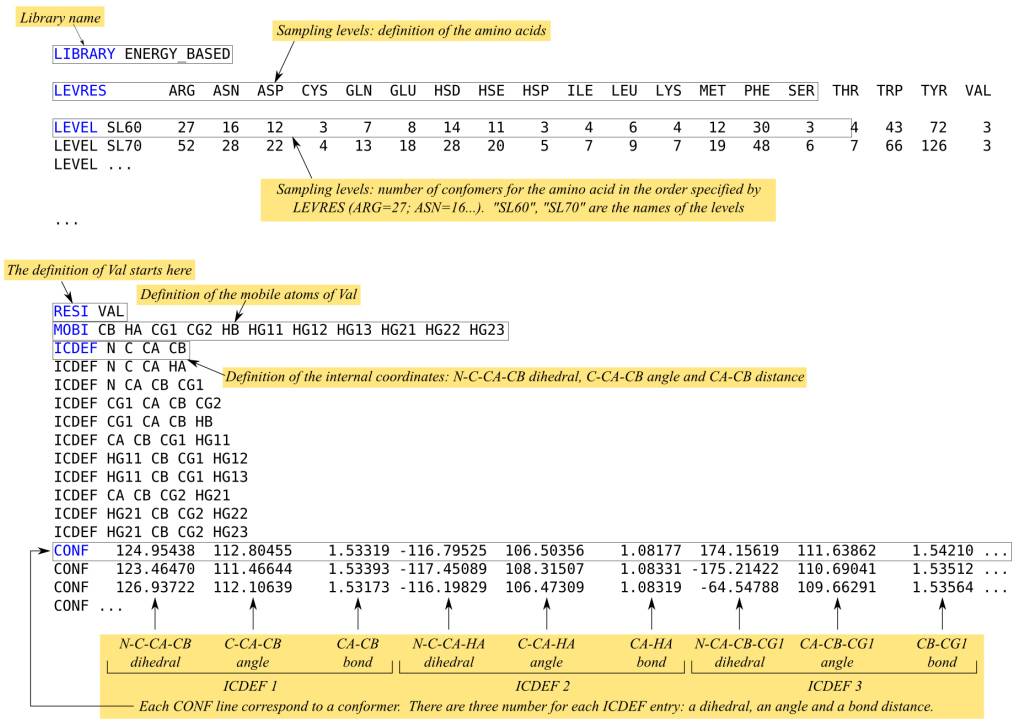


Figure A.2: *File format of the conformer library. The library is distributed in plain text file format. The file starts a library name (LIBRARY). The next section defines the sampling levels (LEVRES: definition of the amino acids; LEVEL: number of conformers for each amino acid in each level). The various amino acids are defined in a section that starts with a RESI declaration, followed by a definition of the mobile atoms (MOBI), a list of the internal coordinates (bonds, angles, dihedral) used to define the conformers (one for each mobile atom), and finally the list of internal coordinates of each conformers (CONF). Each mobile atom can be placed in cartesian coodinates using the logic explained in the code listing above. For example, the CB atom of* Val *is built relative to the position of N, C and CA atoms using the CA-CB bond distance, the C-CA-CB angle and the N-C-CA-CB dihedral (an improper dihedral, in this case). The CG1 atom is built using the CB-CG1 distance, the CA-CB-CG angle and the N-CA-CB-CG1 dihedral (a canonical dihedral).*

best interaction energy in the row (all conformers in the environment) was identified. All the elements of the row were adjusted by subtracting the best energy. The distribution of all the adjusted energy in the entire table was plotted. As shown in Figure A.3, these distributions have a typical peak near the best energy. This peak represents conformers that are near the very best energy. The distance of the modal peak from the minimum is thus indicative of the typical energy spread of conformers that fit the environments favorably.For this reason, we chose the mode of this peak as the threshold to be added to the best environment energy.For example, *Arg* displays a peak at 7.0 kcal mol^{-1} from the best energy, thus the threshold for each *Arg* environment was set 7.0 kcal mol^{-1} above the best energy for that environment.

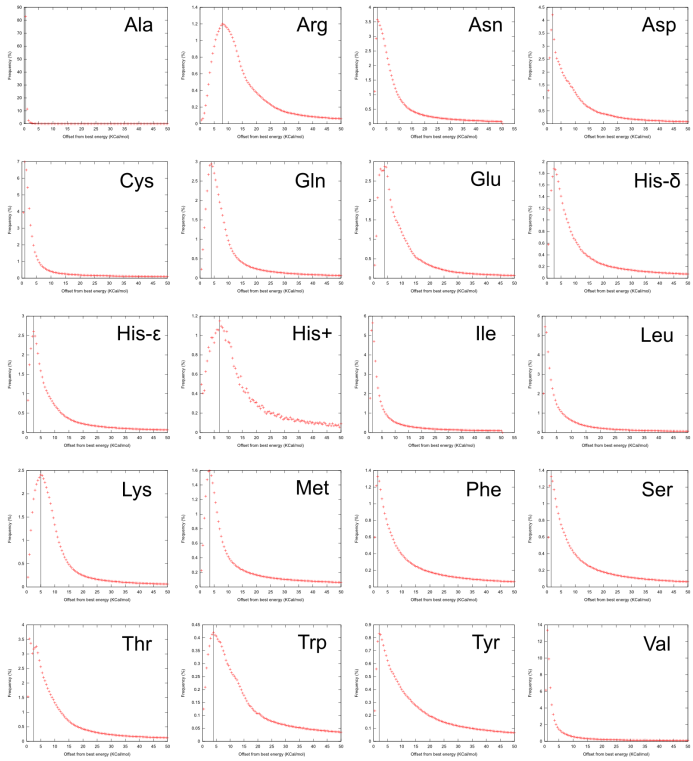


Figure A.3: *Distribution of conformer/environment interaction energies. The graphs show the energy distribution of all conformers relative to the best energy in each environment. The mode of the peaks (marked with a vertical line) was chosen as the tolerance threshold in the conformer sorting procedure (see methods).*

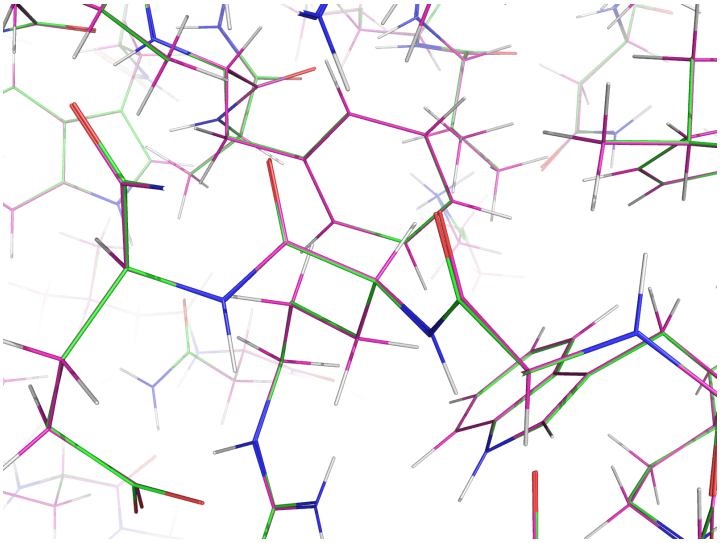


Figure A.4: *Effect of constrained minimization. Superimposed structure of a crystallographic model and the model after constrained minimization. Minimization was necessary to homogenize the bond lengths, which can be affected by the refinement procedures, and to resolve any small clashes in poorly refined regions. However, minimization was performed with strong harmonic constraints to prevent distortions of the experimental structure. The average R.M.S.D. of minimized and unminimized models is 0.052 ± 0.01*Å*, and all side chains retain near native conformation. In the figure a detail of the original (green) and minimized (magenta) structure is shown as a typical example (PDB 1FBQ, RMSD 0.05*Å*).*

Amino acid type	BBD5x	SCL	XCL
Arg	75 x 5 = 375	415	334
Asn	36 x 5 = 180	48	31
Asp	18 x 5 = 90	35	19
Cys	3 x 3 = 9	4	4
Gln	108 x 5 = 540	148	107
Glu	54 x 5 = 270	108	82
His-δ	36 x 5 = 180	62	23
His-ϵ	36 x 5 = 180	62	23
His-p	36 x 5 = 180	62	23
Ile	9 x 5 = 45	18	23
Leu	9 x 5 = 45	36	19
Lys	73 x 5 = 365	195	312
Met	27 x 5 = 135	85	71
Phe	18 x 5 = 90	55	12
Ser	3 x 9 x 3 = 81	8	7
Thr	3 x 9 x 3 = 81	5	6
Trp	36 x 5 = 180	105	18
Tyr	18 x 8 x 5 = 720	88	16
Val	3 x 3 = 9	8	4
Total	3755	1547	1134

Table A.1: *Number of conformers in the benchmark libraries. The Backbone Dependent library was expanded 5 fold (main rotamer, ± 1 standard deviation in χ1, and ± 1 standard deviation in χ2). For amino acids that have only one χ angle the library was expanded 3 fold (main rotamer and ± 1 standard deviation in χ1). The dihedral relative to the hydrogen atom of hydroxyl groups of Ser and Thr was sampled at the canonical -60°, 180° and +60° minima, each one expanded by ±30° (9 total steps) in the BBD5x. The dihedral relative to the hydrogen atom of hydroxyl groups of Tyr was sampled every 45° (8 steps) in the BBD5x*

Amino acid	No. of conformers
Arg	5000
Asn	5000
Asp	5000
Cys	1780
Gln	5000
Glu	5000
His-δ	2906
His-ϵ	4221
His-p	542
Ile	5000
Leu	5000
Lys	5000
Met	5000
Phe	5000
Ser	5000
Thr	5000
Trp	5000
Tyr	5000
Val	3916

Table A.2: *Number of conformers in the energy-based library*

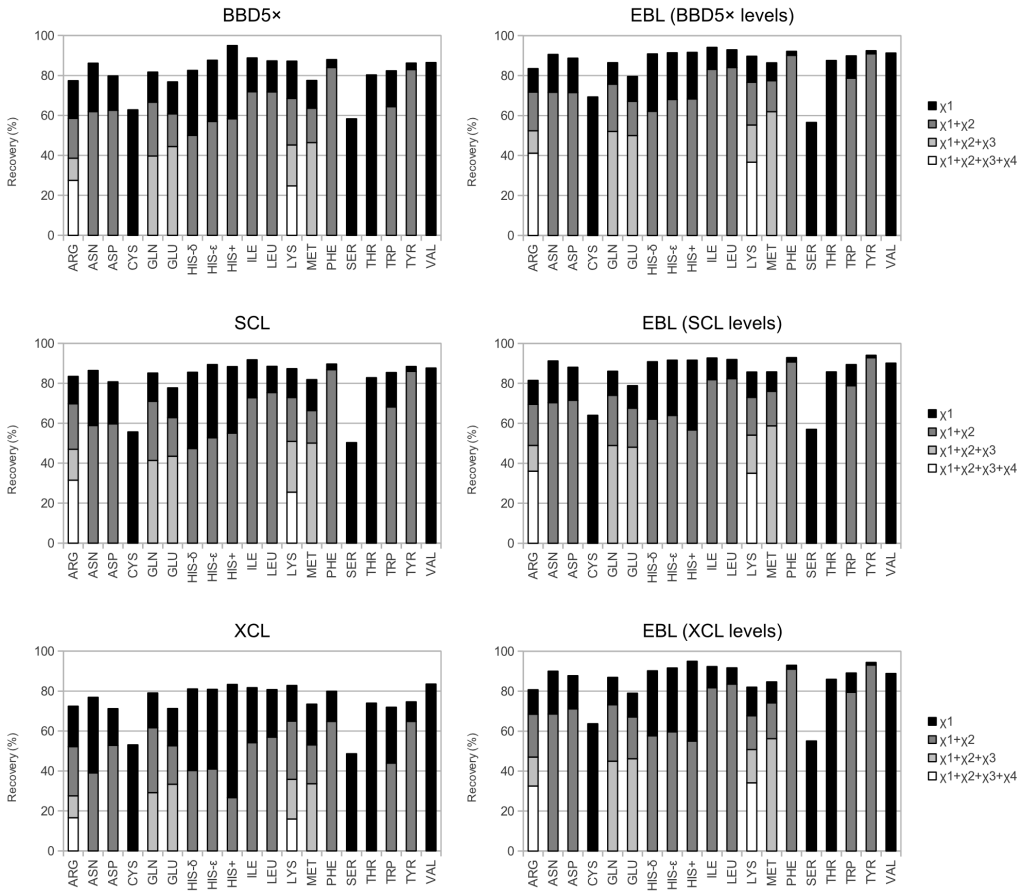


Figure A.5: *Recovery of the crystallographic side chain conformation in total protein repacks for all side chains subdivided by χ1, χ1+χ2, χ1+χ2+χ3, χ1+χ2+χ3+χ4 recoveries. The data is the same as Fig. 2.8, which compares only χ1+χ2 recoveries. Recoveries obtained with the EBL are compared to the BBD5x, the SCL and the XCL. The EBL was evaluated at a sampling level comparable to the benchmark (EBL-lev in 2.7, see legend and main text). The height of the bars is meant to be cumulative (in other words, χ1 recovery > χ1+χ2 > χ1+χ2+χ3 > χ1+χ2+χ3+χ4).*

Level	ARG	ASN	ASP	CYS	GLN	GLU	HIS-δ	HIS-ϵ	HIS-P	ILE	LEU	LYS	MET	PHE	SER	THR	TRP	TYR	VAL	Total
60%	27	16	12	3	7	8	14	11	3	4	6	4	12	30	3	4	43	72	3	282
70%	52	28	22	4	13	18	28	20	5	7	9	7	19	48	6	7	66	126	3	488
75%	73	38	31	6	19	28	39	26	6	9	13	10	25	60	8	10	83	167	4	655
80%	102	51	43	7	27	41	52	37	9	11	17	16	33	76	13	13	111	222	5	886
82.5%	123	60	49	8	33	50	63	44	10	13	21	21	39	87	16	16	126	254	6	1039
85%	149	70	59	9	40	61	76	53	13	16	26	26	47	100	21	20	144	294	7	1231
87.5%	177	83	70	11	49	73	95	63	17	20	32	34	57	116	27	27	164	341	8	1464
90%	222	100	86	16	61	90	121	76	22	26	39	44	72	138	34	34	196	408	10	1795
92.5%	273	122	106	21	77	111	152	94	30	35	52	58	93	169	43	44	237	487	13	2217
95%	354	154	138	32	106	144	226	125	44	50	70	81	126	214	57	62	298	613	16	2910
96%	397	173	152	39	121	163	293	150	51	60	82	94	144	242	65	74	340	687	19	3346
97%	449	201	173	48	143	182	424	172	71	78	98	111	176	278	74	89	382	767	22	3938
98%	498	227	201	71	177	215	649	255	89	108	124	132	206	342	88	108	430	838	28	4786
99%	589	292	231	100	233	261	1107	569	140	165	183	184	281	428	111	152	663	1252	44	6985

Table A.3: *Suggested number of rotamers at each sampling level.The table reports the suggested number of conformers for each amino acid type at different sampling levels. The levels have been obtained by matching the efficiency of repacking single side chain environments. For example, the top 27 Arg conformers satisfy on average 60% of Arg protein environments, and 589 are required to satisfy 99% environments. The top 16 and 292 conformers of Asn provide roughly the same chances to satisfy Asn protein environments. The conformers of His are created separately for the three protonation states, indicated here using the following naming convention: His-*δ*, protonated in ND1; His-*ϵ*, protonated in NE2; His-p, doubly protonated, charged.*

REFERENCES

Altschul, S F, W Gish, W Miller, E W Myers, and D J Lipman. 1990. Basic local alignment search tool. *Journal of molecular biology* 215(3):403–410. PMID: 2231712.

Anfinsen, Christian B. 1973. Principles that govern the folding of protein chains. *Science* 181(4096):223–230. `http://www.sciencemag.org/content/181/4096/223.full.pdf`.

Beauchamp, Kyle A., Yu-Shan Lin, Rhiju Das, and Vijay S. Pande. 2012. Are protein force fields getting better? a systematic benchmark on 524 diverse NMR measurements. *Journal of Chemical Theory and Computation* 8(4):1409–1414.

Berendsen, H. J. C., D. van der Spoel, and R. van Drunen. 1995. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications* 91(1–3):43–56.

Bonneau, R, J Tsai, I Ruczinski, D Chivian, C Rohl, C E Strauss, and D Baker. 2001. Rosetta in CASP4: progress in ab initio protein structure prediction. *Proteins* Suppl 5:119–126. PMID: 11835488.

Bordoli, Lorenza, Florian Kiefer, Konstantin Arnold, Pascal Benkert, James Battey, and Torsten Schwede. 2008. Protein structure homology modeling using SWISS-MODEL workspace. *Nature Protocols* 4(1):1–13.

Brooks, Bernard R., Robert E. Bruccoleri, Barry D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. 1983. CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry* 4(2):187–217.

Carpenter, Elisabeth P, Konstantinos Beis, Alexander D Cameron, and So Iwata. 2008. Overcoming the challenges of membrane protein crystal-

lography. *Current Opinion in Structural Biology* 18(5):581–586. PMID: 18674618 PMCID: PMC2580798.

Chakrabarti, Pinak, and Debnath Pal. 2001. The interrelationships of side-chain and main-chain conformations in proteins. *Progress in Biophysics and Molecular Biology* 76(1–2):1–102.

Consortium, International Human Genome Sequencing. 2004. Finishing the euchromatic sequence of the human genome. *Nature* 431(7011):931–945.

Cornell, Wendy D., Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell, and Peter A. Kollman. 1995. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society* 117(19):5179–5197.

Crick, F. 1970. Central dogma of molecular biology. *Nature* 227(5258): 561–563. PMID: 4913914.

Dunbrack, Jr, Roland L. 2002. Rotamer libraries in the 21st century. *Current opinion in structural biology* 12(4):431–440. PMID: 12163064.

Dunbrack, Roland L., and Martin Karplus. 1994. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Structural & Molecular Biology* 1(5):334–340.

Endres, Nicholas F, Kate Engel, Rahul Das, Erika Kovacs, and John Kuriyan. 2011. Regulation of the catalytic activity of the EGF receptor. *Current opinion in structural biology* 21(6):777–784. PMID: 21868214.

Fagerberg, Linn, Kalle Jonasson, Gunnar von Heijne, Mathias Uhlen, and Lisa Berglund. 2010. Prediction of the human membrane proteome. *Proteomics* 10(6):1141–1149. PMID: 20175080.

Goldsmith-Fischman, Sharon, and Barry Honig. 2003. Structural genomics: Computational methods for structure analysis. *Protein Science* 12(9): 1813âŁ“1821.

Kendrew, J. C., G. Bodo, H. M. Dintzis, R. G. Parrish, H. Wyckoff, and D. C. Phillips. 1958. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature* 181(4610):662–666.

Krivov, Georgii G, Maxim V Shapovalov, and Jr Dunbrack, Roland L. 2009. Improved prediction of protein side-chain conformations with SCWRL4. *Proteins* 77(4):778–795. PMID: 19603484.

Krogh, A, B Larsson, G von Heijne, and E L Sonnhammer. 2001. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of molecular biology* 305(3):567–580. PMID: 11152613.

Kryshtafovych, Andriy, Krzysztof Fidelis, and John Moult. 2013. CASP10 results compared to those of previous CASP experiments. *Proteins: Structure, Function, and Bioinformatics*.

Kulp, Daniel W., Sabareesh Subramaniam, Jason E. Donald, Brett T. Hannigan, Benjamin K. Mueller, Gevorg Grigoryan, and Alessandro Senes. 2012. Structural informatics, modeling, and design with an open-source molecular software library (MSL). *Journal of Computational Chemistry* 33(20):1645–1661.

MacKerell, A. D., D. Bashford, Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. 1998. All-atom empirical potential for molecular modeling

and dynamics studies of proteins. *The Journal of Physical Chemistry B* 102(18):3586–3616.

Moreira, Irina S, Pedro A Fernandes, and Maria J Ramos. 2010. Protein-protein docking dealing with the unknown. *Journal of computational chemistry* 31(2):317–342. PMID: 19462412.

Nayeem, Akbar, Doree Sitkoff, and Stanley Krystek. 2006. A comparative study of available software for high-accuracy homology modeling: From sequence alignments to structural models. *Protein Science* 15(4):808–824.

Onuchic, Jose Nelson, Zaida Luthey-Schulten, and Peter G. Wolynes. 1997. THEORY OF PROTEIN FOLDING: the energy landscape perspective. *Annual Review of Physical Chemistry* 48(1):545–600. PMID: 9348663.

Phillips, James C., Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant KalÃ©, and Klaus Schulten. 2005. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry* 26(16):1781âŁ“1802.

Ponder, Jay W, and David A Case. 2003. Force fields for protein simulations. *Advances in protein chemistry* 66:27–85. PMID: 14631816.

Ramachandran, G. N., C. Ramakrishnan, and V. Sasisekharan. 1963. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology* 7(1):95–99.

Ritchie, David W. 2008. Recent progress and future directions in protein-protein docking. *Current protein & peptide science* 9(1):1–15. PMID: 18336319.

Russ, William P, and Rama Ranganathan. 2002. Knowledge-based potential functions in protein design. *Current opinion in structural biology* 12(4): 447–452.

Samish, Ilan, Christopher M. MacDermaid, Jose Manuel Perez-Aguilar, and Jeffery G. Saven. 2011. Theoretical and computational protein design. *Annual Review of Physical Chemistry* 62(1):129–149. PMID: 21128762.

Schlessinger, J. 2000. Cell signaling by receptor tyrosine kinases. *Cell* 103(2):211–225. PMID: 11057895.

Shapovalov, Maxim V, and Jr Dunbrack, Roland L. 2011. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure (London, England: 1993)* 19(6):844–858. PMID: 21645855.

Shaw, David E., Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, IstvÃ¡n KolossvÃ¡ry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. 2007. Anton, a special-purpose machine for molecular dynamics simulation. In *Proceedings of the 34th annual international symposium on computer architecture*, 1–12. ISCA '07, New York, NY, USA: ACM.

Simons, K T, C Kooperberg, E Huang, and D Baker. 1997. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *Journal of molecular biology* 268(1):209–225. PMID: 9149153.

Staden, R. 1979. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research* 6(7):2601–2610. PMID: 461197 PMCID: PMC327874.

Street, Arthur G, and Stephen L Mayo. 1999. Computational protein design. *Structure* 7(5):R105–R109.

Vasquez, Maximiliano. 1995. An evaluvation of discrete and continuum search techniques for conformational analysis of side chains in proteins. *Biopolymers* 36(1):53–70.

Wallner, Bjorn, and Arne Elofsson. 2005. All are not equal: A benchmark of different homology modeling programs. *Protein Science* 14(5):1315–1327.

Wang, Chu, Ora Schueler-Furman, and David Baker. 2005. Improved side-chain modeling for protein-protein docking. *Protein science: a publication of the Protein Society* 14(5):1328–1339. PMID: 15802647.

Wang, Junmei, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. 2004. Development and testing of a general amber force field. *Journal of Computational Chemistry* 25(9):1157–1174.

Wang, Qiang, Adrian A. Canutescu, and Roland L. Dunbrack. 2008. SCWRL and MolIDE: computer programs for side-chain conformation prediction and homology modeling. *Nature Protocols* 3(12):1832–1847.

Wuthrich, Kurt. 2001. The way to NMR structures of proteins. *Nature Structural & Molecular Biology* 8(11):923–925.

Xiang, Zhexin. 2006. Advances in homology protein structure modeling. *Current protein & peptide science* 7(3):217–227. PMID: 16787261 PMCID: PMC1839925.

Zimmerman, S. Scott, Marcia S. Pottle, George Nemethy, and Harold A. Scheraga. 1977. Conformational analysis of the 20 naturally occurring amino acid residues using ECEPP. *Macromolecules* 10(1):1–9.