SAMPLING-BASED EXPLORATION OF THE FOLDED STATE OF A PROTEIN UNDER KINEMATIC AND GEOMETRIC CONSTRAINTS

A DISSERTATION
SUBMITTED TO THE PROGRAM IN BIOMEDICAL INFORMATICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Zhen (Peggy) Yao
December 2010
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jean-Claude Latombe, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Russ Altman

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Levitt

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

The conformational selection model is emerging as one of the best options to model protein flexibility in ligand binding. According to this model, a protein spans an ensemble of rapidly inter-converting folded conformations and a ligand selects the most favorable conformations to bind to from this ensemble. However, only a small number of folded conformations can be obtained experimentally (e.g., from X-ray crystallography, NMR spectrometry, or cryo-electron microscopy), while existing computational techniques (e.g., Molecular Dynamics and Monte-Carlo simulation) are often too expensive to sample conformations broadly distributed across the folded state of a protein.

This dissertation focuses on designing new computational methods to efficiently explore the folded state of a protein using kino-geometric sampling approach, i.e., an approach that only considers kinematic constraints (fixed bond lengths and angles) and geometric constraints (no collision of atoms modeled as hard spheres). This approach heavily relies on the fact that in a folded state protein atoms are densely packed.

The kino-geometric approach is first applied to loop sampling, where the goal is to explore the conformation space of a loop between two secondary structure elements (helices and/or strands) assuming that the conformation of the rest of the protein is fixed and given as input. Next, it is applied to the considerably more complex problem of sampling folded conformations of an entire protein. In both cases, the proposed methods are validated by showing that the kino-geometric sampler is able to produce a relatively small distribution of conformations (of a loop in the first case, and of an entire protein in the second case) that contains one or more conformations at small
Root Means Square Deviation (RMSD) from a “target” conformation. This target conformation is not given to the kino-geometric sampler; therefore, the sampling process is not biased toward this conformation.

The main computational difficulties in sampling folded protein conformations result from the large dimensionality of the conformation space and the very small relative volume of the folded subspace. While the small volume of the folded subspace is a necessary precondition for any sampling-based exploration method to work (otherwise a prohibitive number of samples would be required to adequately represent the subspace), it also makes it hard to sample conformations that fall into that subspace. The problem is analogous to looking for a pin in a haystack.

For loop conformation sampling, a kino-geometric sampling method based on constraint prioritization is proposed. The idea is to break the loop into several pieces and, for each piece, to achieve the most restrictive constraints (geometric or kinematic) first. The method was tested on loops varying in length from 5 to 25 residues. Its combination with a pre-existing functional site prediction software (FEATURE) makes it possible to compute and recognize calcium-binding loop conformations.

To sample folded conformations of an entire protein, this dissertation proposes a set of algorithms integrated into a sampler called KGS. Inspired by two previous kino-geometric samplers (ROCK and FRODA), KGS explores the folded state of a protein by expanding a distribution of conformations from an input folded conformation (usually extracted from the Protein Data Bank). Like ROCK and FRODA, it avoids unfolding the protein by selecting stable hydrogen bonds (H-bonds) and integrates them as additional kinematic constraints. But H-bonds result in a protein kinematic model containing many (often several dozens) interdependent closed kinematic cycles. These cycles considerably complicate sampling operations, as the rotatable dihedral angles can no longer be perturbed independently. The contributions of KGS are threefold:

1. It uses a Jacobian-based method to simultaneously deform many cycles of a protein kinematic structure without breaking them. This method is faster than that of ROCK, and allows deformation steps of greater amplitude than FRODA.
2. KGS embeds a new non-biased diffusive strategy that expands quickly away from the input conformation and progressively samples conformations more and more densely distributed over the folded state.

3. To predict H-bond stability, KGS uses a protein-independent model where energetic contribution is only one predictor among others. This model, trained on molecular dynamics data, has been shown to be 20% more accurate than models based on energy alone.

Experiments show that KGS can sample functional (binding) conformations of a protein, given a non-functional (non-binding) one, even when the RMSD between the two conformations is large.

This work indirectly demonstrates that kinematic and geometric constraints provide a good characterization of the folded state of a protein, despite the fact that they only implicitly and partially encode electrostatic and van der Waals (vdW) energy terms. On the other hand, it is much faster to handle these kino-geometric constraints computationally than energy functions, which are made of many terms.
Acknowledgements

Many, many thanks to my advisor Professor Jean-Claude Latombe for being such a wonderful mentor, guiding me thinking deeply into the problems, inspiring me with tons of ideas, and helping me grow as a researcher. I would like to thank Russ B. Altman, Michael Levitt, Douglas L. Brutlag, and Leonidas Guibas for their invaluable advices and long-term guidance. I also thank Rhiju Das and Markus Covert for being on my oral committee.

I am grateful to my co-workers, co-authors, and fellow Clarkers: Ankur Dhanik, Liangjun Zhang, Ruixiang Zhang, Tsai-Yen Li, Mikhail Moshkov, Igor Chikalov, Inbal Halperin-Landsberg, Xuhui Huang, Sam Flores, Shirley Wu, and Tianyun Liu, for being stimulating and helpful. I really appreciate Michael Levitt, Jerry Tsai, Lydia Kavraki, Tsung-Han Chiang, and Vijay Pande for sharing MD data with me. I am also grateful to the BMI Executive Committee, especially Mary Jeanne Oliva and Betty Cheng for their enormous help. I thank all BMIsers, including, but not limited to, Ray Lin, Guanglei Xiong, Genaro Hernandez, Nikesh Kotecha, Yael Garten, Samuel Pearlman, Rong Xu, Rong Chen, Marina Sirota, Sarah Aerni, and Tiffany Chen.

I would like to express my thankfulness to NSF grant DMS-0443939, Bio-X Graduate Fellowship, KAUST-Stanford Academic Excellence Alliance program, and BMI for their generous financial support.

Last but not least, I am especially grateful to my dear friends who are always around me: Lan Wei, Qi Liao, Jia Liu, and Hongbo Zhu. I wouldn’t be able to make it today without you girls. I am eternally grateful to my family for their love, unending support, and constant encouragement. Thank you, Mom, Dad and Alex; I dedicate this dissertation to you!
3 Conformation Sampling for Folded Proteins

3.1 Problem Statement ........................................... 28
3.2 Related Work .................................................. 28
3.3 Methods .......................................................... 31
  3.3.1 Overview .................................................. 31
  3.3.2 Protein Model and Notations ............................... 31
  3.3.3 Selection of Stable H-bonds ................................. 34
  3.3.4 Rigidity Analysis ......................................... 36
  3.3.5 Deformation of $q_{seed}$ .................................. 37
  3.3.6 Validation of $q_{new}$ ..................................... 41
  3.3.7 Selection of $q_{seed}$ and Diffusive Strategy .............. 41
  3.3.8 Reduction of Running Time ................................. 42
3.4 Results .......................................................... 43
  3.4.1 Test Proteins .............................................. 44
  3.4.2 Computations Performed by KGS ......................... 45
  3.4.3 Analysis of Computed Distributions ....................... 48
  3.4.4 Checking Cycle Closure ................................. 51
3.5 Comparison to FRODA ......................................... 53
3.6 Conclusion ...................................................... 54
4 Learning Hydrogen Bond Stability from MD Simulation Data

4.1 Introduction ................................................. 57
4.2 Problem Statement ........................................... 59
4.3 Methods .................................................. 59
  4.3.1 General Approach ........................................ 59
  4.3.2 Basic Tree-Construction Algorithm ...................... 62
  4.3.3 Violation of IID Property ................................. 64
  4.3.4 Tree Pruning ........................................... 66
4.4 Results ................................................... 66
  4.4.1 Experimental Setup ....................................... 66
  4.4.2 Experiment 1: Training on one data table, predicting on another 69
  4.4.3 Experiment 2: Training on data from multiple trajectories .. 72
  4.4.4 Experiment 3: Comparison with FIRST-energy model ........ 74
  4.4.5 Experiment 4: Identification of least stable H-bonds ........ 75
  4.4.6 Experiment 5: Models for different averaging and prediction windows ........................................... 77
4.5 Discussion ................................................ 80
4.6 Conclusion ............................................... 83

5 Conclusion .................................................. 85

A List of Predictors ............................................ 87

Bibliography .................................................. 90
List of Tables

2.1 Test set of 20 loops ........................................ 19
2.2 The number of collision-free side-chain placements. .......... 23

3.1 Statistics of the proteins in the experiments. ................. 45
3.2 Minimum RMSD to targets and running time of generating 5000 conformations. ........................................ 50
3.3 Average and maximum cycle opening. ........................ 52
3.4 Comparison of RMSD step size of FRODA and KGS. .......... 53

4.1 Characteristics of the MD simulation trajectories used to create the 6 datasets .................................................. 67
4.2 Mean RBED values for each pair of data tables in Experiment 1 . 70
4.3 Mean RBED values obtained in Experiment 2 ..................... 73
4.4 Mean RBED values computed in Experiment 3 ................... 75
4.5 Mean RBED values for different lengths $l'$ of the predictor averaging window in Experiment 5 ............................ 78
4.6 Mean RBED values for different lengths $l$ of the prediction window in Experiment 5 ........................................ 78
List of Figures

1.1 Timescales of protein motion. ................................. 2
1.2 Protein linkage model. ...................................... 5

2.1 Sampled loop conformations of 4 loops. .................. 20
2.2 Middle Ca positions of 100 sampled loop conformations. 21
2.3 Pairwise RMSD distributions of 100 and 1000 loop samples. 22
2.4 A calcium-binding loop sample for Parvalbumin. ....... 24
2.5 Time comparison of RAPPER and our loop sampler. .... 26

3.1 Overview of the KGS algorithm. .............................. 32
3.2 Graph of rigid groups of a 6-residue protein fragment. ... 33
3.3 Common and different H-bonds in two conformations of the same protein 35
3.4 The geometric criteria of a hydrogen bond. ................ 35
3.5 Two neighboring rigid groups of atoms. .................... 37
3.6 Example of a spanning tree $T_{\text{const}}$ of $L_{\text{const}}$ containing a cycle ... 38
3.7 The initial and goal conformations of test proteins. ...... 46
3.8 The largest rigid groups of atoms of test proteins. ....... 47
3.9 The max RMSD to the initial conformation and the min RMSD to the target conformations. ......................... 49
3.10 Target conformations and the closest sampled conformations. 50
3.11 RMSD to the initial and targets of 5000 sub-sampled conformations generated by FRODA. .............................. 55

4.1 Histogram of the measured stability of H-bond occurrences. 61
4.2 Histogram of the measured stability of H-bonds. . . . . . . . . . . . . 65
4.3 RBED values for the 10 models generated with 1c9oA (Experiment 1) 71
4.4 Regression tree trained with 1c9oA (Experiment 1) . . . . . . . . . . 71
4.5 Regression tree trained with complex (Experiment 1) . . . . . . . . . 72
4.6 Distribution of RBED values trained on multiple trajectories (Experi-
ment 2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
4.7 Top 3 layers of a tree trained with combination of all tables, except
1c9oA (Experiment 2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
4.8 Top 3 layers of a tree trained with combination of all tables, except
1e85A (Experiment 2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
4.9 Identification curves of the least stable bonds (Experiment 4) . . . 76
4.10 Distribution of RBED values for models tested on 1c9oA for different
lengths $l'$ of the predictor averaging window (Experiment 5) . . . . . 79
4.11 Distribution of RBED values for models tested on 1c9oA for different
lengths $l$ of the prediction window (Experiment 5) . . . . . . . . . . 79
4.12 Predictor importance scores . . . . . . . . . . . . . . . . . . . . . . . 82
Chapter 1

Introduction

1.1 Background

Structural biologists, computational biologists and physicists have been exploring the mechanism of protein conformational change for several decades. It is now widely accepted that the ability of a folded protein to adopt different conformations is crucial to its function which requires binding against other molecules. For instance, the model for protein-ligand binding has evolved from the lock-and-key model [59], where the protein has a rigid structure, to the induced-fit model [114], where the protein deforms upon ligand binding, to the now well-accepted conformational selection model [11, 49, 102, 109]. This last model suggests that a protein exists in an ensemble of rapidly inter-converting conformations and that the ligand selects the most favorable conformations to bind to from this ensemble. Recent studies of protein-ligand binding and structure-based drug design stress the need to incorporate protein flexibility in order to correctly predict protein-ligand interaction [1, 18, 97]. In fact, the use of multiple protein structures is emerging as the best option to introduce receptor flexibility in the computational study of ligand binding [18, 19]. Moreover, protein decoys are also useful for testing energy functions [103], gaining insights into key determinants of protein stability [110], and modeling structural heterogeneity in proteins from X-ray crystallography data [105].
Figure 1.1: Timescales of protein motion, along with the experimental and computational methods that can detect them. This figure is drawn from [49].

1.2 Protein Motion

Protein motion spans a wide range of timescales, from the femtosecond to the second, as shown in Figure 1.1. The highest frequency motions are local, small-amplitude, uncorrelated atomic fluctuations. They span structurally similar states. At the opposite end of the spectrum, low-frequency motions are large-amplitude, correlated motions between structurally distinct states. They typically occur on timescales ranging from microseconds to seconds. These “slow” motions are particularly interesting, because they enable key biological processes, such as enzyme catalysis, signal transduction, and protein-molecule interactions [25].

Several experimental techniques have been developed to study protein motion, for example, nuclear magnetic resonance (NMR) spectroscopy [76], time-resolved
Laue X-ray diffraction [87], hydrogen-deuterium exchange [111], fluorescence spectroscopy [88], and cryo-electron microscopy [107] (Figure 1.1). Although these methods can determine structural ensembles directly in solution, their drawbacks include low-resolution, limitation to small proteins, and inability to catch conformations in transitions between energy basins. The highest-resolution experimental data are still provided by X-ray crystallography. As of today, more than 25,000 protein structures with resolution equal to or better than 2Å have been determined by X-ray crystallography and have been deposited in the Protein Data Bank (PDB) [6]. A few hundred of proteins with known multiple stable conformations are recorded in the database MolMovDB [43]. Nevertheless, X-ray crystallography can only “take static snapshots” of stable conformations trapped in homogeneous crystals. The number of structurally distinct conformations of the same protein that can be resolved using this technique is insufficient to characterize low-frequency motions.

1.3 Computational Approaches

Computational approaches to study proteins conformational changes can be classified into two categories: energy-based and kinematics/geometry-based methods.

1.3.1 Energy-Based Methods

Energy-based methods sample conformations or simulate protein motion by traveling in the protein’s energy landscape, which is defined by a set of mathematical formula describing the interaction forces among atoms. Molecular Dynamics (MD) simulation [2, 69, 73], Monte Carlo (MC) simulation [62, 80], and normal mode analysis [14, 16, 39, 70] are examples of popular energy-based methods. MD simulation updates atom positions according to the Newton’s laws of motion integrated with a very small time step (usually on the order of the femtosecond). Some interesting conformational changes crucial to protein function have been simulated, such as the flap motion in the HIV protease [51]. MC simulation samples protein conformations at various energy levels by introducing an energy-dependent weight function into
the Metropolis criterion [75]. Normal mode analysis provides an analytic description of the dynamics in a macromolecular system near an energy minimum by using a harmonic approximation of the energy function. However, due to both the chaotic shape of the energy landscape and high energy barriers between energy basins, it is computationally expensive to employ energy-based methods to study low-frequency motion.

1.3.2 Kinematics/Geometry-Based Methods

Instead of exploring the energy landscape, the kinematics/geometry-based methods directly access the low-frequency motion of a protein without going through the high-frequency motion [86, 92, 94, 99, 115, 118]. They achieve this by characterizing the folded state of a protein by kinematic and geometric constraints that implicitly encode dominant energy terms. They sample conformations by only enforcing these constraints. Hence we call them *kino-geometric* sampling methods.

Here, a protein is represented using a kinematic linkage of atoms, the *linkage model*, where each link connects pairs of covalently bonded atoms [36, 94, 119]. The degrees of freedom (DOFs) are the dihedral angles around the rotatable bonds. Figure 1.2 shows a protein fragment with 3 residues (amino-acids), where the DOFs are the $\phi$, $\psi$, and $\chi$ angles. Each residue provides two dihedral angles $\phi$ and $\psi$ around the bond $N - C_\alpha$ and $C_\alpha - C$, respectively, in the main-chain, and 0 to 6 $\chi$ angles in the side-chain. The linkage model rests on the observation that, once high-frequency motions have been filtered out, bond angles, bond lengths, and dihedral angles around non-rotatable bonds have almost constant values (the averages of their values during the high-frequency motions) [60]. In this way, the linkage model gives a natural, direct access to low-frequency conformational changes. An additional, related advantage of this model is that it considerably reduces the dimensionality of the protein’s conformation space. A conformation is now defined by the values of the dihedral angles around the rotatable bonds, rather than by the 3D coordinates of all the atoms. A conformation defined by the linkage model can thus be seen as a representative of the small region spanned by atomic fluctuations around it in the higher-dimensional.
conformation space parameterized by the 3D coordinates of all atoms.

In the linkage model, each atom is represented by a sphere of van der Waals (vdW) radius. Two atoms are considered in collision if their centers are closer than \( \varepsilon \) times the sum of their vdW radii. The parameter \( \varepsilon \) is usually set to 0.75, which approximately corresponds to the distance where the vdW potential begins to increase steeply. The protein is collision-free if no two atoms collide. Such collision avoidance implicitly eliminates very high energy conformations due to the repulsive vdW forces. Since atoms in a folded state protein are densely packed, the repulsive vdW term is a dominant energy term.

One may encode other key energy terms by imposing additional kinematic and/or geometric constraints on the linkage model. For example, hydrogen bonds (H-bonds) are special interactions combining vdW and electrostatic forces, and are crucial for stabilizing a folded protein. As we will see later in this thesis, they can be modeled in a way similar to the rotatable covalent bonds.
Compared to the energy-based approach, kino-geometric sampling has the advantages of lower dimensionality of the conformation space and simpler encoding of key energy terms.

1.4 Goal and Assumptions

Our goal is to develop algorithms able to efficiently explore the folded state of a protein by kino-geometric sampling methods, so that the sampled conformations are broadly and approximately evenly distributed throughout the folded state.

The success of kino-geometric conformation sampling relies on the satisfaction of three assumptions:

1. A protein folded state must be very small (in both dimensionality and volume) compared to the overall conformation space, so that it can be well represented by a reasonably small distribution of conformations. If this assumption is not satisfied, even huge distributions may fail to cover the neighborhoods around all known biologically interesting conformations.

2. Kino-geometric constraints are sufficient to approximate the folded state of a protein. They implicitly encode several energy terms (such as bond stretching, bond angle, and repulsive vdW energy), but many other energy terms are not taken into account. So, the impact of these latter terms on the shape of the folded state must be small, although they may affect the kinetics and dynamics of the protein deformations.

3. Unlike energy constraints, kino-geometric constraints are sufficiently simple to make it possible to efficiently sample conformations broadly distributed over a protein folded state. Under the previous two assumptions this third assumption is not obvious: even if the folded state is very small and well approximated by kino-geometric constraints, it might be difficult to sample conformations that satisfy these constraints.

One goal of our work will be to indirectly demonstrate that these assumptions are often satisfied.
1.5 Computational Challenges

In spite of its potential advantages, such as dimensionality reduction of the conformation space and simple encoding of dominant energy terms, kino-geometric sampling still raises some computational challenges:

1. The main computational difficulty in sampling folded protein conformations results from the still large dimensionality of the conformation space and the very small relative volume of the folded state. While the small volume of the folded state is a necessary precondition for any sampling-based exploration method to work (otherwise a prohibitive number of samples would be required to adequately represent the state), it also makes it hard to sample conformations that fall into that state. The problem is analogous to looking for a pin in a haystack.

2. Another difficulty results from the fact that the linkage model of a protein may contain closed deformable kinematic cycles. Some cycles are formed naturally by loop structures between $\alpha$-helices and/or $\beta$-strands, which are assumed to keep fixed relative positions and orientations (Chapter 2). Other cycles are due to H-bonds between atoms that are not covalently-bonded, but indirectly connected via multiple covalent bonds. The conformation sampler must choose the dihedral angles in a cycle in a coordinated fashion in order to maintain the cycle closed. The situation becomes even more complicated when multiple cycles share dihedral angles. Then, all the dihedral angles in the cycles are interdependent. There is no simple way to directly pick their values, as they are related by nonlinear equations.

1.6 Applications

In this thesis, we propose two distinct kino-geometric sampling methods to explore the folded state of a protein. One method aims at sampling conformations of a protein loop between secondary structure elements, assuming that the rest of the protein is rigid. The other method addresses the more difficult problem of sampling folded conformations of an entire protein.
1.6.1 Conformation Sampling for a Loop

Here our sampling method is based on constraint prioritization. The idea is to break the loop into several pieces and, for each piece, to achieve the most restrictive constraints (geometric or kinematic) first. The method was tested on loops varying in length from 5 to 25 residues. The experiments show that the method is able to efficiently sample broadly across the closed collision-free (see definitions in Section 2.1) conformation space. The average time to generate one sample varies from less than 0.1 second to about 20 seconds, depending on the length of the loop and its environment. Combination of the sampler with a pre-existing functional site prediction software (FEATURE) [113] makes it possible to compute and recognize calcium-binding loop conformations. This loop sampler is implemented into a toolkit software, LoopTK, available at https://simtk.org/home/looptk. A previous description of this sampler, as well as of the toolkit software, can be found at [117].

1.6.2 Conformation Sampling for an Entire Protein

To sample folded conformations of an entire protein, we propose a set of algorithms integrated into a sampler called KGS. Inspired by two previous samplers (ROCK [118] and FRODA [115]), KGS explores the folded state of a protein by expanding a distribution of conformations from an input folded conformation usually extracted from the PDB. Like ROCK and FRODA, it avoids unfolding the protein by integrating the most stable H-bonds as additional constraints into the linkage model. However, these constraints result into many (often several dozens) interdependent closed kinematic cycles that considerably complicate sampling operations. The values of the rotatable dihedral angles can no longer be sampled independently. To address this difficulty, KGS uses a Jacobian-based method to simultaneously deform the cycles without breaking them. This method is faster than that of ROCK and allows deformation steps of greater amplitude than FRODA. To our knowledge, KGS is the first sampler to apply a Jacobian-based method to deform dozens of inter-dependent cycles simultaneously. In addition, KGS embeds a new non-biased diffusive strategy that expands quickly away from the input conformation and progressively samples
conformations more and more densely distributed over the folded state.

Experiments on proteins of diverse in sizes, structural compositions, and motion styles show that KGS efficiently computes distributions containing conformations at small Root Mean Square Deviation (RMSD) from target (e.g., functional) conformations without having been biased toward these conformations. Comparison with state-of-the-art FRODA shows that the distributions computed by KGS diffuse faster than those by FRODA and that sampling steps are larger.

This work was done in collaboration with Liangjun Zhang.

1.7 H-bond Stability Prediction

Selection of stable H-bonds is an important step in KGS, since it directly affects the kinematic linkage model of a protein at each sampling operation. Previous kinogeometric samplers like ROCK and FRODA estimate the stability of an H-bond by its potential energy. They only retains H-bonds with low energy as stable ones. However, energy alone may not be the best indicator of H-bond stability.

Instead, we train a stability model from MD simulation data. The model is a regression tree built from data tables recording the values of 32 predictors. Each path from the root of the regression tree to a leaf node represents a conjunction of conditions on a subset of predictors, and gives the average stability of all the H-bonds in the training data that satisfy these conditions. Experiments show that a trained stability model can be 20% more accurate than energy alone. Such a model can also reliably recognize the least stable H-bonds.

This work was done in collaboration with Igor Chilakov and Mikhail Moshkov at the King Abdullah University of Science and Technology [23, 24].

1.8 Thesis Contributions

In this section we list the main contributions of our thesis.

Specific Contributions
1. A constraint prioritization method to sample closed collision-free conformations of a protein loop.

2. A Jacobian-based method to deform dozens of interdependent kinematic cycles without breaking them.

3. A novel diffusive sampling strategy to generate distributions of conformations broadly distributed over the folded state of a protein.

4. A machine learning approach to infer probabilistic models of H-bond stability from Molecular Dynamics simulation data.

General Contribution

5. An indirect demonstration (through the experiments conducted with our two samplers) that the three assumptions stated in Section 1.4 are often satisfied.

1.9 Organization of the Dissertation

This dissertation consists of three main chapters describing our work on loop sampling (Chapter 2), protein sampling (Chapter 3), and modeling H-bond stability (Chapter 4). In addition, Chapter 5 summarizes our main contributions and proposes future directions.
Chapter 2

Conformation Sampling for Protein Loops*

2.1 Problem Statement

Given the 3D structure of a protein with a missing loop, our goal is to sample conformations of the loop such that they broadly and approximately evenly distribute over its closed collision-free conformation space. A formal definition is as follows.

Let \( L \) be a loop — a sequence of \( p > 3 \) consecutive residues in a protein \( P \) such that none of the two termini of \( L \) is also a terminus of \( P \). The structure of the rest of the protein, denoted by \( P\setminus L \), is given, and assumed rigid. We model \( L \) using a truncated linkage model (see Section 1.3.2) that only includes the main-chain of \( L \) and the \( C_\beta \) and \( O \) atoms respectively bonded to the \( C_\alpha \) and \( C \) atoms in the main-chain. We denote this truncated model by \( L_B \). It is collision-free if and only if no two atoms, one in \( L_B \), one in \( L_B \) or \( P\setminus L \), are in collision. We attach a Cartesian coordinate frame \( \Omega_1 \) to the N-terminus of \( L_B \) and another frame \( \Omega_2 \) to its C-terminus. When \( L_B \) is connected to its anchors in the rest of the protein, i.e., when it adopts a closed conformation, the pose (position and orientation) of \( \Omega_2 \) relative to \( \Omega_1 \) is fixed.

*This work was presented at the 7th Workshop on Algorithms in Bioinformatics (WABI 2007) [34], and published in IEEE/ACM Transactions on Computational Biology and Bioinformatics [117].
to a predefined value that we denote by $\Pi_g$.

A conformation of $L_B$ is denoted by $q$, which is a vector of all the rotatable dihedral angles in $L_B$ — the $2p$ dihedral angles $\phi_i$ and $\psi_i$ in residues $i = 1, \ldots, p$. If we arbitrarily pick the values of $\phi_i$ and $\psi_i$, then in general we will get an open conformation of $L_B$. The set $Q$ of all open and closed conformations of $L_B$ is a space of dimensionality $n = 2p$. The subspace $Q_{\text{closed}}$ containing only the closed conformations has dimensionality at least $n - 6$. The set of closed and collision-free conformations of $L_B$ form $Q_{\text{free}}^{\text{closed}}$. The goal is to sample broadly and approximately evenly $q \in Q_{\text{free}}^{\text{closed}}$.

Let $\Pi(q)$ denote the pose of $\Omega_2$ relative to $\Omega_1$ when $q \in Q$. The function $\Pi$ and its inverse $\Pi^{-1}$ are the “forward” and “inverse” kinematics map of $L_B$, respectively.

This loop conformation sampling problem requires satisfying two constraints concurrently: closing a kinematic cycle and avoiding collision. Each constraint considered separately is relatively easy to satisfy, but the combination is hard because the two constraints are conflicting. Due to the compactness of a folded protein, although $Q_{\text{free}}^{\text{closed}}$ has the same dimensionality as $Q_{\text{closed}}$, its volume is a small fraction of that of $Q_{\text{closed}}$, especially for long loops. Hence, an arbitrary closed conformation of the loop has small probability to be collision-free. Conversely, an arbitrary collision-free conformation of the loop has null probability to be closed. As a result, existing sampling techniques often have high rejection ratios.

### 2.2 Related Work

The problem considered in this paper is a version of the “cycle closure” problem studied in [20, 27, 33, 55, 64, 112]. Several works have specifically focused on kinematic closure. Analytical Inverse Kinematics (IK) methods are described in [27, 112] to close a fragment of 3 residues. For longer fragments, iterative techniques have been proposed, like the popular CCD (Cyclic Coordinate Descent) [20]. We re-use several of these techniques in our work. In particular, our sampling algorithm applies the

---

2In the robotics literature, this type of problem is called “loop closure”, instead of “cycle closure”. Here, we use the term “cycle closure”, since the word “loop” refers to a protein fragment.
analytical IK method described in [27] in a new way to close loops with more than three residues.

Procedures to sample closed collision-free conformations of loops by varying dihedral angles have been proposed in [29, 33, 55]. The goal of RAPPER [33] and the hierarchical method described in [55] is to generate near-native conformations by minimizing an energy function. Instead, the goal of our method and the one presented in [29] is to explore the closed collision-free conformation space of a loop by sampling conformations broadly distributed across this space.

RAPPER [33] iteratively generates a loop conformation from its N-terminus toward its C-terminus by selecting the values of the dihedral angles $\phi$ and $\psi$ at random from a predefined discrete table of values. It also checks that the $C_{\alpha}$ atom in each residue is sufficiently close to the loop’s C-anchor on the protein. In the end, to close the gap between the loop’s last residue and its anchor on the protein, RAPPER runs an iterative minimization procedure to reduce this gap. Unlike RAPPER, our method does not select dihedral angles from discrete tables, but pick them according to probability distributions input by the user (by default, uniform distributions). In addition, our method retains a sufficient number of dihedral angles (in the middle portion of the loop) to make it possible to close the cycle using an exact IK method.

Like our sampler, the method presented in [55] also exploits the idea of loop decomposition. It breaks a loop into two fragments, then independently sample collision-free conformations for each fragment (by sampling dihedral angles starting from their respective anchors), and finally generates closed conformations by bridging close-enough fragment conformations. Like RAPPER, this method selects dihedral angles from predefined discrete tables. It uses IK and collision avoidance techniques that are very different from ours. Both RAPPER and this method have been tested on relatively short loops having between 2 and 12 residues in length.

The Random Loop Generator (RLG) [29] is used to study the potential mobility of a loop in the presence and absence of certain side-chains. It successively samples closed conformations that it later tests for collisions. To sample closed conformations it divides the loop main-chain into an “active” and a “passive” fragment. The latter has exactly 3 residues (hence, 6 dihedral angles). The dihedral angles in the active
fragment are successively sampled at random using a geometric algorithm that increases the likelihood that a closed conformation will eventually be obtained. The 6 dihedral angles of the passive fragment are used to close the cycle using an IK procedure. The generated closed conformations are then tested for collisions.

Some sampling procedures try to sample conformations using libraries of fragments obtained from previously solved structures [32, 64, 101, 108]. For example, a divide-and-conquer approach is described in [101] that generates a database of fragments of different residue lengths and types, by using a Ramachandran plot distribution. These fragments are then concatenated to build conformations of a longer loop. However, collisions are not taken into account during this process. Other works sample loop conformations directly by minimizing an energy function [5, 33, 37, 55, 91] or running an MD simulation [15] with the goal to identify loop fragments close to native structure. However, in a number of applications it is preferable to explore the closed collision-free conformation space of a loop, i.e., to generate a distribution rather than a single conformation.

In our algorithms, collision detection is done using the efficient grid method previously described in [46]. A similar detection method is also used in RAPPER [33].

2.3 Methods

2.3.1 Overview

The goal of our loop conformation sampler is to generate conformations of $L_B$ broadly distributed over $Q_{\text{free}}$. The challenge comes from the interaction between the kinematic closure and collision avoidance constraints. Computational tests (see Section 2.4) show that the approach (hereafter called the naive approach) that first samples conformations from $Q_{\text{closed}}$ and next rejects those with collisions is often too time consuming, except for short loops, due to its huge rejection ratio. The reverse approach — sampling the angles $\phi_i$ and $\psi_i$ of $L_B$ to avoid collisions — will inevitably end up with open conformations, since $Q_{\text{closed}}$ has lower dimensionality than $Q$.

These insights led us to develop a prioritized constraint-satisfaction approach,
hereafter called the prioritized approach. We partition $L_B$ into three segments, the front-end $F$, the mid-portion $M$, and the back-end $B$. $F$ starts at the N terminus of $L_B$ and $B$ ends at its C terminus. $M$ is the segment between them. Due to the immediate proximity of atoms in $P \setminus L$, the conformations of $F$ and $B$ are more limited by the collision avoidance constraint than by the closure constraint; so, we sample the dihedral angles in $F$ and $B$ to avoid collision, ignoring the closure constraint. Then, for any pair of conformations of $F$ and $B$, the possible conformations of $M$ are mainly limited by the closure constraint; so, we use the naive approach to sample conformations of $M$, by running an IK procedure to close the gap between $F$ and $B$ and testing the collision avoidance constraint afterward. In this way, our prioritized approach reduces the application of the naive approach to a short fragment of the loop. The length of $M$ must be large enough for the IK procedure to succeed with high probability, but not too large since collision avoidance is only tested afterward. In our software, the number of residues in $M$ is usually set to half of that of $L_B$ or to 4, whichever of these two numbers is larger. The number of residues of $F$ and $B$ are then selected equal ($\pm 1$). Tests show that these choices are close to optimal on average for a wide range of loops. For unusually long loops, it may be suitable to set an upper bound on the length of $M$.

The dihedral angles $\phi$ and $\psi$ in each of the three fragments $F$, $M$, and $B$ are picked in the order to sample conformations of $L_B$ broadly distributed over $Q_{\text{free}}^{\text{closed}}$.

### 2.3.2 Sampling Front/Back-End Conformations

Consider the front-end $F$. The angles $\phi$ and $\psi$ closest to the fixed terminus of $F$ are the most constrained by possible collisions with the rest of the protein $P \setminus L$. So, the angles are sampled in the order in which they appear in $F$, that is $\phi_1, \psi_1, \phi_2$, etc. In this order, each angle $\phi_i$ (resp., $\psi_i$) determines the positions of the next two atoms $C_{\beta_i}$ and $C_i$ (resp., the next three atoms $O_i, N_{i+1}$ and $C_{\alpha_{i+1}}$). The angle is sampled so that these atoms do not collide with any atom in $P \setminus L$ or any preceding atom in $F$. Its value is picked at random, either uniformly or according to a user-input probabilistic distribution (e.g., one based on Ramachandran tables). If no value of
the angle prevents the two or three atoms it governs from colliding with other atoms, the algorithm backtracks and re-samples a previously sampled angle. Collision-free conformations of the back-end $B$ are sampled in the same way, by starting from its fixed C terminus and proceeding backward.

### 2.3.3 Sampling Mid-Portion Conformations

Given two collision-free conformations of $F$ and $B$ such that the gap between them does not exceed the maximal length that $M$ can achieve, a conformation of $M$ is sampled as follows.

The values of the $\phi$ and $\psi$ angles in $M$ are picked at random, uniformly or according to a given distribution. This leads to a conformation $q$ of $M$ that is connected to $F$ at one end and open at the other end. To close the gap between $M$ and $B$, we use the IK method described in [27]. This method solves the IK problem analytically, for any sequence of residues in which exactly three pairs of $(\phi, \psi)$ dihedral angles are allowed to vary. These pairs need not be consecutive.

Let us denote the IK method by $\text{ANALYTICAL-IK}(q, i, j, k)$, where argument $q$ is the initial open conformation of $M$ and arguments $i$, $j$, and $k$ are the integers identifying the three residues that contain the pairs of dihedral angles that are allowed to vary. Our experiments show that, on average, the IK method is the most likely to succeed in closing the gap when one pair is the last one in $M$ and the other two are distributed in $M$. Let $r$ and $s$ denote the integers identifying the first and last residue of $M$ in $L_B$. As the IK method is extremely fast, $\text{ANALYTICAL-IK}(q, i, j, s)$ is called for all $i = r, ..., s - 2$ and $j = i + 1, ..., s - 1$, in a random order, until a closed conformation of $M$ has been generated. If this conformation tests collision-free, then the sampling procedure constructs a closed collision-free conformation of $L_B$ by concatenating the conformations of $F$, $M$, and $B$.

If the above operations fail to generate a closed collision-free conformation of $M$, then they are repeated (with new initial values for the $\phi$ and $\psi$ angles in $M$) until a predefined maximal number of iterations have been performed.

We have also experimented with iterative IK techniques, like CCD, to close the
2.3.4 Placing Side-Chains

So far, we have used a truncated linkage model of the loop that does not include side-chains. For each conformation of $L_B$ sampled from $Q_{\text{free}}^{\text{closed}}$, we now run SCWRL [21] to place the side-chains. We may only compute the placements of the side-chains in $L_B$ given the placements of the side-chains in $P\setminus L$. Alternatively, we may (re-)compute the placements of all the side-chains in the protein. In each case, SCWRL minimizes an energy function that penalize collisions among atoms. But it does not fully guarantee that the conformations of the side-chains will be collision-free. Experiments (Section 2.4.2) show that the chances of successful placing collision-free side-chains are higher if we sample dihedral angles $\phi$ and $\psi$ according to the Ramachandran distribution.

2.3.5 Collision Detection

Collision detection is done using the grid method [46]. This method takes advantage of the fact that, to avoid collisions, atoms must spread out, so that any small cubic box of a fixed volume in 3D space contains an upper-bounded number of atom centers, independent of the total number of atoms in the protein.

The method tessellates the three-dimensional space of the protein into an array of equally sized cubes. The edge length of a cube is chosen approximately equal to the largest vdW diameter of the atoms. For a given conformation of the protein, each atom is indexed in the cube that contains its center. Whenever the position of an atom is modified, the grid structure is updated accordingly in constant time. The grid is implemented as a memory-efficient hash table. Only the grid cubes that contain atom centers are represented, each with the corresponding list of atoms.

The collision detection algorithm iterates through all atoms that need to be checked (e.g., the atoms in $L_B$), asking for each atom if it is in collision. The atom only needs to be checked with the atoms indexed in its own grid cube and the 26
cubes surrounding it. The selected size for the cubes guaranteed that there are at most 4 atom centers within one cube. The number of pairs of atoms to check is thus upper-bounded by a constant. In practice, the number of checks for each atom is even smaller and usually less than 6. Hence, collision detection for a single atom runs in $O(1)$ time, and the collision test for all $O(n)$ atoms in $L_B$ or $L$ runs in $O(n)$ time, independent of the total number of atoms in the protein.

2.4 Results

2.4.1 Basic Test

Table 2.1 lists 20 loops, whose sizes range from 5 to 25 residues, which we used to perform computational tests. Each row lists the PDB ID of the protein, the number of residues in the protein, the number identifying the first residue in the loop, the number of residues in the loop, and the average time to sample one closed collision-free conformation of the loop using two distinct procedures (our sampling method and the “naive” method outlined in Section 2.3.1). In some loops the two termini are close, while in others they are quite distant. Some loops protrude from the proteins and have much empty space in which they can deform without collision (e.g., 3SEB), while others are very constrained by the other protein residues (e.g., 1TIB). The loop in 1MPP is constrained in the middle by side-chains protruding from the rest of the protein (see Figure 2.2(b)). In the results presented below, all $\phi$ and $\psi$ angles were picked uniformly at random (i.e., no biased distributions, like the Ramachandran’s ones, were used).

Each picture in Figure 2.1 displays a subset of conformations generated by our sampling method for the loops in 1TIB, 3SEB, 8DFR, and 1THW. The loop in 1TIB, which resides at the middle of the protein, has very small empty space to move in. The PDB conformation of the loop in 1THW (shown green in the picture) bends to the right, but our method also found collision-free conformations that are very different. Each picture in Figure 2.2 shows the distributions of the middle $C\alpha$ atom in 100 sampled conformations of the loops in proteins 1K8U, 1MPP, 1COA, and
### Table 2.1: Test set of 20 loops (see main text for comments).

<table>
<thead>
<tr>
<th>Protein Id</th>
<th>Size</th>
<th>Loop Start</th>
<th>Loop Size</th>
<th>Sampling Ours</th>
<th>Sampling Naive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1XNB</td>
<td>185</td>
<td>SER 31</td>
<td>5</td>
<td>0.22</td>
<td>0.21</td>
</tr>
<tr>
<td>1TYS</td>
<td>264</td>
<td>THR 103</td>
<td>5</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>1GPR</td>
<td>158</td>
<td>SER 74</td>
<td>6</td>
<td>0.38</td>
<td>0.38</td>
</tr>
<tr>
<td>1K8U</td>
<td>89</td>
<td>GLU 23</td>
<td>7</td>
<td>0.21</td>
<td>0.20</td>
</tr>
<tr>
<td>2DRI</td>
<td>271</td>
<td>GLN 130</td>
<td>7</td>
<td>0.42</td>
<td>0.46</td>
</tr>
<tr>
<td>1TIB</td>
<td>269</td>
<td>GLY 172</td>
<td>8</td>
<td>2.49</td>
<td>13.03</td>
</tr>
<tr>
<td>1PRN</td>
<td>289</td>
<td>ASN 215</td>
<td>8</td>
<td>0.33</td>
<td>0.66</td>
</tr>
<tr>
<td>1MPP</td>
<td>325</td>
<td>ILE 214</td>
<td>9</td>
<td>0.53</td>
<td>99.85</td>
</tr>
<tr>
<td>4ENL</td>
<td>436</td>
<td>LEU 136</td>
<td>9</td>
<td>1.46</td>
<td>19.35</td>
</tr>
<tr>
<td>135L</td>
<td>129</td>
<td>ASN 65</td>
<td>9</td>
<td>0.77</td>
<td>1.54</td>
</tr>
<tr>
<td>3SEB</td>
<td>238</td>
<td>HIS 121</td>
<td>10</td>
<td>0.50</td>
<td>3.80</td>
</tr>
<tr>
<td>1NLS</td>
<td>237</td>
<td>ASN 216</td>
<td>11</td>
<td>1.30</td>
<td>5.51</td>
</tr>
<tr>
<td>1ONC</td>
<td>103</td>
<td>MEC 23</td>
<td>11</td>
<td>2.26</td>
<td>5.66</td>
</tr>
<tr>
<td>1COA</td>
<td>64</td>
<td>VAL 53</td>
<td>12</td>
<td>19.02</td>
<td>67.49</td>
</tr>
<tr>
<td>1TFE</td>
<td>142</td>
<td>GLU 158</td>
<td>12</td>
<td>0.48</td>
<td>8.14</td>
</tr>
<tr>
<td>8DFR</td>
<td>186</td>
<td>SER 59</td>
<td>13</td>
<td>2.02</td>
<td>39.36</td>
</tr>
<tr>
<td>1THW</td>
<td>207</td>
<td>CYS 177</td>
<td>14</td>
<td>1.48</td>
<td>9.84</td>
</tr>
<tr>
<td>1BYI</td>
<td>224</td>
<td>GLU 115</td>
<td>16</td>
<td>2.52</td>
<td>&gt;800</td>
</tr>
<tr>
<td>1G5A</td>
<td>628</td>
<td>GLY 433</td>
<td>17</td>
<td>3.28</td>
<td>&gt;800</td>
</tr>
<tr>
<td>1HML</td>
<td>123</td>
<td>GLY 51</td>
<td>25</td>
<td>17.74</td>
<td>&gt;800</td>
</tr>
</tbody>
</table>

1G5A along with a few entire conformations. The loops in 1K8U and 1COA have relatively large empty space to move in, whereas the loops in 1MPP and 1G5A are restricted by the surrounding protein residues. These figures illustrate the ability of our sampling procedure to generate conformations broadly distributed across the closed collision-free conformation space of a loop.

The average running time (in seconds) of our sampling procedure to compute one closed collision-free conformation of each loop is shown in column 5 of Table 2.1. Each average was obtained by running the procedure until it generated 100 conformations of the given loop and dividing the total running time by 100.\(^3\) The last column of

\(^3\)The algorithms are written in C++ and runs under Linux. Running times were obtained on a 3GHz Intel Pentium processor with 1GB of RAM.
Table 2.1 gives the average running time of the “naive” procedure that first samples closed conformations of the loop main-chain and next rejects those which are not collision-free. Our sampling procedure does not break a loop into 3 segments if it has fewer than 8 residues. So, the running times of both procedures for the first 5 proteins are essentially the same. For all other proteins, our procedure is faster, sometimes by a large factor (188 times faster for the highly constrained loop in 1MPP), than the naive procedure. For the last three proteins, this latter procedure failed to sample 100 conformations after running for more than 80,000 seconds.

Not surprisingly, the running times vary significantly across loops. Short loops with much empty space around them take a few 1/10 seconds to sample, while long loops with little empty space can take a few seconds to sample. The loops in 1COA and 1HML take significantly more time to sample than the others. In the case of 1COA, it is difficult to connect the loop’s front-end and back-end (3 residues each).
Figure 2.2: Positions of the middle Cα atom (red dots) in 100 loop conformations sampled for four protein loops: 1K8U, 1MPP, 1COA, and 1G5A.

with its mid-portion (6 residues). The termini of the loop are far apart and the protein constrains the loop all along. Due to the local shape of the protein at the two termini of the loop, many sampled front-ends and back-ends tend to point in opposite directions, which then makes it often impossible to close the mid-portion without collisions. In this case, we got a better average running time (4 seconds, instead of 19) by setting the length of the mid-portion to 8 (instead of 6). The loop in 1HML is inherently difficult to sample. Not only is it long, but there is also little empty space available for it. Other experiments not reported here indicate that the running times reported in Table 2.1 vary moderately when parameters like the factor $\varepsilon$ and the number of residues in the loop’s mid-portion $M$ are slightly modified.

Figure 2.3 displays RMSD histograms generated for the loop in 3SEB. The purple (resp., white) histogram was obtained by sampling 100 (resp. 1000) conformations of the corresponding loop and plotting the frequency of the RMSDs between all pairs of
conformations. The almost identity of the two histograms indicates that the sampled conformations spread quickly in $Q_{\text{free}}$. Similar histograms were generated for other loops.

### 2.4.2 Using Ramachandran Distribution

Our sampler calls SCWRL [21] to place side-chains. The placements computed by SCWRL, however, are not guaranteed to be collision-free, which may result in discarding the generated conformation of $L_B$. This leads to the idea of biasing dihedral angle sampling using the Ramachandran distribution. We ran our sampler to generate conformations of the main-chains of the loops in 1K8U, 2DRI, 1TIB, 1MPP, and 135L, with the uniform and Ramachandran distributions for the dihedral angles (Sections 2.3.2 and 2.3.3). For each loop, we sampled 50 conformations with the uniform distribution and 50 with the Ramachandran distribution. We then ran SCWRL to place side-chains in the loop (with the side-chains in the rest of the protein fixed) and checked each conformation for collisions. Table 2.2 reports the number of collision-free
conformations (out of 50) for each loop. As expected, the main-chain conformations generated using the Ramachandran distribution facilitate the collision-free placement of the side-chains.

<table>
<thead>
<tr>
<th>Protein</th>
<th>1K8U</th>
<th>2DRI</th>
<th>1TIB</th>
<th>1MPP</th>
<th>135L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>7</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Ramachandran plots</td>
<td>18</td>
<td>14</td>
<td>6</td>
<td>4</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 2.2: The number of collision-free side-chain placements on conformations sampled with uniform and Ramachandran distributions.

2.4.3 Calcium-Binding Site Prediction

Calcium-binding proteins play a key role in signal transduction. Many such proteins share the same functional domain, a helix-turn-helix structural motif called EF-hand [61]; the calcium ion binds at the loop region in this motif. As a loop is often flexible, its conformation with calcium bound (called the holo state) and its conformation without calcium (the apo state) can be significantly different [3].

Many functional site prediction methods, for example FEATURE [113], are based on structural properties of the binding site. However, if the conformation of the functional site changes upon calcium binding, these methods may not be able to recognize the binding site in the apo state due to the absence of the binding structural properties. One way to overcome this problem is to sample many closed collision-free conformations of the loop and run the functional site prediction method on each of them. If a sampled conformation is recognized by the method, not only does this indicate that the loop may be a possible calcium-binding site, it also tells us what the holo conformation would look like. In fact, molecular dynamics simulation has already been used successfully to generate conformations starting with apo proteins in order to identify unrecognized calcium binding sites in them [42]. But MD simulation is computationally expensive.

§This part of the work was done in collaboration with Russ B. Altman and Inbal Halperin-Landsberg.
CHAPTER 2. CONFORMATION SAMPLING FOR PROTEIN LOOPS

For example, Parvalbumin [17] is a calcium-binding protein, where the loop ALA51-ILE58 is a binding site that flips up upon calcium-binding. The PDB IDs for its apo and holo structures are 1B8C and 1B9A, respectively. In Figure 2.4, these conformations are shown blue and green, respectively; the black dot is the center of the calcium ion in the holo PDB file. We sampled successive conformations of this loop using our sampling procedure and ran FEATURE on each of them, until FEATURE recognized a loop conformation as a calcium-binding site. The recognized conformation, shown red in Figure 2.4, is close to the holo structure 1B9A. The red dot represents the position of the calcium ion predicted by FEATURE in this recognized conformation. Similarly, the two green dots represent positions of the calcium ion predicted by FEATURE for the green holo conformation. Note that all these dots are all very close to the calcium position recorded in the PDB. Correctly, FEATURE did not recognize the apo conformation shown blue as a binding-conformation; hence, there is no blue dot in the figure.

Figure 2.4: Parvalbumin loop ALA51-ILE58: The apo and holo conformations recorded in the PDB are shown in blue and green, respectively. The loop conformation in red is the conformation generated by our sampling method and recognized by FEATURE as a calcium-binding site. The black dot is the position of the calcium ion recorded in the PDB. The green and red dots are the calcium positions predicted by FEATURE for the loop conformations of the same color.
2.5 Comparison with Previous Methods

Comparing methods is delicate because, as discussed in Section 2.2, these methods have different purposes. Thus, preferences in their solutions and evaluation metrics differ. RAPPER [33] and the hierarchical method in [55] focus on generating near-native conformations, while our sampler aims at exploring the closed collision-free conformation space. The results of Section 2.4.3 demonstrates the ability of our methods to generate both native conformations and other biologically important conformations that significantly differ from the native ones. Such results would be difficult to obtain with the methods presented in [33, 55].

Figure 2.5 plots the average running times of RAPPER (as reported in [33]) and those of our sampling procedure to get one conformation of one loop for different loop lengths. Although the absolute running times are subject to differences in computer speed and software coding, the trends shown in the figure suggest that our sampling method scales better than RAPPER when loop length increases. There is not enough data in [55] to provide a similar comparison.

Using discrete sets of $\phi$ and $\psi$ values derived from protein structure databases certainly reduces the size of the search space. In RAPPER, each residue has 5184 states, and the method in [55] assigns 215 to 866 states to each residue. On the other hand, it may also make it more difficult to sample collision-free conformations, especially non-native conformations. Furthermore, the methods in [33, 55] also incur the cost of running an energy minimization algorithm to generate near-native conformations. Overall, we believe that the fact that our sampling procedure seems to be faster than RAPPER and to scale better with loop length is mainly due to the constraint prioritization scheme embedded in our procedure.

The paper on RLG [29] only reports tests on a single 17-residue loop (named loop 7, between Gly433 and Gly449) of protein 1G5A. The goal of the work was to study the mobility of this loop in the presence and absence of certain side-chains. About 1 hour was needed to generate a tree of 1000 nodes using RLG (see Section 2.2), which amounts to 3.6 seconds per conformation. On this same loop, our sampling method takes 3.28 seconds per conformation. However, in [29] a less stringent overlap factor
was used to test atomic collisions. Moreover, it is unknown how quickly the tree generated by RLG expands across the loop’s closed collision-free conformation space.

### 2.6 Conclusion

In this chapter, we have described a method to sample broadly distributed conformations in the closed collision-free conformation space of a protein loop. It is based on a novel prioritized constraint-satisfaction approach that interweaves the treatment of collision avoidance and cycle closure constraints. Experiments show that it can efficiently handle loops ranging from 5 to 25 residues in length. Additional tests demonstrate its ability to generate biologically interesting loop conformations, such as calcium-binding conformations. This critical ability could be used in the future
CHAPTER 2. CONFORMATION SAMPLING FOR PROTEIN LOOPS

to predict loop conformations and improve other structure prediction techniques, like homology, when functional information is known in advance. Moreover, the broadly-distributed samples can be used as “seeds” to explore interesting neighborhoods in the conformation space. These neighborhoods can then be finely explored using another sampler developed by Ankur Dhanik, called deformation sampler [117].

In the future, it would be interesting to develop methods to simultaneously sample conformations of several loops on the same protein, assuming the rest of the protein is rigid. This will be helpful to the design of antibody therapeutics, where the 6 loops known as complementarity determining regions (CDRs) in the variable domain of an antibody are responsible for recognizing the specific antigen [63].

One limitation of loop sampling is that it requires the two anchors of the loop to keep exactly the same relative position and orientation while the loop deforms. However, in practice, when a loop deforms, its two anchors also move somewhat, even if it is by a small amount. This now leads us to consider the more general problem of sampling folded conformations of entire protein.
Chapter 3

Conformation Sampling for Folded Proteins*

3.1 Problem Statement

In this chapter, we address the computational problem of exploring the folded state of a given protein by sampling conformations broadly and approximately evenly distributed over the folded state under kino-geometric constraints. This is a much harder problem than sampling loop conformations, as it requires dealing with larger sets of atoms and more complex kinematic models containing multiple interdependent closed kinematic cycles. Again, our sampling method does not address the problem of recognizing functional conformations in the computed distribution of conformations. We assume that this problem is handled by another software program, such as FEATURE [113].

3.2 Related Work

Two previous samplers, ROCK [118] and FRODA [115], have particularly influenced our work:

*This work was done in collaboration with Liangjun Zhang.
1. ROCK (for Rigidity Optimized Conformational Kinetics) transforms covalent bonds, H-bonds (with potential energy less than a given threshold) and hydrophobic contacts into distance constraints among atoms. Using the 3D Pebble Game algorithm [54], it determines rigid groups of atoms. The result is a kinematic model of the protein that is made of rigid groups connected by variable dihedral angles around rotatable covalent bonds. This model usually contains many closed cycles. To sample new conformations, ROCK performs a random walk. At each step of the walk, it perturbs at random dihedral angles around rotatable bonds not contained in any cycle. In addition, it perturbs all variable dihedral angles in each cycle, except six, which are then solved using the cycle closure equations. Since it treats cycles sequentially, deformation of each cycle result in breaking previously treated cycles sharing variable dihedral angles with it. So, once all cycles have been treated, ROCK iteratively minimizes to zero a “potential” function measuring the gaps in the broken cycles. Due to conflicting cycle closure constraints, this function can have many local minima. The minimization process tends to be slow and is not guaranteed to succeed. Once the cycles have been closed, the obtained conformation is checked for collision.

2. FRODA (for Framework Rigidity Optimized Dynamic Algorithm) performs the same rigidity analysis as ROCK. It also performs a random walk, but differs from ROCK in the way it samples each new conformation. The goal of FRODA is to be much faster than ROCK. The positions of all the atoms are first perturbed at random. Then iterative optimization is used to fit the relative positions of the atoms in every rigid group $R$ back to the geometric template associated with $R$, while avoiding collision between atoms from different groups. This has the indirect effect of achieving cycle closure. Experiments with FRODA show that each step of the random walk is 100 to 1000 times faster than with ROCK. However, our own experience with FRODA has been that the RMSD magnitude of each step is very small. Indeed, to easily fit back atoms to rigid body templates, the random perturbations of the atoms must be small (less than 0.1Å), while the subsequent process of fitting back the atoms to the templates tends to cancel
out the initial deformation. In addition, the method has difficulty generating deformations in which large groups of atoms perform correlated moves.

Both ROCK and FRODA suffer from the inherently slow diffusion rate of random walks. The iterative methods used by ROCK to generate each new sample are slow. Those used by FRODA are much faster, but produce sampling steps of small magnitude. The non-biased, diffusive sampling strategy and the Jacobian-based cycle deformation method of KGS are specifically aimed at avoiding these drawbacks. KGS’s diffusive strategy guides exploration toward less visited space. Its Jacobian-based cycle deformation method naturally keeps cycles closed, so does not require an iterative procedure to close them later; it is fast and allows larger steps than FRODA. On the other hand, the rigidity analysis module of KGS is the same as in ROCK and FRODA; only the selection of stable H-bonds differs.

Another sampling strategy using rigidity analysis (again, with the Pebble Game algorithm) is proposed in [99]. The goal of this method is to sample the nodes of a “roadmap” model (a network of sampled conformations connected by simple trajectories) of protein motion. Here, sampling is not limited to the folded state; so, the generated samples are retained in the roadmap model according to their potential energy. The results of rigidity analysis are used to guide each sampling operation by selecting which dihedral angles to perturb. For instance, angles in rigid groups of atoms are perturbed with much smaller probability than other angles. But, unlike ROCK, FRODA, and KGS, the sampler allows rigid atom groups to be slightly deformed and cycle closure constraints to be broken.

The Pebble Game algorithm is only one possible method to determine rigid groups of atoms. For instance, other methods compare known folded conformations [10, 78, 116] or predict rigidity from a single conformation [56, 72, 79]. Such methods could also be used in KGS, but the Pebble Game algorithm is fast and not restricted to any type of protein or conformation.
CHAPTER 3. CONFORMATION SAMPLING FOR FOLDED PROTEINS

3.3 Methods

3.3.1 Overview

The inputs to KGS consist of an initial protein conformation \( q_{\text{initial}} \) and an RMSD threshold \( \rho \). The goal of KGS is to sample conformations broadly and approximately evenly distributed over the region \( F_\rho \) defined as the subset of the protein’s folded state lying within RMSD \( \rho \) from \( q_{\text{initial}} \). The sampling process ends when an independently-defined termination criterion is achieved. For instance, this criterion may recognize that a target conformation has been generated. Usually, it also specifies an upper bound on the running time and/or the number of sampled conformations. In the experiments of Section 3.4, KGS simply terminates when a certain number of conformations has been sampled.

Figure 3.1 provides a chart of the components of KGS. The sampler expands a distribution \( \Delta \) of sampled conformations iteratively, starting from the input conformation \( q_{\text{initial}} \). At each step a seed conformation \( q_{\text{seed}} \) within RMSD \( \rho \) from \( q_{\text{initial}} \) is picked from the current distribution and is deformed into a new conformation \( q_{\text{new}} \). If \( q_{\text{new}} \) is free of collision, then it is added to the distribution, otherwise it is discarded. The key components of KGS are described below.

3.3.2 Protein Model and Notations

A protein conformation is represented using the linkage kinematic model denoted by \( L \), as described in Section 1.3.2. Bond lengths, bond angles, and non-rotatable dihedral angles are all fixed to their values in \( q_{\text{initial}} \).

Deforming \( q_{\text{seed}} \) consists of perturbing at random the variable dihedral angles in \( L \). However, just doing this would lead to exploring a space of much higher dimensionality than the protein’s folded state since it would break key H-bonds stabilizing the folded structure [4]. So, like in [115, 118], we identify the most stable H-bonds present in \( q_{\text{seed}} \) and we require that they be not broken during the deformation. This requirement leads to imposing additional bond lengths and bond angles constraints to the linkage model of the protein, in the same way as covalent bonds. It freezes some variable
Figure 3.1: Overview of the KGS algorithm.
dihedral angles in $L$, thus creating rigid groups of atoms. It also creates closed kinematic cycles, due to H-bonds connecting atoms from distant residues along the main-chain. Each kinematic cycle is a circular sequence of rigid groups of atoms $R_1, \ldots, R_p, R_{p+1}$, where $R_{p+1} = R_1$, and any two consecutive groups $R_i$ and $R_{i+1}$ $(i = 1$ to $p$) share exactly two atoms bonded by a rotatable bond — either a covalent bond from the original model $L$ or an H-bond (see Section 3.3.4 for more details). We denote the resulting constrained kinematic model of $q_{seed}$ by $L_{const}$.

To illustrate, Figure 3.2(a) shows a 6-residue protein fragment with three internal H-bonds. The graph in Figure 3.2(b) depicts the constrained kinematic model $L_{const}$ for this fragment. Each node of $L_{const}$ corresponds to a rigid group of atoms and each edge corresponds to a remaining rotatable bond (a covalent bond or an H-bond). $L_{const}$ contains three interdependent cycles colored red, cyan, and yellow, respectively. These three cycles share rotatable bonds, so cannot be deformed independently of each other.
3.3.3 Selection of Stable H-bonds

H-bonds are weaker than covalent bonds, but the most stable H-bonds are critical to the stabilization of both the secondary structure elements and the tertiary structure of a folded protein [4, 35, 90]. On the other hand, less stable H-bonds in two distinct folded conformations of the same protein can be very different. Figure 3.3 displays the common (in blue) and different (in red and green) H-bonds in two known conformations of the Escherichia coli catabolite gene activator protein (PDB ID: 1G6N). As one can see, most of the commonly-believed-to-be-stable H-bonds, such as those in α-helices or β-sheets, are present in both conformations. However, some other H-bonds are present in only one of the two conformations. If all H-bonds in one conformation were modeled as constraints, then it would be impossible for the protein to adopt the other conformation.

In order to constrain the deformation of q_seed appropriately, only the most stable H-bonds present in q_seed must not be broken. If we retain too many H-bonds, \( L_{\text{const}} \) will be too rigid for KGS to successfully explore the folded state of the protein. But if we allow stable H-bonds to break, KGS will waste time exploring a region larger than the actual folded state.

We first identify all H-bonds present in q_seed using the geometric criteria given in [74] and illustrated in Figure 3.4. Wells et al [115] identifies an H-bond as stable if its potential energy (defined according to [31]) is lower than a certain threshold. However, energy is only one possible predictor of H-bond stability. It does not take the local environment of the H-bond into account. In KGS, we use a probabilistic model of H-bond stability that has been trained on MD simulation data (see Chapter 4). For each H-bond present in q_seed, this model returns a number in [0,1] measuring the probability that this bond will remain present over an extended period of time.

For each H-bond present in q_seed, let \( \pi \) be the stability probability returned by the stability model. KGS retains this H-bond as stable with probability \( \pi \). Non-retained H-bonds are allowed to break during the deformation of q_seed, hence are not used to constrain the linkage model of the protein.
Figure 3.3: Common (in blue) and different (in red and green) H-bonds in two known conformations of the Escherichia coli catabolite gene activator protein (PDB ID: 1G6N).

Figure 3.4: The geometric criteria of a hydrogen bond.
3.3.4 Rigidity Analysis

To construct the constrained model $L_{\text{const}}$, we need to determine the rigid groups of atoms created by the selected stable H-bonds. Here, we use the method implemented in ROCK [118] and FRODA [115]. A distance graph is first computed, in which each node represents a protein’s atom and each edge corresponds to an equality distance constraint between two atoms. As described in [53], the distance constraints encoded in this graph are derived from both the kinematic linkage model (lengths of covalent bonds and angles between adjacent covalent bonds) and the geometry of the selected H-bonds:

1. Each covalent bond yields one distance constraint. Each pair of two consecutive bonds, between atoms A and B and between B and C, yields one distance constraint between A and C to maintain the bond angle A-B-C fixed. Each non-rotatable covalent bond yields an additional distance constraint to rigidify the group of four atoms that define the fixed dihedral angle around the non-rotatable bond.

2. Each H-bond yields three distance constraints (see Figure 3.4 for atom identifiers): one between atoms H and A, one between atoms D and A (to keep the angle D-H-A fixed to its value in $q_{\text{seed}}$), and one between atoms H and AA (to keep the angle H-A-AA fixed to its value in $q_{\text{seed}}$). No distance constraint is generated to keep the angle D-A-AA fixed. This encoding of the geometry of an H-bond relies on the observation that a stable H-bond is highly directional, that is, its four atoms D, H, A, and AA are almost aligned. So, the angles D-H-A and H-A-AA are much larger than 90° and almost fixed (often 150° or larger). As a result, the dihedral angle (D-H)-(A-AA) can vary freely without breaking the constraint on angle D-A-AA. In the following, we will call this angle the H-bond’s dihedral angle.

The 3D Pebble Game algorithm implemented in ROCK and FRODA is then applied to identify the rigid groups of atoms in $q_{\text{new}}$. Every pair of adjacent rigid groups of atoms identified by this algorithm shares exactly two atoms. Either these
two atoms are connected by a rotatable covalent bond, or they are the H-A atoms of an H-bond. Only the dihedral angles around these shared bonds can vary in $L_{\text{const}}$ (see Figure 3.5).

Figure 3.8 shows the largest rigid groups of atoms identified in the initial conformations of the three proteins used in the experiments of Section 3.4.

### 3.3.5 Deformation of $q_{\text{seed}}$

$L_{\text{const}}$ is a kinematic linkage formed by the identified rigid groups of atoms and the variable dihedral angles around the rotatable bonds — both covalent bonds and H-bond — connecting them. As this linkage usually contains closed kinematic cycles, some of its dihedral angles can only be perturbed in a coordinated way. The space $C_{\text{const}}$ of conformations achievable by $L_{\text{const}}$ is implicitly defined by the closure equality constraints. Since these constraints are nonlinear, $C_{\text{const}}$ is a nonlinear sub-manifold of the space $C$ of conformations achievable by $L$. Unlike for $L$, there is no parameterization of $C_{\text{const}}$ that allows us to directly sample conformations of $L_{\text{const}}$ [65, 84].

In order to sample $q_{\text{new}}$, we approximate $C_{\text{const}}$ in the neighborhood of $q_{\text{seed}}$ by its tangent space $T C_{\text{const}}(q_{\text{seed}})$ and we perturb $q_{\text{seed}}$ by a small vector $u$ picked in this space. By definition, $T C_{\text{const}}(q_{\text{seed}})$ is the linear space of all vectors $v$ such that an
Figure 3.6: The left part of the figure shows a spanning tree $T_{\text{const}}$ of $L_{\text{const}}$. The red edge between the rigid groups $R_1$ and $R_2$ represents a rotatable bond of $L_{\text{const}}$ not contained in $T_{\text{const}}$. This bond completes a cycle (boxed in dashed line), and is called a cycle-closing bond. The right part of the figure illustrates the cycle in more detail. The two dots in cyan depict the two atoms $A$ and $B$ shared by the rigid groups $R_1$ and $R_2$ and connected by the cycle-closing bond.

infinitesimal change along $v$ of the dihedral angles of $L_{\text{const}}$ does not cause a first-order-magnitude break-up of the kinematic cycles. Its axes define the instantaneous\(^2\) DOFs of $L_{\text{const}}$. This approach has been previously used in robotics [48, 93]. But, while in our case $L_{\text{const}}$ has often hundreds of instantaneous DOFs and several dozen cycles, robot models are usually much simpler. For instance, the relatively complicated 6-legged robot in [48] has at most 6 cycles with 24 instantaneous DOFs.

To easily pick $u$, we compute a basis of $TC_{\text{const}}(q_{\text{seed}})$ as follows. We first identify a complete non-redundant set of cycles by creating a spanning tree $T_{\text{const}}$ of $L_{\text{const}}$ in the following way. We pick a rigid group of atoms $G$ to be the root of $T_{\text{const}}$ (refer to Figure 3.6). We install every rigid group sharing a rotatable bond with $G$ and not already in $T_{\text{const}}$ as a child of $G$ in $T_{\text{const}}$. We proceed in the same way with every descendant of $G$ until all rigid groups are in $T_{\text{const}}$, which is guaranteed to happen

\(^2\)The DOFs of $L_{\text{const}}$ depend on the conformation of $L_{\text{const}}$, since the orientation of the tangent space $TC_{\text{const}}(q)$ changes when the conformation $q$ of $L_{\text{const}}$ is deformed. So, we call them here “instantaneous”. The number of instantaneous DOFs remains constant over all non-singular conformations of $L_{\text{const}}$. 
since the original model \( L \) is connected. Every rotatable bond \( l \) of \( L_{\text{const}} \) (either a covalent bond or an H-bond) not present in \( T_{\text{const}} \) completes a cycle \( \lambda \) (boxed in dashed line) of \( L_{\text{const}} \). We call this bond a cycle-closing bond (shown in red). Let this bond be between two atoms \( A \) and \( B \) (shown as cyan dots) shared by rigid atom groups \( R_1 \) and \( R_2 \). Let \( R \) be the closest common ancestor of these two nodes in \( T_{\text{const}} \). We divide cycle \( \lambda \) into two chains connecting \( R \) to \( R_1 \) (marked as a green path) and to \( R_2 \) (marked as a pink path), respectively. For cycle \( \lambda \) to stay closed during the perturbation, each of the two atoms \( A \) and \( B \) must undergo exactly the same displacement along these two chains. So, each cycle identified in the spanning tree yields six equations, each of the form:

\[
f_{i,1}(q) - f_{i,2}(q) = 0 \tag{3.1}
\]

where \( q \) is the vector of dihedral angle values generated by perturbing the variable dihedral angle values in \( q_{\text{seed}} \). \( f_{i,1}(q) \), \( i = A, B \), stands for either one of the \( x \), \( y \), or \( z \) coordinates of the center of atom \( A \) or \( B \) calculated by propagating the deformation of \( q_{\text{seed}} \) along the chain connecting \( R \) to \( R_1 \), and \( f_{i,2}(q) \) stands for the same coordinate calculated by propagating the deformation along the chain connecting \( R \) to \( R_2 \). Note that the six equations 3.1 are slightly redundant since the distance between \( A \) and \( B \) is fixed.

The cycles identified as above form a complete set of cycles, i.e., it covers all the possible cycles in \( L_{\text{const}} \). In addition, this set is non-redundant, i.e., removing any cycle would make it incomplete. Let \( n \) be the number of variable dihedral angles in \( L_{\text{const}} \) and \( m \) the number of cycles detected by comparing the spanning tree \( T_{\text{const}} \) to \( L_{\text{const}} \). By differentiating the resulting \( 6m \) equations 3.1, we obtain the linear system of equations:

\[
J \times dq = 0 \tag{3.2}
\]

where \( J \) is a \( 6m \times n \) matrix. The elements of \( J \) are constants that can be calculated directly from the geometry of \( q_{\text{seed}} \) [93], without having to write equations 3.1 explicitly. The solutions \( dq \) of this linear system are all the tangent vectors of \( C_{\text{const}} \) at \( q_{\text{seed}} \).
i.e., all the vectors in $TC_{\text{const}}(q_{\text{seed}})$. In the general case where $q_{\text{seed}}$ is non-singular, they form an $(n - 5m)$-dimensional linear space. Indeed, each detected cycle removes only 5 dimensions since, as already mentioned, cycle-closing bonds have fixed lengths. Let $U\Sigma V^T$ be the singular value decomposition (SVD) of $J$ [45], where $\Sigma$ is a $6m \times n$ matrix whose left $6m \times 6m$ sub-matrix is diagonal and right $6m \times (n - 6m)$ sub-matrix is filled with 0. The rows of $\Sigma$ are sorted from the top in decreasing order of the singular values. Because of the redundancy in equations 3.1, the last $m$ computed singular values are zero. The last $n - 5m$ column vectors in matrix $V$ form a basis $N$ of $TC_{\text{const}}(q_{\text{seed}})$ [106].

To deform $q_{\text{seed}}$ into $q_{\text{new}}$, we generate a vector $u \in TC_{\text{const}}(q_{\text{seed}})$ by picking its $n - 5m$ components along $N$ independently at random, over a $360^\circ$ interval. We then scale $u$ so that its modulus and largest component are no greater than pre-specified thresholds. We set $q_{\text{new}}$ to be $q_{\text{seed}} + u$. Since $TC_{\text{const}}(q_{\text{seed}})$ is only a linear approximation of $C_{\text{const}}$ at $q_{\text{seed}}$, the new conformation $q_{\text{new}}$ does not lie exactly in $C_{\text{const}}$, i.e., some cycles may be slightly broken. The small gaps could easily be removed using exact inverse kinematics [30]. However, our experiments show that gaps stay reasonably small (see Section 3.4.4) even after long sequences of sampling operations. So, this correction is not made in KGS.

The computational complexity of sampling $q_{\text{new}}$ is dominated by the cost of computing the SVD of matrix $J$. This operation can be done in $O(nm^2)$ time [45]. Fortunately, it does not need to be performed at every sampling step (see Section 3.3.8). Moreover, the Jacobian matrix must only be constructed for the variable dihedral angles that occur into cycles, since the other variable dihedral angles can be independently perturbed. For instance, for the protein conformation of PDB ID 1G6N (see Section 3.4.1), $L_{\text{const}}$ contains 80 cycles, and only 577 of the 1110 variable dihedral angles of $L_{\text{const}}$ appear in cycles. The tangent space reduced to these 577 angles has $577 - 5 \times 80 = 177$ dimensions. When cycles can be grouped into several subsets such that no two cycles from distinct subsets share variable dihedral angles, the computational cost can be reduced further by forming a separate Jacobian matrix for each group of cycles.
3.3.6 Validation of $q_{\text{new}}$

Before being inserted into the distribution $\Delta$, the newly sampled conformation $q_{\text{new}}$ is tested for collision among atoms. Recall from Chapter 1 that two atoms are considered to be colliding with each other if the distance between their centers is smaller than $\varepsilon$ times the sum of their van der Waals radii (see Section 1.3.2). In KGS, $\varepsilon$ is set to 0.6. This value is a bit smaller than the values used in previous works (e.g., [115, 117]), so that the conformations $q_{\text{initial}}$ drawn from the PDB are free of collision. It leads KGS to explore a slightly larger space, but this may actually be an advantage when some regions of a folded state are difficult to reach because of narrow passages. Collisions are checked using the grid method already presented in Section 2.3.5, which runs in time linear in the number of atoms in the protein.

The current version of KGS makes no other validation test. But other tests are possible. For example, it would be easy to check that $q_{\text{new}}$ satisfies inequality distance constraints. Such tests would be useful to maintain certain geometric relations among atoms (e.g., hydrophobic contacts among residues) that cannot be defined properly by equality distance constraints. KGS could also compute the potential energy of $q_{\text{new}}$ and use the Metropolis criterion to decide whether $q_{\text{new}}$ should be retained.

3.3.7 Selection of $q_{\text{seed}}$ and Diffusive Strategy

The choice of $q_{\text{seed}}$ from the current distribution $\Delta$ at each sampling step is crucial. As already mentioned at the beginning of this chapter and in Section 3.1, the goal of KGS is to sample conformations broadly distributed over the $F_\rho$ region. We would like KGS to first quickly expand $\Delta$ throughout $F_\rho$ and then to increase sampling density roughly evenly over $F_\rho$.

A purely random pick of $q_{\text{seed}}$ from $\Delta$ would fail to achieve this goal. The iterative sampling process would then be akin to performing random walks, which are known to expand at a very slow rate. To address a similar issue, two types of diffusive strategies have been proposed in the field of robot motion planning [52, 67]. The strategy proposed in [52] aims at sampling regions that have been less densely sampled so far. Within KGS, this strategy would consist of estimating the sampling density
around each conformation in $\Delta$ and selecting $q_{\text{seed}}$ with a probability inverse to its sampling density. For each conformation $q$ in $\Delta$, it would thus require updating the number of other conformations in $\Delta$ contained in a ball of a given RMSD radius centered at $q$. This would lead to computing a number of RMSD between sampled conformations quadratic in the size of $\Delta$, which would be prohibitively expensive. The other strategy [67] would consist of sampling a conformation $q$ (not necessarily a feasible one) by picking every variable dihedral angle in $L$ (the non-constrained kinematic model of the protein) uniformly at random and selecting $q_{\text{seed}}$ to be the conformation in $\Delta$ that is closest RMSD-wise from $q$. This strategy would first expand $\Delta$ quickly from $q_{\text{initial}}$ (something that we want), but then it would lead to sampling most new conformations near the boundary of $F_\rho$ since this region is tiny relative to the conformation space $C$ of $L$. At every sampling step, this strategy would also require computing a number of RMSD linear in the size of $\Delta$ to select $q_{\text{seed}}$.

The diffusive strategy implemented in KGS is inspired from the second strategy, but avoids its drawbacks. Whenever a new conformation is added to $\Delta$, we compute its RMSD to $q_{\text{initial}}$ and we maintain a list of all conformations in $\Delta$ sorted in increasing RMSD values. To select $q_{\text{seed}}$, we first sample a conformation $q$ by drawing every variable dihedral angle in $L$ uniformly at random, then we pick two values $r_1$ and $r_2$ uniformly at random over the range of RMSD values spanned by the sorted list of conformations in $\Delta$, and finally we select $q_{\text{seed}}$ to be the conformation in $\Delta$ that is closest RMSD-wise from $q$ among those that lie between $r_1$ and $r_2$ from $q_{\text{initial}}$. This strategy ensures that $\Delta$ first expands quickly until it reaches the boundary of $F_\rho$ and then increases its density broadly across $F_\rho$. Moreover, it reduces the average number of RMSD computations at each sampling step. If the RMSD between a sampled conformation $q_{\text{new}}$ and $q_{\text{initial}}$ is greater than $\rho$, then $q_{\text{new}}$ is still inserted in $\Delta$, but it is not considered later as a possible seed conformation.

### 3.3.8 Reduction of Running Time

Rigidity analysis and SVD are the two most computationally intensive operations at each sampling step. In order to reduce the average running time per sample, KGS
performs these operations only for a relatively small number of conformations in $\Delta$, which have been designated as root conformations. Every other conformation in $\Delta$ has one of these conformations as its root. Every root conformation has itself as its root.

Conformation $q_{\text{initial}}$ is initially inserted into $\Delta$ as a root conformation. When a conformation $q_{\text{seed}}$ is selected at a sampling step, it is deformed into a new conformation $q_{\text{new}}$ using the tangent space $T C_{\text{const}}(q_{\text{root}})$ computed at its root conformation $q_{\text{root}}$. If $q_{\text{new}}$ satisfies the validation test, then it is added to $\Delta$. If the RMSD between $q_{\text{new}}$ and $q_{\text{root}}$ is smaller than a specified threshold, then $q_{\text{root}}$ is set as the root of $q_{\text{new}}$. Otherwise, $q_{\text{new}}$ is designated as a new root conformation in $\Delta$. A new tangent space $T C_{\text{const}}(q_{\text{new}})$ is generated for $q_{\text{new}}$.

Keeping the same stable H-bonds in conformations that have the same root makes sense since these conformations are close apart. However, the basis of the tangent space of $C_{\text{const}}$ varies between conformations with the same kinematic model $L_{\text{const}}$. This variation turns out to be small between conformations that are close apart. Our tests reported in Section 3.4.4 show that using the same basis for all conformations with the same root does not lead to any significant cycle break-up.

### 3.4 Results

In this section we present and discuss experimental results obtained with three very different test proteins. For each protein, the initial conformation is drawn from the PDB. We specify one or several target conformations either obtained from MD simulation or drawn from the PDB. Section 3.4.1 presents the three proteins. Section 3.4.2 describes the computation carried out with each test protein. Section 3.4.3 analyzes the distributions of conformations generated. Section 3.4.4 checks cycle closure over these distributions. Section 3.5 will compare results obtained using KGS with results obtained using FRODA.
3.4.1 Test Proteins

Experiments were carried out using the following three proteins:

1. *The mating pheromone Er-1 from Euplotes raikovy* (PDB ID: 2ERL): It contains 567 atoms. The folded structure is made of three \( \alpha \)-helices. Molecular Dynamic simulation reported in [58] shows that the first \( \alpha \)-helix (starting from the N-terminal) moves relative away from the rest of the protein, which remains almost unchanged. Figure 3.7(a) displays the initial conformation (grey) and the conformations at 100ps (yellow), 200ps (green), and 300ps (blue). These three conformations are within 2\( \AA \) of the initial conformation. We consider them as target conformations in the analysis of Section 3.4.3.

2. *The Escherichia coli catabolite gene activator protein* (PDB ID: 1G6N): It contains 3211 atoms. The folded structure is a mix of \( \alpha \)-helices and \( \beta \)-strands. From its initial conformation, this protein undergoes an allosteric transition upon binding of cyclic AMP (cAMP) and then binds to DNA specifically to activate transcription [83]. The allosteric transition is mainly a hinge motion between its two domains, as shown in Figure 3.7(b), where the grey conformation is the initial one and the blue conformation is the one after the hinge motion. The two conformations are 2.66\( \AA \) apart. They are the chain A and chain B of the PDB entry 1G6N, respectively. The conformation after the hinge motion is the target conformation in the analysis of Section 3.4.3.

3. *Cyanovirin-N* (PDB ID: 2EZM): This potent HIV-inactivating protein, which contains 992 atoms, consists of two sequence repeats with 32% sequence similarity. The folded structure is made mainly of \( \beta \)-strands. A domain swapping happens between two Cyanovirin-N molecules when one repeat of one molecule binds a repeat of the other molecule. The domain swapped dimer has higher anti-viral affinity than the monomer [9]. Both the monomer and the dimer can exist in solution with a high-energy transition barrier between them. We consider the monomer (shown grey in Figure 3.7(c)) as the initial conformation. The domain-swapped conformation (shown in blue, PDB ID: 1L5E) is 16\( \AA \) away
CHAPTER 3. CONFORMATION SAMPLING FOR FOLDED PROTEINS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2ERL</td>
<td>567</td>
<td>153</td>
<td>184</td>
<td>15</td>
<td>83</td>
</tr>
<tr>
<td>1G6N</td>
<td>3211</td>
<td>931</td>
<td>1110</td>
<td>80</td>
<td>577</td>
</tr>
<tr>
<td>2EZM</td>
<td>992</td>
<td>221</td>
<td>502</td>
<td>47</td>
<td>279</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics of the proteins in the experiments.

and is the target conformation in the analysis of Section 3.4.3.

We refer to these proteins by the PDB IDs of their initial conformations in the rest of the text.

3.4.2 Computations Performed by KGS

For each test protein, KGS performed rigidity analysis only once for a set of H-bonds selected in the initial conformation. So, all sampled conformations were generated using the same constrained kinematic model $L_{\text{const}}$ derived from this analysis. However, in order to avoid any significant break-up of kinematic cycles, the basis of the tangent space at a seed conformation $q_{\text{seed}}$ was recomputed whenever the RMSD between $q_{\text{seed}}$ and its root conformation was greater than 0.5Å (see Section 3.3.8).

For both 2ERL and 1G6N, KGS used the stability model introduced in Section 3.3.3 to select the stable H-bonds in $q_{\text{initial}}$. The RMSD radius $\rho$ of the region $F_{\rho}$ was set to 6Å. For 2EZM, only the H-bonds present in both the initial and the domain-swapped conformations were retained as stable and $\rho$ was set to 25Å. The values of $\rho$ were set much larger than the RMSD between the initial and target conformations to avoid providing any strong indication about the locations of the target conformations to KGS.

Table 3.1 gives the number of rigid groups of atoms and the number of variable dihedral angles in the constrained kinematic model of each protein, along with the number of cycles and the number of variable dihedral angles in cycles. So, the three models have quite different kinematic complexity. Figure 3.8 shows the largest rigid groups of atoms identified in the initial conformations of the three proteins.
Figure 3.7: The initial and target conformations of test proteins. For each of the three proteins, the initial conformation is shown in grey. For 2ERL, the three target conformations are shown in green, yellow, and blue, respectively. For 1G6N and 2EZM, their single target conformations are shown in blue.
Figure 3.8: The largest rigid group of atoms in each of the three test proteins: 52 atoms for 2ERL, 48 for 1G6N, and 20 for 2EZM.
For each protein, we ran KGS multiple times with different seeds of the random number generator and we generated distributions of various sizes. In all cases, we obtained very similar results independent of the seed. Distributions bigger than 5000 conformations did not achieve significantly better results than those reported below. So, here we only present three “typical” runs of KGS. In each run, KGS sampled 5000 conformations of one of the three test proteins. The total running times\(^3\) (including H-bond selection and rigidity analysis) were 10 minutes, 13 hours, and 93 minutes for 2ERL, 1G6N, and 2EZ, respectively. The average running times per sample were 0.12, 9.36, and 1.12 seconds.

### 3.4.3 Analysis of Computed Distributions

For each of the three considered distributions, the curves plotted in Figure 3.9 show:

1. The maximum RMSD between the initial conformation and the conformations sampled so far during the iterative sampling process. The purpose is to check that the distribution quickly expands away from the initial conformation. The asymptote for two curves are greater than ρ because all sampled conformations are inserted in ∆ (see end of Section 3.3.7).

2. The minimum RMSD distance between the conformations sampled so far and the target conformations (three for 2ERL, one for 1G6N, one for 2EZM). The purpose is to check that KGS eventually samples conformations close to the targets.

Table 3.2 also gives the exact values of RMSD and the depth of the path leading to the conformation which is closest to the target. In addition, Figure 3.10 shows the initial conformation (grey), the target conformation (blue) and the sampled conformation (magenta) that is closest to the target. For 2EZM, two intermediate conformations on the path from the initial to the closest are displayed in yellow and orange.

---

\(^3\)KGS is implemented in C++ and runs under Linux. Our tests were performed on a dual quad-core 3GHz computer with 16GB of memory.
Figure 3.9: The max RMSD between the initial conformation and the conformation sampled so far is plotted in blue. The min RMSD to the target conformations so far is plotted in other colors.
### Table 3.2: Minimum RMSD to targets and running time of generating 5000 conformations.

<table>
<thead>
<tr>
<th>Protein</th>
<th>Target</th>
<th>Initial RMSD to target (Å)</th>
<th>Min RMSD to target (Å)</th>
<th>Best Path length</th>
<th>Max RMSD to initial (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ERL</td>
<td>100ps</td>
<td>1.74</td>
<td>0.79</td>
<td>2</td>
<td>7.55</td>
</tr>
<tr>
<td></td>
<td>200ps</td>
<td>1.83</td>
<td>1.00</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300ps</td>
<td>1.94</td>
<td>1.09</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1G6N</td>
<td>hinge motion</td>
<td>2.66</td>
<td>1.31</td>
<td>16</td>
<td>6.44</td>
</tr>
<tr>
<td>2EZM</td>
<td>domain swapping</td>
<td>16</td>
<td>6.31</td>
<td>59</td>
<td>21.36</td>
</tr>
</tbody>
</table>

Figure 3.10: Target conformation (blue) and the closest sampled conformation (magenta) aligned with the initial conformation (grey) for each of the three test proteins. For 2ERL, only the target conformation at 300ps is shown. For 2EZM, the figure also shows two intermediate conformations (yellow and orange) sampled by KGS.
These results show that the distribution of conformations expands quickly toward the boundary defined by the RMSD radius $\rho$. This boundary is reached or almost reached after about 130 samples for 2ERL, 1000 samples for 1G6N, and 2000 samples for 2EZM. For 2ERL and 1G6N, we also ran experiments with $\rho = 3\text{Å}$ so that $F_\rho$ is just big enough to include their target conformations. The boundaries are reached sooner. However, the minimum distances to targets are about the same as those in experiments with $\rho = 6\text{Å}$. In the case of 2EZM, it is not clear whether the $\rho$ boundary can be reached under the selected H-bond constraints. These results suggest that the ability for KGS to sample conformations close to a target is not sensitive to the value of $\rho$, as long as the target is inside $F_\rho$.

The distances to the target conformations also decrease rapidly to reach non-zero (but small) asymptotes. Separate tests show that these non-zero asymptotes are mostly caused by the fact that bond lengths and angles are not exactly the same in the initial and target conformations. For 2EZM, the minimum RMSD to the target conformation is higher than for the other two proteins, but still much smaller than the distance between the initial and target conformation. Indeed, for 2EZM to deform from its initial conformation to the target conformation, one domain must undergo a large rotation relative to the other. So, small geometric differences near the main rotation hinge lead to bigger global RMSD differences.

Overall, these results validate our initial assumption that the folded state can be well covered with a relatively small distribution of samples (a few thousands). They also demonstrate that KGS is capable of computing such a distribution quite efficiently. We must stress again that KGS did not know any of the target conformations (except for the selection of the stable H-bonds in 2EZM), hence sampling was not biased toward them. As mentioned in Section 3.1, our goal is to explore the folded state of a protein, not to predict any of the target conformations.

### 3.4.4 Checking Cycle Closure

Recall that we sample each conformation $q_{\text{new}}$ by deforming slightly a seed conformation $q_{\text{seed}}$. The deformation is done by perturbing the variable dihedral angles of
Table 3.3: Average and maximum cycle opening in all cycles in 5000 samples. The number of variable dihedral angles in the cycle having the MAX opening is also given.

<table>
<thead>
<tr>
<th>Protein</th>
<th>AVG (Å)</th>
<th>MAX (Å)</th>
<th>Length of most opened cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ERL</td>
<td>0.02</td>
<td>0.32</td>
<td>18</td>
</tr>
<tr>
<td>1G6N</td>
<td>0.10</td>
<td>2.11</td>
<td>206</td>
</tr>
<tr>
<td>2EZM</td>
<td>0.07</td>
<td>0.77</td>
<td>41</td>
</tr>
</tbody>
</table>

$L_{\text{const}}$ in the tangent space of $C_{\text{const}}$ calculated at the root conformation of $q_{\text{seed}}$. Since this tangent space is only an approximation of $C_{\text{const}}$ at $q_{\text{seed}}$, there is no guarantee that the cycles are not eventually significantly broken, especially after long sequences of sampling operations.

To check whether errors accumulate prohibitively, we examine the atom positions on all the cycle-closing bonds (see Section 3.3.5). Let us take the cycle $\lambda$ in Figure 3.6 as an example. Let $A_1$ and $B_1$ be the positions of atom $A$ and $B$ computed from the chain connecting $R$ to $R_1$, and $A_2$ and $B_2$ be the positions computed from the chain connecting $R$ to $R_2$. Let us define $\text{open}(\lambda) = \max(\|A_1 - A_2\|, \|B_1 - B_2\|)$. For each protein, we compute the average and maximum values of $\text{open}$ of all the cycles in all 5000 samples:

$$AVG = \frac{\sum_{i=1}^{5000} \sum_{j=1}^{N} \text{open}(\lambda_j)}{5000 \times N} \quad (3.3)$$

$$MAX = \max_{i=1}^{5000} \max_{j=1}^{N} \text{open}(\lambda_j) \quad (3.4)$$

where, $N$ is the number of cycles in $L_{\text{const}}$, and $\lambda_j$ denotes the $j$th cycle. Table 3.3 gives the values of $AVG$ and $MAX$ in Å, as well as the number of rotatable bonds in the cycle that yields the $MAX$ value. $AVG$ is very small for all 3 proteins. So are the values of $MAX$ for proteins 2ERL and 2EZM. The larger $MAX$ value for 1G6N occurs for a very long cycle containing 206 variable dihedral angles, in which small errors may accumulate. But the average value of $\text{open}$ for this cycle is only about 1Å, which is quite acceptable. These results indicate that, on average, cycle closure is well-maintained during deformation.
### 3.5 Comparison to FRODA

FRODA [115] is arguably the best kino-geometric conformation sampler available so far. Our sampler KGS actually reuses the rigidity analysis module from FRODA. But, as described in Section 3.2, it differs significantly in the way it samples each new conformation. KGS chooses each seed conformation in order to generate a distribution of sampled conformations that diffuses quickly throughout the protein’s folded state (Section 3.3.7). In addition, it deforms each selected seed conformation in tangent space (Section 3.3.5) in order to avoid breaking kinematic cycles.

To compare KGS with FRODA, we ran FRODA on each protein for the same amount of time as KGS did\(^4\). The bond constraints (both covalent and H-bonds) were exactly the same as in the KGS experiments to ensure that FRODA uses the same kinematic models as KGS. As on average FRODA performs deformation steps much faster than KGS, it generated many more samples than KGS in the same amount of time. To compare the distributions generated by the two samplers, we uniformly sub-sampled 5000 conformations from the distribution generated by FRODA. Table 3.4 gives the total number of iterations performed by FRODA, the average RMSD step size (in Å) over all iterations performed by FRODA, the average RMSD between two sub-sampled conformations generated by FRODA, and the average RMSD step size over all iterations performed by KGS. The last two columns show that KGS makes much larger steps by unit of time than FRODA for two of the test proteins (2ERL and 2EZM), and steps of the same size for the other protein (1G6N).

\(^4\)These computations were performed on the same dual quad-core 3GHz computer with 16GB of memory that was used to test KGS.

<table>
<thead>
<tr>
<th>Protein</th>
<th>Total # of FRODA iter.</th>
<th>Avg. step size of FRODA iter.</th>
<th>Avg. step size of 5000 conf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ERL</td>
<td>65,000</td>
<td>0.06</td>
<td>0.20</td>
</tr>
<tr>
<td>1G6N</td>
<td>620,000</td>
<td>0.06</td>
<td>0.46</td>
</tr>
<tr>
<td>2EZM</td>
<td>85,000</td>
<td>0.07</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Table 3.4: Comparison of RMSD step size of FRODA and KGS.
Figure 3.11 contains the plots of the RMSD from the sub-sampled 5000 conformations generated by FRODA to the initial and target conformations. These plots show that the distributions generated by FRODA expand from the initial conformations much more slowly than the distributions generated by KGS (see Figure 3.9) and that they do not get as close to the target conformations. This is not surprising since random walks in high-dimensional spaces, such as those performed by FRODA, are known to diffuse slowly. At each step, they have a greater probability to move away from a target conformation than closer to it. We believe that the superior performance of KGS is due not only to its diffusion strategy (Section 3.3.7), but also to its deformation method (Section 3.3.5). As mentioned in Section 3.2, the deformation method used by FRODA that first perturbs all atoms independently at random and then fits them back to the templates defined by the rigid groups of atoms tends to cancel out the initial deformation. Moreover, the method is not suitable to generate deformations in which large non-rigid groups of atoms perform correlated moves (e.g., as in 2EZM). Such deformations are easier to generate by perturbing dihedral angles around bonds.

Nevertheless, FRODA may be better suited than KGS for certain applications. In particular, the ability of FRODA to perform many small steps very quickly can be useful to explore a small neighborhood of a given initial conformation, as demonstrated in the experiments reported in [115]. By biasing the deformations toward a given target conformation, FRODA can also be used to generate finely sampled plausible pathways between an initial and a target conformation. Along this line, the Database of Macromolecular Movements (MolMovDB) [43] uses FRODA to compute a morphing between two given conformations [38]. KGS may not be able to perform such tasks as well.

3.6 Conclusion

In this chapter we described a kino-geometric conformation sampler, KGS, to explore the folded state of a protein. The two main contributions of KGS are (1) a robotics-inspired, Jacobian-based method to simultaneously deform a large number
Figure 3.11: RMSD to the initial (blue) and to the targets (other colors) of 5000 subsampled conformations generated by FRODA. These conformations are uniformly drawn from their FRODA trajectories. For example, the 5000th sample of 2ERL in the first plot is the 65000th sample in its trajectory.
of interdependent kinematic cycles without any significant break-up of the closure constraints, and (2) a diffusive strategy to sample conformations broadly distributed over a protein folded state. The Jacobian-based method is fast and makes it possible to perform relatively large steps at each sampling operation. The diffusive strategy expands quickly away from the input conformation $q_{\text{initial}}$ and progressively samples conformations more and more densely distributed over the protein’s folded state.

Experiments demonstrate that KGS is able to efficiently compute distributions containing conformations at small RMSD from target (e.g., functional) conformations without having been biased toward these configurations. Comparison with state-of-the-art FRODA shows that the distributions computed by KGS diffuse faster than FRODA and that sampling steps are larger.

In all experiments performed so far, the stable H-bonds are selected in the initial conformation $q_{\text{initial}}$ and remain the same throughout the sampling process. However KGS allows us to both select stable H-bonds and perform a new rigidity analysis in every new sampled conformation. So, it would be interesting to experiment with KGS on proteins for which there is clear evidence that some H-bonds must break and later re-form to move from $q_{\text{initial}}$ to a target conformation. At this stage, it is unknown if KGS would still be able to efficiently generate distributions containing conformations close to the targets in those scenarios.

Some proteins contain multiple flexible loops (between secondary structure elements) that deform independently. In this case, the first assumption stated in Section 1.4 would not be well satisfied. It would be interesting to explore an approach where (1) KGS would compute a distribution of $h_1$ conformations, and (2) a loop sampler, such as the one described in Chapter 2, would generate $h_2$ conformations for each loop in each of the $h_1$ conformations sampled by KGS (keeping the rest of the protein conformations fixed). In this way, we would get a distribution of $h_1 h_2^r$ conformations in time and space both roughly proportional to $h_1 + r \times h_2$, where $r$ is the number of separately deformed loops.
Chapter 4

Learning Hydrogen Bond Stability from MD Simulation Data*

4.1 Introduction

Recall from the previous chapter that KGS must reliably identify stable H-bonds in a given conformation before deforming it. But more generally, modeling H-bond stability is an important topic in itself.

An H-bond corresponds to the attractive electrostatic interaction between a covalent pair D-H of atoms, in which the hydrogen atom H is bound to a more electronegative donor atom D, and another non-covalently bound, electronegative acceptor atom A. Most H-bonds in a protein are of the form N-H...O or O-H...O, but other forms are possible. Due to their strong directional character, short distance ranges, and relatively large number in a folded protein, H-bonds play a key role in both the formation and stabilization of protein structures [4, 35, 90]. On the one hand, H-bonds between atoms from nearby main-chain residues are essential to the

---

*This work was done in collaboration with Igor Chikalov and Mikhail Moshkov at the King Abdullah University of Science and Technology (KAUST). The work will be presented at the 3rd International Conference on Machine Learning and Computing (ICMLC 2011) [23] and at the 9th Asia Pacific Bioinformatics Conference (APBC 2011) [24].

2We only consider H-bonds inside a protein. We ignore H-bonds between a protein and the solvent.
CHAPTER 4. LEARNING HYDROGEN BOND STABILITY

formation of secondary structure elements. On the other hand, H-bonds between atoms from distant main-chain residues are critical to the stabilization of a protein’s tertiary structure; they strengthen the inevitable burial of polar residues, as the failure of these residues to find hydrogen-bonding partners in the protein would have destabilizing effect [4, 50, 82, 90]. In particular, such H-bonds shape loops and other irregular features that may contain functional sites.

Unlike covalent bonds, H-bonds greatly vary in stability. They form and break while the conformation of a protein deforms. For instance, the transition of a folded protein from a non-functional meta-stable state into a functional (e.g., binding) state may require certain H-bonds to break and others to form [7]. The intrinsic strength of an individual H-bond has been studied from an energetic viewpoint [31, 68, 77, 85, 96]. However, potential energy alone may not be a very good indicator of H-bond stability. Other local interactions may reinforce (or weaken) an H-bond. Moreover, several “redundant” H-bonds may contribute to rigidify the same group of atoms [100]. Besides the fact that folded protein conformation samplers, such as KGS (see Chapter 3), need a tool to reliably select stable H-bonds, it is also desirable to identify the important determinants of H-bond stability in order to better understand protein flexibility.

We apply inductive learning methods to train a protein-independent probabilistic model of H-bond stability from a training set of MD trajectories of various proteins. The input to the training procedure is a data table in which each row gives the value of several (32) attributes, called predictors, of an H-bond and its local environment at a given time $t$ in a trajectory, as well as the measured stability of this H-bond over an interval of time $(t, t + \delta)$. The output is a function $\sigma$ of a subset of predictors that estimates the probability that an H-bond present in the conformation $c$ achieved by a protein will be present in any conformation achieved by this protein within an interval of time of duration $\delta$. The value of $\delta$ defines the timescale of the prediction.

MD simulation trajectories provide huge amount of data yielding training data tables made of several hundred thousand, or more, rows. To build regression trees from such tables we propose methods that run in $O(ab \log a)$ time, where $a$ is the number of rows and $b$ is the number of predictors. Tests demonstrate that the models trained with these methods can predict H-bond stability roughly 20% better than
models based on H-bond energy alone. The models can also accurately identify a large fraction of the least stable H-bonds in a given conformation. In most tests, about 80% of the 10% H-bonds predicted as the least stable are actually among the 10% truly least stable.

4.2 Problem Statement

Let \( c \) be the conformation of a given protein \( P \) at some time \( t_0 \) and \( h \) be an H-bond present in \( c \). Let \( M(c) \) be the set of all physically possible trajectories of \( P \) passing through \( c \) and \( \pi \) be the probability distribution over this set. We define the stability of \( h \) in \( c \) over the time interval \( \delta \) by:

\[
\bar{\sigma} : (H, c, \delta) \rightarrow [0, 1], \quad \bar{\sigma}(H, c, \delta) = \sum_{q \in M(c)} \left[ \frac{1}{\delta} \int_0^\delta I(q, H, t) dt \right] \pi(q) \tag{4.1}
\]

where \( I(q, H, t) \) is a Boolean function that takes value 1 if \( H \) is present in the conformation \( q(t) \) at time \( t \) along trajectory \( q \), and 0 otherwise. The value \( \bar{\sigma}(H, c, \delta) \) can be interpreted as the probability that \( H \) will be present in the conformation of \( P \) at any specified time \( t \in (0, t_0 + \delta) \), given that \( P \) is at conformation \( c \) at time 0.

Our goal is to design methods for generating good approximations \( \sigma \) of \( \bar{\sigma} \). We also want these approximations to be protein-independent, i.e., the argument \( c \) may be a conformation of any protein.

4.3 Methods

4.3.1 General Approach

We use machine learning methods to infer \( \sigma \) from a training set \( Q \) of MD simulation trajectories generated with various proteins. Each trajectory \( q \in Q \) is a discrete sequence of conformations of a specified protein. These conformations are reached at times \( t_i = i \times \mu, i = 0, 1, 2, \ldots \), called ticks, where \( \mu \) is typically on the order of
the picoseconds.\textsuperscript{3} We detect H-bonds which are present in every conformation $q(t_i)$ using criteria given in [74] (see Figure 3.4). Note that an H-bond in a given protein is uniquely identified (across different conformations) by its donor, acceptor, and hydrogen atoms. So, we call the presence of a specific H-bond $H$ in a conformation $q(t_i)$ an occurrence of $H$ in $q(t_i)$.

For each occurrence of an H-bond $H$ in $q(t_i)$ we compute a fixed list of predictors, some numerical, others categorical. Some are time-invariant, like the types of the donor and acceptor atoms and the number of residues along the main-chain between the donor and acceptor atoms. Others are time-dependent. Among them, some describe the geometry of $H$ in $q(t_i)$, e.g., the distance between the hydrogen and the donor atoms and the angle made by the donor, hydrogen, and acceptor atoms. Others describe the local environment of $H$ in $q(t_i)$, e.g., the number of other H-bonds within 5Å of the mid-point of the hydrogen atom and the acceptor of $H$. The complete list of predictors used in our work is given in Appendix A. In total, it contains 32 predictors.

We train $\sigma$ as a function of these predictors. The predictor list defines a predictor space $\Sigma$ and every H-bond occurrence maps to a point in $\Sigma$. As some predictors vary over time, two occurrences of the same H-bond at two different ticks usually map to two distinct points. Given the input set $Q$ of trajectories, we build a data table in which each row corresponds to an occurrence $h$ of an H-bond present in a conformation $q(t_i)$ contained in $Q$. So, many rows may correspond to the same H-bond at different ticks. In our experiments, a typical data table contains several hundred thousand rows (see Section 4.4.1). Each column, except the last one, corresponds to a predictor $p$ and the entry $(h, p)$ of the table is the value of $p$ for $h$. The entry in the last column is the measured stability $y$ of the H-bond occurrence in conformation $q(t_i)$. More precisely, let $H$ be the H-bond of which $h$ is an occurrence. In addition, let $l = \delta/\mu$, where $\delta$ is the duration over which we wish to predict the stability of $h$ (see Section 4.2), and let $m \leq l$ be the number of ticks $t_k$, $k = i+1, i+2, ..., i+l$, such that $H$ is present in $q(t_k)$. The measured stability $y$ of $h$ is the ratio $m/l$. Figure 4.1 plots a (typical) histogram of the measured stability of all H-bond occurrences in

\textsuperscript{3}MD simulation trajectories are computed by integrating the equations of motion with a time step on the order of the femtoseconds ($10^{-15}$s). However, to reduce the amount of stored data, they are usually sub-sampled at a time step on the order of the picoseconds ($10^{-12}$s).
one protein trajectory. This histogram indicates that H-bond occurrences tend to be quite stable: over 25% have measured stability 1, about 50% have measured stability higher than 0.8, and only 15% have measured stability less than 0.3.

We build $\sigma$ as a binary regression tree [12]. This well-studied machine learning approach has been one of the most successful in practice. Regression trees are often simple to interpret. Not only may this simplicity eventually lead to pertinent insights to better understand H-bond stability; it also allows us to perform many experiments, compare the generated trees, and analyze the relative importance of the predictors. Furthermore, the method can work with both categorical and numerical predictors in a unified way, as shown in Section 4.3.2.

Each non-leaf node $N$ in a regression tree is a Boolean test, called a split. Each split on a numeric predictor $p$ divides the predictor space $\Sigma$ into two half-spaces separated by a hyper-plane perpendicular to the coordinate axis representing $p$. Each arc outgoing from $N$ corresponds to one of these half-spaces. So, each node $N$ of the tree determines a region of $\Sigma$ which is obtained by intersecting all the half-spaces
associated with the arcs connecting the root of the tree to \( N \). We say that an H-bond occurrence falls into a node \( N \) if it is contained in this region. The predicted stability value stored at a leaf node \( L \) is the average of the measured stability values computed for all the H-bond occurrences in the training data table that fall into \( L \). We expect this averaging, which is done over many pieces of trajectories, to approximate well the averaging defined in Equation 4.1. To avoid over-fitting the input data, only a relatively small subset of predictors (selected by the training algorithm, as described in Section 4.3.2-4.3.4) is eventually used in a regression tree.

Once a regression tree has been generated, it is used as follows. Given an H-bond \( H \) in an arbitrary conformation \( c \) of an arbitrary protein, the leaf node \( L \) of the tree into which \( H \) falls is identified by calculating the values of the necessary predictors for \( H \) in \( c \). The predicted stability value stored at \( L \) is returned. (Note that by construction of the tree, any H-bond \( H \) falls into one and only one leaf node.)

### 4.3.2 Basic Tree-Construction Algorithm

We construct a model \( \sigma \) as a binary regression tree using the CART (Classification And Regression Tree) method [12]. The tree is generated recursively in a top-down fashion, i.e., starting from the root. When a new node \( N \) is created, it is inserted as a leaf of the tree if a predefined recursion depth has been reached or if the number of H-bond occurrences (from the training data table) falling into \( N \) is smaller than a predefined threshold. Otherwise, \( N \) is added as an intermediate node, its split is computed, and its left and right children \( L \) and \( R \) are created. A split \( s \) is defined by a pair \((p, r)\), where \( p \) is the split predictor and \( r \) is the split value. If \( p \) is a numerical predictor, then \( r \) is a threshold on \( p \), and \( s \triangleq p < r \). If \( p \) is a categorical predictor, then \( r \) is a subset of categories, and \( s \triangleq p \in r \). We define the score \( w(p, r) \) of split \( s = (p, r) \) at a node \( N \) as the reduction of variance in measured stability that results from \( s \). More formally:

\[
    w(p, r) = \text{Var}(Y_N) - \left[ \frac{n_L}{n} \text{Var}(Y_L) + (1 - \frac{n_L}{n}) \text{Var}(Y_R) \right]
\]

(4.2)

where: \( Y_N \) is the distribution of the measured stability of the H-bond occurrences in
the training data table falling into $N$; $Y_L$ and $Y_R$ are the distributions of the measured stability of the H-bond occurrences falling into $L$ and $R$, respectively, when split $s$ is applied; $Var(Y)$ is the variance of distribution $Y$; $n$ is the number of H-bond occurrences falling into $N$; $n_L$ is the number of H-bond occurrences falling into $L$ when split $s$ is applied.

The algorithm chooses the split — both the predictor and the split value — that has the largest score. The computation of the split value for each predictor is done as follows (where we denote by $H_N$ the subset of H-bond occurrences in the training data table that fall into $N$):

1. For a numerical predictor, the values of this predictor in $H_N$ are sorted in ascending order. All midpoints between two consecutive values are used as candidate split values. The one with the best split score is used as the split value. This value is clearly optimal.

2. For a categorical predictor, for every possible value $v$ of this predictor in $H_N$, we first compute the mean $\bar{Y}_N(v)$ of the measured stability of all the H-bond occurrences in $H_N$ where the predictor has value $v$. We then sort the possible values of the predictor into a list $(v_1, v_2, \ldots, v_K)$ ordered by $\bar{Y}_N(v_i)$. All the $K-1$ splits that divide this list into two contiguous sub-lists — e.g., $(v_1, v_2, \ldots, v_j)$ and $(v_{j+1}, v_{j+2}, \ldots, v_K)$ — are considered. The one with the best score is selected. Statement 8.16 in [12] proves that no other split can give a better score.

Since the number of values of a numerical predictor in $H_N$ may often be quite large, it is worth noticing that there is an efficient procedure for computing split scores. Consider two consecutive candidate split values $s_i$ and $s_{i+1}$ of a numerical predictor. Assume that we have computed the split score for $s_i$ and that we now want to compute the score for $s_{i+1}$. We can easily identify the H-bond samples that are shifting from $L$ to $R$. Then we can update $Var(Y_L)$ and $Var(Y_R)$ by only considering these samples, in time linear in their number. As a result we can compute the scores of all the candidate split values in time linear in the number of values of the considered numerical predictor in $H_N$. For a categorical predictor, the computation
of the scores of all the candidate split values is also linear in the number of categorical values.

At each layer of the tree the total number of samples does not exceed the number of rows in the training table. So, building each layer takes linear time in the table size. Since we limit the depth of a regression tree by a relatively small constant (see Section 4.3.4), the complexity of the tree construction algorithm is dominated by the initial sorting of the table rows for each predictor. So, a tree is built in $O(ab \log a)$ time, where $a$ is the number of rows in the training data table and $b$ is the number of predictors. This makes it possible to process tables with dozens of attributes and several hundred thousand rows using an off-the-shelf computer.

### 4.3.3 Violation of IID Property

One important issue to deal with is the violation of the IID (independent, identically distributed) property in the training data. The IID property would require that H-bond samples follow a certain fixed probability distribution, and that each line of a data table input to the learning algorithm is sampled according to this distribution, independent of the other lines. The satisfaction of this property is critical for the inferred stability model $\sigma$ to predict reliably the stability of H-bonds in new protein conformations. However, it is likely to be violated, mainly because several H-bond samples in a data table correspond to the same H-bond. More specifically, two samples of the same H-bond along the same trajectory are more likely to be similar (or even the same, in the case of time-independent predictors) along several dimensions of the predictor space $\Sigma$ than two samples of distinct H-bonds, especially if these bonds belong to different proteins. This may result into correlations between predictor values and measured stability that are bond-specific and thus do not extend to other bonds. To illustrate the point, we computed the mean measured stability of all samples of the same H-bond in an MD simulation trajectory. Figure 4.2 plots the (typical) histogram of the mean measured stability of all the H-bonds in a selected MD simulation trajectory. The figure shows that distinct H-bonds can have very different mean measured stability. It also shows that many H-bonds are unstable.
These bonds contribute few bond occurrences in the training data table, which leads the histograms in Figures 4.1 and 4.2 to have “inverse” shapes. While Figure 4.1 indicates that most H-bond occurrences are quite stable, Figure 4.2 indicates that many H-bonds are unstable.

To address this issue, we apply a two-step split calculation procedure [104]. The training dataset of MD simulation trajectories is first divided at random into two subsets $T_1$ and $T_2$. The split predictor $p$ and the split value $r$ at a node $N$ are computed separately, using one of these two subsets:

1. The best split value $r^*_p$ is computed for each predictor $p$ using $T_1$:
   $$r^*_p = \argmax_r \{ w_1(p, r) \},$$
   where $w_1(p, r)$ denotes the score of split $(p, r)$ on the dataset $T_1$.

2. The best split predictor $p^*$ is computed using $T_2$ with the best split values computed at the previous step:
   $$p^* = \argmax_p \{ w_2(p, r^*_p) \},$$
   where $w_2(p, r^*_p)$ denotes the score of split $(p, r^*_p)$ on the dataset $T_2$.

3. The selected split is $(p^*, r^*_p)$. 
CHAPTER 4. LEARNING HYDROGEN BOND STABILITY

Assume that the best split value computed in the first step is obtained for some predictor \( p' \). If this best value results from a bond-specific correlation between \( p' \) and measured stability in \( T_1 \), then this correlation is unlikely to happen again in \( T_2 \), since \( T_1 \) and \( T_2 \) have been separated at random. So, in the second step, predictor \( p' \) is likely to have a small score \( w_2(p', r^*_{p'}) \) and thus not to be selected as the split predictor.

4.3.4 Tree Pruning

To prevent model overfitting, we limit the size of a regression tree by bounding its maximal depth by a relatively small constant (5 in most of our experiments). We also define a minimal number of H-bond occurrences that must fall into a node for this node to be split. However, it is usually better to set these thresholds rather liberally and later prune the obtained tree using an adaptive algorithm, as described below.

We initially set aside a fraction \( T_3 \) of the training data table that has no overlap with the subsets \( T_1 \) and \( T_2 \) used in Section 4.3.3. Once a tree has been constructed using \( T_1 \) and \( T_2 \), pruning is an iterative process. At each step, one non-leaf node \( N \) whose split has minimal score (on \( T_1 \)) becomes a leaf node by removing the sub-tree rooted at \( N \). This process continues until the pruned tree only contains the root node. It creates a sequence of trees with decreasing numbers of nodes. We then estimate the prediction error of each tree as the mean square error of the predictions made by this tree on \( T_3 \). The tree with the smallest error is selected.

4.4 Results

4.4.1 Experimental Setup

MD Trajectories

In the experiments reported below, we used 6 MD simulation trajectories picked from different sources. We call these trajectories \( 1c9oA, 1e85A, 1g9oA_1, \) and \( 1g9oA_2 \) from [58], \( \text{complex} \) from [47], and \( 1eia \) generated by ourselves. In all of them the time
CHAPTER 4. LEARNING HYDROGEN BOND STABILITY

interval $\mu$ between two successive ticks is 1ps. Table 4.1 indicates the protein simulated in each trajectory, its number of residues, the force field used by the simulator, and the duration of the trajectory. Each trajectory starts from a folded conformation resolved by X-ray crystallography.

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>Protein</th>
<th>Num. Res.</th>
<th>Force field</th>
<th>Duration</th>
<th>H-bonds</th>
<th>occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1c9oA</td>
<td>Cold shock protein</td>
<td>66</td>
<td>ENCAD [66] with F3C explicit water model</td>
<td>10ns</td>
<td>263</td>
<td>363463</td>
</tr>
<tr>
<td>1e85A</td>
<td>Cytochrome C</td>
<td>124</td>
<td>Same as above</td>
<td>10ns</td>
<td>525</td>
<td>1253879</td>
</tr>
<tr>
<td>1g9oA_1</td>
<td>PDZ1 domain of human Na(+)/H(+) exchanger regulatory factor</td>
<td>91</td>
<td>Same as above</td>
<td>10ns</td>
<td>374</td>
<td>558761</td>
</tr>
<tr>
<td>1g9oA_2</td>
<td>Same as above</td>
<td>91</td>
<td>Same as above</td>
<td>10ns</td>
<td>397</td>
<td>544491</td>
</tr>
<tr>
<td>1eia</td>
<td>EIAV capsid protein P26</td>
<td>207</td>
<td>Amber 2003 with SPC/E water model</td>
<td>2ns</td>
<td>757</td>
<td>379573</td>
</tr>
<tr>
<td>complex</td>
<td>Efb-C/C3d complex formed by the C3d domain of human complement component C3 and one of its bacterial inhibitors</td>
<td>362</td>
<td>Amber 2003 with implicit solvent using the General Born solvation method [98]</td>
<td>2ns</td>
<td>1825</td>
<td>348943</td>
</tr>
</tbody>
</table>

Table 4.1: Characteristics of the MD simulation trajectories used to create the 6 datasets

Trajectories obtained with different proteins allow us to test if a model trained with one protein can predict H-bond stability in another protein. Similarly, trajectories generated with different force fields allow us to test a model $\sigma$ trained with one force field can predict H-bond stability in trajectories generated with another force field. We did additional experiments with a larger set of trajectories, but the results were similar to those reported below.
Data Tables

From each of the 6 trajectories we derived a separate data table in which the rows represent the detected H-bond occurrences and the columns give the values of the predictors and H-bond measured stability. The last 2 columns of Table 4.1 lists the number of distinct H-bonds detected in each trajectory and the total number of H-bond occurrences extracted. Since most H-bonds are not present at all ticks, the number of H-bond occurrences is smaller than the number of distinct H-bonds multiplied by the number of ticks. So, for example, the data table generated from trajectory 1e85A consists of 1,253,879 rows, 32 columns for predictors, and one column for measured stability. Note that complex was generated for a complex of two molecules. All H-bonds occurring in this complex are taken into account in the corresponding data table.

The measured stability $y$ of an H-bond $H$ in $q(t_i)$ is computed as described in Section 4.3.1, as the ratio of the number of ticks where the bond is present in the time interval $[t_i, t_i + l \times \mu]$ in trajectory $q$ divided by the total number of ticks $l$ in this interval.

The values of the time-varying predictors are subject to thermal noise. Since a model $\sigma$ will in general be used to predict H-bond stability in a protein conformation sampler using a linkage model ignoring thermal noise (such as KGS described in Chapter 3), we chose to average the values of these predictors over $l'$ ticks to remove thermal noise. More precisely, let $h$ be an H-bond occurrence in $q(t_i)$. The value of a predictor stored in the row of the data table corresponding to $h$ is the average value of this predictor in $q(t_{i-l'+1}), q(t_{i-l'+2}), \ldots, q(t_i)$, where $t_{i-l'+k} = t_i - (l' - k) \times \mu$.

The values of $l$ and $l'$ are chosen according to different criteria. The choice of $l$ is based on two considerations. It must be large enough for the measured stability $m/l$ to be statistically meaningful. It must also correspond to the timescale over which one wants to predict H-bond stability. The choice of $l'$ should be just enough to remove thermal noise from the predictor values. Experiment 5 in Section 4.4.6 shows that $l' = 50$ is near optimal. We also chose $l = 50$ in most of the tests reported below, as this value both provides a meaningful ratio $m/l$ and corresponds to an interesting prediction timescale (50ps). In Experiment 5, we will compare the performance of
models generated with several values of \( l \).

**Performance measures**

The performance of a regression model can be measured by the *Root Mean Square Error* (RMSE) of the predictions on a test dataset. For a data table

\[
T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\},
\]

where each \( x_i, i = 1, \ldots, n \), denotes a vector of predictor values for an H-bond occurrence and \( y_i \) is the measured stability of the H-bond, and a model \( \sigma \), the RMSE is defined by:

\[
RMSE(\sigma, T) = \sqrt{\frac{1}{n} \sum_{i} (y_i - \sigma(x_i))^2}
\] (4.3)

As RMSE depends not only on the accuracy of \( \sigma \), but also on the table \( T \), some normalization is necessary in order to compare results on different tables. So, in our tests we compute the decrease of RMSE relative to a base model \( \sigma_0 \). The *Relative Base Error Decrease* (or RBED) is then defined by:

\[
RBED(\sigma, \sigma_0, T) = \frac{RMSE(\sigma_0, T) - RMSE(\sigma, T)}{RMSE(\sigma, T)} \times 100\%
\] (4.4)

In most cases, \( \sigma_0 \) is simply defined by \( \sigma_0(x) = \frac{1}{n} \sum_i y_i \), i.e., the average measured stability of all H-bond occurrences in the dataset. In other cases, \( \sigma_0 \) is a model based on the H-bond energy.

**4.4.2 Experiment 1: Training on one data table, predicting on another**

Here we trained 10 models on each one of the 6 data tables (i.e., 60 models total). We tested every model separately on each of the other 5 data tables. For each model, the corresponding training data table was partitioned into three tables \( T_1, T_2, \) and \( T_3 \), as described in Sections 4.3.3 and 4.3.4: 60% of the data went to \( T_1 \), 20% to \( T_2 \), and
In addition, to achieve greater independence between the three tables, no two tables contain occurrences of the same H-bond. The 10 models trained with the same data table were generated with different partitions generated at random, but still satisfying the previous ratios and condition. In all cases the maximal depth of a tree was set to 5.

Table 4.2 gives the mean value of RBED for each pair of data tables. More specifically, the chart of Figure 4.3 shows the distribution of the RBED values for the 10 models trained with 1c9oA on each data table (so, each of the 10 models contributes 6 points in the chart). These results show that, on average, a model trained with one trajectory predicts H-bond stability in another trajectory reasonably well, even if the two trajectories were generated using different energy functions. However, we note that the variance of the 10 RBED values for each test data table is rather large.

We also note that the mean RBED values are generally lower for models tested on complex, while the mean RBED values for models trained on complex and tested on other tables (last column of Table 4.2) are comparable to other values. Recall that the trajectory complex was generated for a complex made of a protein and a ligand, while every other trajectory was generated for a single protein. So, it is likely that complex contains H-bonds in situations that did not occur in any of the other trajectories. Figures 4.4 and 4.5 show trees trained with the data table 1c9oA and complex. We will comment on generated trees in Section 4.5.
Figure 4.3: RBED values for the 10 models generated with 1c9oA (Experiment 1). The horizontal line shows that the average of all 50 RBED values.

Figure 4.4: Regression tree trained with 1c9oA (Experiment 1).
4.4.3 Experiment 2: Training on data from multiple trajectories

Here, we trained models on data tables obtained by mixing subsets of 5 data tables and we tested these models on the remaining data table. For each combination of 5 data tables, we trained 10 models by mixing different fractions of the 5 data tables. For each model, the mixed data table was partitioned into three tables $T_1$, $T_2$, and $T_3$ as in Experiment 1: 60% of the data went to $T_1$, 20% to $T_2$, and 20% to $T_3$. Again, no two tables contain occurrences of the same H-bond. Furthermore, we trained 4 groups of models varying in the tree’s maximal depth (5 or 15) and in the fraction of H-bond occurrences taken from each data table (10% or 50%). So, in total, 240 models were generated in this experiment.

Table 4.3 shows the mean RBED value for each combination of data tables and each group of models. In columns 3 through 8 we indicate the data table used for testing the models trained on a combination of the 5 other data tables. Figure 6 shows the distribution of the RBED values for the models built with the settings of in the first row of Table 4.3 (i.e., maximal depth of 5 and 10% from each data table).

We note that the RBED values are significantly higher than in Experiment 1, meaning that models trained using data from several trajectories are more accurate than models trained using data from a single trajectory. This is not surprising, since a
training data table generated from several trajectories is likely to provide richer data about H-bond stability than a table derived from a single trajectory. Furthermore, the variance of RBED values is now very small, meaning that the training process yields models that are stable in performance. Finally, like in Experiment 1, the RBED values are again lower for models tested on complex. All these results suggest that we should try to train models with a larger set of trajectories. We actually did some experiments using a few additional trajectories, but with no noticeable improvement. Most likely these trajectories did not contain enough H-bonds in situations that did not already occur in the trajectories of Table 4.2.

Another observation is that both deeper trees and larger data fractions tend to
CHAPTER 4. LEARNING HYDROGEN BOND STABILITY

Figure 4.7: Top 3 layers of a tree trained with combination of all tables, except 1c9oA (Experiment 2). The actual tree has 5 layers containing 55 nodes.

Figure 4.8: Top 3 layers of a tree trained with combination of all tables, except 1e85A (Experiment 2). The actual tree has 5 complete layers containing 63 nodes.

4.4.4 Experiment 3: Comparison with FIRST-energy model

Here, the models are the same as those generated in Experiment 2 in the first row of Table 4.3 (maximal depth of 5 and 10% from each data table). But we now compare them to a regression tree $\sigma_0$ built from the same training data using FIRST_energy.
as the only predictor (predictor 32 in Appendix A). FIRST\textsubscript{energy} is the value of the function used in FIRST [85] to evaluate the energy of an H-bond occurrence; it is a slightly modified version of the Mayo energy [31]. We compute RBED values as defined in Section 4.4.1, where $\sigma_0$ is the simple regression tree.

Table 4.4 shows the mean RBED values. Tests on all 6 data tables show that the more complex models are significantly more accurate that the model based on FIRST\textsubscript{energy} only. Overall, these results confirm that the stability of an H-bond occurrence depends not only on its energy, but also on other parameters. See Section 4.5 for more comments.

![Table 4.4: Mean RBED values computed in Experiment 3.](image)

### 4.4.5 Experiment 4: Identification of least stable H-bonds

Most H-bond occurrences tend to be stable. So, accurately identifying the weakest ones is important if one wishes to predict the possible deformation of a protein [100]. Here, we measure how well the models generated in Experiment 2 (again, in the first row of Table 4.3) identify the least stable H-bonds occurrences in the test data table. In each test table $T$, we first identify the subset $S$ of the 10% least stable H-bond occurrences (i.e., the H-bond occurrences with the smallest measured stability). Using a regression tree $\sigma$ trained with a combination of data from the 5 other tables, we then sort the H-bond occurrences in $T$ in ascending order of predicted stability and we compute the fraction $w \in [0,1]$ of $S$ that is contained in the first $100 \times u\%$ occurrences in this sorted list, for successive values of $u \in [0,1]$. We call the function $w(u)$ the identification curve of the least stable H-bonds for $\sigma$.

Figure 4.9 plots identification curves for each of the 6 test tables. Each plot consists of three curves: the red curve is the ideal identification curve (the one that would be obtained with a model that perfectly predict the 10% least stable H-bonds), the
Figure 4.9: Identification curves of the least stable bonds for different models in Experiment 4.
blue curve is obtained with one (randomly picked) regression tree computed in Experiment 2, and the green curve is obtained by sorting H-bond occurrences in decreasing values of FIRST_energy. One can see that the models computed in Experiment 2 perform well in general. For models tested on data tables other than complex, about 80% of the 10% H-bond occurrences predicted as the least stable are actually among the 10% truly least stable. However, several curves show a rather long tail of poorly ranked unstable bonds. For example, the set of the 50% least stable bonds predicted by the model tested on 1eia still misses about 5% of the truly least stable bonds. Not surprisingly, the results for complex are much less satisfactory. The regression models generated in Experiment 2 perform consistently better than the FIRST_energy-only models, but for 1eia the difference is small.

4.4.6 Experiment 5: Models for different averaging and prediction windows

Here the testing setup is the same as for Experiment 2 (first row of Table 4.3), but we let the numbers of ticks \( l \) and \( l' \) vary. Recall from Section 4.4.1 that \( l \) is the number of ticks over which bond stability is measured, while \( l' \) is the number of ticks over which predictor values are averaged.

First, we set \( l \) to 50 ticks (50ps), and we built and tested models for predictor averaging windows of successive lengths \( l' = 2, 5, 10, 20, 50, 100, 200, \) and 500 ticks. So, in total we built 400 models. Table 6 shows the mean RBED value for each test table and each value of \( l' \). Figure 4.10 shows the distribution of the RBED values for the models tested on 1c9oA (10 models for each value of \( l' \)). An averaging window length of \( l' = 50 \) ticks gives the best results. Shorter lengths fail to eliminate thermal noise in predictor values, but longer windows tend to smooth out important changes in predictor values.

Next, we set \( l' \) to 50 ticks (50ps) and we built and tested models for prediction windows of successive lengths \( l = 10, 20, 50, 100, 200, \) and 500 ticks (so, here we built 300 models). Table 4.6 shows the mean RBED value for each test table and each value of \( l \). Figure 4.11 shows the distribution of the RBED values for the
models tested on 1c9oA. Again, a 50-tick prediction window gives the best results. With shorter windows measured stability is less reliable. But longer windows lead to making predictions too far beyond an observed H-bond occurrence; the pertinence of the predictor values decreases over time. Nevertheless, the RBED values are still good for $l = 200$ ticks (200ps), showing that models can be trained to predict H-bond stability over rather long timescales. For $l = 500$ ticks, the mean RBED values start declining more significantly, while the plot of Figure 4.11 indicates that the variance of the RBED values also increases sharply. This is not surprising since a window of 500 ticks represents a large fraction of each of the trajectories.

Table 4.5: Mean RBED values for different lengths $l'$ of the predictor averaging window in Experiment 5.

<table>
<thead>
<tr>
<th>$l'$</th>
<th>1c9oA</th>
<th>1e85A</th>
<th>1eia</th>
<th>1g9oA_1</th>
<th>1g9oA_2</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>28.15</td>
<td>39.55</td>
<td>22.71</td>
<td>36.39</td>
<td>24.72</td>
<td>7.18</td>
</tr>
<tr>
<td>5</td>
<td>38.71</td>
<td>48.65</td>
<td>31.08</td>
<td>43.75</td>
<td>38.05</td>
<td>22.03</td>
</tr>
<tr>
<td>10</td>
<td>42.78</td>
<td>54.18</td>
<td>36.31</td>
<td>46.50</td>
<td>42.37</td>
<td>29.53</td>
</tr>
<tr>
<td>20</td>
<td>45.58</td>
<td>57.43</td>
<td>40.48</td>
<td>49.40</td>
<td>44.66</td>
<td>34.78</td>
</tr>
<tr>
<td>50</td>
<td>47.13</td>
<td>59.72</td>
<td>43.88</td>
<td>51.48</td>
<td>45.76</td>
<td>39.05</td>
</tr>
<tr>
<td>100</td>
<td>46.58</td>
<td>59.44</td>
<td>43.81</td>
<td>50.96</td>
<td>45.52</td>
<td>38.54</td>
</tr>
<tr>
<td>200</td>
<td>45.18</td>
<td>58.91</td>
<td>43.21</td>
<td>49.95</td>
<td>43.38</td>
<td>36.20</td>
</tr>
<tr>
<td>500</td>
<td>41.40</td>
<td>56.25</td>
<td>41.81</td>
<td>46.94</td>
<td>37.99</td>
<td>30.69</td>
</tr>
</tbody>
</table>

Table 4.6: Mean RBED values for different lengths $l$ of the prediction window in Experiment 5.

<table>
<thead>
<tr>
<th>$l$</th>
<th>1c9oA</th>
<th>1e85A</th>
<th>1eia</th>
<th>1g9oA_1</th>
<th>1g9oA_2</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>27.13</td>
<td>34.60</td>
<td>21.45</td>
<td>30.54</td>
<td>26.17</td>
<td>17.61</td>
</tr>
<tr>
<td>5</td>
<td>36.98</td>
<td>46.65</td>
<td>30.49</td>
<td>41.34</td>
<td>35.99</td>
<td>26.58</td>
</tr>
<tr>
<td>10</td>
<td>42.49</td>
<td>53.16</td>
<td>36.17</td>
<td>47.02</td>
<td>41.50</td>
<td>32.23</td>
</tr>
<tr>
<td>20</td>
<td>45.75</td>
<td>57.28</td>
<td>40.55</td>
<td>50.57</td>
<td>44.91</td>
<td>36.39</td>
</tr>
<tr>
<td>50</td>
<td>47.09</td>
<td>59.78</td>
<td>43.87</td>
<td>51.50</td>
<td>45.79</td>
<td>38.81</td>
</tr>
<tr>
<td>100</td>
<td>45.89</td>
<td>59.29</td>
<td>43.47</td>
<td>49.17</td>
<td>44.82</td>
<td>38.24</td>
</tr>
<tr>
<td>200</td>
<td>43.32</td>
<td>56.76</td>
<td>42.17</td>
<td>45.38</td>
<td>42.55</td>
<td>35.24</td>
</tr>
<tr>
<td>500</td>
<td>38.13</td>
<td>49.77</td>
<td>37.08</td>
<td>41.43</td>
<td>38.32</td>
<td>31.30</td>
</tr>
</tbody>
</table>
Figure 4.10: Distribution of RBED values for models tested on 1c9oA for different lengths $l'$ of the predictor averaging window (Experiment 5).

Figure 4.11: Distribution of RBED values for models tested on 1c9oA for different lengths $l$ of the prediction window (Experiment 5).
4.5 Discussion

In all our regression trees the root split was done with predictor Dist_H_A (the distance between the H and acceptor atoms), which therefore appear as the single most discriminative attribute to predict H-bond stability. The mean measured stability of the two children of the root node differs by a ratio ranging from 1.5 to 2 depending on the specific tree. The importance of the distance between the H and the acceptor is consistent with previous findings. Levitt [68] found that most stable H-bonds have Dist_H_A less than 2.07 Å. Jeffrey and Saenger [57] also suggested that Dist_H_A is a key attribute affecting H-bond stability, with a value less than 2.2 Å for moderate to strong H-bonds. Consistent with these previous findings, the split values of the deepest Dist_H_A nodes in our regression trees vary slightly around 2.1 Å. This distance was observed in [68] to sometimes fluctuate by up to 3 Å in stable H-bonds, due to high-frequency atomic vibration. This observation supports our decision to average predictor values over windows of \( t' \) ticks, as it would be easy to incorrectly predict the stability of an H-bond from the value of Dist_H_A at a single tick.

Predictor FIRST_energy, a modified Mayo potential [31] implemented in FIRST (a protein rigidity analysis software) [85], is often used in splits close to the root. This is not surprising since it is a function of several other pertinent predictors: Dist_H_A, Angle_D_H_A, Angle_H_A_AA, and Hybrid_state (hybridization state of the bond). Some other distance-based predictors (Dist_D_AA, Dist_D_A, Dist_H_D), angle-based predictors and Ch_type predictor appear often in regression trees, but closer to the leaf nodes. They nevertheless play a significant role in predicting H-bond stability. For example, as shown in Figures 4.7 and 4.8, if Angle_H_A_AA is at least 105°, an H-bond has very high stability (about 0.96); otherwise, the stability drops to 0.71. The preference for larger angle matches well with the well-known linearity of H-bonds [26, 57].

Other predictors that are used in splits only occasionally have a less obvious role. A number of predictors (such as, Atom_type_A, Atom_type_AA, Resi_type_H, Rgd_type) never appeared in our trees. Either they have no or very small impact on H-bond stability, or they are highly correlated with other more discriminative
In order to get a more quantitative measure of the relative impact of the predictors on H-bond stability, let us define the importance of a predictor $p$ in a regression tree by:

$$I(p) = \sum_{s \in N_p} \frac{w(s)}{n(s)}$$  \hspace{1cm} (4.5)

where $N_p$ is the set of nodes where the split is made using $p$, $w(s)$ is the score of the split $s$, and $n(s)$ is the number of H-bond occurrences falling into the node where split $s$ is made\textsuperscript{4}. We trained 10 models on data tables combining 10% of each the 1c9oA, 1e85A, 1eia, 1g9oA_1, 1g9oA_2 and complex data tables. Importance scores for each predictor were averaged over these models and then linearly scaled to adjust the score of the least important predictor (with non-zero average importance) equal to 1. The average importance of every predictor appearing in at least one model is shown in Figure 4.12. The figure confirms that distance-based and angle-based predictors, as well as FIRST\textsubscript{energy}, are the most important. It also shows that a number of other predictors — including Resi\textsubscript{name}_H, Resi\textsubscript{name}_A, and Range (difference in residue numbers of donor and acceptor) — have less, but still significant importance.

Overall, we observe that predictors that describe the local environment of an H-bond occurrence play a relatively small role in predicting its stability. In particular, we had expected that descriptors such as the 29\textsuperscript{th} (Num\_hb\_spaceNbr) and the 30\textsuperscript{th} (Num\_hb\_spaceRgdNbr), which count the number of other H-bonds located in the neighborhood of the analyzed H-bond, would have had more importance. However, this may reflect the fact that the MD simulation trajectories used in our tests are too short to contain enough information to infer the role of such predictors. Indeed, while transitions between meta-stable states are rare in those trajectories, predictors describing local environments may have greater influence on the stability of H-bonds that must break for such transitions to happen. So, longer trajectories may eventually be needed to better model H-bond stability.

\textsuperscript{4}This measure is not perfect because some predictors are correlated. For instance, the value of FIRST\textsubscript{energy} is correlated with other predictors. A better, but more complicated, measure of predictor importance uses ensembles of trees of a special form [104].
Figure 4.12: Predictor importance scores.
4.6 Conclusion

In this chapter, we described machine learning methods to train regression trees modeling H-bond stability in a protein. The training and test data are in the form of tables whose rows describe H-bond occurrences at successive times along MD simulation trajectories and columns give the values of various predictors. Each node in a regression tree is a Boolean test on a predictor. Each row (H-bond occurrence) in a data table determines a path in the tree from the root to a leaf node. A predicted stability is associated with each leaf node. The generated trees are relatively small and easily understandable. Trees can be built to predict H-bond stability over different time scales.

Test results demonstrate that trained models can predict H-bond stability quite well. In particular, we have shown that their performance is significantly better (roughly 20% better) than that of a model based on H-bond energy alone. We have also shown that they can accurately identify a large fraction of the least stable H-bonds in a given conformation. However, our results also suggest that better results could be obtained with a richer set of MD simulation trajectories. In particular, the trajectories used in our experiments might be too short to characterize the stability of H-bonds that break and form during a transition between meta-stable states.

We believe that the training methods could be improved in several ways:

1. To eliminate thermal noise, predictor values are averaged over time windows of 50 ticks, independent of the elapsed time between two ticks (see Section 4.4.5). It would be better to averaging predictor values before sub-sampling MD simulation trajectories (see the footnote in the first paragraph of Section 4.3.1). This would result in a much shorter averaging window, hence it would greatly reduce the risk of filtering out changes in predictor values that are important for H-bond stability. Unfortunately, in our trajectories we only had access to the data after sub-sampling.

2. More sophisticated learning techniques could be used. For example, instead of generating a single tree, we could generate an ensemble of trees, such as
Gradient Boosting Trees [40] or Random Forests [13]. A regression tree could also be enriched by using splits on linear combinations of predictors and by fitting linear regression models at the leaves.

3. We could use rigidity analysis methods such as those described in [100] to decompose a protein into rigid groups of atoms (based on distance constraints imposed by covalent and hydrogen bonds present in the current conformation). This would allow us to apply Bayesian techniques to align the predicted stability of individual H-bonds in the same rigid group. By doing so, we could better predict the collective behavior of related H-bonds and avoid solitary incorrect predictions.

4. Finally, the notion of stability itself could be refined, for example by distinguishing between the case where an H-bond frequently switches on and off during a prediction window and the case where it rarely switches.

Overall, we believe that considerable progress can still be made in learning more accurate and robust models of H-bond stability.
Chapter 5

Conclusion

In this dissertation, we proposed two kino-geometric conformation sampling methods, respectively for protein loops and for entire proteins, to explore their folded conformation spaces. Our loop sampler builds new conformations in fragments by focusing on the most restricting constraints first, either geometric or kinematic. Our protein conformation sampler, KGS, is the first to our knowledge to apply a Jacobian-based method to deform dozens of interdependent cycles simultaneously, and to demonstrate that this method is able to maintain cycle closure well in such a complicated kinematic structure. KGS also embeds a new diffusive strategy that expands the distribution of sampled conformations quickly away from the initial conformation and then makes the distribution increasingly denser. Experiments show that both sampling methods are able to efficiently generate conformations broadly and approximately evenly distributed over the spaces of interest. This efficiency of our samplers relies on several key algorithms: the grid method to check collisions among atoms, the constraint prioritization for the loop conformation sampler, and the Jacobian-based method and the diffusive strategy for KGS.

The results obtained with the two implemented samplers also provide evidence that, in general, the three assumptions listed in Section 1.4 are satisfied, i.e.:

1. The folded state of a protein is small enough to be represented by a reasonably small distribution of sampled conformations.
2. Kinematic and geometric constraints are sufficient to approximate the folded state of a protein (due to the fact that atoms are densely packed).

3. Kinematic and geometric constraints are sufficiently simple to be handled efficiently.

However, kino-geometric sampling is inherently limited to exploring the folded state of a protein. Roadmap and Markov model methods [22] could be used in addition to such samplers for studying kinetic and dynamic properties of protein motion. For instance, conformations computed by our samplers could be used as the start conformations of short MD simulations, and Markov models could be built from these simulations, like in [22, 95]. Kinetic and dynamic properties, e.g., mean transition times between pairs of sub-states, could then be extracted from these models using standard Markovian tools.
Appendix A

List of Predictors

Please refer to Figure 3.4 for atom identifiers. Type N stands for Numerical and Type C stands for Categorical. Some predictors involve rigid groups of atoms that are identified in the same way as in Section 3.3.4.
## APPENDIX A. LIST OF PREDICTORS

<table>
<thead>
<tr>
<th>#</th>
<th>Feature Name</th>
<th>Feature Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distance-related</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Dist,H,D</td>
<td>Distance between H and D</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>Dist,H,A</td>
<td>Distance between H and A</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>Dist,A,AA</td>
<td>Distance between A and AA</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>Dist,D,A</td>
<td>Distance between D and A</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>Dist,D,AA</td>
<td>Distance between D and AA</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>Dist,H,AA</td>
<td>Distance between H and AA</td>
<td>N</td>
</tr>
<tr>
<td><strong>Angle-related</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ang,D,H,A</td>
<td>Angle D-H-A</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>Ang,H,A,AA</td>
<td>Angle H-A-AA</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>Ang,D,A,AA</td>
<td>Angle D-A-AA</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>Ang,planar</td>
<td>Angle between the planes D-H-A and H-A-AA</td>
<td>N</td>
</tr>
<tr>
<td><strong>Atom</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Atom_type,D</td>
<td>D atom type (e.g., O, N, S, C)</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>Atom_type,A</td>
<td>A atom type (e.g., N, O, S)</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>Atom_type,AA</td>
<td>AA atom type (e.g, P, C, S)</td>
<td>C</td>
</tr>
<tr>
<td><strong>Residue</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Resi_name,H</td>
<td>D residue name (3 letter code)</td>
<td>C</td>
</tr>
<tr>
<td>15</td>
<td>Resi_name,A</td>
<td>A residue name (3 letter code)</td>
<td>C</td>
</tr>
<tr>
<td>16</td>
<td>Resi_type,H</td>
<td>D residue type. Nonpolar (Ala, Val, Leu, Ile, Trp, Met, Pro), Polar_acidic (Asp, Glu), Polar_uncharged (Gly, Ser, Thr, Cys, Tyr, Asn, Gln), Polar_basic (Lys, Arg, His)</td>
<td>C</td>
</tr>
<tr>
<td>17</td>
<td>Resi_type,A</td>
<td>A residue type</td>
<td>C</td>
</tr>
<tr>
<td>18</td>
<td>Resi_sch_size,H</td>
<td>D residue side-chain size, i.e., number of atoms in the side-chain</td>
<td>N</td>
</tr>
<tr>
<td>19</td>
<td>Resi_sch_size,A</td>
<td>A residue side-chain size</td>
<td>N</td>
</tr>
</tbody>
</table>

(To be continued on the next page)
### Bond structure type

<table>
<thead>
<tr>
<th>No.</th>
<th>Predictor</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Sec_type</td>
<td>Secondary structure of the H-bond. MA (H atom and A atom are in same helix, middle portion), MB (same strand, middle), EA (same helix, end), EB (same strand, end), AL (helix-loop), BL (helix-loop), DA (different helices), SL (same loop), DL (different loops). Don’t have DB (different strands) because it’s hard to know which strand pairs with which strand to form the sheet.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Ch_type</td>
<td>H and A are on main-chain (mch) or side-chain (sch): MM (mch-mch), MS (mch-sch), SS (sch-sch)</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Rgd_type</td>
<td>SR (H and A are in the same rigid body), DR (different rigid body)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Range</td>
<td>Difference in the residue number of D and A, i.e., $</td>
<td>ResId(D) - ResId(A)</td>
</tr>
<tr>
<td>24</td>
<td>Hybrid_state</td>
<td>Hybridization state (sp2-sp2, sp2-sp3, sp3-sp2, sp3-sp3)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Num_furcated_H</td>
<td>Number of H-bonds sharing the H atom</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Num_furcated_A</td>
<td>Number of H-bonds sharing the A atom</td>
<td></td>
</tr>
</tbody>
</table>

### Environment

<table>
<thead>
<tr>
<th>No.</th>
<th>Predictor</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Num_potential_As</td>
<td>Number of potential acceptors (N, O, or S) in 3Å of H (but not covalently bonded to it) besides the current A</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Num hb_seqNbr</td>
<td>Number of sequence-neighboring H-bonds, i.e., number of H-bonds of residues $\pm 2$ of $ResId(D)$ and $ResId(A)$</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Num hb_spaceNbr</td>
<td>Number of space-neighboring H-bonds, i.e., number of H-bonds within 5Å of the mid-point of this H-bond</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Num hb_spaceRgdNbr</td>
<td>Number of space-neighboring H-bonds in the same rigid group, i.e., number of Num hb_spaceNbr in the same rigid group as this H-bond</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Surface</td>
<td>Average surface percentage of the H and A atom</td>
<td></td>
</tr>
</tbody>
</table>

### Energy

<table>
<thead>
<tr>
<th>No.</th>
<th>Predictor</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>FIRST_energy</td>
<td>Modified Mayo potential implemented in FIRST [85]</td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


