Scaling Up Robotics-Inspired Conformational Sampling Algorithms

by

Jayvee R. Abella

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree Master of Science

Approved, Thesis Committee:

Lydia E. Kavraki, Chair
Noah Harding Professor of Computer Science and Bioengineering

Suay K. Nachleh
Professor of Computer Science

Devika Subramanian
Professor of Computer Science

Houston, Texas
November, 2016
Scaling Up Robotics-Inspired Conformational Sampling Algorithms

Jayvee R. Abella

ABSTRACT

The ability to efficiently sample a protein’s conformational space allows one to understand how a protein may interact with different partners. Algorithms from sampling-based robot motion planning have been used for conformational sampling of small-sized systems. These algorithms keep track of “coverage” in conformational space based on what has been sampled and aim to intelligently perturb the protein’s degrees of freedom to bias search in less densely explored areas of conformational space. However, these algorithms were not designed for large proteins or complexes. These algorithms depend heavily on defining useful perturbation strategies, which is a very difficult task for large proteins because such systems are typically more constrained and exhibit complex motions. Additionally, conformational sampling generally becomes a harder problem as the size of the considered system increases, so these algorithms need to take advantage of significant computational resources when needed.

This thesis describes SIMS 2.0, a new framework for conformational sampling built from prior work called the Structured Intuitive Move Selector (SIMS). We introduce an automated construction of perturbation strategies derived from B-factors, secondary structure, and rigidity analysis. We also introduce a new algorithm for conformational sampling that can take advantage of large-scale computational resources while still keeping the geometric reasoning that robotics-inspired algorithms excel at. This work pushes the limits of the size of systems that can be studied by robotics-inspired conformational sampling.
Acknowledgement

I would like to thank Dr. Lydia Kavraki and Dr. Mark Moll for their guidance and encouragement that I continue to receive since the day I started the program. I would also like to thank the members of the Kavraki Lab for helping me overcome the various challenges I faced throughout the course of this work. Finally, I would like to thank my family and friends who make my day to day possible.

This work is supported in part by NSF CCF 1423304, Rice University funds, and a training fellowship from the Gulf Coast Consortia on the Training Program in Biomedical Informatics, National Library of Medicine T15LM007093.
# Contents

1 Introduction  
1.1 Contributions .................................................. 4  
1.2 Organization .................................................... 5  

2 Background  
2.1 Energy Models and RMSD ........................................... 6  
2.2 Standard Methods: MD and MCMC ............................... 7  
2.3 Enhanced Sampling ................................................ 8  
  2.3.1 Coarse-grained Energy Models ............................... 8  
  2.3.2 Generalized Ensemble Methods ......................... 9  
  2.3.3 Markov State Models .................................... 10  
2.4 Geometric Reasoning for Conformational Sampling ............ 10  
2.5 Structured Intuitive Move Selector ............................... 12  
  2.5.1 Sampling a Conformation using Coverage Estimates ...... 13  
  2.5.2 Perturbing Conformations using the Schema ............... 14  
  2.5.3 Accepting/Rejecting Conformations ...................... 16  

3 Incorporating Flexibility Information  
3.1 Motivation .......................................................... 18  
3.2 Automatically constructing the projection using B-factors ...... 19
3.3 Automatically constructing the schema using B-factors, secondary structure, and rigidity analysis

4 Implementing a Global Coverage Grid

4.1 Motivation

4.2 Computing Coverage

4.3 Solution: Summaries generated from the Global Coverage Grid

5 Results

5.1 Proteins Used in Experiments

5.1.1 Cyanovirin-N

5.1.2 Calmodulin

5.1.3 Ribose-Binding Protein

5.1.4 GroEL Subunit

5.2 Improved scalability

5.3 Improved conformational space coverage

5.3.1 Nearest Neighbor Distances

5.3.2 Expansiveness

5.3.3 Isolating each improvement

6 Conclusion and Future Directions
List of Figures

2.1 Pictorial representation of the sampling process in SIMS. SIMS cycles between three steps. 1) Sampling a conformation based on coverage estimates. 2) Perturbing the structure using the schema. 3) Accepting the new conformation if it is below a user defined energy threshold. The projection is needed for computing coverage, which is used in selecting conformations for Step 1. The schema defines SIMS’ perturbation strategy for Step 2. .......................... 13

2.2 How SIMS computes coverage using the projection. ..................... 14

2.3 Types of moves in SIMS. The randomOne and randomAll moves consist of dihedral angle perturbations. ................................. 16

3.1 The first contribution is an automatic construction of the projection and schema using B-factors, secondary structure information, and KINARI web. ............................................................... 19
4.1 Pictorial representation of the effect that memory limits have on the sims coverage grid data structure. Conformations are the blue dots. Before hitting memory limits, the search is proceeding in some direction depicted by the black arrow. When the search is restarted, coverage estimates are lost so the search could enter areas of the space that have already been sampled. .................................................. 26

4.2 My second contribution is an algorithm that can take advantage of computational resources while maintaining the geometric reasoning that make robotics-inspired methods efficient by implementing a global coverage grid into a centralized database. Each instance of sims periodically synchronizes with the database. .................................................. 28

4.3 Pictorial representation of the effect that the use of summarized coverage statistics from sims 2.0 has on a local coverage grid. Conformations are the blue dots. Before hitting memory limits, the search is proceeding in some direction depicted by the black arrow. When the search is restarted, summary coverage statistics are maintained and depicted in shades of gray. Darker shades indicate more densely sampled areas that the search can use to reduce sampling redundancy. ................................................................. 30

5.1 Cyanovirin-N. a) Colored by B-factors; b) Colors based on kinari web output: clusters (white), hinges (black). The B-factors and hinges reflect the flexibility of the three loop regions at residues 24–28, 50–55, and 75–80. ................................................................. 33
5.2 Calmodulin. a) Colored by B-factors; b) Colors based on KINARI web output: clusters (white), hinges (black). Note that the B-factors reflect the flexibility of the middle alpha helix which would have been treated as stable in previous versions on SIMS.

5.3 Ribose-Binding Protein. a) Colored by B-factors; b) Colors based on KINARI web output: clusters (white), hinges (black). The hinges computed with KINARI do not correspond to the three loop regions in the center. However the B-factors for these loop regions are indeed weighted more heavily than the two domains.

5.4 A single subunit of GroEL. a) Colored by domain: equatorial (blue), intermediate (white), apical (red); b) Colored by B-factors; c) Colors based on KINARI web output: clusters (non-black), hinges (black). The B-factors reflect the flexibility of the apical domain while hinges roughly define regions between major domains.

5.5 Density of nearest neighbor distances (averaged over ten runs) for the GroEL subunit. SIMS 2.0 produces more conformations that are farther apart from each other.

5.6 Density of nearest neighbor distances (averaged over ten runs) for CVN (top), CaM (middle), and RBP (bottom). SIMS 2.0 produces more conformations that are farther apart from each other.

5.7 Density of distances to start conformation (average over ten runs) for the GroEL subunit. SIMS 2.0 produces more conformations that are farther from the start conformation.
5.8 Density of distances to start conformation (averaged over ten runs) for CVN (top), CaM (middle), and RBP (bottom). SIMS 2.0 produces more conformations that are farther from the start conformation.

5.9 Density of distances to start conformation (average over ten runs) for the GroEL subunit. woSync (blue) corresponds to the runs where the synchronization was turned off. Since the distribution shifted to the left, the search was less expansive than SIMS 2.0.

5.10 Density of distances to start conformation (average over ten runs) for the GroEL subunit. randomProj (green) corresponds to the runs where a random projection was used. ssProj (purple) corresponds to the runs where secondary structure information was used to construct the projection. Since the distributions barely shifted, the definition of the projection doesn’t appear to have much impact on the expansiveness of the search.

5.11 Density of distances to start conformation (average over ten runs) for the GroEL subunit. The new schema appears to have the greatest impact since the run that that uses a schema with only secondary structure information (Naive Schema, red) has a density curve that is most similar to the original version of SIMS (old, gray).
List of Tables

5.1  Number of Conformations Produced (averaged over 10 runs) in the GroEL Subunit runs. Standard deviations shown in parentheses. As the number of cores increase, SIMS 2.0 produces conformations at a faster rate. ................................................................. 37

5.2  Number of Conformations Produced (averaged over 10 runs) for three different proteins. Standard deviations shown in parentheses. As the number of cores increase, the new version of SIMS produce conformations at a faster rate. ................................................................. 38
Chapter 1

Introduction

Proteins are responsible for a wide array of functions and are vital for the operation of a healthy cell. A necessary component of understanding protein function ultimately includes a detailed characterization of proteins on a molecular level. Molecular level understanding of protein function and how diseases disrupt function can lead to the creation of new therapeutics [1].

Protein function is related to structure. Starting from a linear chain of amino acids, proteins fold up into particular three dimensional arrangements. A particular arrangement of the atoms that make up the protein is called a conformation. Proteins were initially viewed as folding into a unique conformation that stays mostly stable throughout the protein’s lifetime. In this model, a single conformation would be sufficient to begin understanding protein function.

However, proteins are dynamic [2]. Every residue is constantly bombarded with the surrounding cellular environment and simultaneously trying to maintain favorable internal interactions. There are even so called intrinsically disordered proteins that have evolved to purposely remain unstable in order to achieve proper function [3]. Therefore, a single conformation is no longer sufficient to understand a given protein. We need at least a set of conformations that accurately span the relevant parts of conformational space. In other words, we want to find the set of conformations that a given protein is most likely to adopt.

Experimental techniques can provide vital structural information. Techniques
such as X-ray crystallography and cryo-EM are used to get detailed information from a single conformation [4, 5]. Other methods like hydrogen-deuterium exchange get only indirect dynamical information from parts of the protein [6]. Nuclear magnetic resonance (NMR) can be used to get multiple conformations but is usually limited to small proteins below about 250 amino acids [7].

Computational methods for conformational sampling provide a necessary complement to experiment. These methods generate conformations on the basis of an energy model and some type of sampling protocol (described in detail in the next chapter). Molecular dynamics (MD) and Markov chain Monte Carlo methods (MCMC) are the most popular sampling algorithms. MD simulations follow the evolution of each atom in a defined system (protein plus environment) using classical mechanics [8]. MCMC methods apply particular moves to propose new conformations that are accepted based on the Metropolis criterion [9]. However, these methods do not scale well in the size of the studied protein. MD simulations are typically done with timesteps on the order of femtoseconds ($10^{-15}$s) while many interesting protein motions like protein folding or ligand binding occur on the order of milliseconds to even seconds. MCMC methods face the difficulty of diminishing acceptance rates as the size of the protein increases [9]. Anton 2, a supercomputer specifically made for MD simulations, takes about two weeks to run a single millisecond simulation of a 35 residue protein [10]. While this is an impressive feat, we would like to be able to do conformational sampling for larger proteins with modest computational resources.

The focus of this work is on a relatively newer class of methods that originate from sampling-based robot motion planning [11–13]. These methods are closer in spirit to MCMC in that they apply moves or perturbations to propose new conformations, but in this case the moves can be applied to any of the previously sampled conformations. In
other words, the set of conformations already sampled is used to compute some notion of coverage to bias search to unexplored parts of the conformational space. In this work, we will call this idea geometric reasoning for protein conformational sampling. Essentially, the set of conformations itself is used to generate more conformations. We believe that this idea could be the basis for scalable conformational sampling algorithms that still provide molecular level details of protein function.

However, robotics-inspired conformational sampling has not reached the point to where systems larger than 300 residues could be studied without making major simplifying assumptions [14]. There are two major issues that must be addressed to push this barrier. First, as with MCMC methods, proposed conformations become rejected more often for large proteins. Larger proteins typically have more correlation among residues that result in more intricate movements. Perturbation strategies are then less likely to capture these movements, and the probability of proposing a valid conformation diminishes greatly. Second, current robotics-inspired methods for conformational sampling were not built explicitly for use on multicore systems, which becomes a requirement for running on large proteins. The complicating issue arises from the fact that robotics-inspired methods make regular use of previously sampled conformations. Conformations for larger proteins take more memory to store and keeping sampled conformations becomes memory intensive. For example a conformation with 300 residues contains about 5700 atoms and takes 68.4 KB per conformation (each atom takes $4*3=12$ bytes). Then for a million conformations we would need at least 68.4 GB of memory. This type of memory usage clearly isn’t scalable for protein systems that can have tens of thousands of residues. This work aims to address these issues that will push the boundary of the size of protein systems that robotics-inspired conformational sampling can handle.
1.1 Contributions

This thesis describes SIMS 2.0, a new framework for conformational sampling built from prior work called the Structured Intuitive Move Selector (SIMS) [15]. While the improvements are presented in the context of SIMS, they are also applicable to other robotics-inspired conformational sampling approaches.

The first contribution of this thesis is an automated characterization of relative flexibility based on B-factors, secondary structure, and rigidity analysis. Our previous work relied only on secondary structure, supplemented by expert insight. The new approach reduces the need for expert input (although it is still possible to take this information into consideration) and allows SIMS 2.0 to focus more on the relevant parts of the conformational space.

The second contribution is a new algorithm for dealing with a critical data structure in the sampling approach. This algorithm enables SIMS 2.0 to be executed across multiple cores while addressing the memory issue described in the previous section. The algorithm directs each core to sample conformations while periodically syncing with a central database. The database coordinates across the cores such that each core is aware of the work done by other cores. Our new algorithm can take advantage of multicore systems while still keeping the geometric reasoning that robotics-inspired algorithms excel at.

We will show that with our contributions we are more successful in generating diverse conformations. This work shows results for four different proteins of various sizes.
1.2 Organization

The rest of the thesis is organized as follows. In the next chapter, we will give an overview of the various methods used for conformational sampling and give a more detailed description of SIMS. In Chapter 3, we will describe our use of B-factors, secondary structure, and rigidity analysis for conformational sampling. In Chapter 4, we will describe our new algorithm for running SIMS 2.0 across multiple cores. In Chapter 5, we describe the results of our simulations. Finally, we conclude with a brief summary and a discussion of directions for future work.
Chapter 2

Background

In this section, various methods for conformational sampling will be discussed along with a detailed description of our previous framework, SIMS [15]. It will be important to keep in mind the degree to which previously sampled conformations are used during the search. Molecular dynamics (MD) and Markov chain Monte Carlo methods (MCMC) were the first methods for protein conformational sampling, followed by a multitude of enhancements that aim to scale towards more complex applications [8, 16]. Then independently, algorithms from the robot motion planning field that relied more on geometric reasoning came into the picture and have since been applied to various applications in structural biology [14, 17]. SIMS is one such method, and the final section in this chapter will set the stage for the contributions that follow.

2.1 Energy Models and RMSD

Before we discuss conformational sampling methods, we will first discuss how conformations are distinguished from one another. For a given protein, each conformation can be assigned an energy according to some energy model. Most energy models are based on approximations of the underlying quantum mechanics, also known as physics-based potentials [18]. These models compute the potential energy of the system given the interactions between its atoms. Knowledge-based potentials are also popular which are based on interactions from a database of known protein structures.
Rosetta is a popular knowledge-based potential that we use in this work [19].

Root mean square deviation is one of the standard distance metrics used to compare conformations. This value is typically computed over every atom but in this work we only use alpha carbons in the protein backbone because we are not explicitly manipulating degrees of freedom in the side chains. Conformations that are considered “close” are within 2 Angstroms, which is the typical experimental error in X-ray crystallography studies.

2.2 Standard Methods: MD and MCMC

Molecular simulation of proteins started to become popular when the first force fields were developed and applied to proteins [20]. The Hamiltonian (potential plus kinetic energy) is computed and its negative gradient gives rise to the forces, where Newton’s second law is integrated to evolve the system along a timestep. This process is repeated for many timesteps, and this simple algorithm is the basis of MD simulations.

MCMC came about which introduced the idea of biased sampling [21]. Moves or proposals are applied to perturb the system, and the resulting conformation is accepted or rejected according to the Metropolis criterion. One important advantage that this method had was that a gradient was not strictly required, which meant that energy functions with discontinuities could be used. Since gradients were not computed and the perturbations were somewhat independent of the underlying energy landscape, MCMC could in principle perform faster exploration of conformational space [9, 22, 23].

MD and MCMC remain popular today because they produce the desired thermodynamic ensembles. Sampling from these ensembles mean that conformations are distributed according to its Boltzmann weight where low-energy conformations are
exponentially more likely to be sampled than higher-energy conformations. This also means that the estimator for thermodynamic observables (which could be measured experimentally) is simply the sample mean across the whole simulation [8].

However, there are well known issues when using the pure versions of MD and MCMC. Physiologically interesting events typically occur on the order of milliseconds or seconds while typical timestep lengths in MD simulations are on the order of femtoseconds [17]. This means that MD is quite inefficient at sampling rare events, which could lie on far parts of the relevant conformational space. MCMC methods rely on how new conformations are proposed. As the size of the system increases, the rejection rate becomes too high, and the method struggles to find valid conformations [9]. Additionally, both methods are prone to get stuck in local minima, which wastes computational resources that could be spent sampling other parts of the space. These problems are only exaggerated for larger systems with more degrees of freedom, which lead to rougher energy landscapes with more local minima. This lead to a multitude of additional approaches that aim to speed up conformational sampling.

2.3 Enhanced Sampling

2.3.1 Coarse-grained Energy Models

One way of improving conformational sampling is to reduce the number of degrees of freedom that is considered in the simulation [18]. This idea is called coarse-graining. Coarse-graining is not completely new as the first MD simulations were done in vacuum (water was not taken into account), and important insights into the protein folding process came from “bead” models (residue-level interactions) or even discrete spaces (lattice models). One modern example is Rosetta’s centroid representation, where the
side chain of every residue in the protein is reduced to a single super-atom [19]. There are a variety of such coarse-graining approaches that lead to faster energy computation and a smoother energy surface, which reduces the number of local minima.

### 2.3.2 Generalized Ensemble Methods

Another approach under the term *generalized ensemble methods* aims to modify the underlying thermodynamic ensemble in order to favor the sampling of rare events and explore the conformational space more evenly [16, 24–30]. There are two such algorithms that have gained the most traction over the years. The first is referred to as Replica Exchange Molecular Dynamics (REMD) [24]. This method usually involves running simulations (or replicas) in parallel and then exchanging the states between replicas to encourage faster exploration of conformational space. The replicas are usually at different temperatures where higher temperature replicas produce more variable conformations that trickle down to the replica at the temperature of interest. The other algorithm is known as Metadynamics [30]. Metadynamics has been informally described as “filling in the free energy landscape with computational sand.”

A few variables known as collective variables are carefully chosen to represent the energy landscape. During the course of the simulation, the conformations sampled are projected down to the space of collective variables, where a Gaussian centered at the conformation is added. The sum of the Gaussians are then used to compute a force that biases the simulation away from previously explored areas of the projected space. Thus, the sampled conformations are used indirectly in improving the exploration of conformational space.
2.3.3 Markov State Models

The final approach worth noting here arose from the need to create methods that analyze simulations in a statistically-rigorous manner. The main method here is called Markov State Models (MSM) [31, 32], although other related methodologies are outlined here [33]. MSMs provide a way of combining several short-time trajectories together by clustering conformations into macrostates and counting the transitions between them. The interesting use of MSMs comes from the ability to identify underexplored areas of the conformational space. This could mean clusters with low conformation counts or transition rates based on a low number of counts. The short-time trajectories usually come from classical MD but there are now ways to combine trajectories that were obtained from enhanced sampling methods [34]. MSMs are yet another step in the direction of using sampled conformations to improve the exploration of conformational space.

2.4 Geometric Reasoning for Conformational Sampling

Along a somewhat different direction, an analogy was discovered between robots and proteins that opened the door to the use of sampling-based motion planning algorithms [14, 17]. The defining characteristic these motion planning methods have is the direct use of the sampled conformations to bias search to unexplored parts of the conformational space. MD and MCMC based methods for conformational sampling work in a sequential manner, while motion planning methods can perturb any of the previously sampled conformations to obtain new conformations. The methods described earlier could be seen as slowly approaching this characteristic that motion planning used all along. The idea of using the sampled conformations as a bias has
even led to the rise of genetic algorithms for conformational sampling that evolve a population of conformations during the exploration [35, 36].

Motion planning algorithms cannot be applied as they were to robot configurations without modification. Robots needed to avoid “obstacles” that correspond to a binary attribute for each configuration as to whether they are in collision or not. Protein conformations have the potential energy, which is a continuous attribute. Obtaining lower energy conformations has been handled for example through a Metropolis-like criterion [37] or simply through an energy threshold [15]. Realistic conformations are also usually highly constrained, which means a greater correlation between degrees of freedom. This means that conformations cannot be obtained through a uniform sampling across each degree of freedom. This has led to algorithms such as NMA-RRT [38], which samples conformations using normal mode analysis, or SIMS, which relies on a set of multi-resolution moves to explore the space [15].

Robotics-inspired methods have been used in a variety of applications, including protein folding [39–41], loop sampling [42, 43], identifying low-energy transitions between known conformations [15, 38, 44, 45], and exploring conformational space [15, 46, 47].

Robotics-inspired methods are still evolving, and there are several open problems. While the increased emphasis on geometry has allowed greater computational efficiency, there is no known way to relate the resulting sampled conformations to a particular thermodynamic ensemble. Additionally, these methods rely on moves or perturbations to explore conformational space, which is a bottleneck for increasingly large and more constrained proteins. Finally, the use of all the sampled conformations could be memory intensive, especially in the context of running these algorithms in parallel. In this work, we address the last two problems.
2.5 Structured Intuitive Move Selector

SIMS works by repeatedly applying small perturbations called *moves* to previously generated conformations [15]. The choice of which previously sampled conformation to perturb is determined from the use of *coverage estimates*, which are encoded inside a data structure called the *coverage grid*. The coverage grid data structure currently relies on a *projection* as input, which is used to map conformations to the coverage grid.

SIMS’ perturbation strategy is defined in the *schema*, which specifies what type of moves to use, how they are applied, and how often to apply them. The moves are applied to sets of residues called *fragments*, where each move-fragment pair is assigned a *weight* to reflect how often to apply the move-fragment pair. Ideally, the schema captures which fragments of residues might be involved in coordinated motion and how flexible they are (through the use of the weights).

The conformational sampling framework can be summarized as: (1) sample a previously generated conformation, (2) sample a fragment of this conformation along with a move with a probability proportional to its weight, and (3) accept the new conformation if its energy is below some user-specified threshold. There are different robotics-inspired sampling techniques that can be used for step 1 [14]. This process is schematically represented in Fig. 2.1. Previous results [15, 48] show that SIMS is good at finding low-energy paths between states for proteins up to about 300 residues.
Figure 2.1: Pictorial representation of the sampling process in SIMS. SIMS cycles between three steps. 1) Sampling a conformation based on coverage estimates. 2) Perturbing the structure using the schema. 3) Accepting the new conformation if it is below a user defined energy threshold. The projection is needed for computing coverage, which is used in selecting conformations for Step 1. The schema defines SIMS’ perturbation strategy for Step 2.

2.5.1 Sampling a Conformation using Coverage Estimates

Rather than always perturbing the last conformation (like MCMC sampling), SIMS and many other robotics-inspired sampling techniques heuristically bias the selection of a conformation to push the search towards parts of the conformational space that are less densely sampled. To keep track of coverage of conformational space, SIMS projects conformations using the inputted projection through the process described
by Fig. 2.2, which are then mapped to the coverage grid, and coverage estimates are updated. Conformations are sampled from the coverage grid by focusing on less densely populated areas specified by the coverage estimates (described in more detail in chapter 4). In the initial work, random projections were used [15]. Later work showed the choice of projection significantly affects performance [48].

Figure 2.2 : How SIMS computes coverage using the projection.

### 2.5.2 Perturbing Conformations using the Schema

Once a conformation has been sampled using coverage, the schema is used to generate a new conformation. The schema essentially contains statements like “perturb this set of residues using this type of perturbation with a particular weight.” The “set of residues” are called fragments. Fragments need not correspond to consecutive residues (e.g., several loop regions may connect two domains and may need to coordinate their motion). Fragments may also overlap (e.g., a set of residues may be involved in different coordinated motions). Previous work showed that secondary structure can
be used to partition the protein into flexible loops, which should be perturbed more often, and relatively rigid helices and sheets, which should be perturbed less often.

The “type of perturbation” is known as a move. For each type of fragment, different sets of moves can be defined. SIMS currently has 5 major moves: minimization, loop, rigidbody, randomOne, and randomAll illustrated in Fig. 2.3.

1. Minimzation involves a few steps of a minimization protocol on the fragment.

2. Loop involves sampling a new loop conformation using a few steps of Monte Carlo sampling.

3. Rigidbody involves rotating and translating one part of the domain relative to another and filling in the linking residues in between. The linking residues are chosen randomly within a fragment.

4. randomOne involves randomly perturbing a single residue’s dihedral angles from the fragment

5. randomAll involves perturbing all the dihedral angles in the fragment.

The weight determines the probability that a particular fragment and move are chosen. Once the move is applied to a fragment, side chain positions are determined by Rosetta’s side chain minimization protocol [49, 50] to obtain a complete conformation. Since the moves are only explicitly sampling degrees of freedom restricted to the backbone dihedral angles, SIMS is classified as a coarse-grained method and provides a residue-level characterization of the underlying energy landscape. In the next chapter, we describe how SIMS 2.0 automatically constructs a useful projection and schema.
Figure 2.3: Types of moves in SIMS. The \textit{randomOne} and \textit{randomAll} moves consist of dihedral angle perturbations.

### 2.5.3 Accepting/Rejecting Conformations

The new conformation is automatically rejected if its computed energy is above a user-defined threshold. In this work, we use Rosetta’s “\texttt{score12\_full}” energy function for our smaller test runs and Rosetta’s “\texttt{score3}” energy function in “centroid” mode for our larger runs, although other energy functions could be used as well. Centroid mode computations in Rosetta are faster because side chains are approximated as a single atom of varying size, which provides additional computational benefit for larger proteins while still maintaining molecular detail. Energy thresholds are chosen to filter out conformations with steric clashes and other highly unfavorable interactions. Energy thresholds for this work are set to the value 0 because past experiments tend to show that conformations with a positive Rosetta score have some degree of
steric clashes. One could always lower the energy threshold or filter out high-energy conformations in a post-processing step to obtain sampled conformations with lower energy. One could also use a more fine-grained energy function if more detailed physics need to be modelled. However, issues dealing with the energy function and the energy threshold are outside the scope of this thesis.
Chapter 3

Incorporating Flexibility Information

3.1 Motivation

Previous iterations of SIMS had a default sampling strategy ill-suited for large, highly constrained systems. SIMS depends heavily on a projection as well as a schema (see Chapter 2). In the absence of expert knowledge, the default projection is randomly generated while a default schema file will generate fragments based on secondary structures. These defaults contain virtually no information on a protein’s inherent flexibility. Therefore as the system becomes larger and more constrained, SIMS is unlikely to keep track of the meaningful conformational space using a random projection [48] and struggles with finding new conformations given only secondary structure information. Below we will show that readily available flexibility information improves conformational sampling for SIMS 2.0.

First, we can use experimental residue-level flexibility information to automatically define a projection. Note then that any robotics-inspired method that makes use of a projection can benefit from this projection definition. Next, we include information about which residues are involved in coordinated motion, which is a key component of SIMS’ fragment-based perturbation approach. We automatically generate a schema that biases perturbations towards fragments that are more flexible using a combination of B-factors, secondary structure, and rigidity analysis. The rigidity analysis is done using KINARI web [51]. This construction effectively increases the number of conformations
generated and can be used for any conformational sampling method that relies on the use of moves or perturbations to generate new conformations. This contribution is pictorially depicted in Fig. 3.1. We describe this in detail below.

3.2 Automatically constructing the projection using B-factors

We use B-factors from experiment to automatically generate a projection that weights residues in flexible regions over stable ones, resulting in a more accurate proxy to meaningful conformational space. SIMS relies on a starting conformation which is derived from experiment. These experiments will have some measure of uncertainty, which is usually correlated with flexibility or movement. For X-ray crystallography experiments, B-factors (also known as temperature factors) describe the displacement
of the atomic positions from their mean values [52]. These B-factors can be easily extracted from a PDB file (coordinates of a protein conformation derived from experiment) to generate the projection. B-factors can also be generated from prediction tools [53].

Recall from Chapter 2 that SIMS uses a projection that helps map a conformation to the coverage grid. For a system with \( n \) residues, the projection will be of dimension \( d \times 4n \), where \( d \) is the dimension of the projection. SIMS uses the sine and cosine of each dihedral angle in the system (two per residue) for a total of \( 4n \) degrees of freedom. The sine and cosines are done to embed the angles to a Euclidean space.

In SIMS 2.0, the B-factors are used to explicitly construct the first dimension of the projection, which is the first row of size \( 1 \times 4n \). First, the \( n \) B-factors corresponding to the alpha carbon atoms in the backbone are extracted from the PDB file. Then for each factor \( b_i \), a user defined range \( [b_{\text{low}}, b_{\text{high}}] \) of the B-factors is imposed using the following transformation.

\[
t_i(b_i) = \begin{cases} 
  b_{\text{low}}, & \text{if } b_i < b_{\text{low}} \\
  b_{\text{high}}, & \text{if } b_i > b_{\text{high}} \\
  b_i, & \text{otherwise}
\end{cases} \quad (3.1)
\]

This transformation is done to suppress B-factors that overly dominate. For our experiments, we normally set \( b_{\text{low}} = 20 \) and \( b_{\text{high}} = 50 \). Then the following transformation is applied to each \( t_i \).

\[
f(t_i) = \exp(t_i/\alpha) \quad (3.2)
\]

This transformation essentially spreads the values farther apart from one another. The amount of spreading can be controlled using another user defined parameter \( \alpha \).
The use of an exponential function here is to space the larger B-factors away even further from the smaller B-factors such that the flexible residues are sampled more often. All of our experiments use $\alpha = 10$. Next, a vector of size $1 \times 4n$ is created by replicating each element four times consecutively. This operation is done because every four elements in a single row of the projection correspond to a single residue.

The other $d - 1$ dimensions are generated randomly. For each extra dimension, we generate a $1 \times 4n$ random vector, where each element is drawn from a standard Normal distribution. Finally, the vectors are made orthonormal using the Gram-Schmidt process. The full projection is constructed vertically with the B-factor row at the top and the other randomly generated rows below to get a $d \times 4n$ matrix that is inputted into SIMS 2.0.

3.3 Automatically constructing the schema using B-factors, secondary structure, and rigidity analysis

In previous iterations of SIMS, the default schema made use of only secondary structure information. Alpha helices and beta sheets were made more stable than loops. However, secondary structure is essentially local information because every secondary structure element consists of a few consecutive residues. While in general helices and sheets are more stable than loops, helices and sheets from different parts of the protein may have vastly different flexibility [48]. The previous default schema lacked global structure information, like tertiary structure.

Tertiary structure information is not available directly from experiment but an approximation can be derived from the atom coordinates using rigidity analysis. This is done with KINARI web, a suite of tools for computing rigidity and flexibility of
biomolecules [51]. KINARI web uses the pebble game algorithm to compute groups of residues that are expected to move together, and we treat these groups of residues as tertiary structures [54]. A PDB file is inputted into the web server to get the residue groupings. All default parameters are used in the computation. KINARI outputs a file that specifies residue clusters. Each cluster consists of a set of residue intervals. We use each residue interval from each cluster as a separate residue grouping. We use these residue groupings in SIMS 2.0 as a proxy to tertiary structure boundaries.

SIMS uses a schema that specifies fragments, which consist of groups of residues, and moves, which define perturbations on the fragments. As mentioned in Chapter 2, SIMS currently has 5 major moves: minimization, loop, rigidbody, randomOne, and randomAll. Now in SIMS 2.0, each fragment is weighted using B-factors. The B-factors are processed the same way as the previous section to produce \( f(t_i) \) for each residue, and the weight is computed by summing each \( f(t_i) \) in the fragment. Then, SIMS 2.0 generates a schema that has three major sets of fragments.

The first is a fragment that is defined over the whole protein. This fragment is sampled 10% of the time and is used to occasionally generate structures with a lower energy or try disruptive whole protein perturbations. When this fragment is sampled, minimization is done 90% of the time and rigidbody is done 10% of the time.

The second major set of fragments is generated using secondary structures. This set of fragments constituted the majority of the fragments in previous iterations of SIMS. We place less overall influence on this set since we only sample this set 40% of the time (compared to 90% previously). The secondary structure information is extracted from the PDB file. Alpha helices and beta sheets are treated as loops if they are less than 5 residues in length. A fragment is then defined for each consecutive interval of residues with the same secondary structure classification. Loops can be
perturbed using loop (10%), randomOne (30%), randomAll (30%), or rigidbody (30%). Helices and sheets are generally more stable so we use randomOne (50%) or rigidbody (50%). In the previous default, SIMS manually defined loops to be sampled with greater weight than helices and sheets. In SIMS 2.0, fragments are now defined using B-factors.

Finally, the third major set of fragments is generated using the residue groupings from KINARI. Since each residue grouping is predicted to moved together, a fragment is defined for each interval of residues between the residue groupings (intervals at the ends are not counted). In other words, the parts of the protein that we wish to perturb are the residues that lie in between tertiary structures, analogous to hinges. When these hinges are perturbed, the surrounding parts move together as a rigid body. Each fragment is weighted using B-factors and can be perturbed using loop (10%), randomOne (35%), randomAll (20%), or rigidbody (30%). This set of fragments is sampled 50% of the time.

The increased emphasis on tertiary structures is more aligned with the intended use of SIMS for sampling large backbone motions. All of the percentages given above were determined empirically and further research may fine tune these values. We end by noting that unlike the original SIMS, the projection used for the coverage grid and the schema used for sampling and perturbing fragments are now coupled in SIMS 2.0 through the use of B-factors.
Chapter 4

Implementing a Global Coverage Grid

4.1 Motivation

Conformational sampling generally becomes a harder problem as the size of the considered system increases, so we would like to be able to take advantage of more computational resources when needed. Energy computation takes longer so we must run simulations longer in order to sample a given number of conformations. The conformational space is also larger so we may need more conformations to accurately characterize the space. Additionally for robotics-inspired methods, keeping all of the sampled conformations means that the rate of memory usage increases.

Initially, one may consider trying to parallelize Sims, but this turns out to be more complicated than it seems. We could parallelize the energy computation, which is the single most expensive operation. However, robotics-inspired methods involve other steps as shown in Chapter 2, which are not easily parallelizable, meaning that many cores would be idle. One interesting direction is to have each core sampling conformations. But this means each core must have access to all of the sampled conformations because any previously sampled conformation could be perturbed in a given iteration. The only way this is possible, in the context of running Sims on a high performance cluster, is if a central database on disk is maintained because keeping conformations in memory becomes prohibitive. This may be expensive if every iteration involves reading and writing to disk.
If directly parallelizing SIMS is difficult, we need a new algorithm that instead tries to combine multiple running copies of SIMS. Each copy would only work with its own sampled conformations, and then the solution would rely on coordination between SIMS runs. However, there is one subtle issue that must be addressed. During a SIMS run, conformations are being accumulated in memory. At some point, the memory resources for the SIMS run will be exhausted, and the conformations must be written into disk. If the search is not completed, we must restart the search with a subset of the conformations. But the coverage estimates depended on the use of sampled conformations through the coverage grid, so restarting essentially makes the search “forget” what has been sampled before (Fig. 4.1). Any algorithm must address this unavoidable problem that only worsens for larger proteins. While there has been work on parallelizing robotics-inspired sampling techniques [55–57], these approaches do not address the fact that memory use becomes a bottleneck for large molecular complexes.

We want to build an algorithm that combines multiple instances of SIMS, where a successful implementation means that every instance of SIMS has some notion of what other instances of SIMS have sampled, as well as address the memory issue from the previous section. Each individual SIMS instance still maintains a local coverage grid. In SIMS 2.0, we solve this by implementing a global coverage grid, whose scope reaches across instances of SIMS. We will first describe in more detail how coverage estimates are computed.
Figure 4.1: Pictorial representation of the effect that memory limits have on the SIMS coverage grid data structure. Conformations are the blue dots. Before hitting memory limits, the search is proceeding in some direction depicted by the black arrow. When the search is restarted, coverage estimates are lost so the search could enter areas of the space that have already been sampled.

4.2 Computing Coverage

In SIMS, conformations sampled are added to the coverage grid data structure through the use of a projection. The grid contains cells with conformations mapped to them and by counting how many conformations map into each cell, we can estimate the sampling density or coverage. Different robotics-inspired sampling techniques such as Expansive Space Trees [58] and Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE) [59] use this information to guide the sampling towards
less-densely sampled parts of the conformational space. In this work we use KPIECE since it has been shown to significantly outperform EST [59].

KPIECE keeps track of various statistics for each grid cell and uses these statistics to compute a heuristic called importance for each cell. Conformations in cells with higher importance are perturbed more often. Importance is computed for each cell using four statistics.

(1) The number of projected conformations mapped into the cell
(2) The number of times the cell has been chosen for expansion
(3) The iteration in which the cell had its first conformation mapped to it
(4) The number of cell neighbors that have conformations mapped to it

An increase in (1), (2), and (4) produce lower importance while a high value for (3) produces greater importance. Indeed, these are the statistics that are lost when conformations must be written to disk. Fewer conformations are accounted for in the coverage grid, and the importance heuristic loses the ability to differentiate cells based on sampling density.

4.3 Solution: Summaries generated from the Global Coverage Grid

In SIMS 2.0, we assign every available core an instance of SIMS to perform its own conformational sampling. Our algorithm maintains a global coverage grid, whose scope reaches across every instance of SIMS. Global grid cell statistics are saved into a central database along with conformations sampled from each SIMS run. When a new SIMS run is started, a subset of conformations is loaded along with “summarized” coverage statistics about the global coverage grid. These “summarized” coverage statistics are
our indirect way of accounting for the sampling done by other SIMS instances. When a SIMS instance is finished, the new conformations are then written to the database and the global grid cell statistics are updated. Each core runs its own SIMS instance and communicates with the database. This algorithm provides coordination between SIMS runs as well as solves the memory issue. My contribution is illustrated in Fig 4.2.

![Diagram of Contribution 2]

**Figure 4.2 :** My second contribution is an algorithm that can take advantage of computational resources while maintaining the geometric reasoning that make robotics-inspired methods efficient by implementing a global coverage grid into a centralized database. Each instance of SIMS periodically synchronizes with the database.
4.3.1 Global Grid Cell Statistics

Global grid cell statistics on the global coverage grid in SIMS 2.0 are maintained centrally in a similar manner to how an individual SIMS run computes grid cell statistics on a local coverage grid. As referred to in section 4.2, each grid cell computes an importance heuristic that determines how often conformations from that cell are perturbed. In the context of the global coverage grid, (3) is no longer used to compute importance because there is no meaningful way to define iteration when multiple cores are sampling simultaneously. However, (1), (2), and (4) are still used.

4.3.2 Updating the Global Grid Cell Statistics

When a SIMS run is finished, the new conformations are written to the database and the global grid cell statistics are updated. This is done by computing the change in (1) and (2) during the course of the SIMS run. These values are then added to the global values of (1) and (2) in the database. (4) is subsequently updated based on the new grid cell statistics. These statistics are global in the sense that all the cores provide an update when their SIMS run is finished.

4.3.3 Inputting Summarized Coverage Statistics

When a SIMS run is started, the KPIECE sampling strategy is used to select new starting conformations based on the global coverage grid. Additionally, the local coverage grid is initialized to the current values in the global coverage grid. While each core is aware of the global coverage statistics, each core can only perturb conformations that are in memory (i.e., generated since the start of the SIMS run). However, when a conformation is generated in a cell, the sampling density is determined not only by the conformations generated from the SIMS run but also the sampling density loaded
at the start of the run (Fig. 4.3). The presence of all other sampled conformations is thus accounted for indirectly.

Figure 4.3: Pictorial representation of the effect that the use of summarized coverage statistics from SIMS 2.0 has on a local coverage grid. Conformations are the blue dots. Before hitting memory limits, the search is proceeding in some direction depicted by the black arrow. When the search is restarted, summary coverage statistics are maintained and depicted in shades of gray. Darker shades indicate more densely sampled areas that the search can use to reduce sampling redundancy.

4.3.4 Role of the Restart Frequency

At this point, it should be clear that each core achieves some sort of coordination through the use of the global coverage estimates. We now claim that this algorithm also solves the memory issue. Each core will cycle through three steps: reading
conformations and statistics from the database, performing a SIMS run, then writing conformations and updating the global statistics. The frequency in which each core does this process is a user defined parameter called the restart frequency. The memory issue is avoided through the use of the summarized coverage and a restart frequency that is not too low. That is, we must restart often enough such that a core will not run out of memory from sampling too many conformations. However interestingly, we also have incentive to restart often as this is the mechanism in which the database is updated with the work that the core has done. We leave it as future work to determine an optimal value for this parameter.
Chapter 5

Results

The previous two chapters featured contributions that aim to scale robotics-inspired conformational sampling. We run a variety of experiments to show that our new methods are essential if we are to sample diverse conformations. First, we show the extent to which the use of B-factors and KINARI reflect the natural flexibility for the four proteins that we use in this work. Next, we show that SIMS 2.0 produces more conformations at a faster rate than a naive version of SIMS. Finally, we show that SIMS 2.0 produces more diverse conformations. We also quantify the importance of each contribution.

All of our experiments are run on a single compute node with two Intel E5-2650v2 Ivy Bridge EP processors for a maximum of 16 cores. All runs are done for 100 minutes and write conformations to disk every 10 minutes (restart frequency is 6 restarts per hour). Energy thresholds for all the runs are set to the value 0 because past experiments tend to show that conformations with a positive Rosetta score have some degree of steric clashes. All the projections used are 2-dimensional.

5.1 Proteins Used in Experiments

We illustrate the benefits of our new methods on four proteins of varying sizes: Cyanovirin-N (CVN) [60], Calcium-loaded Calmodulin (CaM) [61], Ribose-Binding Protein (RBP) [62, 63], and a single subunit of GroEL [64, 65]. CVN, CaM, and RBP
are smaller sized proteins that we have previously studied in the context of evaluating random projections [48]. The GroEL subunit is a larger and more constrained system that SIMS 2.0 can now handle more easily.

5.1.1 Cyanovirin-N

CVN is a 101 residue bacterial protein (PDB 3EZM) that exhibits antiviral activity towards the human immunodeficiency virus. CVN shows large scale motions from the correlated activity of three loop regions at residues 24–28, 50–55, and 75–80. These same loop regions are found as flexible from Fig. 5.1a. From Fig. 5.1b, KINARI classifies two of these these loop regions as hinges (residues 24–28 and 40–54). When constructing the schema as described in the previous chapter, the range of the B-factors are 10–20, instead of the default 20–50.

Figure 5.1: Cyanovirin-N. a) Colored by B-factors; b) Colors based on KINARI web output: clusters (white), hinges (black). The B-factors and hinges reflect the flexibility of the three loop regions at residues 24–28, 50–55, and 75–80.
5.1.2 Calmodulin

CaM is a 144 residue protein (PDB 1CLL) involved with interactions between calcium ions and other proteins. Flexible parts of the protein are found in residues 5–20, 35–41, 52–57, 67–80, 87–93, 107–116, and 126–129 which is represented in Fig. 5.2a. Note that the flexible helix at residues 67-80 would have been treated as a more stable part of the structure (and hence, not perturbed frequently) in the previous SIMS default. The computed hinges in Fig. 5.2b are loops located at residues 41–43, 57–62, and 114–116.

Figure 5.2: Calmodulin. a) Colored by B-factors; b) Colors based on KINARI web output: clusters (white), hinges (black). Note that the B-factors reflect the flexibility of the middle alpha helix which would have been treated as stable in previous versions on SIMS.

5.1.3 Ribose-Binding Protein

RBP is a 271 residue protein (PDB 1URP, chain A) that consists of two domains connected by three loop regions located at 91–104, 226–237, and 253–269. The first
two regions are more constrained and have to move in a coordinated way. Interestingly, the B-factor distribution in Fig. 5.3a shows that the most flexible parts are mainly the alpha helices at the end. The KINARI output in Fig. 5.3b also predicted that most of the protein move together (residues 3–205 and 211–268), leaving a single hinge at residues 206–210 that is not part of the three main loop regions. Nevertheless, the B-factors for the main loop regions indeed show greater flexibility than the surrounding two domains.

Figure 5.3 : Ribose-Binding Protein. a) Colored by B-factors; b) Colors based on KINARI web output: clusters (white), hinges (black). The hinges computed with KINARI do not correspond to the three loop regions in the center. However the B-factors for these loop regions are indeed weighted more heavily than the two domains.

5.1.4 GroEL Subunit

GroEL (PDB 1XCK) is a molecular chaperone consisting of 14 identical subunits forming two heptameric rings. We focus this work on a subunit of GroEL because it is a well studied system, and we envision SIMS tackling systems of this size and
beyond. We extract out chain A and use this as input to KINARI web to obtain the tertiary structure information. Each subunit consists of 524 residues arranged into three domains (Fig. 5.4a): equatorial (1–133, 409–524), intermediate (134–190, 377–408), and apical (191–376). The apical domain has the most movement, facilitated by hinges located in the intermediate domain [64, 65]. The equatorial domain remains largely stable.

In Fig. 5.4b, notice that the high B-factors correlate to apical domain which is known to be the most flexible part. Fig. 5.4c shows the parts of the subunit from which we treat as hinges. Note how the hinges are mostly loops located between major helices/sheets in the system. Perturbing these hinges will in turn affect the alpha helices and beta sheets, analogous to a rigid body transform.

Figure 5.4 : A single subunit of GroEL. a) Colored by domain: equatorial (blue), intermediate (white), apical (red); b) Colored by B-factors; c) Colors based on KINARI web output: clusters (non-black), hinges (black). The B-factors reflect the flexibility of the apical domain while hinges roughly define regions between major domains.
5.2 Improved scalability

In this section, we want to get a sense of how SIMS 2.0 scales with respect to how many conformations are produced. Note that all of the conformations produced are required to be below an energy threshold. Producing more conformations is more efficient in the sense that the method is rejecting conformations less often.

In the following experiments, we compare SIMS 2.0 with a version of SIMS that allow comparison for multicore runs. We will denote this version as “Naive SIMS.” This version uses the old default schema (based on secondary structure) and the old default projection (randomly generated). Additionally for multicore runs, Naive SIMS refers to our new algorithm but with no update on global coverage statistics. Naive SIMS is running almost independent SIMS runs from conformations chosen using the KPICE sampling strategy. We run experiments on a varied number of cores (1, 4, and 16) to assess the effect that our new methods have on scalability. Table 5.1 shows the average number of conformations produced.

Table 5.1: Number of Conformations Produced (averaged over 10 runs) in the GroEL Subunit runs. Standard deviations shown in parentheses. As the number of cores increase, SIMS 2.0 produces conformations at a faster rate.

<table>
<thead>
<tr>
<th>GroEL Subunit</th>
<th>1 Core</th>
<th>4 Cores</th>
<th>16 Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMS 2.0</td>
<td>3026 (162)</td>
<td>11120 (623)</td>
<td>38763 (2729)</td>
</tr>
<tr>
<td>Naive SIMS</td>
<td>850 (39)</td>
<td>3495 (223)</td>
<td>13739 (392)</td>
</tr>
<tr>
<td>Difference</td>
<td>+2176</td>
<td>+7625</td>
<td>+25024</td>
</tr>
</tbody>
</table>

Table 5.1 clearly shows that as the number of cores used increases, the number of conformations produced by SIMS 2.0 increases at a faster rate for the GroEL subunit
system. Note that all of the conformations produced are below the user-defined energy threshold. Thus, SIMS 2.0 produces low-energy conformations at a faster rate. This improvement becomes vital as we begin to do conformational search for larger systems because the energy computation becomes more expensive. SIMS 2.0 would make more efficient use of resources when the search requires more cores. Table 5.2 shows similar results for the other three proteins.

Table 5.2: Number of Conformations Produced (averaged over 10 runs) for three different proteins. Standard deviations shown in parentheses. As the number of cores increase, the new version of SIMS produce conformations at a faster rate.

<table>
<thead>
<tr>
<th></th>
<th>1 Core</th>
<th>4 Cores</th>
<th>16 Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVN Naive sims</td>
<td>10710 (1501)</td>
<td>43136 (3064)</td>
<td>176813 (6721)</td>
</tr>
<tr>
<td>Difference</td>
<td>+17584</td>
<td>+78931</td>
<td>+139176</td>
</tr>
<tr>
<td>CaM Naive sims</td>
<td>6822 (789)</td>
<td>27383 (1314)</td>
<td>111247 (4348)</td>
</tr>
<tr>
<td>Difference</td>
<td>+19155</td>
<td>+78938</td>
<td>+161357</td>
</tr>
<tr>
<td>RBP Naive sims</td>
<td>2183 (177)</td>
<td>8729 (803)</td>
<td>32125 (1024)</td>
</tr>
<tr>
<td>Difference</td>
<td>+10895</td>
<td>+41964</td>
<td>+132463</td>
</tr>
</tbody>
</table>

5.3 Improved conformational space coverage

The results from the previous section say nothing about the diversity of the conformations produced. We now show that SIMS 2.0 does not simply produce more
conformations that are all similar with respect to $C_{\alpha}$ RMSD. In this set of experiments, we fix the number of cores to 16 and assess how well SIMS 2.0 covers the conformational space.

### 5.3.1 Nearest Neighbor Distances

We first measure the closeness of the conformations from each other. This is done by tracking the distance of each conformation to its nearest neighbor. This is a measure of how “spread out” the conformations are. From Fig. 5.5 we see that SIMS 2.0 produces GroEL subunit conformations that are farther apart from each other. Fig. 5.6 shows similar results for the other three proteins.

![Figure 5.5](image)

**Figure 5.5**: Density of nearest neighbor distances (averaged over ten runs) for the GroEL subunit. SIMS 2.0 produces more conformations that are farther apart from each other.
Figure 5.6: Density of nearest neighbor distances (averaged over ten runs) for CVN (top), CaM (middle), and RBP (bottom). SIMS 2.0 produces more conformations that are farther apart from each other.
5.3.2 Expansiveness

We now measure the expansiveness of the conformational search. A more expansive search means that farther parts of the conformational space are sampled given the same starting point. Fig. 5.7 shows a density plot of the distances of each conformation from the start conformation for the GroEL subunit. In addition to producing more conformations, SIMS 2.0 also produces more conformations that are farther from the start. Thus, SIMS 2.0 produces a more expansive search. Fig. 5.8 shows similar results for the other three proteins.

Figure 5.7: Density of distances to start conformation (average over ten runs) for the GroEL subunit. SIMS 2.0 produces more conformations that are farther from the start conformation.
Figure 5.8: Density of distances to start conformation (averaged over ten runs) for CVN (top), CaM (middle), and RBP (bottom). SIMS 2.0 produces more conformations that are farther from the start conformation.
5.3.3 Isolating each improvement

These results taken together show that SIMS 2.0 produces a more efficient conformational search. We are now able to produce more diverse, low-energy conformations compared to the previous versions of SIMS. We now end this section by investigating which specific improvement contributed most to the efficiency.

We first ran an experiment with SIMS 2.0, where the global coverage grid did not send summarized coverage estimates when a SIMS run restarted. This run is similar to “Naive SIMS ” except with the improved projection and schema. Fig. 5.9 shows the effect this had on expansiveness. It appears that since no synchronization was occurring, the search was not as expansive likely due to the increased amount of repeated work done amongst the cores.
Figure 5.9: Density of distances to start conformation (average over ten runs) for the GroEL subunit. *woSync* (blue) corresponds to the runs where the synchronization was turned off. Since the distribution shifted to the left, the search was less expansive than *sims 2.0*. 
Next we focus on the projection improvement. We ran two additional experiments. The first is SIMS 2.0 using a random projection \((randomProj)\) [15]. The second is SIMS 2.0 using a projection constructed from secondary structure as mentioned in [48] \((ssProj)\). The results in Fig. 5.10 imply that the projection definition is not vital to the search. The search using a random projection only appears to be marginally worse than one from SIMS 2.0. Additionally, the projection using secondary structure information doesn’t provide much improvement over the runs using a random projection.

Figure 5.10: Density of distances to start conformation (average over ten runs) for the GroEL subunit. \textit{randomProj} (green) corresponds to the runs where a random projection was used. \textit{ssProj} (purple) corresponds to the runs where secondary structure information was used to construct the projection. Since the distributions barely shifted, the definition of the projection doesn’t appear to have much impact on the expansiveness of the search.
Finally, we focus on the schema improvement. We ran an experiment with Sims 2.0 using a schema defined using only secondary structure information. The new schema appears to contribute the greatest since the density curve with a *Naive schema* is most similar to the one corresponding to the naive version of Sims even though the other improvements are included. The density curve with a *Naive schema* also implies that the improvement in the number of conformations produced was also due to the new schema, since the height of the density curve of *Naive schema* is lower than the one from Sims 2.0. This demonstrates how important the schema is to the search because it essentially encodes how Sims 2.0 *explores* the space.

![Figure 5.11](image)

Figure 5.11: Density of distances to start conformation (average over ten runs) for the GroEL subunit. The new schema appears to have the greatest impact since the run that uses a schema with only secondary structure information (*Naive Schema*, red) has a density curve that is most similar to the original version of Sims (*old*, gray).
Chapter 6

Conclusion and Future Directions

This thesis described SIMS 2.0, a new framework for conformational sampling that offers scalable conformational sampling for large proteins. Our contributions enabled us to do conformational sampling on proteins up to 500 residues in size with molecular level detail. This was done through two main directions.

First, we presented a new default perturbation strategy in SIMS that can automatically generate a projection and a schema that includes important flexibility information derived from B-factors, secondary structure, and tertiary structure. Larger proteins are more constrained, so information is needed to prioritize which degrees of freedom are more important. We showed that this new model of flexibility allowed SIMS 2.0 to increase the number of conformations produced within a given time as well as create a more diverse distribution of samples.

Second, we introduced an algorithm that can take advantage of large-scale computational resources while still keeping the geometric reasoning that robotics-inspired algorithms excel at. The algorithm assigns every available core an instance of SIMS to perform its own conformational sampling. Then, the algorithm maintains a global coverage grid, whose scope reaches across every instance of SIMS. Global grid cell statistics are saved into a central database along with conformations sampled from each SIMS run. When a new SIMS run is started, a subset of conformations is loaded along with “summarized” coverage statistics about the global coverage grid. We showed that this approach is vital for use on multicore systems due to its scalability.
and maintenance of a global coverage grid.

For future work, we can begin to investigate new ways to keep track of coverage. Our experiments showed that the projection definition doesn’t greatly affect the expansiveness of the search. One can immediately look to non-linear projections but we can abstract this further and instead look for interesting distance metrics to be used.

Additionally, this thesis showed the most significant improvement in terms of expansiveness came from the new schema definition. The schema essentially defines how the search is traversing through the space so if the schema is defined with even more flexibility information, we can expect the overall search to produce a more expansive set of conformations. In future work, we can investigate how a dynamically changing schema would perform that perturbs conformations differently as a function of the currently chosen conformation. We can also investigate more complicated moves, such as the use of a brief molecular dynamics simulation, for expensive, yet intricate, perturbations.

The contributions of this work push the limits of the kinds of systems that can be studied by robotics-inspired conformational sampling. We also want to push our work to even larger systems, particularly symmetric systems like viruses. Viruses can undergo processes that involve massive conformational change, and SIMS 2.0 may be one of the only conformational sampling frameworks to study them. We can use Rosetta symmetry [66] to aid in the conformational search and future research would focus on how much approximation from symmetry can be leveraged to still produce meaningful results. We are hopeful that this work brings robotics-inspired conformational sampling closer to reaching their full potential.
Bibliography


[34] G. R. Bowman, X. Huang, and V. S. Pande, “Using generalized ensemble sim-
ulations and Markov state models to identify conformational states,” *Methods*,

[35] T. Maximova, R. Moffatt, B. Ma, R. Nussinov, and A. Shehu, “Principles and
Overview of Sampling Methods for Modeling Macromolecular Structure and

maps and structural excursions of proteins,” *BMC Genomics*, vol. 17, no. 4,

in continuous cost spaces,” in *2008 IEEE/RSJ International Conference on

[38] I. Al-Bluwi, M. Vaisset, T. Siméon, and J. Cortés, “Modeling protein confor-
mational transitions by a combination of coarse-grained normal mode analysis and

folding landscapes and analyze folding kinetics of known native structures,” *J.


[41] L. Tapia, S. Thomas, and N. M. Amato, “A motion planning approach to studying
molecular motions,” *Communications in Information and Systems*, vol. 10, no. 1,


[55] D. Devaurs, T. Simeon, and J. Cortes, “Parallelizing RRT on large-scale


