TECHNIQUES FOR IMPROVED PROBABILISTIC INFERENCE IN PROTEIN-STRUCTURE DETERMINATION VIA X-RAY CRYSTALLOGRAPHY

by

Ameet Bharat Soni

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2011

To Karen, Bharat, and Sharmistha.

Acknowledgments

Borrowing from an old African proverb states which states, “It takes a village to raise a child,” I believe it takes a village to raise a PhD dissertation. This document is not only a culmination of my graduate career, but also the result of the support and training I have received from those around me.

First, I would like to thank my advisor, Jude Shavlik, for being a supportive mentor and teacher during my graduate career. Not only has he imparted a great deal of knowledge upon me concerning the fields of machine learning and computational biology, but he has also trained me to be an effective communicator and practitioner of science in general.

I would also like to acknowledge my committee members: George Phillips, Mark Craven, David Page, and Vikas Singh. Meeting and discussing my work with George has taught me the importance of interdisciplinary research – not only as an application for computational work, but as a framework for learning new perspectives and approaches to difficult problems. Much of what I have learned in my field has come from courses taken with Mark and David. Both have been encouraging of my interests in becoming a teacher and have provided, along with Vikas, great feedback on my research as well as career development. In addition, it is important to recognize that all of my committee members have always been friendly and encouraging, going out of their way to welcome me to their homes or to have a friendly conversation during tea time.

Without my many collaborators, I probably would not have anything to say after Chapter 2. First and foremost, I must thank Frank DiMaio for taking me under his wings as a young graduate student and guiding me through the initial steps of performing graduate research. Not only did our many conversations further my understanding of the project, but his advice was very helpful for my graduate career. My discussions with members of the Center for Eukaryotic Structural Genomics – including Craig Bingman, Sethe Burgie, and Dmitry Kondrashov – always left me learning something new and exciting about biochemistry. Sriraam Natarajan always amazed me with his ability to grasp a concept instantly, and I truly thank him for his friendship and support. In addition, I would like to thank other collaborators on Acmi: Stuart Ballard, Tuo Wang, and Siddharth Puthur. I am also thankful for all of the great people who I have come to know in the machine learning group here at the University of Wisconsin.

I acknowledge the importance of my funding and training at the University of

Wisconsin. My research has been supported by NLM training grant T15-LM007359, and NLM grant R01- LM008796. In addition, support for my collaborators at the University of Wisconsin Center for Eukaryotic Structural Genomics (CESG) has been provided by NIH Protein Structure Initiative Grant GM074901. I am truly thankful to CIBM, and in particular Louise Pape, for all of their support and the opportunity to train in the field of computational biology.

Lastly, I would like to acknowledge my family. These few sentences do not do justice to what you have provided me. To Bharat, Sharmistha, Asha, and Ankoor, thank you for always being there with love and support. Thank you to all of the additions to my family: Dale, Mary, Laura, and Emily. Thank you to my loving wife, Karen, for being my best friend, and my constant through the ups and downs of my graduate career.

The contributions of my research are part of the Acmi (Automated Crystallographic Map Interpretation) computer package. Acmi and the data set of experimentally phased density maps used in my experiments are available online at http://pages.cs.wisc.edu/∼dimaio/acmi/get_acmi.htm.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Å	Angstroms
AA	amino acid
ACMI	Automated Crystallographic Map Interpretation
ACMI-BP	Acmi's Belief Propagation inference algorithm
ACMI-FF	Acmi's Fast Fourier local match algorithm
ACMI-PF	Acmi's Particle Filter sampling algorithm
ACMI-SH	Acmi's Spherical Harmonic local match algorithm
ARP	Automated Refinement Procedure
ASU	asymmetric unit
BLAST	Basic Local Alignment Search Tool
BP	(loopy) belief propagation
CESG	Center for Eukaryotic Structural Genomics at the University of Wisconsin-Madison
DSSP	Dictionary of Protein Secondary Structure
EDM	electron-density map
FFT	fast Fourier transform
MCMC	Markov Chain Monte Carlo
MRF	Markov random field
PDB	Protein Data Bank
PEA	Probabilistic Ensembles in Acmi
RBP	Residual Belief Propagation
R_{free}	free residual factor
RMSE	root-mean-squared error

SHED	Structural Homology using Electron Density
SIR	Statistical Importance Resampling
SVM	Support Vector Machine

Glossary

alpha carbon (Cα) – the central atom in an amino acid structure, connecting the backbone to the side chain.

amino acid – the building blocks of protein structures, 20 varieties in all. Each amino-acid type contains the same backbone atoms, but a unique set of side-chain atoms.

amino acid sequence or **sequence –** the linear chain of amino acids forming a protein structure. The sequence is typically represented as a string of letters, with each letter representing one of twenty amino-acid types.

asymmetric unit – a partition of the unit cell related to all other partitions by a symmetry operation.

backbone – the set of atoms linking one amino acid to the next in a protein structure. Each amino acid's backbone consists of an amino group on one end and a carboxylic group on the other with the alpha carbon (Cα) connecting the two groups together. The side chain, not part of the backbone, is also connected to the Cα.

completeness – akin to *recall*; of all possible events to predict (e.g., amino-acid locations in a PDB solution), the percentage that were predicted accurately (e.g., within 2 Å).

conditional probability – $P(A|B)$; the probability of event A given event B has occurred.

correctness – akin to *precision*; of all predictions by a model (e.g., Acmi-produced amino-acid locations), the percentage that were predicted accurately (e.g., within 2 Å).

domain knowledge – knowledge about the environment of a task or entire discipline; knowledge available to experts in a field.

electron-density map or **density map** or **map –** a three-dimensional image of a molecule resulting from X-ray crystallography.

homology – a similarity between two sequences or structures implying common ancestry.

interpreting or **tracing** a density map **–** placing atoms in an electron-density map to determine the protein structure that produced the image.

joint probability – the probability of multiple events occurring, $P(A, B)$.

likelihood – the probability value associated with a particular outcome. In this document, the likelihood represents the probability a model assigns to the correct solution.

marginal probability – the probability of some event, B, when ignoring other events, A, in a model; this calculation is obtained by summing over all possible outcomes of A in a joint distribution.

Markov random field (MRF) – a type of undirected graphical model; models the full-joint probability of a set of random variables as a product of potential functions associated with cliques (connected subgraphs) in the graph. A pairwise MRF contains potential functions with no more than two random variables; that is, potential functions are associated with either a vertex or edge.

peptide bond – the chemical bond formed between two amino acids, joining the carboxylic group of one to the amino group of the other.

percentile rank – the percentage of solutions below the true solution in a descending list of sorted probabilities.

phases – the angular portion of reflection data. The **phase problem** arises because diffraction data only measures the intensities of reflections; additional experiments are needed to estimate phases, introducing error into the electron-density calculation.

posterior probability – the probability of a variable (or state) after being given evidence or knowledge of the state of other variables in a model.

primary structure – the linear chain of amino acids forming a protein molecule.

prior probability – the probability of a variable (or state) before any evidence or knowledge is obtained.

probabilistic graphical model – a probability model defined on a graph where vertices represent random variables and edges represent relationships between variables.

probabilistic inference – the process of computing posterior probabilities for a random variable when given evidence in a probabilistic model; algorithms for answering queries about hidden variables in a probabilistic model.

probability – a measure expressing belief or likelihood of an event occurring (or having occurred). Probabilities range from a value of 0 to 1, and the sum of all possible states of a random variable sum to 1.

protein or **polypeptide –** a polymer chain of amino acid molecules; these molecules play an essential role in almost all cellular functions of living organisms.

protein disorder – a local description of amino acids in a protein structure characterized as unstable, existing as an ensemble of widely diverging conformations, with no specific equilibrium state.

random variable – a variable whose state is a result of a random process.

reflection – the spot on a collection plate in X-ray crystallography resulting from the diffraction of an X-ray beam by a protein crystal. The reflection data is related to the electron-density map via a Fourier transform, but only the magnitude of the complex-valued reflection are collected, meaning phases must be estimated.

residue – an amino acid after it has lost a water molecule when linking with another amino acid. For this document, the terms amino acid and residue are used interchangeably.

resolution – in X-ray crystallography, resolution is the highest resolvable peak in the reflection data; more generally, a measure of the resolvability of molecules in an electron-density image. Resolution is measured in Angstroms (Å), with higher values indicating poorer resolution.

secondary structure – one of a set of commonly occurring, three-dimensional structural motifs in a protein structure.

spherical harmonics – the angular portion of a set of solution's to Laplace's equation. That is; a set of orthogonal basis functions in spherical coordinates, used to describe a three-dimensional object in this dissertation.

tertiary structure – a protein's full, three-dimensional structure.

unit cell – basic repeating unit of a protein crystal; a density map is usually one unit cell, representing the average image of approximately a trillion unit cells in the crystal.

X-ray crystallography – a molecular imaging technique where X-ray beams are shot through a crystallized molecule, creating a diffraction pattern; the most common technique for determining protein structures.

Abstract

Over the past decade, the field of machine learning has seen a large increase in the use and study of probabilistic graphical models due to their ability to provide a compact representation of complex, multidimensional problems. Graphical models have applications in many areas, including natural language processing, computer vision, gene regulatory-network modeling, and medical diagnosis. Recently, the complexity of problems posed in many domains has stressed the ability of algorithms to reason in graphical models. New techniques for *inference* are essential to meet the demands of these problems in an efficient and accurate manner.

One such area of application is in the area of structural genomics. The task of determining protein structures has been a central one to the biological community, with recent years seeing significant investments in structural-genomic initiatives. X-ray crystallography, a molecular-imaging technique, is at the core of many of these initiatives as it is the most popular method for determining protein structures. In creating a high-throughput crystallography pipeline, however, the final step of constructing an all-atom protein model from an *electron-density map* – a three-dimensional image of a molecule produced as an intermediate product of X-ray crystallography – remains a major bottleneck in need of computational methods. In difficult cases where the image is poor, this can take months of manual effort by an experienced crystallographer.

In this thesis, I develop new inference techniques for the use of probabilistic graphical models for the automated determination of protein structures in electron-density maps. The first, guided belief propagation using domain knowledge, prioritizes messages in the popular belief propagation algorithm for approximate inference. Second, I propose Probabilistic Ensembles in Acmi (Pea), a framework for leveraging multiple, diverse executions of approximate inference to produce more accurate estimations of a variable's posterior probability distribution. Lastly, I present work on the use of statistical sampling (*particle filtering*) for the purpose of providing physically feasible, all-atom protein structures.

I demonstrate that my new methods not only improve the accuracy of the probabilistic model in terms of log-likelihood values, but also produce protein structures with higher completeness, lower RMS error, and better fit to the density map according to R_{Free} factor. My methods interpret difficult electron-density maps (3-4 Å resolution) better than prior inference approaches. Across a set of poor-quality density maps, my work outperforms all related work in the field by improving the

state-of-the-art technique, Acmi. In addition, I show that the ability to incorporate biochemical domain knowledge is an important aspect to probabilistic modeling, creating more accurate modeling functions and influencing algorithmic design of belief propagation.

I also describe my contributions on the subtask of three-dimensional shape matching in electron-density maps by utilizing spherical-harmonic decompositions to quickly align two 3D objects over rotations. I show that spherical-harmonic decompositions, when applied to the task of matching small amino-acid fragments, are more efficient and accurate than previous work. I also extend spherical harmonics to two other shape-detection tasks: homologous structure detection in electron-density maps and feature generation for 3D shape classification of local density regions.

While the application of my work specifically targets the problem of protein-structure determination, the issues I pose generalize to computational problems seen in many areas of the field of artificial intelligence. Throughout this work, I will refer to, and develop, techniques to solve problems seen in probabilistic inference, three-dimensional shape matching, and statistical sampling, among others.

1 Introduction

The task of determining protein structures has been a central one to the biological community for several decades. Given the structure of a protein, biologists are able to obtain insight into several important properties of the protein. In particular, researchers can obtain information about the protein's chemical interactions and mechanisms of action, which aid in understanding underlying biology. Determining a protein's structure helps researchers' understanding of protein structure-function relationships as well as aiding in a wide set of applications, including disease treatment, drug design, industrial catalyst design, and protein design.

In an effort to increase the number of known protein structures, recent years have seen significant investments in structural-genomic initiatives [81]. X-ray crystallography, a molecular-imaging technique, is at the core of many of these initiatives as it is the most popular method for determining protein structures. In creating a high-throughput crystallography pipeline, however, the final step of constructing an all-atom protein model from an *electron-density map* – a three-dimensional image of a molecule produced as an intermediate product of X-ray crystallography – remains a major bottleneck in need of automation. In difficult cases, this can take months of manual effort by a crystallographer.

My thesis concerns developing computational methods for automating this task. In particular, I discuss the use of a probabilistic framework for combining visual feature recognition in the image with known biochemical constraints in terms of the dynamics of molecular structures. In the pursuit of a protein-structure determination method, I address many algorithmic challenges, including the problem of performing approximate probabilistic inference in models with complex interactions.

The methods I propose both rely on and make contributions to many computational fields, including machine learning, computer vision, and signal processing. The result is a combination of methods that forms the current state-of-the-art approach for determining protein structures in low-quality electron-density maps. These methods also contribute general algorithmic techniques for improved probabilistic inference, three-dimensional shape matching, and statistical sampling.

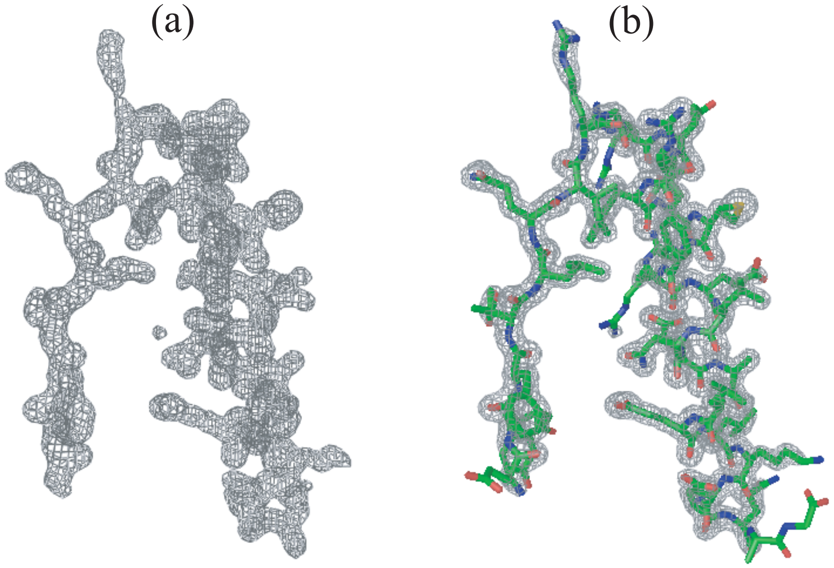


Figure 1.1: The last step in the protein X-ray crystallography pipeline takes a) the electron-density map (a 3D image) of the protein and finds b) the most likely protein structure that explains the map. Here, the electron density is contoured and the chemical structure of the protein is designated with a stick model showing all of the non-hydrogen atoms.

1.1 Protein-Structure Determination in Electron-Density Maps

As this document lays out, the process of elucidating a protein structure remains a laborious task filled with potential for automated methods. Of interest to my work is X-ray crystallography, which accounts for 88% of structures in the Protein Data Bank (PDB) [72]. The last step in the crystallography process, producing a protein structure from the electron-density map, remains a resource-intensive and time-consuming task and is a major hurdle in creating a high-throughput pipeline for determining protein structures. My work seeks to create and build upon automated methods for determining protein structures from electron-density maps. Specifically, the main objective of my work is:

> Given both the electron-density map (a three-dimensional image) and a primary sequence of a target protein, computationally produce a three-dimensional, physically feasible, all-atom model of the target protein's structure.

Figure 1.1 depicts this task pictorially. Figure 1.1a shows a contoured electron-density map, similar to what a crystallographer would see at the beginning of interpretation. In b) we see the resulting protein structure with all non-hydrogen atoms in a stick representation.

The process of determining protein structures via X-ray crystallography remains challenging for several reasons, as I discuss in Section 2.1.2. The key challenge addressed in this document is the *resolution* of the density map. Resolution is a measure of image quality, with poorer resolutions producing maps where individual atoms are no longer visible. Many proteins have difficulty crystallizing, yielding poor crystal formations which produce low resolution images. While advances in image technology may help improve quality, resolution is mainly a property of the protein itself. In addition to resolution, *phasing error* (see Section 2.1.2) as well as the inherent flexibility of proteins further reduces the interpretability of density maps.

This thesis introduces probabilistic techniques for automatically determining protein structures from electron-density maps. My techniques specifically address poor-quality maps, utilizing probabilistic models and biochemical domain knowledge to determine protein structures in low-resolution maps where all other methods fail.

1.2 Probabilistic Reasoning

Over the past decade, the field of machine learning has seen a large increase in the use and study of probabilistic graphical models such as Bayesian networks and Markov random fields [6, 33]. These models represent the uncertainty inherent in data and event outcomes. Their popularity owes to the ability of these models to provide a compact representation of complex, multidimensional problems and handle noisy, incomplete (e.g., missing values, hidden variables) data. Graphical models have applications in many areas, including natural language processing [54], computer vision [29], gene regulatory network modeling [31], and medical diagnosis [38].

An important problem in probabilistic modeling is *inference*, or the ability to reason probabilistically about the value of certain outcomes given evidence of the states of other variables. Recently, the complexity of problems has exceeded the ability of algorithms to reason in graphical models. As this complexity continues to increase, an important task for the field of machine learning is to develop more efficient and accurate approximate-inference methods.

My thesis work on automatic density-map interpretation is a good example of a complex problem requiring probabilistic graphical models to provide a solution. The complexity of this task serves not only as a case study in effective probabilistic modeling, but provides a test bed for developing improved machine learning approaches for performing probabilistic inference.

My task poses problems of complexity for several reasons. First is the size of the probabilistic model. Each protein structure has thousands of atoms, meaning there are thousands of variables to model. In addition, the state space of each variable is the entire electron-density map, which contains on the order of 10^6 possible values – one for each (x, y, z) location in the map. A further complication is the globular nature of proteins, which creates potential pairwise interactions between almost every atom in the structure. In modeling terms, this means there are N^2 interactions to model, where N is the number of variables.

This work proposes several novel approaches to improved approximate probabilistic inference for such complex domains. One important contribution is the utilization of domain knowledge to inform existing approximate-inference methods. Using this expert knowledge exploits information about a model's domain that, prior to my work, could not be utilized to inform inference. My second central contribution is the use of ensemble-learning methods in approximate inference to mitigate the impact of assumptions introduced by model simplification, producing multiple, probabilistic perspectives to a complex solution. In this dissertation, I also describe several other, more secondary, contributions to the problem of approximate inference.

1.3 Thesis Statement

This dissertation investigates the following statement:

> *Using biochemical domain knowledge and enhanced algorithms for probabilistic inference will produce more accurate and more complete protein structures.*

Specifically, I first hypothesize that guiding belief propagation using biochemical domain knowledge will improve the quality of approximate probabilistic inference in terms of the accuracy of probability distributions as well as the accuracy of resulting protein structures. Second, I propose extending the concept of ensemble methods in supervised learning to the problem of approximate inference to leverage multiple, diverse runs of belief propagation. In addition to inference, I hypothesize that spherical-harmonic decomposition is a superior method for performing shape-matching tasks in protein-structure images, producing high-fidelity descriptions of small amino-acid fragments as well as whole-protein structures, efficiently. Lastly, I propose that the use of statistical sampling techniques will produce physically feasible, all-atom protein structures that explain the underlying electron-density map better than all existing automated density-map interpretation methods, across a set of difficult, poor-quality maps.

1.4 Thesis Organization

My thesis is organized as follows:

- **Chapter 2** provides the pertinent background material for my work. I begin with essential biochemical definitions and concepts for my application, including a description of protein structures, X-ray crystallography, and related techniques for automated protein-structure determination. I then describe the algorithmic background for my work, including a formal explanation of probabilistic graphical models and the task of performing inference. Other algorithmic techniques I overview include spherical-harmonic decompositions, particle filtering, and ensemble-learning methods.

- **Chapter 3** presents the Acmi (Automated Crystallographic Map Interpretation) framework. I begin with a description of prior work on the Acmi project by Frank DiMaio et al. Next, I provide an abstract description of Acmi's three-phase pipeline that provides a good reference for later discussions. This roadmap describes the role of each phase of Acmi as well as highlight my contributions within this framework. Lastly, I provide an analogy to my application by demonstrating how Acmi's pipeline would approach the related problem of face detection in images.

- **Chapter 4** describes the general experimental methodology of my work. First, I describe the test set of proteins used to evaluate my algorithms. Second, I present the various metrics used to evaluate protein-structure quality in our system, borrowing from both the information retrieval and crystallography communities. Lastly, I explain other metrics used throughout my work to evaluate probabilistic-inference methods.

- **Chapter 5** investigates the influence of a message-passing scheduler on belief propagation. Specifically, I propose a general method for guiding belief propagation using domain knowledge to assess the priority of messages and/or variables. I apply this technique to the inference process (i.e., Phase 2) of Acmi by utilizing a concept known as protein-disorder prediction to predict the stability of each amino acid. Results show this scheduling technique produces more accurate inference solutions in Acmi relative to a naive, round-robin algorithm as well as a proposed information-theoretic measure.

- **Chapter 6** reviews work on improving approximate-inference techniques in difficult domains such as as Acmi. I apply the concept of ensemble methods from the supervised machine learning literature to the problem of inference in graphical models. My proposed technique, Pea (Probabilistic Ensembles in Acmi), leverages multiple runs of approximate inference to provide a more complete picture of the underlying protein structure in the density map. I show that this ensemble method provides statistically significant improvements in inference quality as well as more complete and accurate protein structures.

- **Chapter 7** presents my research on shape matching in electron-density maps, where I employ spherical-harmonic decompositions to efficiently describe two three-dimensional objects and align them across all rotations. First, I describe a new Phase 1 method, Acmi-SH, which generates more accurate protein structures than the previous technique. Second, I explore the training of a "first-pass filter" to further reduce the search burden of Phase 1, *a priori* eliminating large portions of the map from consideration for having protein structure. Lastly, I use spherical-harmonic decompositions to develop Shed, an algorithm that detects homologous structures to an unsolved protein's electron-density map with greater accuracy than sequence-only methods, such as Blast [2].

- **Chapter 8** describes my work on producing all-atom proteins structures using a probabilistic sampling method known as particle filtering. With Frank DiMaio, I develop Acmi-PF (Phase 3 of Acmi), which samples both backbone and side-chain atoms to create a physically feasible protein structure. Results show Acmi is the state-of-the-art method for determining structures in difficult electron-density maps. In addition, I explore the idea of incorporating domain knowledge – or well-described biochemical concepts – into Acmi-PF's probabilistic framework by informing backbone-atom sampling using secondary-structure information.

- **Chapter 9** is the conclusion of my thesis. I review my contributions and outline directions of future research.

2 Background

This chapter presents the biochemical and computational background behind my work. I begin with a description of protein structures, including the most popular technique for determining structures, X-ray crystallography. I also present three popular algorithms for automated density-map interpretation with pseudocode and high-level overviews. The second half of this chapter concentrates on the algorithmic concepts underpinning my methods and contributions. This includes a formal presentation of probabilistic reasoning in graphical models and a discussion of the important problem of statistical inference. I also describe techniques used to solve subtasks in the Acmi framework, including spherical-harmonic decompositions for 3D-object recognition, particle filtering for sequential variable sampling, and ensemble-learning methods.

2.1 Biochemical Background

This section provides an overview of biochemical concepts and terms used throughout my work. Much of these descriptions appear in joint work with Frank DiMaio and Jude Shavlik [24].

2.1.1 Protein Structures

Proteins (also known as *polypeptides*) are polymer chains that play an essential role in almost all functions of cells in living organisms. These include enzymatic roles in catalyzing reactions, cell signaling, and structural support. Proteins are encoded in an organism's genome, such that a sequence of a protein is determined (for the most part) by the sequence of a gene. Proteins form a linear chain, or *sequence*, based on a set of building blocks known as *amino acids*. There are twenty natural occurring amino acids, each having the same backbone structure in addition to a unique side-chain group. Figure 2.1 shows the chemical layout of an amino acid. The *backbone* of each amino acid consists of an amino group on one end, a carboxylic group on another, and a central carbon atom, known as the *alpha carbon* (Cα for short) connecting the two groups[1]. An amino acid's *side chain*, the only portion that varies from one amino acid to another, is also connected to the Cα. The side chain of each

[1] The Cα is chemically identical to other carbon atoms, but plays a special role by linking the major components of an amino acid together.

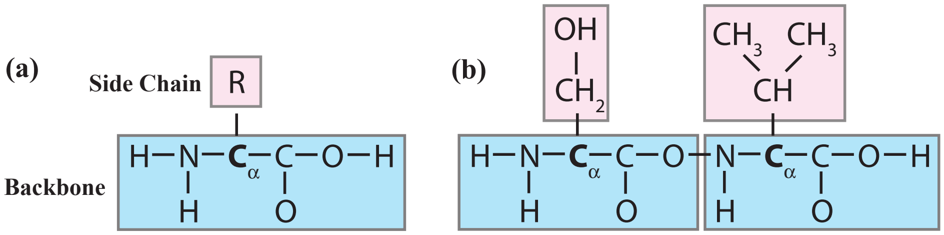


Figure 2.1: Chemical structure of amino acids. A single, generic amino acid is shown in a). **R** denotes the side-chain atoms – unique to each amino-acid type. The backbone atoms are shown in the bottom (blue) box. In b) I show two specific amino-acid residues (serine and valine) linked by a *peptide bond*.

amino-acid type varies in length, polarity, and charge. These properties influence an amino acid's interactions with solutions and other molecules and account for the variation in structure from one protein to the next.

To form a protein, adjacent amino acids condense via the amino group of one and the carboxylic group of the other. The bond is called a *peptide bond* and the reaction produces a water molecule in addition to the linked amino acids. After this reaction, amino acids are often referred to as *residues* (short for amino-acid residues). This pattern repeats, forming a linear polymer chain with side chains dangling off the backbone. The linear combination of amino acids in this matter results in the protein's *primary structure*.

The protein's primary structure often contains continuous segments that form commonly occurring three-dimensional structural motifs, known as the protein's *secondary structure*. Figure 2.2 shows the two most common motifs: α-helices and β-strands. An α-helix is a motif where a portion of the peptide chain folds into a corkscrew formation. The spiral pattern is stabilized by hydrogen bonds between non-adjacent residues. β-strands, also stabilized by hydrogen bonds, are stretched out segments of the peptide chain which line up with other segments in either a parallel or antiparallel fashion. *Random coils*, or loops, do not form regular formations like the above motifs but are used to describe the segments of the protein chain that link motifs together. Finally, *tertiary structure* refers to a protein's full three-dimensional conformation. With a few exceptions, a protein's primary sequence of amino acids determines the protein's tertiary structure.

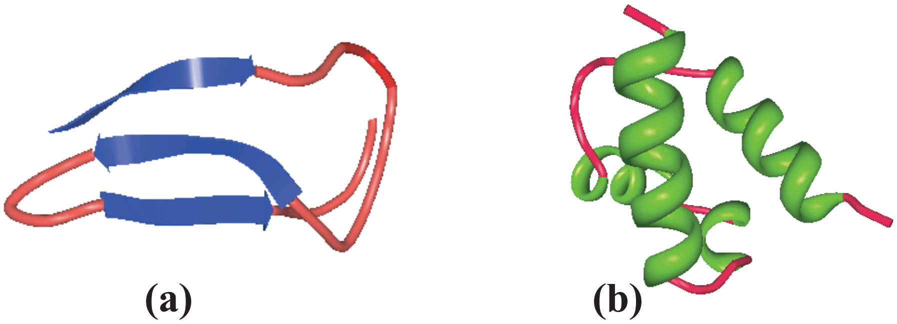


Figure 2.2: Ribbon representation of two sample proteins, showing secondary structure; a) consists of antiparallel β-strands (blue arrows), connected by random coils (red strings); b) shows a set of α-helices (green spirals) connected by random coils.

2.1.2 Protein X-Ray Crystallography

The task of *protein-structure prediction* often refers to determining the tertiary structure of a protein given a primary sequence. While researchers have made advances in developing fully automated methods for predicting a protein's 3D structure, no computer algorithm yet exists that can accurately map protein sequences into 3D structures. Instead, structures deposited into the Protein Data Bank (PDB) are determined by "wet lab" techniques for measuring a protein's structure. The vast majority of deposited structures, about 88% [72], have been produced using a technique known as *X-ray crystallography*. While X-ray crystallography remains the most popular method for determining large macromolecular structures, it is a time-consuming and resource-intensive process with many bottlenecks. This section describes how X-ray crystallography produces electron-density maps of protein structures and the difficulties present in creating a high-throughput structure-determination pipeline.

Figure 2.3 provides an overview of the protein crystallography process. The end goal is to produce an *electron-density map* (a three-dimensional image) of the target protein in high resolution such that an accurate protein structure can be determined from the image. The first, and most difficult, step for a crystallographer is to take a target protein and produce a suitable *protein crystal* containing a regular, repeating lattice of a purified form of the target protein. This process is difficult due to the large number of variables involved in successfully crystallizing the protein and the difficulty of maintaining a large crystal needed for large macromolecules such as proteins. Once a protein crystal forms, crystallographers expose the crystal to a beam of X-rays and collect the resulting diffracted beams on a collection plate. The

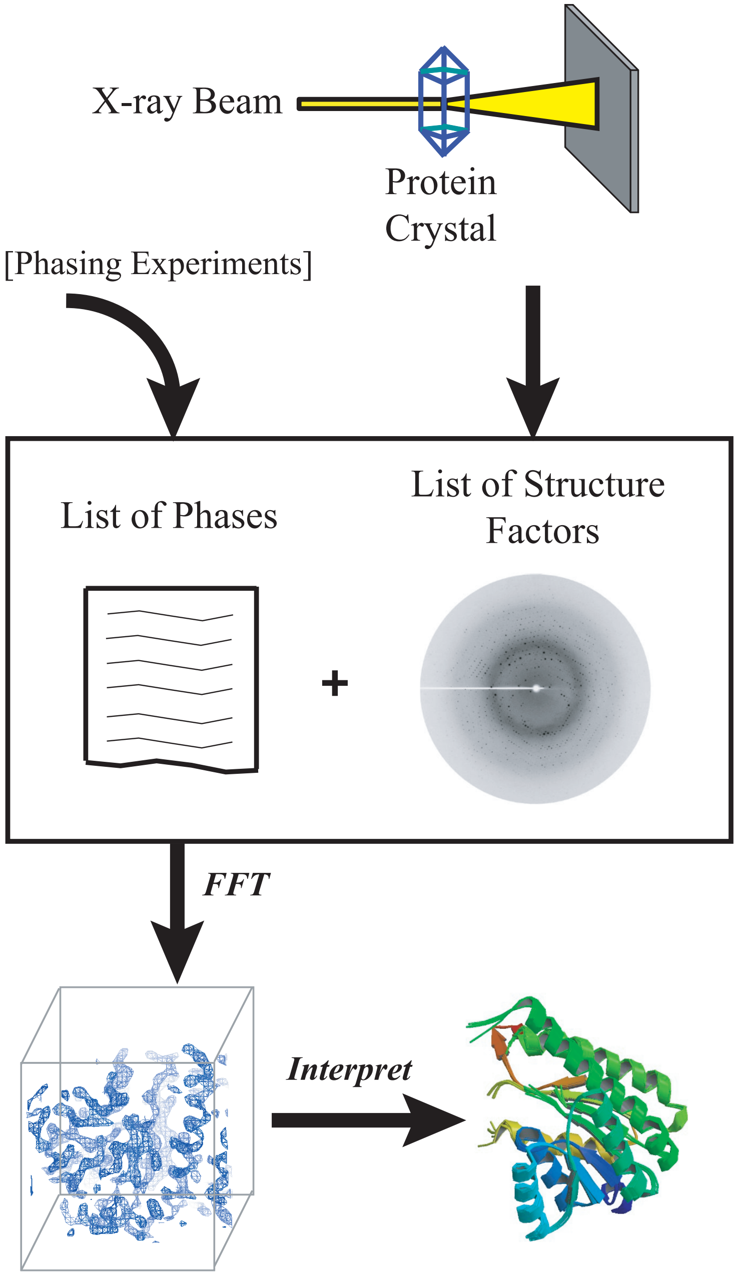


Figure 2.3: An overview of the X-ray crystallography pipeline, starting with the crystallization of the protein and ending with the interpretation of an electron-density map. Here, the brackets surrounding "Phasing Experiments" denote this information comes from additional experiments.

collection plate will record a pattern of spots known as *reflections* or *structure factors* which represent the crystal lattice in reciprocal space. The collected structure factors are related to the original atomic structure via a Fourier transform. Thus, a Fourier transform of the reflections can produce an electron-density map. Unfortunately, the structure factors represent only the intensities of the Fourier series – all phase information is lost. This is known as the *phase problem*. Many methods exist for making approximations of the true phases [74], but the process remains error prone. Once phases are estimated, a crystallographer can construct an electron-density map of the crystallized protein.

An electron-density map is a three-dimensional lattice of points, regular spaced covering the unit cell. In this document, I alternatively refer to the electron-density map as *density map* or *map*. The *unit cell* represents the basic repeating unit of the protein crystal. The unit cell may contain many copies of the target protein. Many of these copies may be related by crystallographic symmetry – one of 65 commonly occurring patterns for packing proteins in a crystal. Each crystallographic symmetry group describes a set of symmetry operators that relate one area of the unit cell to another via translation and/or rotation. The *asymmetric unit* refers to just one of the symmetric copies in a unit cell. In addition to symmetric protein copies, asymmetric units may contain multiple copies of a protein due to the target protein forming multimeric complexes such as dimers and trimers when placed in solution. A crystallographer's task is to determine the structure of just one copy of the protein, although this usually requires complete interpretation of all copies in the asymmetric unit to ensure quality.

The last step in the crystallographic process is *interpreting* the density map, whereby a crystallographer fits a protein molecular model to the density map. This phase is alternatively referred to as *tracing* the protein. Referring to the image back in Figure 1.1, Figure 1.1a is a contoured electron-density map, similar to what a crystallographer would see at the beginning of interpretation. In b) we see the resulting protein structure with all non-hydrogen atoms in a stick representation.

The crystallographer's task is: *given a protein's amino-acid sequence and an electron-density map of the protein, produce the underlying protein structure.* This end goal is the same as in automated *ab initio* protein-structure prediction, with the difference being that a crystallographer also possesses a fuzzy image of the protein structure. This last step of interpreting the density map represents the second bottleneck in creating a high-throughput molecular determination pipeline, and is the focus of my thesis work.

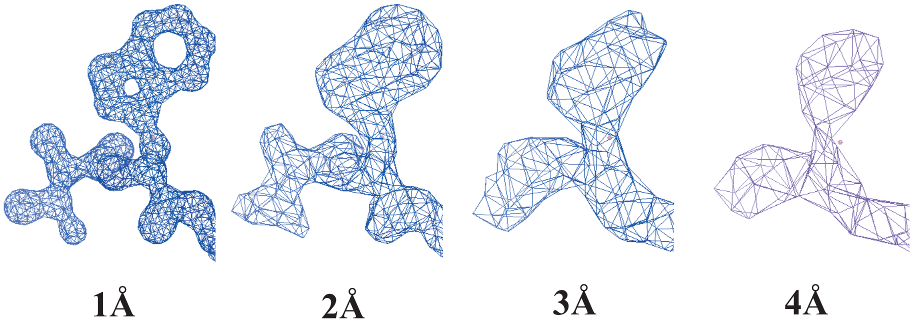


Figure 2.4: The electron density of tryptophan as resolution varies from 1 Å (high resolution) on the left to 4 Å (low resolution) on the right. As the resolution decreases, the characteristic double-ringed side chain of tryptophan becomes impossible to see and can be confused as part of the backbone.

Several factors make tracing the protein a difficult and time-consuming process, mainly by affecting the quality of the electron-density map. The most significant factor is crystallographic resolution which describes the highest spatial frequency terms used to assemble the electron-density map. The main determinant of electron-density resolution is the quality of the protein crystal. A crystal that produces narrow diffraction patterns will result in poor-quality density maps where much of the image is "smoothed" out due to the elimination of higher frequency information. Figure 2.4 demonstrates the effect of resolution when visualizing a portion of structure in an electron-density map. As the resolution worsens, we quickly lose evidence of tryptophan's distinguishing features. In the X-ray crystallography community, the resolution is measured in angstroms (Å), with higher values indicating poorer-quality maps. Thus, low-numbered values indicate a high-quality map and are described as having high resolution, while larger angstrom values represent poor-quality, low-resolution density maps.

The second factor in electron-density map quality relates to the previously discussed phase problem. Experimentally determined phases are often inaccurate, and make interpretation difficult by creating inaccuracies in the density map. As a crystallographer builds a model, phases are iteratively improved, thus improving the quality of the density map. However, this usually requires large portions of the density map to be traced first, which may be difficult if the phases are inaccurate and/or the map has low resolution.

Further factors affecting the ability to interpret a density map include the sheer number of atoms contained in a map as well as the quantity of noise present in the signal. In addition, the electron-density map is not an image of just one copy of the unit cell, but rather an *average* over all unit cells in the crystal lattice. Side chains with multiple possible conformations as well as disordered, loopy regions of the protein result in a blurry smear of density in certain portions of the map where structure is not even visible in the extreme case.

It should be noted that while technology and additional experiments can mitigate some of these complicating factors (e.g., the phase problem), the major determinant of the image quality is the protein itself. Large, active proteins are often difficult to stabilize out of solution and form imperfect crystals, if any at all.

When a protein crystallizes and produces a high-resolution density map, crystallographers traditionally perform the task of interpreting the density map with assistance of specialized graphical software suites such as O [44] and Coot [28]. However, poor-quality structures can be magnitudes slower for a crystallographer to interpret [48]. In some cases, determining the structure is not feasible or is shelved, due to limited resources, to work on easier targets. The remainder of this chapter will focus on methods which seek to automatically interpret electron-density maps with low resolution (greater than 2.3 Å) or poor experimental phases.

2.1.3 Related Work in Automated Density-Map Interpretation

Several research groups have developed automated methods for interpreting protein structures in electron-density maps. This section presents a high-level overview of some of these methods, describing algorithmically how each has approached this problem.

2.1.3.1 ARP/wARP

The most commonly used method for automatic density-map interpretation is ARP/wARP [65, 71]. The ARP/wARP (Automated Refinement Procedure) software suite is a crystallographic tool for the interpretation and refinement of electron-density maps. An overview of ARP/wARP is shown in Figure 2.5. Given a list of structure factors, phases, and the primary protein sequence, the central algorithm begins by creating a free atom model – a model containing only unconnected, unlabeled atoms – to fill in the density map of the protein. It then connects some of these atoms using a heuristic, creating a hybrid model. This hybrid model consists of a partially connected backbone, together with a set of unconstrained atoms. ARP/wARP then

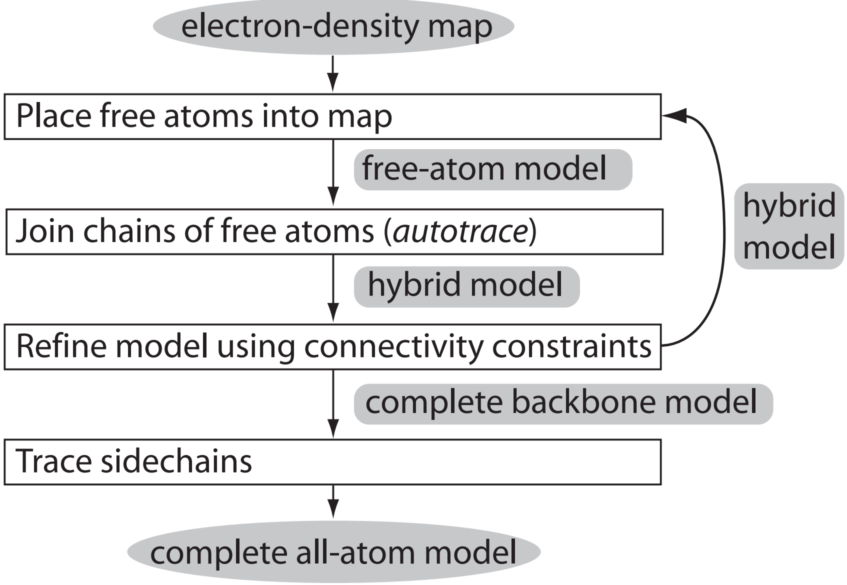


Figure 2.5: An outline of ARP/wARP, a method for tracing all-atom protein structures based on a bottom-up approach of placing free atoms and introducing constraints.

refines this model, producing a map with improved phase estimates. The process iterates using this improved map. At each iteration, ARP/wARP removes every connection, restarting with a free-atom model. A rotamer search is then applied to place side chains, followed by a loop-building method which fills in the best matching templates of five-consecutive Cα's from structures in the PDB [45].

ARP/wARP contains an atom placement method based on ARP, an interpretation method for general molecular models. ARP places unconnected atoms into the density map, producing a free atom model. To initialize the model, ARP begins with a small set of atoms in the density map. It slowly expands this model by looking for areas above a density threshold, at a bonding distance away from existing atoms. The density threshold is slowly lowered until ARP places the desired number of free atoms.

The next phase of ARP/wARP, wARPntrace, takes this free-atom model and adds connectivity to create a hybrid model of backbone atoms and free atoms. Given a free-atom model of a protein, one can form a crude backbone trace by looking for pairs of free atoms the proper distance apart. wARPntrace formalizes this procedure, called autotracing, using a heuristic method. wARPntrace assigns a score based on density values to each free atom. The highest scoring atom pairs 3.8 ± 0.5 Å apart become candidate Cα's. The algorithm verifies candidate pairs by overlaying them with a peptide template. If the template matches the map, wARPNTRACE saves the candidate pair. Candidate pairs are then linked based on matches to a database of known backbones. The longest built chain is added to the model, and the process repeats with the rest of the candidate Cα pairs being linked until no chains of length 5 remain.

Autotracing produces a hybrid model containing a set of connected chains together with a set of free atoms. Autotracing identifies some atom types and connectivity, which enables the use of stereochemical information in refinement. A modified version of ARP refines this hybrid model, and the process iterates from the beginning with a new free-atom model. Since the map is better-phased, autotracing produces a more complete model. This, in turn, provides a better refinement, improving the phases further.

This cycle continues for a fixed number of iterations, or until a complete trace is available. Finally, wARPNTRACE adds on side chains by identifying patterns of free atoms around Cα's. It aligns these free-atom patterns to the sequence to produce a complete model. To address issues with missing loops in models, ARP/wARP adds an optional step for their loop construction algorithm LOOPY [45]. Given an incomplete structure model, LOOPY will fill in gaps in the protein chain by extending Cα traces one residue at a time. To extend a trace, the new Cα is treated as the last in a pentapeptide fragment. Based on templates in the PDB that match this pentapeptide, LOOPY can predict the proper angle and distance of the missing Cα. This procedure is iterated, creating a tree of loop models which are pruned based on likelihood of fitting the model.

ARP/wARP is the preferred method for automatically interpreting electron-density maps, assuming sufficiently high-resolution data is available. Its placement of individual atoms, followed by atom-level refinement, produces an accurate trace with no user action required in 2.3 Å or better density maps. After recent improvements, the researchers claim success on structures up to 2.7 Å in resolution. Unfortunately, many protein crystals fail to produce maps of sufficient resolution, and ARP/wARP suffers due to the reduced number of observations available for refinement.

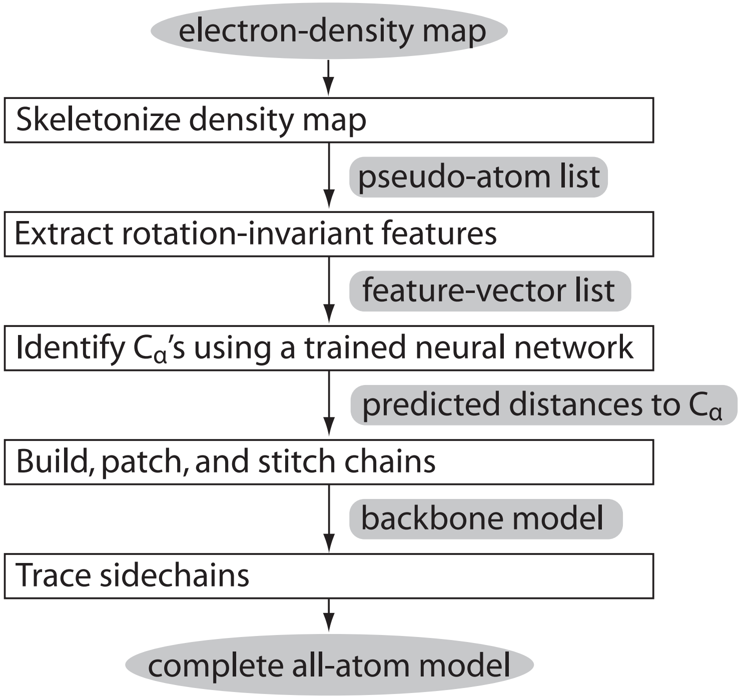

electron-density map
Skeletonize density map
pseudo-atom list
Extract rotation-invariant features
feature-vector list
Identify Cα's using a trained neural network
predicted distances to Cα
Build, patch, and stitch chains
backbone model
Trace sidechains
complete all-atom model

Figure 2.6: An outline of TEXTAL, a pattern-recognition based method which utilizes rotation-invariant features to train a neural network to predict backbone locations. In the last step, these features are also used to quickly match side-chain templates to the density map to create an all-atom model.

2.1.3.2 TEXTAL

TEXTAL [42] uses a four-step approach based on pattern-recognition techniques in order to interpret maps in the 2.2 to 3.0 Å resolution range. A flowchart of TEXTAL is shown in Figure 2.6. The first step, FINDMOL, identifies the region(s) of the map containing the protein target in order to limit the search space. Then CAPRA [41], a trained neural-network classifier, identifies Cα locations using a set of 19 rotation-invariant features from grid points in the density map. LOOKUP identifies side chains by comparing regions of density around each predicted Cα to a database of known side chains and places the best matching side-chain atoms in the trace. Finally, a set of heuristic methods are run to align the structure to the sequence and refine the structure.

The most important component of TEXTAL is its extraction of a set of numerical features from a region of density. These numerical features allow rapid identification of similar regions from different (solved) maps. A key aspect of TEXTAL's feature set is invariance to arbitrary rotations of the region's density. This eliminates the need for an expensive rotational search for each fragment. TEXTAL uses 76 such numerical features to describe a region of density in a map. These features include 19 rotationally invariant features, sampled at four different radii: 3, 4, 5 and 6 Å. The use of multiple radii is critical for differentiation among side chains: large residues often look similar at smaller radii but greatly differ at 6 Å, while small amino acids may have no density in the outer radii and thus are only differentiated at small radii. The 19 rotation-invariant features fall into four basic classes. The first class describes statistical properties of neighborhoods of density: mean, standard deviation, skewness, and kurtosis – the last two of which provide descriptions of the lopsidedness and peakedness of the distribution of density values. The second class of features is just a single feature: the distance from the center of mass to the center of the sample. A third class of descriptors includes moments of inertia (MOI), which provide six features describing how elliptical the density distribution is. Moments of inertia are calculated as the Eigenvectors of the inertia matrix. The final class of features represents higher-level geometrical descriptors of the region. Three "spokes of maximal density" are extended from the center of the region, spaced $> 75^\circ$ apart. These aim to approximate the direction of the backbone N-terminus, the backbone C-terminus, and the side chain. Rotation-invariant features derived from these spokes include the minimum, middle, maximum, and sum of the angles, the density sum along each spoke, and the area of the triangle formed by connecting the end points of the spokes.

After running a subroutine to identify the region of the map containing a molecule, CAPRA produces the initial Cα trace by using a feed-forward neural network. First, CAPRA skeletonizes the map, creating a trace of pseudo-atoms that identifies the medial axis of some density map contour. This trace is a crude approximation of the backbone, and may traverse the side chains or form multiple distinct chains. A feed-forward neural network – a nonlinear function approximator used for both classification and regression – is trained to learn which pseudo-atoms on the skeleton correspond to actual Cα's. Specifically, the network is trained on a set of previously solved maps to predict the distance of each pseudo-atom to the nearest Cα. The rotation-invariant features above are inputs to the network; a single output node estimates the distance to the closest Cα. A hidden layer of 20 sigmoidal units fully connects input and output layers. Given a predicted distance for each pseudo-atom, CAPRA performs a greedy trace to find a linear chain linking Cα's together to give us a backbone trace.

Textal must now identify the residue type associated with each Cα. This identification is performed by a subroutine Lookup. The subroutine compares the density around each Cα to a database of solved maps to identify the residue type. Lookup uses the rotation-invariant feature to build a database of feature vectors corresponding to side chain in solved maps. To determine the residue type of an unknown region of density, Lookup finds the nearest neighbors in the database using weighted Euclidean distance. Since information is lost when representing a region as a rotation-invariant feature vector, the nearest neighbor in the database does not always correspond to the best-matching region. Therefore, Lookup keeps the top k side chains and considers these for a more time-consuming correlation computation. It then quickly approximates the optimal rotation and translation for each side chain by aligning the moments of inertia between the template density region and target density region. Lookup computes the real-space correlation at each alignment, and selects the highest-correlated candidate. Finally, Lookup retrieves the translated and rotated coordinate atoms of the top-scoring candidate and places them in the model.

Textal's final step is improving the model using a few simple post-processing heuristics. First, Lookup often reverses the backbone direction of a residue; Textal's post-processing makes sure that all chains are oriented in a consistent direction. Refinement, as in ARP/wARP, corrects improper bond lengths and bond angles, iteratively moving individual atoms to fit the density map better. Finally, Textal takes into account the target protein's sequence to fix mismatched residues.

2.1.3.3 Resolve

While the two previous methods use a bottom-up approach, Resolve [82] uses a top-down procedure in which secondary-structure elements are located in the map with the best model being chosen for refinement and extension. An outline of the procedure is provided in Figure 2.7. Given an electron-density map, Resolve begins its interpretation by searching all translations and rotations in the map for a model 6-residue α-helix and a model 4-residue β-strand. Resolve constructs these fragments by aligning a collection of helices (or strands) from solved structures; it computes the electron density for each at 3 Å resolution, and averages the density across all examples. Given these model fragments, Resolve considers placing them at each position in the map. At each position, it considers a set of rotations and computes a standardized squared-density difference between the fragment's electron density

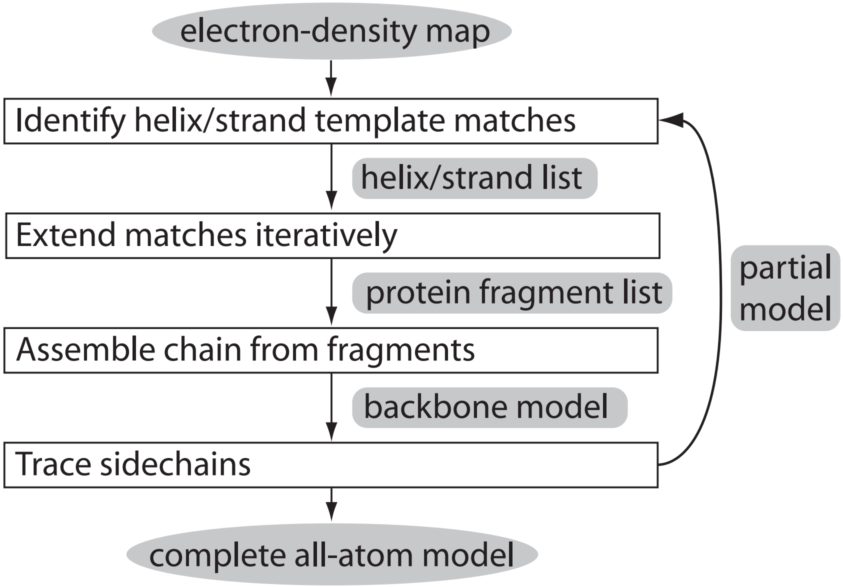


Figure 2.7: An outline of RESOLVE, which uses templates representing common secondary-structure elements – helices and strands – to identify putative locations of secondary structure in the map and create an initial list of backbone atoms. RESOLVE produces an all-atom protein model by extending and combining identified chains and finally adding side chains based on a rotamer library.

and the map. This is the same function used by ACMI-FF in Equation 3.1. Each fragment is refined, and then kept if it exceeds a threshold for correlation to the map.

At this point, RESOLVE has a set of putative helix and strand locations in the density map. The next phase of the algorithm extends these using a much larger library of fragments. Specifically, RESOLVE makes use of four such libraries for fragment extension: one for extending helices, another for extending strands, and two for extending backbones using tripeptide fragments based on the direction of chain extension. RESOLVE first attempts to extend using either of the first two libraries. Once no extensions exist above a certain correlation factor to the map, the last two libraries are utilized to create loops between the secondary-structure fragments. This is similar to the LOOPY method from Section 2.1.3.1.

Given this set of candidate model segments, RESOLVE's next step is to assemble a continuous chain. To do so, it uses an iterative method. The outermost loop repeats until no more candidate segments remain. At each iteration, the algorithm chooses the top-scoring candidate segment not overlapping any others. It considers all other segments in the model as extensions: if at least two Cα's between the candidate and extension overlap, Resolve accepts the extension. Finally, the extension becomes the current candidate chain.

Resolve's final step is, given a set of Cα positions in some density map, to identify the corresponding residue type, and to trace all the side-chain atoms. Resolve's side-chain tracing uses a probabilistic method, finding the most likely layout conditioned on the input sequence. Resolve's side-chain tracing relies on a rotamer library. This library consists of a set of low-energy conformations that characterizes each amino-acid type. A cross-correlation score is calculated for rotamer at each Cα to the map. These scores are converted to probabilities and then run through a Bayesian calculation for providing the sequence-to-chain alignment that best matches the rotamer correlation-coefficient calculations to the chain. Resolve has successfully interpreted density maps from 1.1 to 3.2 Å in quality. More recently, Resolve has been incorporated into the Phenix software package, forming the core of the Autobuild procedure [1].

2.1.3.4 Buccaneer

A recent method, Buccaneer [13] takes a similar approach as Textal by first finding likely Cα locations in the electron-density map and then extending them into a chain. The main difference is that while Textal utilizes rotation-invariant features to infer Cα positions, Buccaneer utilizes orientation-based features in order to not only infer Cα positions, but the likely orientation of the backbone at each position. First, a set of highly probably "seed" Cα positions are chosen using a density-dependent likelihood function. Next, these Cα are extended into chain fragments under angular constraints by optimizing the likelihood of a fragment given the local density. Lastly, Buccaneer joins consistent chain fragments by searching for the longest chain path possible among the fragments and then prunes inconsistent fragments from shorter chain fragments. Buccaneer currently only performs a backbone trace, and thus does not provide a complete protein model. Nonetheless, results have shown promise on maps ranging up to 3.2 Å in resolution.

2.2 Algorithmic Background

This section describes the algorithmic background underpinning my work which is largely rooted in the machine learning and statistics fields. A more in-depth discussion of these topics can be found in Bishop's textbook [6] or Mitchell's textbook [63].

I also rely on concepts from the computer-vision and signal-processing communities with further reading found in the cited literature.

2.2.1 Probability

Probability describes a branch of mathematics concerned with handling uncertainty, or randomness, in systems. This section provides a brief description of the terminology and notational conventions employed in this document. *Random variables* describe events or processes with multiple possible outcomes (discrete or continuous), with each outcome having a weight, or *probability*. Random variables are usually denoted with capitalized letters, specific outcomes in lower case, and a probability distribution described by the function $P()$. For example, for some random variable, A, the probability of a specific outcome, or state, a, is denoted $P(A = a)$ or $P(a)$. $P(A)$ describes a probability distribution of all possible outcomes. The probability of a specific state is between 0 and 1 and the sum of all possible outcomes equals 1:

$$\sum_a P(A = a) = 1. \tag{2.1}$$

In the example above, $P(A)$ is known as a *prior probability* on the random variable A. That is, absent any other information or data, this function encodes the probability of each value of A. For example, consider the task of modeling a coin flip. Before executing a series of trials to determining the propensity of outcomes with a "heads" versus "tails", one could model the prior probability of flipping a heads as uniform (i.e., a fair coin), $P(heads) = P(tails) = 0.5$.

A *conditional probability*, or *posterior probability* in Bayesian terms, describes the probability distribution of a random variable given information about a previously unknown portion of the world (e.g., the state of another variable; empirical data). For example, given the value of a variable, B, the effect on random variable A is encoded with the function $P(A \mid B)$. The conditional probability can model more than two variables; in general, all undetermined (or hidden) variables are placed before the bar, $\mid$, and all *evidenced* variables are placed after. For example, to model the effect of variables B and C on the outcome of random variable A, we express $P(A \mid B, C)$. Two variables are considered conditionally independent if the evidence does not alter the prior belief; that is $P(A \mid B) = P(A)$.

A *joint probability* refers to a distribution over the outcomes of multiple variables in combination and is expressed, for variables A and B, as $P(A, B)$. Joint probabilities can be expressed conditionally to indicate the influence of evidence on two variables, such as with the probability of A and B given evidence of C, $P(A, B \mid C)$. Additionally, two variables are said to be independent of one another if their joint probability is equivalent to the product of their prior probabilities, $P(A, B) = P(A)P(B)$.

Often, it becomes computationally intractable to calculate the full-joint probability distribution over a large set of variables since the quantity grows exponentially in state space. In my work, I often calculate the *marginal probability* of a variable which is the result of summing out all other variables from the joint probability distribution. For example, if we have a joint distribution $P(A, B, C)$, we can obtain the marginal probability of A by summing out B and C:

$$P(A) = \sum_{b \in B} \sum_{c \in C} P(A, B = b, C = c). \tag{2.2}$$

These summations are over all states of B and C and reflect the value of A in the joint distribution.

Lastly, in important concept in probabilistic modeling is *Bayes' rule*, which transforms conditional probability distributions. Bayes' rule states:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}. \tag{2.3}$$

This reversal of the conditional probability proves useful when the quantity $P(A \mid B)$ is difficult to estimate directly from the data, but the quantity $P(B \mid A)$ is more easily available. Additionally, if one wishes to calculate the relative importance of outcomes of A, the denominator can be omitted since it is independent of the value of A and cancels out in the ratio.

As an example application of Bayes' rule, imagine a scenario where a medical lab wants to report a disease test result. The test has been validated as 99% accurate in identifying diseased and non-diseased patients; that is, the test correctly identifies a non-diseased patient as negative 99% of the time and a diseased patient as positive 99% of the time. The lab also knows that only 1 in 200 individuals in the public are positive (0.5%). If a patient's test result comes back positive, what is the probability the patient actually has the disease? Intuition may lead us to think the value is 99%, but that is incorrect.

To see why, I'll introduce notation. We want to know the probability a person has a disease given a positive result, $P(Disease = true|Test = positive)$, or more simply $P(dis \mid pos)$. Tests are positive (pos) or negative (neg) and individuals are diseased

(dis) or not (no_dis). Given this, the test accuracy above is written: $P(pos \mid dis) = P(neg \mid no\_dis) = 0.99$. The background, or prior, rate of disease is $P(dis) = 0.005$. To determine the probability a person has a disease given a positive result, we use Bayes' rule:

$$P(dis \mid pos) = \frac{P(pos \mid dis)P(dis)}{P(pos)}.$$

The denominator is unknown, but we can normalize by summing the probabilities of the two possible values the disease state can take:

$$P(dis \mid pos) = \frac{P(pos \mid dis)P(dis)}{P(pos \mid dis)P(dis) + P(pos \mid no\_dis)P(no\_dis)}.$$

We know all of the values in both the numerator and denominator, yielding:

$$P(dis \mid pos) = \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \approx 0.332.$$

This is much lower than intuition would imply, and provides a representative example of how evidence can alter probabilistic estimates.

2.2.2 Undirected Graphical Models

From the discussion in the previous section, we could proceed to develop and solve probabilistic models using algebraic manipulation. However, it is beneficial to consider a diagrammatic representation of probability models using *probabilistic graphical models* [6]. Graphical models, such as Bayesian networks and Markov random fields, provide a data structure for compactly representing full-joint probability models while also providing modelers the flexibility to encode and inspect relationships between random variables (e.g., conditional dependencies). In addition, an important task in probabilistic modeling is efficiently answering queries about the world – a process known as *statistical inference* (in the remainder of this document, I will use *inference* for short). Graphical models pair with several algorithms for performing inference which can be expressed in terms of manipulations of the graphical model.

Of particular interest to my work are *pairwise Markov random fields* (MRF) [33], a type of *undirected* graphical model where *vertices*, or *nodes*, represent random variables and *edges* encode dependencies between two variables. Edges are undirected, hence the categorization of undirected graphical models. Associated with nodes and edges are *potential functions*, where the multiplication of all such functions represents

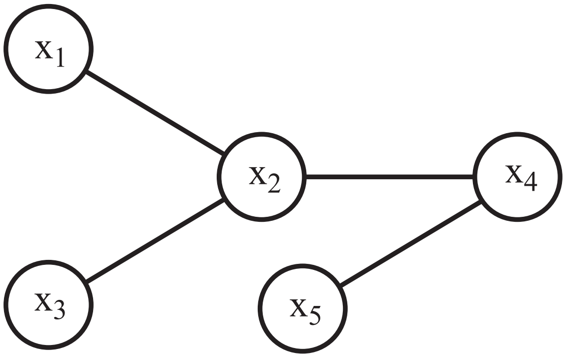


Figure 2.8: A sample pairwise Markov random field model. Vertices represent random variables and edges represent constraints between two random variables.

the full-joint probability of the variables in the graphical model. Thus, the joint distribution over all variables is decomposed into a product of local factors over a small subset of variables. In general, potential functions are defined over cliques, or a subset of fully connected variables. In pairwise MRFs, however, the maximum clique considered is two variables so we only consider potentials associated with edges and nodes.

Formally, a pairwise Markov random field model $G = (V, E)$ consists of vertices $i \in V$ connected by undirected edges $(i, j) \in E$. The full-joint probability over a set of variables, $\mathbf{x} = x_1, x_2, \ldots, x_N$ is

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \times \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j) \tag{2.4}$$

where Z is the *partition function* acting as a normalizing constant. The second term is a product over potentials, $\psi_i(x_i)$, associated with each vertex i. These are local potentials, akin to a prior probability function on variable x_i. The second product term is over edge potentials, $\psi_{ij}(x_i, x_j)$, which encode the dependency between variables x_i and x_j. These dependencies can be thought of as soft constraints on the respective variables.

Figure 2.8 shows a sample MRF model. This simple graph represents a probabilistic model with five random variables, $\mathbf{x} = \{x_1, x_2, x_3, x_4, x_5\}$, and a set of pairwise edges connecting (some) pairs of variables. Using Equation 2.4, the full-joint probability for the model in Figure 2.8 is

$$\begin{aligned}P(x_1, x_2, x_3, x_4, x_5) &= \frac{1}{Z} \times \psi_1(x_1) \times \psi_1(x_2) \times \psi_1(x_3) \times \psi_1(x_4) \times \psi_1(x_5) \\ &\quad \times \psi_{12}(x_1, x_2) \times \psi_{23}(x_2, x_3) \times \psi_{24}(x_2, x_4) \times \psi_{45}(x_4, x_5).\end{aligned}$$

In the above discussion, we assumed the variables are discrete. The framework works for continuous variables as well, with the substitution of integrals for summations. Also, it should be noted that potential functions are not restricted to be probabilities as in directed graphs. This results in a need to calculate the (expensive) partition function, which limits the application of undirected graphical models when model parameters must be learned. If one is not interested in learning model parameters but rather in calculating local marginal probabilities, this term can be omitted since marginal probabilities can be normalized after summing out variables.

2.2.2.1 Inference in Graphical Models

While a probabilistic graphical model provides a representation of the joint probability over all variables, a modeler typically is interested in querying probability distributions of specific subsets of variables, particularly when certain variables are observed, or provided as evidence. This process of reasoning over variables is known as *inference*. Two specific quantities of interest in probabilistic inference are marginal probabilities:

$$P(\mathbf{x_A}) = \sum_{\mathbf{x_B}} P(\mathbf{x_A}, \mathbf{x_B}) \tag{2.5}$$

for some partition of node indices (A, B); and second, the most probable assignment of outcomes to each variable, known as the *maximum a posteriori* (MAP) probability:

$$\mathbf{x}^*_{\mathbf{A}} = \max_{\mathbf{x_B}} P(\mathbf{x_A}, \mathbf{x_B}). \tag{2.6}$$

If the set of variables and states is small, these calculations can be executed by simply enumerating all possible states, in essence calculating the full-joint probability distribution. This calculation, however, grows exponentially. With discrete variables, the size of the joint probability table is $O(K^N)$, with K being the number of states (or outcomes) for each variable and N being the total number of variables in the model. In many instances, the conditional independencies in the probabilistic model suggest methods for performing inference in more manageable time. For example, inference in a linear chain, where nodes are lined up left to right and each node

Algorithm 2.1: Belief Propagation

```
input : vertex potentials $\psi_i(x_i)$ for each $i \in V$,
        edge potentials $\psi_{ij}(x_i, x_j)$ for each $(i, j) \in E$
output: (approximate) marginal probabilities $\hat{p}_i(x_i)$ for each $i \in V$

// Initialize all messages to uniform
foreach $(i, j) \in E$ do
    $m^0_{i \to j}(x_j) \leftarrow U(x_j)$
end
while Stop Criteria Not Met do
    foreach $i \in V$ do
        // Accept messages from all neighbors
        $\hat{p}^n_i(x_i) \leftarrow \psi_i(x_i) \times \prod_{j \in \Gamma(i)} m^n_{j \to i}(x_i)$

        // Calculate messages to all neighbors
        foreach $j \in \Gamma(i)$ do
            $m^n_{i \to j}(x_j) \leftarrow \sum_{x_i} \psi_{ij}(x_i, x_j) \times \psi_i(x_i) \times \prod_{k \in \Gamma(i) \setminus j} m^{n-1}_{k \to i}(x_i)$
        end
    end
end
```

connects only to the node directly preceding and following it, can be performed in linear time. In tree-structured graphs, algorithms such as variable elimination and junction trees exist for performing exact inference in tractable time.

In large graphs with loops, however, exact inference is often not possible. Instead, we must look towards *approximate-inference* methods such as variational methods or Montel Carlo algorithms (e.g., Markov Chain Montel Carlo and Gibbs Sampling) for solutions. The next section will discuss the method utilized in my work, (loopy) belief propagation [70]. For an in-depth discussion of inference, see Koller and Friedman's textbook [50] or Jordan et al. [46].

2.2.2.2 Belief Propagation

Belief propagation (BP) is an inference algorithm that calculates marginal probabilities (Equation 2.5) by utilizing a local message-passing scheme to propagate information across a graphical model [70]. In tree-structured graphs, this inference is exact and efficient. In cyclical graphs, such as the model for Acmi in later chapters, convergence to the exact solution is not guaranteed. In practice, belief propagation in graphs with cycles (loopy belief propagation) tends to produce good approximations, particularly under certain conditions [67, 87]. This section provides a general introduction to belief propagation; Section 3.2.3 and Chapter 5 provide a more detailed description of belief propagation in Acmi's framework.

Belief propagation proceeds in an iterative fashion, with each iteration updating a node's (i.e., variable's) belief by receiving information from neighboring nodes. Conceptually, evidence flows across edges in the graph through message passing, allowing each variable to receive new information on each iteration. Formally, at each iteration, a vertex computes an estimate of its marginal probability distribution (or *belief*), as a product over all associated potential functions, marginalizing out other random variables. The vertex then calculates outgoing messages to each of its connected neighbors by combining its belief with the edge potential function shared with that particular neighbor. BP, at iteration n for each vertex i, computes an estimate, $\hat{p}_i^n(x_i)$, of variable x_i's marginal distribution (or *belief*) over all possible outcomes for x_i by combining its local potential function and incoming messages:

$$\hat{p}_i^n(x_i) \propto \psi_i(x_i) \times \prod_{j \in \Gamma(i)} m_{j \to i}^n(x_i) \tag{2.7}$$

where $\Gamma(i)$ is the set of vertices connected to vertex i (i.e., neighbors of i). Messages from vertex i to vertex j are calculated by convoluting the edge potential $\psi_{ij}(x_i, x_j)$ with vertex i's belief

$$m_{i \to j}^n(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j) \times \psi_i(x_i) \times \prod_{k \in \Gamma(i) \setminus j} m_{k \to i}^{n-1}(x_i). \tag{2.8}$$

In essence, a message from vertex i to j is stating, "Based on my current belief, here are the probabilities of your outcomes". Note that the the values after the first potential are equivalent to x_i's belief in the previous iteration, $p_i^{n-1}(x_i)$, with the influence of x_j's last message to x_i removed:

$$m_{i \to j}^n(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j) \times \frac{\hat{p}_i^n(x_i)}{m_{j \to i}^{n-1}(x_i)}. \tag{2.9}$$

Belief propagation proceeds iteratively, updating beliefs and messages until the beliefs converge or some stopping criteria is met. Algorithm 2.1 provides pseudocode for belief propagation. An important design choice is the order messages are processed in the inner loop. While the pseudocode enumerates variables based on some arbitrary index of variables, Chapter 5 will explore the importance of this design choice in detail.

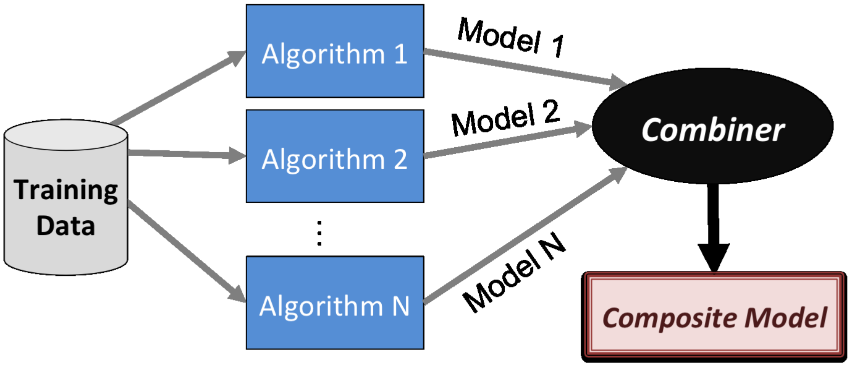


Figure 2.9: Ensemble-learning methods for supervised learning. Multiple algorithms construct models based on the training data. The resulting models are combined to produce a more comprehensive, composite model.

2.2.3 Ensemble Methods

Ensemble-learning methods come primarily from the supervised machine-learning community. The goal of supervised learning is to develop a model (or classifier) with high predictive performance on future instances of a problem. Traditional learning methods involve a search through a hypothesis space that returns a single-best model, $\hat{f}(x)$, to estimate the underlying (but unknown) true function, $f(x)$. Ensemble-learning methods, shown in Figure 2.9, aim to develop a collection of models, $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^N(x)$, that, in aggregate, produce a classifier with better performance than any single constituent model. Empirical evaluations of ensemble-learning methods (or *ensembles*) show that such methods outperform the best individual constituent models under two conditions [5, 17, 59]. First, the ensemble must be *diverse*. A lack of diversity means each model will produce the same answer to a given instance and thus the collective performance will mirror individual performance. Second, the individual members must be *weak classifiers*; that is, each member must be accurate, performing better than random guessing.

There are two primary design choices in developing an ensemble-learning method. First, the learner must generate models that meet the two criteria above; i.e., the learner must generate diverse, accurate models. The machine learning literature contains a variety of techniques for accomplishing this, including the popular methods *bagging* and *boosting*. Second, the learner must aggregate the decisions (or predictions) of each model. This is often accomplished with majority voting, where each model gets a weighted or unweighted “vote” on the answer to a query instance.

While most work on ensembles is on supervised machine learning problems, I am interested in structured-prediction problems. Rather than classification, this task involves performing inference to estimate the probability densities of several unknown and connected variables. Weiss et al. [86] proposed Structural Ensemble Cascades (SEC), an iterative, hierarchical method for structured-prediction problems. Their model learns a sequence of coarse-to-fine models to filter possible output locations in a vision task (e.g., tracking human pose across frames of video). Ensembles are used in inference, where an intractable graph is converted to a set of tractable (i.e., tree-structured) graphs. Wainwright et al. [84] took a similar approach to the problem of intractable inference by decomposing the graph into a set of overlapping, tractable models. These techniques, however, tie parameter learning between the tractable models, imposing an expensive increase in inference time. SEC loosens this contraint since it is not needed for the filtering task.

2.2.4 Spherical-Harmonic Decomposition and the Fast Rotation Function

In Chapter 7, I present a series of methods for recognizing molecular objects in electron-density maps. These methods rely on a 3D shape descriptor known as *spherical harmonics*. Spherical harmonics $Y_l^m(\theta, \phi)$, with order $l = 0, 1, \ldots, L$ and degree $m = -l, -(l-1), \ldots, l$, are the solution to Laplace's equation in spherical coordinates. They are analogous to a Fourier transform, but on the surface of sphere. They form an orthogonal basis set on the sphere's surface. Any spherical function $f(\theta, \phi)$ can be written:

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} a_{lm} \cdot Y_l^m(\theta, \phi) \tag{2.10}$$

where a_{lm} represents the complex coefficient for each harmonic. The key property of a spherical-harmonic decomposition is that for any rotation to a function, the spherical harmonics at some frequency are a linear combination of all spherical harmonics at that same frequency:

$$\mathbf{R}(\vec{r}) \cdot Y_l^m(\theta, \phi) = \sum_{k=-l}^{l} Y_l^k(\theta, \phi) \cdot D_{km}^l(\vec{r}). \tag{2.11}$$

In other words, the amount of energy in a particularly frequency remains the same, just as a translation shift in a Fourier series does not change the power in any frequency. Here $D_{km}^l(\vec{r})$ is an entry in the Wigner D-matrix.

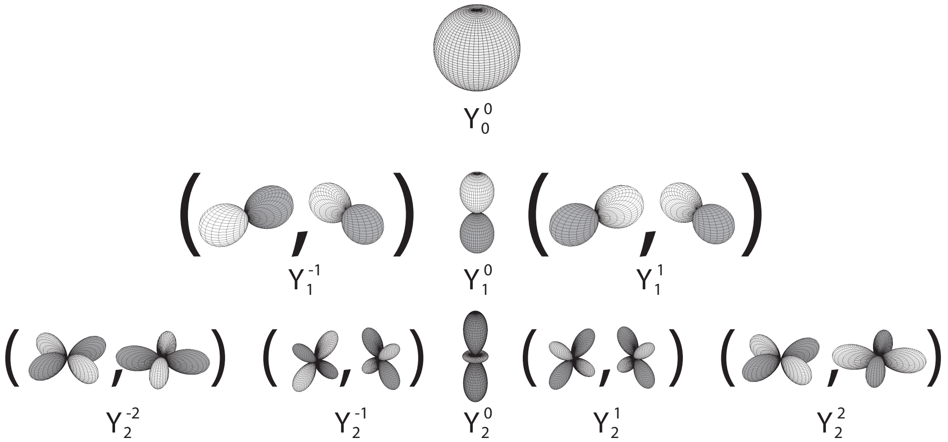


Figure 2.10: The real and imaginary components of several low-order spherical harmonics. These basis functions are illustrated such that the radius at a particular angular coordinate indicates the value at that particular location. Colors indicate sign (gray positive and white negative).

Figure 2.10 illustrates the real and imaginary components of some low-order spherical harmonics. The methods I describe below are based on spherical-harmonic decompositions of three-dimensional objects or images. Spherical-harmonic decompositions have been applied to similar shape-matching problems in other domains [36, 40], and is also similar to the fast rotation function used for molecular-replacement solutions [15, 83]. The key advantage of such a representation is that several different "fast rotation" algorithms exist to quickly compute the cross correlation of two functions on a sphere as a function of rotation [52, 83]. That is, given (real) functions $f(\theta, \phi)$ and $g(\theta, \phi)$ on the sphere, we want to compute the cross correlation between them as a function of rotation angles $\vec{r}$:

$$C_{fg}(\vec{r}) = \int\int f(\theta, \phi) \cdot \mathbf{R}(\vec{r}) \cdot g(\theta, \phi) \cdot sin\,\theta\, d\theta\, d\phi. \tag{2.12}$$

If the functions f and g are band-limited to some maximum bandwidth B, then these fast rotation functions quickly compute this cross correlation given the spherical-harmonic decomposition of f and g, running in $O(B^4)$ or $O(B^3\ \log\ B)$ as opposed to the naive $O(B^6)$ [52, 75]. A full derivation is shown by Kostelec and Rockmore [52].

For computational settings, a bandwidth B is chosen to limit the harmonic transformation to the B lowest-frequency components. This limit truncates the signal in higher frequencies, possibly eliminating high-frequency phenomena. In general, choosing too low of a value for B will lose important information in the signal, while setting B too high results in significant slowdown. Furthermore, eliminating some high frequency components in the signal may be desirable (for example, it may reduce noise). Similar design issues are seen in handling other basis functions, such as wavelets or Fourier series.

2.2.5 Particle Filtering

In Section 2.2.2.2, I described belief propagation, a method for performing approximate inference on a graphical model. Belief propagation is part of a large class of approximate-inference methods, which attempt to approximate probability distributions over (hidden) random variables given a set of evidence. One group of methods, called *variational methods* [46], approach the problem by simplifying the graph structure and performing deterministic inference on the new model. Another set of methods, *Markov Chain Monte Carlo* (MCMC) methods, approximate probability measures using *stochastic sampling* to produce a large set of instantiations of all variables in a Markov Chain. MCMC methods are usually exact representations of the model, but take a long time to reach convergence.

I present an overview of *particle filters*, a set of methods that are a sequential analogue of Markov Chain Monte Carlo (MCMC) methods. In particular, Chapter 8 utilizes sequential importance resampling (SIR) [4, 25] to sample a protein structure. SIR approximates a posterior probability distribution over a state sequence $x_{1:K} = \{x_1, \ldots, x_K\}$ given observations $y_{1:K}$, as the weighted sum of a finite number of point estimates $x_{1:K}^{(i)}$,

$$p(x_{1:K} \mid y_{1:K}) \approx \sum_{i=1}^{N} w^{(i)} \delta(x_{1:K} - x_{1:K}^{(i)}). \tag{2.13}$$

Here, $i = 1, \ldots, N$ is the particle index, $w^{(i)}$ is particle i's *importance weight*, and $\delta()$ is the Dirac delta function. In my work, the technical term "particle" refers to one specific 3D layout of all the non-hydrogen atoms in the protein. SIR represents the distribution of protein configurations as a set of distinct layouts. Particles are sampled sequentially, that is one state (i.e., amino acid) at a time. Each x_k is the position of every non-hydrogen atom in a single amino acid, and each y_k is the region of density in the map providing evidence to amino acid k. Each hidden variable, x_k, has a state space of an entire electron-density map. Therefore, the Dirac delta function, for sequence position k leads to a probability of $w^{(i)}$ being placed on the location sampled by $x_k^{(i)}$ (i.e., a point estimate). The collection of importance weights thus serve as approximations to the relative posterior probabilities of the particles such that:

$$\sum_{i=1}^{N} w^{(i)} = 1. \tag{2.14}$$

SIR relies on the choice of a *proposal distribution*, $\pi(x_k \mid x_{1:k-1}^{(i)}, y_{1:k})$, for sampling point estimates for each variable:

$$x_k^{(i)} \sim \pi(x_k \mid x_{1:k-1}^{(i)}, y_{1:k}). \tag{2.15}$$

The optimal proposal function is the end target function; that is, x_k's posterior transition distribution:

$$\pi(x_k \mid x_{1:k-1}, y_{1:k}) = p(x_k \mid x_{k-1}, y_k). \tag{2.16}$$

SIR is based on the assumption that this posterior is too difficult to sample from directly, but can be used to evaluate a given estimate up to a normalization constant. Instead, SIR typically employs the transition prior probability as the sampling function (called the *importance function*) to approximate the posterior:

$$\pi(x_k \mid x_{1:k-1}, y_{1:k}) = p(x_k \mid x_{k-1}). \tag{2.17}$$

To correct for the bias of this importance function, SIR's importance weights are set to be the the ratio of the optimal proposal distribution (Equation 2.16) to the importance function (Equation 2.17). The weight update comes out to:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{\pi(x_k \mid x_{1:k-1}, y_{1:k})}. \tag{2.18}$$

When Equation 2.17 is the importance function, this simplifies to

$$w_k^{(i)} \propto w_{k-1}^{(i)} p(y_k \mid x_k^{(i)}). \tag{2.19}$$

If the importance function is close to the optimal proposal distribution, weights will have low variance. If there is a large bias, however, weights will diverge quickly causing a few particles to maintain extremely high weights and pushing most particles to nearly 0 weight values (due to normalization). This problem is known as *degeneracy*. To overcome this problem, SIR incorporates an optional resampling step at the end of each sequential sampling round. If the effective number of particles, defined as:

$$M_{eff} = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2} \tag{2.20}$$

falls below a given threshold, resampling with replacement occurs. That is, a new set of particles is drawn one by one from the old set of particles. The chance of selecting a particle is proportional to its old weight. When a particle is drawn, it is placed in the new set but not removed from the old set, so it can be selected multiple times. When N particles have been sampled, all weights are set to uniform. Thus, each step of SIR extends all N particles by sampling x_k from the proposal distribution, updates the weights of particles based on Equation 2.18, normalizes weights, and then optionally resamples particles before moving to the next variable in the sequence.

3 The Acmi System

3.1 Introduction

As outlined in Chapter 2, the most popular technique for determining protein structures is via electron-density maps produced in X-ray crystallography. To this end, a central goal for current structural genomics efforts is to develop automated approaches for interpreting these density maps efficiently and accurately [81]. Section 2.1.3 outlined four current efforts for automated density-map interpretation: ARP/wARP, Textal, Resolve, and Buccaneer. While these techniques have found success in a large range of protein structures, they tend to fail on more difficult protein structures with low resolution (>2.7 Å), high disorder, poor crystal formations, poor phasing information, and/or large numbers of amino acids.

To overcome these shortfalls, our group has been developing the Automated Crystallographic Map Interpretation (Acmi) system. Acmi is a probabilistic framework for combining visual features detected in an electron-density map with biochemical constraints to infer a protein structure. Acmi combines methods from several fields of study, relying primarily on novel approximate-inference approaches to scale to the large state-space of protein-structure determination. This chapter serves two purposes. First, I provide a description of *prior work* on Acmi; that is, previous contributions to the software by other members of our group. Second, I provide a roadmap for my contributions to the Acmi framework. This latter section will serve as a comparison point between my contributions and previous work. More importantly, it will provide a reference point for later discussions to understand where my contributions fit conceptually in the larger Acmi framework. Last, I present a simple face-detection problem as an illustrative analogy for the Acmi pipeline.

3.2 Prior Work on Acmi

DiMaio et al. [21] developed an alternative approach to density-map interpretation which uses a probabilistic model to trace the protein backbone in poor-quality maps ($\sim$3 to 4 Å resolution). Much of this section is also detailed in Frank DiMaio's thesis work [18]. At the center of Acmi's backbone trace algorithm is a pairwise Markov random field model (MRF) [33] constructed from the protein sequence where nodes represent the location of each residue's Cα and edges enforces chemical constraints. Acmi maintains two properties that distinguish it from other methods in the field and

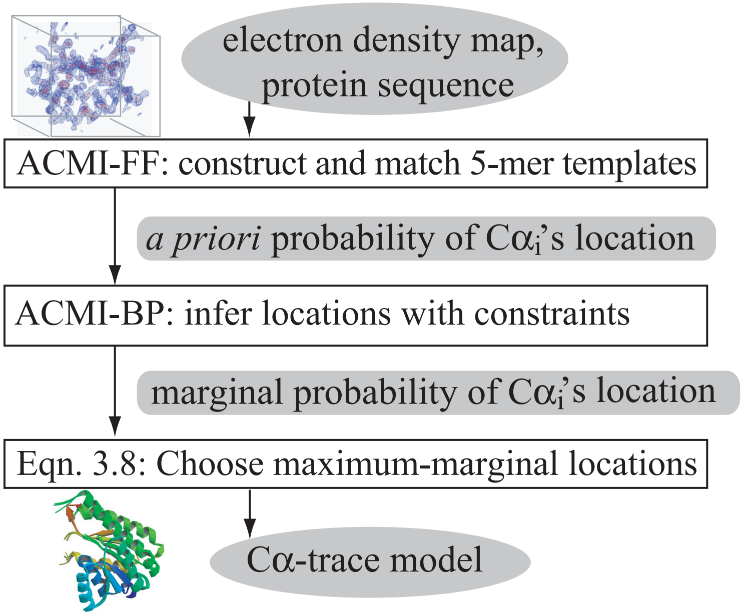

electron density map,
protein sequence
ACMI-FF: construct and match 5-mer templates
a priori probability of $C\alpha_i$'s location
ACMI-BP: infer locations with constraints
marginal probability of $C\alpha_i$'s location
Eqn. 3.8: Choose maximum-marginal locations
$C\alpha$-trace model

Figure 3.1: An outline of Acmi from DiMaio et al. [21]. Acmi utilizes a probabilistic framework to first independently match templates of each residue to the density map and then enforce global bond constraints to create a backbone model of a protein.

allow it to perform inference on difficult maps. First, Acmi simultaneously ties local density information and global constraints to infer possible locations of residues. Second, rather than represent each residue as one or a set of possible locations in the map, Acmi represents each residues location as a distribution over the entire electron-density map. This allows the algorithm to overcome poor, early decisions while also allowing weaker evidence to linger and possibly be utilized in later stages.

This section will outline the two main components of Acmi as originally constructed in DiMaio et al. [21]. An overview of the Acmi method is show in Figure 3.1. The first component, Acmi-FF, scores *local matches* for each residue across all locations in the electron-density map. The result is a probability distribution over the electron-density map for each residue. The second component, Acmi-BP, builds a pairwise Markov random field model to enforce *global constraints* on the the local matches. Acmi-BP performs inference over the likelihood function used to model these constraints to produce a posterior marginal probability describing the location of each residue. From there, a $C\alpha$ trace can be constructed by taking the maximum probability location for each residue's marginal probability distribution. Throughout this document, Acmi will refer to the program as a whole. Individual components of Acmi will be named Acmi-XX, such as Acmi-BP or Acmi-FF.

3.2.1 Local Matching with Acmi-FF

The local matching step of Acmi computes, for each residue i in the protein, a probability distribution $P_i(\vec{u_i})$ over all locations and orientations $\vec{u_i}$ in the unit cell. This phase of Acmi can be considered an "amino-acid finder" task, where the goal is to find likely locations in the map for each residue. The probability distributions are calculated using an *independent* search for each residue that scores proxy templates for the residue against regions around each grid point of the density map, converting the score into a probability and storing the most likely orientation of the residue for each grid point[1]. The particular method used for creating templates, performing a match search, and creating a probability distribution is referred to as Acmi-FF (for fast-Fourier matching). An overview of the method is shown in Algorithm 3.1.

To detect where an amino acid is likely to be found in the electron-density map, the structure of an amino acid is compared against each region of the map and scored based on the match. The particular conformation for a residue (i.e., the shape the residue takes in this protein), however, is unknown. In addition, each of the twenty amino acids take on several different possible conformations. Acmi-FF uses pentapeptide fragments from the Protein Data Bank (PDB) as proxy templates in this search. These pentapeptide fragments are short polypeptide segments five amino acids in length. Throughout this document they may also be referenced as fragments or 5-mers.

For each residue, Acmi-FF first queries previously solved structures in the PDB for fragments with a similar 5-mer context as the residue in question. That is, the amino-acid sequence starting two upstream ($i - 2$) from the residue of interest through two downstream ($i + 2$) forms a 5 amino-acid long sequence with the residue of interest (i) in the center. The rationale for including neighboring amino acids is that local chemical interactions often influence the particular conformation an amino acid may take, therefore Acmi-FF seeks to query fragments with a similar context as the target residue. A set of no fewer than fifty pentapeptide fragments are retrieved based on a nearest-neighbor search, where the PAM-120 distance is used as the measure of similarity. Acmi-FF clusters the retrieved set of fragments into distinct conformations and stores a single representative instance (centroid) and weight for each cluster.

[1] The number of grid points varies for each map, and for each dimension (i.e., x, y, z). Typically, the points range anywhere from 30 to 250 in each dimension.

Algorithm 3.1: Local Template Matching with Acmi-FF

```
input  : amino-acid sequence $Seq$
         density map $\mathbf{M}$
output : Vertex potentials $\psi_i(\vec{u}, \vec{r})$ for $i = 1 \ldots N$
foreach residue $i$ do
    // Find fragments with a similar sequence; cluster by structure similarity
    $\mathbf{PDBfrags}_i \leftarrow$ lookup-in-PDB($Seq_{i-2:i+2}$)
    $\mathbf{centroids}_i \leftarrow$ cluster($\mathbf{PDBfrags}_i$)
    foreach $frag \in \mathbf{centroids}_i$ do
        $template \leftarrow$ compute-dens($frag$)
        foreach rotation $\vec{r}_k \in \mathbf{R}$ do
            // Calculate the mismatch between the map and the rotated template
            $\mathbf{t} \leftarrow$Fffear($template(\vec{r}_k), \mathbf{M}$)
            // Convert the mismatch scores to probabilities based on expected values
            foreach point $\vec{u}_j \in \mathbf{M}$ do
                $(\mu_t, \sigma_t) \leftarrow$ learn-from-tuneset($\mathbf{PDBfrags}_i$,$\vec{u}_j$,$\vec{r}_k$)
                $z_k \leftarrow (\mu_t - t(\vec{u}_j))/\sigma_t$
                $p_{null} \leftarrow$ normCDF $(z_k)$
                $\psi_i(\vec{u}_j, \vec{r}_k) \leftarrow (1 - p_{null})/p_{null}$
            end
        end
    end
end
```

This set of cluster representatives enumerates the conformational space that will be considered for the target residue.

Given an electron-density map, protein sequence, and set of 5-mer cluster representatives for an amino acid in the sequence, Acmi-FF must now measure the correlation of each cluster with each location and orientation in the map to produce the required probability distribution. This template-matching process is outlined in Figure 3.2. Acmi-FF constructs an expected electron density for each centroid fragment. This electron density is then compared to the region around each location in the electron-density map using a mean squared electron-density difference function $t(\vec{u_i})$ where $\vec{u_i} = (x_i, y_i, z_i)^T$ is the 3D grid point location. Acmi defines the mismatch score as:

$$t(\vec{u_i}) = \sum_y \varepsilon_f(\vec{y}) \left(\rho'_f(\vec{y}) - \frac{1}{\sigma_\rho(\vec{u_i})} [\rho(\vec{y} - \vec{u_i}) - \bar{\rho}(\vec{u_i})] \right)^2 \tag{3.1}$$

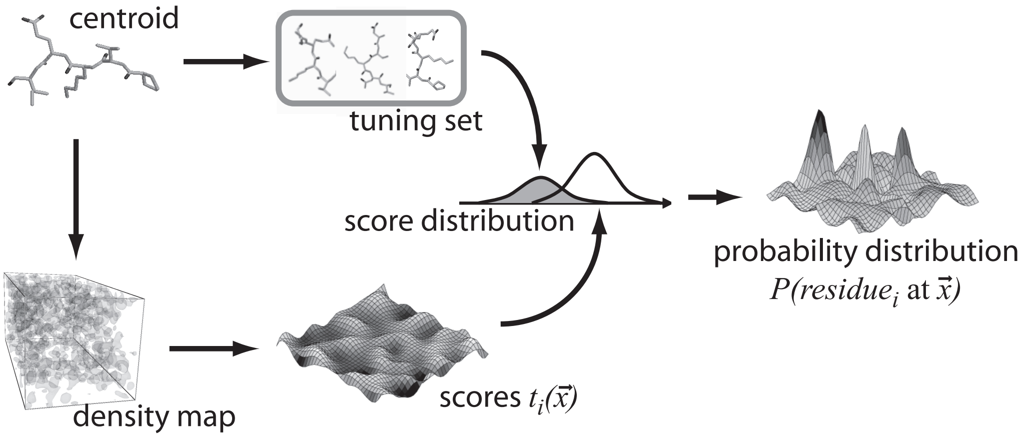


Figure 3.2: An outline of ACMI-FF's method for estimating a residue's probability distribution over all locations in the density map. This figure shows one centroid being compared against the density map to generate match scores. These match scores are converted into a probability distribution by comparing the scores to randomly generated match scores.

where $\rho(\vec{u})$ is the electron-density function of the target map, $\rho'_f(\vec{y})$ is the standardized fragment electron-density function, $\varepsilon_f(\vec{y})$ is a masking function that is non-zero only for grid points near the fragment, and $\bar{\rho}(\vec{u})$ and $\sigma_\rho(\vec{u})$ standardize the map in the masked region $\varepsilon_f(\vec{y})$ centered at $\vec{u}$.

Equation 3.1 is similar to the mismatch function utilized by RESOLVE [82]. In addition to searching across the three dimensions of the electron-density map, ACMI must consider all possible rotations of the fragment. This poses a computationally expensive 6D search problem (i.e., three translation dimensions, three rotation dimensions). ACMI-FF uses a fast Fourier transform (FFT) convolution to efficiently perform a search over all locations at a particular angle. Using Cowtan's FFFEAR [12] package, ACMI-FF discretizes the angular components of the search and performs an efficient FFT calculation of the mismatch score at each angular combination. For each position, ACMI-FF stores the score for the best pentapeptide fragment and the corresponding orientation.

With the mean squared electron-density difference scores calculated, ACMI-FF converts these scores into a probability distribution, $P(\text{res } i \text{ at } \vec{u_i} \mid t(\vec{u_i}))$, which measures the probability that a particular residue, i, is at position $\vec{u_i}$ given the mean square electron-density difference score $t(\vec{u_i})$. To calculate this probability, Bayes rule is employed to give us:

$$P(\text{res } i \text{ at } \vec{u_i} \mid t(\vec{u_i})) = P(t(\vec{u_i}) \mid \text{res } i \text{ at } \vec{u_i}) \times \frac{P(\text{res } i \text{ at } \vec{u_i})}{P(t(\vec{u_i}))}. \tag{3.2}$$

The denominator is simply the distribution of all match scores obtained in the map. The prior probability $P(\text{res } i \text{ at } \vec{u_i})$ can be dropped as Acmi-FF will normalize the final probability scores to sum to the number of copies of the particular 5-mer in the target protein. The first term measures the probability of seeing score $t(\vec{u_i})$ given residue i is at position $\vec{u_i}$. To calculate this, Acmi-FF collects a tuning set of 5-mers from the original pentapeptide search and calculates the distribution of match scores between a) the centroid that matched the map best and b) each member of the tuning set. By doing this, Acmi-FF simulates the expected distribution of scores if a residue is present.

3.2.2 Enforcing Global Constraints with Acmi-BP

At the end of Acmi-FF, Acmi possesses a set of *a priori* probability distributions describing the likelihood of the location and orientation of each amino acid's Cα in the target protein. In a generative model sense, each distribution describes the probability that the density at a grid point $\vec{u_i}$ was generated by residue i. These distributions take into account only local information and do not depend on results of other residues. This lack of considering global interactions leads to clashing information (e.g., two amino acids have a high probability of being at the same location) and inconsistencies (e.g., neighboring amino acids have probability peaks that are too far apart to be chemically feasible). The second phase of Acmi, Acmi-BP (for **B**elief **P**ropagation, seeks to ameliorate these errors by enforcing global constraints on the local-match information to produce a physically feasible backbone trace. Given a protein's linear amino-acid sequence, Acmi-BP uses a *pairwise Markov random field model* to accomplish this task. A pairwise Markov random field, a type of undirected graphical model introduced in Section 2.2.2, defines a probability distribution on a graph, where *vertices* (or nodes) are associated with random variables, and *edges* enforce pairwise constraints on those variables. The probability of a particular setting of a random variable is the product of all potential functions associated with vertices and edges. In Acmi-BP, each vertex corresponds to an amino acid i, and the random variables describe the location and orientation, $\vec{u_i}$, of each Cα_i. Edges enforce pairwise structural constraints on the protein.

Figure 3.3 shows the Markov random field model associated with an arbitrary

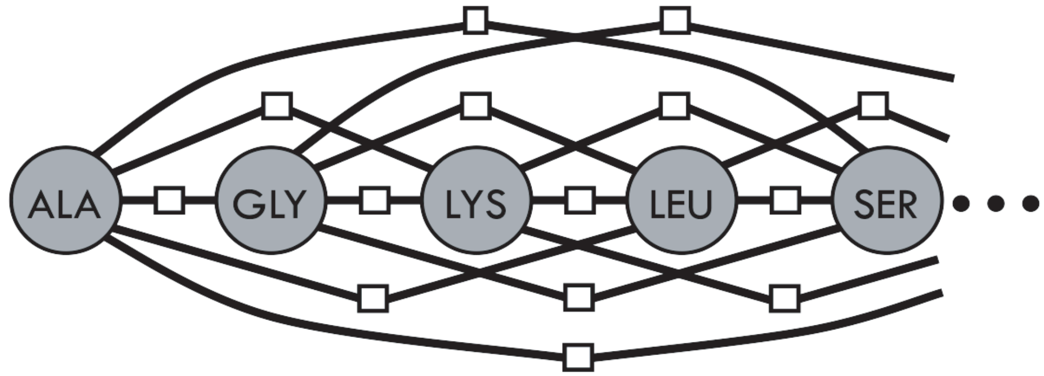


Figure 3.3: A portion of a pairwise Markov random field model for an example protein sequence. Each node represents a random variable for the configuration (location and orientation) of a residue, represented by a three-letter amino-acid code. Each edge represents a pairwise constraint between those two random variables. Boxes on an edge represent the existence of a potential function modeling a constraint between the two variables.

protein. Formally, Acmi-BP's pairwise Markov random field model is similar to the definition in Section 2.2.2, with the addition of an evidence variable, $\mathbf{M}$, representing the fact that our amino-acid locations are conditionally dependent on the evidence in the density map.

Specifically, Acmi-BP produces a graph, $G = (V, E)$ consisting of vertices $i \in V$ connected by undirected edges $(i, j) \in E$. Each vertex is associated with a hidden random variable $\vec{u_i} \in \mathbf{U}$. Each vertex has an associated potential function $\psi_i(\vec{u_i} \mid \mathbf{M})$, referred to as an *observation* potential function. Edge potential functions, denoted $\psi_{i,j}(\vec{u_i}, \vec{u_j})$, represent one of two *conformation* potentials. With these potential functions, the full joint probability of all amino-acid conformations, $\mathbf{U}$, is defined as

$$P(\mathbf{U} \mid \mathbf{M}) = \prod_{i \in V} \psi_i(\vec{u_i} \mid \mathbf{M}) \times \prod_{(i,j) \in E} \psi_{i,j}(\vec{u_i}, \vec{u_j}). \tag{3.3}$$

The edge potentials can be further separated into two types of edges – those between adjacent amino-acid residues and those between non-adjacent residues. Incorporating these into the Acmi-BP model expands Equation 3.3 to:

$$P(\mathbf{U}|\mathbf{M}) = \prod_{\text{amino acid } i} \psi_i(\vec{u_i} \mid \mathbf{M}) \times \prod_{\substack{\text{amino acids } i,j \\ |i-j|=1}} \psi_{adj}(\vec{u_i}, \vec{u_j}) \times \prod_{\substack{\text{amino acids } i,j \\ |i-j|>1}} \psi_{occ}(\vec{u_i}, \vec{u_j}). \tag{3.4}$$

There are three types of potential functions in Equation 3.4. The first, $\psi_i(\vec{u_i} \mid \mathbf{M})$, represents a function over possible locations and orientations for amino acid i, which is proportional to the probability calculations obtained from Acmi-FF. The second function, $\psi_{adj}(\vec{u_i}, \vec{u_j})$, is on edges between neighboring amino acids and represents the *adjacency potential function*. This potential encodes the constraint that adjacent residues must maintain an approximate 3.8 Å spacing and the angle between consecutive Cα-Cα-Cα atoms must follow previously observed distributions. Briefly, both conditions are represented as probability distributions. The Cα-Cα distance is a tight Gaussian distribution where the mean and standard deviation are learned from adjacent Cα distances in the PDB. The angular constraint is modeled as a four-dimensional Gaussian centered on the optimal orientation for each location from Acmi-FF. The width of each Gaussian is also taken from structures in the PDB. The third function, $\psi_{occ}(\vec{u_i}, \vec{u_j})$, occurs on edges between non-neighboring amino acids i and j. This function is the *occupancy potential*. This potential enforces the constraint that no two residues can occupy the same region of the electron-density map. Since occupancy is only an issue of location (and not orientation), this function is simply a step function where all values outside of a 3 Å radius are 1 and inside are 0.

3.2.3 Approximate Inference in Acmi-BP

The model in Equation 3.3 represents the full joint probability distribution over all possible configurations (location and rotation) for all residues in the target protein. Calculating this probability exactly is intractable in large graphs with loops. The goal of Acmi-BP, however, is only to find the trace $\mathbf{U}^* = \{\vec{u_i^*}\}$ that maximizes the observation and edge potentials:

$$\mathbf{U}^* = \arg\max_{\mathbf{U}} \prod_{i \in V} \psi_i(\vec{u_i} \mid \mathbf{M}) \times \prod_{(i,j) \in E} \psi_{i,j}(\vec{u_i}, \vec{u_j}). \tag{3.5}$$

Acmi-BP uses *loopy belief propagation* (BP) to compute an approximate marginal probability $P(\vec{u_i})$ for each amino acid i. With this distribution, Acmi-BP can approximate the maximum *a posteriori* trace in Equation 3.5 by instead taking the maximum-marginal label for each amino acid.

Belief propagation, detailed in Section 2.2.2.2, is an inference algorithm that calculates marginal probabilities by utilizing a local message-passing scheme to propagate information across a graphical model [70]. Recall that in cyclical graphs, such as the

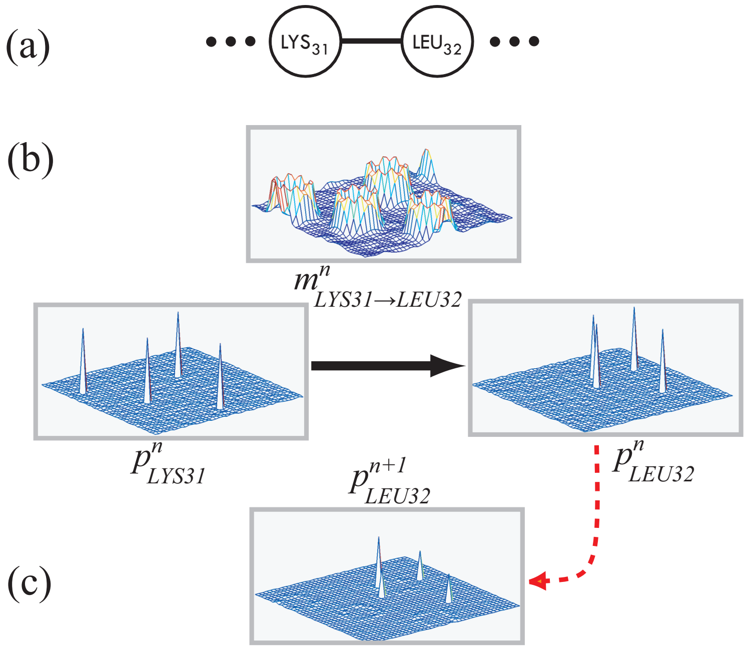


Figure 3.4: A sample message being sent in ACMI's belief-propagation algorithm. In a) we show the portion of interest in the protein's MRF. In b) we show the current state of beliefs and the calculated message to be sent from lysine (amino acid 31 in the sequence) to leucine (32). Finally, c) shows leucine's updated belief after receiving the message, which has boosted the confidence in one of the original four peaks. For simplicity, these distributions represent probabilities on a 2-dimensional plane.

model for Acmi-BP, convergence for belief propagation to the exact solution is not guaranteed. In practice, however, a variation known as *loopy belief propagation* tends to produce good approximations, particularly under certain conditions [67]. As originally shown in Algorithm 2.1, at each iteration a vertex computes an estimate of its marginal probability distribution as a product over all associated potential functions, marginalizing out other random variables. The vertex then calculates outgoing messages to each of its connected neighbors by combining its marginal probability estimate with the edge potential function shared with that particular neighbor. Acmi-BP, at iteration n for each vertex (i.e., amino acid) i, computes an estimate, $\hat{p}_i^n(\vec{u}_i)$, of amino acid i's marginal distribution (or *belief*) over location in the unit cell by combining its local probability (from Acmi-FF) and incoming messages:

$$\hat{p}_i^n(\vec{u}_i) = \psi_i(\vec{u}_i \mid \mathbf{M}) \times \prod_{k \in \Gamma(i)} m^n_{k \rightarrow i}(\vec{u}_i) \tag{3.6}$$

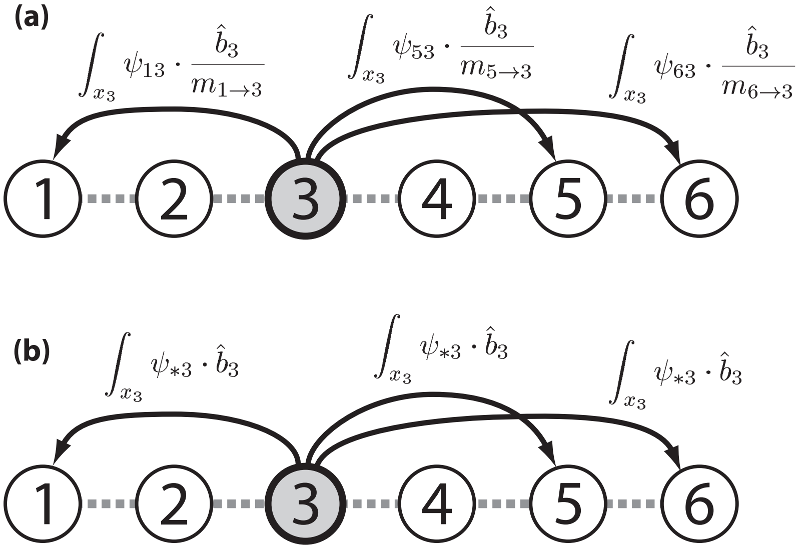

(a)
$\int_{x_3} \psi_{13} \cdot \frac{\hat{b}_3}{m_{1\to 3}}$
$\int_{x_3} \psi_{53} \cdot \frac{\hat{b}_3}{m_{5\to 3}}$
$\int_{x_3} \psi_{63} \cdot \frac{\hat{b}_3}{m_{6\to 3}}$
1
2
3
4
5
6
(b)
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
$\int_{x_3} \psi_{*3} \cdot \hat{b}_3$
1
2
3
4
5
6

Figure 3.5: An example of occupancy messages passed from node 3 in a graph. a) shows the basic method, where unique messages must be calculated, one for each neighbor of node 3. b) shows the use of an aggregate occupancy model, where the same message is passed to all neighbors.

where $\Gamma(i)$ is the set of vertices connected to vertex i.

Messages from amino acid i to amino acid j are calculated by convoluting the edge potential $\psi_{i,j}(\vec{u_i}, \vec{u_j})$ (i.e., adjacency potential or occupancy potential) with amino acid i's belief

$$m_{i\to j}^{n}(\vec{u_j}) = \int_{\text{EDM}} \psi_{i,j}(\vec{u_i}, \vec{u_j}) \times \frac{\hat{p}_i^n(\vec{u_i})}{m_{j\to i}^{n-1}(\vec{u_i})}\, d\vec{u_i}. \tag{3.7}$$

The convolution occurs over the entire distribution, denoted EDM (for Electron-Density Map). The denominator inside the integral removes the influence of the previous message sent across the edge. In essence, message passing from vertex i to j is amino acid i stating, “Based on my belief in my location, I would expect you to be located (with probability) here.” A sample iteration of message passing is shown in Figure 3.4. This process continues iteratively, one amino acid at a time, until all amino acids converge to a stable solution, or some stopping criteria is met. The belief after the final iteration becomes the posterior probability of each amino acid’s location.

Acmi-BP makes two more approximations to improve computational efficiency.

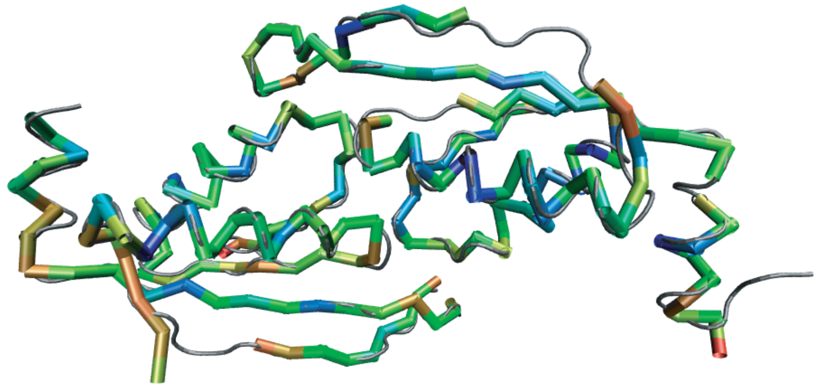


Figure 3.6: A comparison of predicted versus actual structure on DiMaio et al.'s [21] sixth-best (of ten) interpretation at 3.5 Å resolution. The thin continuous coil is the actual structure, while the thicker segmented chain is ACMI's prediction. The predicted structure is colored by log-likelihood, where the least probable residue placements are shown in red, and most probable in blue.

The calculation of messages and beliefs are done using a Fourier representation of each probability distribution and message, allowing an efficient nonparametric belief propagation (NBP) implementation. This improves message passing efficiency, since calculating convolutions can be done quickly using FFTs.

Second, rather than passing $O(N^2)$ occupancy messages, where N is the number of amino acids, Acmi-BP utilizes the fact that the numerator in Equation 3.7 for amino acid i is the same for all occupancy messages sent out. In addition, accumulated incoming occupancy message for any two vertices, i and j, differ only in one component –their own occupancy message. Acmi-BP, therefore, aggregates all occupancy messages as one large message [20]. At each iteration, each node sends and receives messages with three neighbor – the amino acids before and after it in the linear sequence, and the aggregator node. When a node sends a message to the aggregator (denoted $*$), the aggregator multiplies the incoming message to its current aggregator belief. Then, when node i calculates its own belief, it receives a message from the aggregator and removes its own contribution from the previous iteration $m_{i \to *}^{n-1}$. This occurs in place of accepting all individual occupancy messages, thus reducing the number of messages calculated and sent from $O(N^2)$ to $O(N)$ per iteration. An illustration of this aggregation is shown in Figure 3.5.

The last step for Acmi-BP is to return a backbone trace – the item of interest to biologists – using the approximate marginal probabilities. This is done by choosing the most probable (location), $\vec{u_i}^*$, for each amino acid i according to the final belief of its location:

$$\vec{u_i}^* = \arg\max_{\vec{u_i}} \hat{p}_i(\vec{u_i}). \tag{3.8}$$

One advantage of this approach is the ability to assign a confidence measure to the trace based on the probability of the chosen argument. Figure 3.6 shows a sample trace, with the relative probability of each amino acid location indicated by color scale. Results show the backbone traces produced by Acmi have greater accuracy and completeness than traces by Textal [42] and Resolve [82] in a test bed of maps truncated to 3 Å and 4 Å resolution (Section 4.1). However, like Buccaneer [13], Acmi-BP only produces a Cα backbone trace. While useful to biologists for certain tasks, the lack of all-atom protein models hinders the usefulness of the final model.

3.3 Roadmap for Acmi and Thesis Contributions

The previous section provides a base for further development of the Acmi system. The remainder of this document will describe several of my contributions to Acmi, some of which replace modules and others that augment modules in the prior Acmi framework. Figure 3.7 provides a conceptual overview of Acmi in its current form (i.e., inclusive of my contributions) as a reference for future discussions. Acmi is a three-phase pipeline for automatically determining protein structures in low quality density maps, beginning with a density map and linear protein sequence and ending with an all-atom protein structure. The following paragraphs detail the goal of each phase.

Phase 1 – Local Match

Given:	Electron-density map Protein sequence
Do:	Score each amino acid i's match to every location in the map to produce observation potentials, $\Psi_{obs} = \{\psi_i(\vec{u_i})\}$

Phase 1 estimates the observation potential function, a distribution of the probable location of each amino acid in the density map independent of information about other amino acids. Section 3.2.1 described the prior work's appcoach to this problem, Acmi-FF. This phase is conceptually an "amino-acid finder" and is similar to tasks in

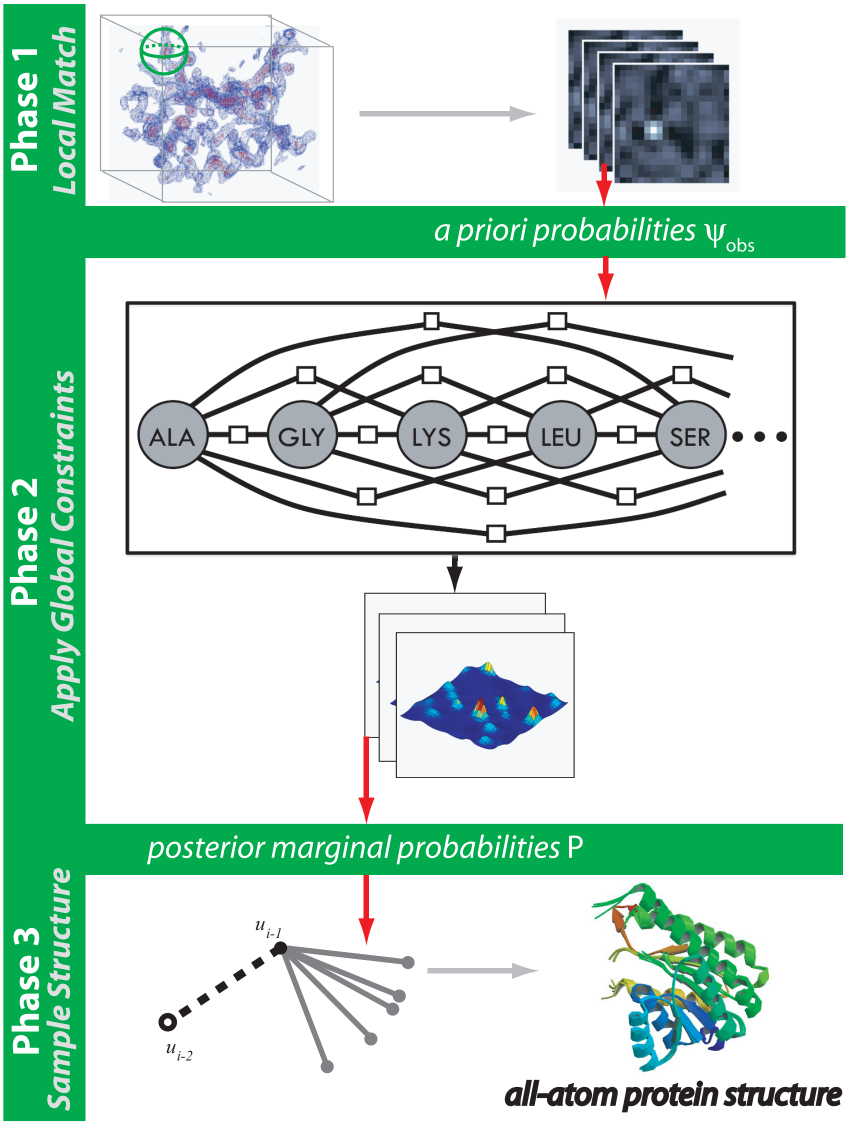


Figure 3.7: The three-phase ACMI pipeline. Given an electron-density map and amino-acid sequence, Phase 1 performs a local-match search independently for each amino acid. Phase 2 combines these local-search results with global constraints to create posterior probabilities of each amino acid's location. Finally, Phase 3 uses these marginals to sample physically feasible, all-atom protein structures. The upper box in Phase 2 shows a portion of a Markov random field for an example protein sequence.

3D shape matching and object recognition in the computer vision community. The term local refers to the local context of the search; that is, ignoring global concerns such as chemical constraints and the location of distant amino acids.

Phase 2 – Apply Global Constraints

Given: Protein sequence
Observation potentials, Ψ_{obs}
Pairwise structural constraints (e.g., adjacency constraint, Ψ_{adj}, occupancy constraint, Ψ_{occ})

Do: Produce a posterior probability for each amino acid i's location in the density map given all available information, $\mathbf{P} = \{\hat{p}(\vec{u}_i)\}$

Phase 2 estimates the posterior probability of an amino acid's location given the local match information of all amino acids in addition to known biochemical constraints on protein structures. While Phase 1 is independent for each amino acid, Phase 2 models the dependencies between amino acids in the protein structure and thus takes a global view of the entire structure. Prior work developed ACMI-BP, which constructs a Markov random field (Section 3.2.2) to probabilistically model the highly connected nature of a protein structure. ACMI-BP employs an approximate-inference technique known as loopy belief propagation, described in Section 3.2.3, to handle the difficult task of performing inference in a complex MRF model.

Phase 3 – Sample Possible Protein Structures

Given: Protein sequence
Electron-density map
Posterior probability of each amino acid's location in the density map, $\mathbf{P}$

Do: Produce a physically feasible, all-atom protein structure that explains the electron-density map

Phase 3 is the final phase of ACMI and estimates a protein structure model. While ACMI is a probabilistic framework, biologists are interested in an actual protein structure, not the probability space of structures. Thus, Phase 3's output is a *point estimate* of the most likely structure given the probability model. In previous work, Phase 3 is a fairly basic post-processing step in ACMI-BP – Equation 3.8 assigns the

	Acmi Phase		
My Contributions	1	2	3
Guided Belief Propagation using Domain Knowledge (Chapter 5)		✓	
Probabilistic Ensembles in Acmi (Chapter 6)		✓	✓
Statistical-Sampling to Produce All-Atom Protein Structures (Chapter 8)			✓
Spherical-Harmonic Decompositions for Template Matching (Section 7.2)	✓		
Filtering Methods for Reducing Search Space (Section 7.3)	✓		
Structural Homology Search in EDMs (Section 7.4)			

Table 3.1: Thesis contributions in the Acmi roadmap. Each item in the first column lists a contribution of this thesis, outlined in Section 1.4. The next three columns attribute whether the contribution applies to Phase 1, 2, and/or 3 respectively. A check-mark in a cell implies that the contribution applies to that phase. The last contribution, Shed, is outside the Acmi framework but extends work on Phase 1.

maximum probability location according to the posterior probabilities as an amino acid's location. These coordinates are only for the Cα atom of each amino acid, yielding only an intermediate result known as a backbone trace. Ideally, biologists want a physically feasible structure with all backbone and side-chain atoms.

These three phases encapsulate three general machine learning problems: shape matching (Phase 1), approximate probabilistic inference (Phase 2), and statistical sampling (Phase 3). In Chapter 1, I outlined the contributions and structure of my thesis. Given this high-level overview of Acmi, I further ascribe the contributions in my thesis to addressing one or more of the three phases of Acmi and subsequently a general area of study in machine learning. Table 3.1 describes which phase(s) of Acmi each contribution of my work applies to. The exception to this framework is my contribution in Section 7.4, structural homology detection in electron-density maps, which is an extension of a novel method for Phase 1 to a different problem in X-ray crystallography. Table 3.1 and Figure 3.7 provide a reference point for placing later discussions in context of the full Acmi framework.

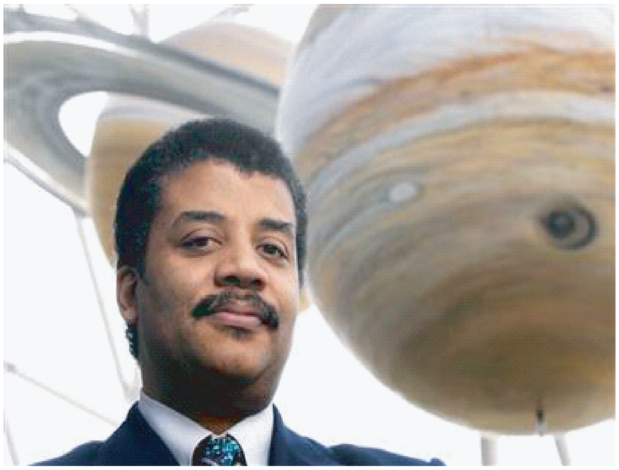


Figure 3.8: An example of the input image to the face-detection algorithm based on the Acmi pipeline. Photo of Dr. Neil deGrasse Tyson courtesy of David Britt-Friedman / MSNBC.com file.

3.4 Analogy to Face Detection

To further illustrate the high-level concepts involved in the Acmi pipeline, this section addresses an analogous problem: face detection[2]. I define the face-detection task as: *given a two-dimensional image and a list of face-parts, identify the location of a person's face in the image*. Here, the two-dimensional picture plays the role of the electron-density map, while the list of face parts is analogous to a protein sequence. While the sequence of amino acids differs from protein to protein, we can assume for this analogy that all faces have two eyes, one nose, and one mouth. Figure 3.8 shows an example image with a face. While this example seems trivial, it will illustrate the role each phase of Acmi plays in processing an image.

In Phase 1, we seek to perform a local-match detection. In Acmi, we seek to measure the probability of individual amino acids being at every location in the map. In face detection, we similarly look to determine the probability of each *face part* being at a location in the image. Assuming we have some part-detection algorithm to score matches, we independently search for a nose, two eyes, and a mouth in our image. Figure 3.9 illustrates the results of such an algorithm projected on our original image. For all three panels, ovals indicate contoured levels of high probability; that

[2] This analogy is for illustrative purposes only, it is not to be taken as representative of current state-of-the-art methods for face detection, nor a complete description of the face-detection task.

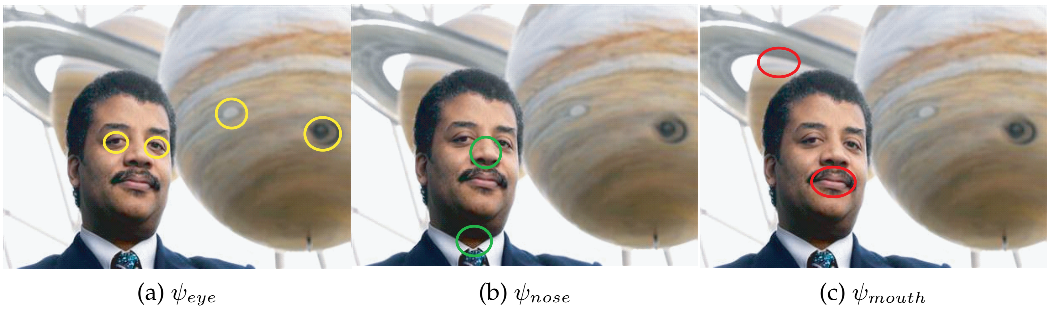


Figure 3.9: Phase 1 observation potentials for the face-detection task. Portions of the image inside contoured ovals indicate high probability values.

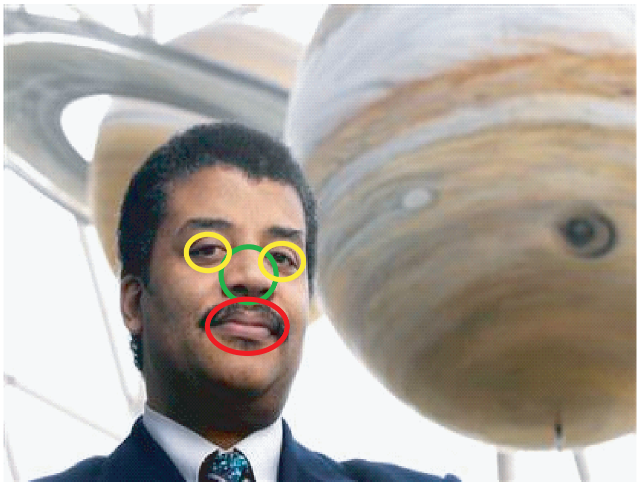


Figure 3.10: Phase 2 posterior probabilities for the face-detection task, with all four face parts projected on the image with contoured probabilities. Yellow represents high probability of an eye, red for a mouth, and green for a nose.

is, the portion of the image inside the circle exceeds some threshold. Figure 3.9a shows the observation potential for the random variable representing the location of an eye, $\psi_{eye}(\vec{u}_{eye})$, with high probabilities encircled in yellow. Figures 3.9b and 3.9c show similar probability maps for the nose (green) and mouth (red) variables. As the image shows, the algorithm has done a good job of identifying the correct locations, but also placed high probabilities on some false locations.

While local features can be powerful, they cannot alone infer the best location for a face. Phase 2 seeks to apply global constraints to introduce structural consistency into our probability model. In Acmi, Phase 2 models pairwise constraints, such

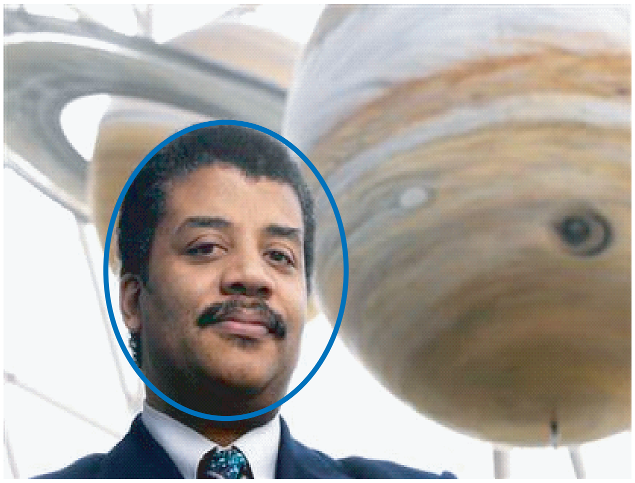


Figure 3.11: Phase 3 output of the most likely face estimate for the face-detection task.

as the fact that neighboring amino acids must be within a constrainted distance. Similarly, we can model pairwise constraints between face parts. For example, no two parts can be in the same location. As an aside, face detection, in reality, poses one problem not seen in protein-structure determination – face parts are not scale invariant. That is, the size of faces vary from person to person, and by distance to lens and magnification of the lens. We'll assume that all images are standardized so that face parts are the same size. With that assumption, we can also have a global constraint that each pair of parts have a limited range of distances between each other (e.g., two eyes cannot be a foot apart). Figure 3.10 shows the Phase 2 posterior probabilities, $\mathbf{P} = \{\hat{p}_{eyel}, \hat{p}_{eyer}, \hat{p}_{nose}, \hat{p}_{mouth}\}$ projected on to the original image. Notice that the location of the false positives have disappeared (i.e., now have low probabilities).

While probability models are extremely powerful in modeling complex problems, many applications require an actual point estimate. In Phase 3, we sample such an estimate. In Acmi, Phase 3 samples a protein structure from the posterior probabilities of Phase 2. In face detection, we again have a set of posterior probabilities from which we want to sample the best face location. In our illustrative example, Phase 3 would output an answer similar to that show in Figure 3.11, where the location of a face is circled in blue. To summarize, face detection provides a simple analogy to the Acmi pipeline. It first identifies individual parts in the image, then applies constraints to introduce structurally consistent estimates, and finally outputs the single best estimate.

4 Data Sets and Protein-Structure Validation Methods

This chapter describes the data sets I use to validate the methods I develop in my thesis. In addition, I provide the details for many of the assessment techniques used for both determining the accuracy of predicted protein structures as well as probability distributions over amino-acid locations.

4.1 Protein Sets for Algorithm Evaluation

This section describes the proteins I utilize for evaluating the performance of my proposed techniques. All experiments in this thesis use one of the two data sets described below. The proteins described below were provided by the Center for Eukaryotic Genomics (CESG) at the University of Wisconsin.

4.1.1 Model-Phased Structures

The first data set of proteins collected, chronologically, consists of ten *model-phased* electron-density maps. Natively, these maps have fairly good resolution – 1.5 to 2.5 Å – and all have crystallographer-determined solutions. Since Acmi is designed for low-resolution maps, Frank DiMaio and CESG computationally downsampled the data to simulate poor-resolution maps. Table 4.1 provides a description of each test-set protein, including the PDB (Protein Data Bank) access number for the deposited solution, the size of the protein structure, and the size of the unit cell in the map. Throughout this document, I refer to this data set as the *model-phased*, protein-structure set since maps were constructed using the final, deposited structure to determine phasing information.

To simulate difficult maps, Frank DiMaio and CESG smoothly truncated the structure factors at 3 Å and 4 Å resolution, and then recomputed the electron-density maps. Truncating in this fashion gives maps virtually identical to maps natively at a particular resolution. To avoid truncation effects, and give a more realistic model of low-resolution data, they scaled structure factors by $exp(-K/R^2)$, where R is the resolution of the structure factor and K is a scaling constant chosen based on the desired resolution (higher values of K smooth the map more). K was chosen to be $K = R_0^2$, where R_0 is the desired final resolution (e.g., 3.0 or 4.0 Å). The result is the signal strength was weakened by $1/e$ at the point of truncation.

PDB ID	Amino Acids in ASU	Molecules in ASU	Resolution (Å)	Unit-Cell Size (Å)
1Q4R	112	1	1.9	$55 \times 55 \times 58$
1VJH	244	2	2.1	$46 \times 34 \times 79$
1VK5	157	1	1.7	$83 \times 83 \times 61$
1VMO	260	2	1.9	$60 \times 79 \times 44$
1XFI	367	1	1.7	$40 \times 43 \times 53$
1XM8	508	2	1.8	$68 \times 59 \times 69$
1XMT	103	1	2.0	$27 \times 61 \times 29$
1XQ1	266	1	2.1	$56 \times 77 \times 112$
1XY7	332	2	1.8	$56 \times 56 \times 147$
1YDH	432	2	2.3	$122 \times 80 \times 51$

Table 4.1: Protein structures in the model-phased protein data set. ASU stands for asymmetric unit.

4.1.2 Experimentally Phased Structures

The second data set used in this document is a set of ten *experimentally phased* electron-density maps. Unlike the *model-phased* set above, however, this data set contains maps in their original form before any interpretation was attempted. Thus, this set more realistically simulates the task of interpreting density maps as a crystallographer would first experience it; e.g., without the optimal phasing information from the final structure.

CESG crystallographers initially phased these maps using either the Solve [82] or Sharp [53] packages, with non-crystallographic symmetry averaging used to improve the map quality where possible. The ten maps were selected as the "most difficult" from a larger data set of twenty maps made available by CESG. The difficulty assessment of these maps was based on expert judgment of the electron-density quality (by E. Bitto and other members of CESG), as well as quantitative estimate of phase error (see Table 4.2). These structures were previously solved by a crystallographer and deposited to the PDB, enabling a direct comparison with the correct model. However, all ten required a great deal of human effort to build the final atomic model.

Table 4.2 provides a summary of the protein structures in this data set, including the resolution and phase error values indicating the quality of density maps. The resolution refers to that available from the initial phasing, which may not reach the

PDB ID	Amino Acids in ASU	Molecules in ASU	Resolution (Å)	Mean Phase Error	Unit-Cell Size (Å)
2NXF[a]	322	1	1.9	58°	64 × 87 × 157
2Q7A[a]	316	2	2.6	49°	82 × 82 × 107
3BUS	566	2	2.65	54°	119 × 119 × 84
1XRI	430	2	3.3	39°	124 × 124 × 124
1ZTP	753	3	2.5	42°	63 × 117 × 124
1Y0Z	660	2	2.4 (3.7[b])	58°	145 × 61 × 115
2A3Q	340	2	2.3 (3.5[b])	66°	74 × 74 × 236
2IFU	1220	4	3.5	50°	84 × 91 × 265
2BDU	594	2	2.35	55°	134 × 134 × 39
2AB1	244	2	2.6 (4.0[b])	66°	46 × 58 × 89

[a] A different data set was used to solve the PDB structure

[b] Phasing was extended from lower resolution

Table 4.2: Protein structures in the experimentally phased protein data set. ASU stands for asymmetric unit.

resolution limit of the data set. In three structures, the initial low-resolution phasing was computationally extended to higher-resolution shells, using the algorithm implemented in Resolve. The mean phase error was computed by comparing the phases calculated from the final all-atom model with those in the initially phased data set using the CCP4 suite [11] of programs. Included in this table are the size of protein structure (i.e., amino acids in the asymmetric unit (ASU)) and unit-cell size which also influence the difficulty of tracing a protein structure.

4.2 Assessing Protein-Structure Quality

As introduced in Chapter 3, Acmi is a three-phase system whose final product is a protein-structure model that explains a provided electron-density map. To validate techniques introduced in this document, I propose in this section various metrics of assessing protein-structure quality. These methods borrow concepts from the information-retrieval and crystallography communities.

4.2.1 Correctness and Completeness

When comparing a predicted protein structure to its ground-truth (i.e., PDB-deposited) structure, one set of complementary metrics I employ is *completeness* and *correctness*. Correctness describes the portion of amino acids in the predicted structure that are within 2 Å of their corresponding true-solution location. This is similar to the *precision* metric used in information retrieval, rewarding algorithms that make good predictions, and ignoring how many predictions were actually made. 2 Å was chosen by our group since it is approximately half the distance between adjacent Cα atoms. The completeness of the predictions is the percent of amino acids available in the PDB solution that are accurately predicted (within 2 Å). This is akin to a *recall* metric and reflects how much of the true solution an algorithm recovered, while ignoring extraneous predictions.

Correctness and completeness are calculated from the coordinates of the true solution and the predicted solution. Specifically, given a list of coordinates for the true structure, $\mathbf{T} = \left\{\vec{t_i}\right\}$, and a list of predicted coordinates by an algorithm, $\mathbf{A} = \{\vec{a_i}\}$, for $i = 1, \ldots, N$ where N is the number of amino acids in the protein, I compute the correctness as

$$Correctness = \frac{\sum_{i=1}^{N} f(\vec{t_i} - \vec{a_i})}{|\mathbf{A}|} \tag{4.1}$$

where the numerator is an indicator function that returns 1 if the distance between $\vec{t_i}$ and $\vec{p_i}$ is less than 2 Å:

$$f(\vec{d}) = \begin{cases} 1 & \text{if } ||\vec{d}|| \leq 2 \text{ Å} \\ 0 & else \end{cases} . \tag{4.2}$$

Completeness is defined as

$$Completeness = \frac{\sum_{i=1}^{N} f(\vec{t_i} - \vec{a_i})}{|\mathbf{T}|}. \tag{4.3}$$

Crystallographers often cannot place all amino acids in a protein. In this case, I ignore the predictions for amino acids not in the true solution since there is no ground truth to evaluate their accuracy. In other words, i only indexes over amino acids in the true solution. If an amino acid is not predicted by an algorithm (but there is a true solution) then $\vec{a_i}$ is undefined and the indicator function in Equation 4.2 returns 0. Unless otherwise noted, the coordinate vectors represent the three-dimensional (i.e., (x, y, z)) location of an amino acid's Cα atom.

4.2.2 Root-Mean-Squared (RMS) Error

Another metric of precision used in statistics and bioinformatics is the *root-mean-squared error* (RMS error), or *root-mean-squared deviation*, of an estimate. RMS error measures the difference between the predicted and the actual values of some estimate. Better estimates produce lower RMS error values, with a perfect prediction yielding a value of 0.

In my application, RMS error measures the deviation in the (x, y, z) coordinates between the predicted location of an atom and the actual (i.e., PDB) coordinates. Each pair of values for a single atom form a *residual* error, and the root of sums of all residual errors for a protein-structure prediction define the RMS error:

$$\begin{aligned} RMSE(\mathbf{A}, \mathbf{T}) &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\vec{a}_i - \vec{t}_i||^2} \\ &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} (a_{i_x} - \vec{t}_{i_x})^2 + (a_{i_y} - \vec{t}_{i_y})^2 + (a_{i_z} - \vec{t}_{i_z})^2}. \end{aligned} \tag{4.4}$$

The vectors **A** and **T** are lists of (aligned) coordinates (e.g., predicted structure and true structure, respectively). In my results, I often only look at the Cα atom for each residue, so the lists will only contain the coordinates of those atoms. In addition, an alignment search is first performed to minimize the RMS error. If either the true structure or predicted structure do not make a prediction for a coordinate, it is not included in either list of vectors since no comparison can be made. Thus, RMS error is a measure of only *predicted* atoms, not all possible predictions. RMS error is best used in conjunction with the completeness and correctness measures of Section 4.2.1 to obtain a total picture of structure error.

4.2.3 R-factor

In interpreting an electron-density map, it is essential to have a metric to measure the quality of the resulting protein structure. While the metrics in Section 4.2.1 validate a protein model against a manual "ground truth" solution, crystallographers need a metric that validates a structure on a novel protein structure. This is particularly important when using computational methods where crystallographers must judge the validity of a "black box" interpretation of a density map. To produce such an assessment, crystallographers typically calculate a protein model's *R-factor* (for *residual* factor) to evaluate the quality of an interpretation. The R-factor provides a measurement of how well a protein model fits the observed values in the electron-density map.

Given the experimentally determined structure factors, F_{obs}, and the model-determined structure factors, F_{calc}, the R-factor is defined as:

$$R = \frac{\sum ||F_{obs}| - |F_{calc}||}{\sum |F_{obs}|}. \tag{4.5}$$

Structure factors are the initial diffraction values accumulated on the collection plate in Figure 2.3, and roughly correspond to the intensity values of the diffracted X-ray beams. Crystallographers usually strive to obtain R-factors under 0.2 (or lower, depending on map resolution), while also building a physically feasible (i.e., valid bond distances, torsion angles, etc.) model. In large, difficult proteins, however, it is not uncommon to see values up to 0.6.

One problem with the R-factor is that Equation 4.5 is susceptible to overfitting – one can always reduce the R-factor by placing extra water molecules in the density map. This explains more of the density values, thus improving the model score, but does not provide any meaningful contribution to the interpretation. To avoid this problem, crystallographers measure the *free R-factor*, or R_{free}, which is akin to a held-aside test set in machine learning. Typically, 5-10% of reflections are randomly held out during refinement of the model. Refinement optimizes the R-factor of the remaining observations (now called R_{work}). Once modifcations to the model are complete, R_{free} is calculated as the R-factor for these held-aside reflections, providing a check against overfitting of the diffraction data.

4.3 Assessing Accuracy of Probability Distributions

While the accuracy of predicted protein structures is the ultimate metric for assessing the performance of the entire Acmi system, the intermediate outputs of the system are probability distributions over the protein structure. Specifically, the Phase 1 observation potentials and Phase 2 posterior probabilities are distributions over the entire density map, produced for each amino acid in the sequence. To test the performance of proposed shape-matching and inference techniques in these phases, I propose two metrics in this section for reporting the accuracy of probability distributions: log-likelihood probability and percentile rank of the true solution. In each of these metrics, the gold standard is the deposited PDB solution. That is, I am comparing a point-estimate[1] to the probability distribution estimate.

4.3.1 Log-Likelihood Probability

One technique for assessing the accuracy of a probability distribution is to view the probability value of the true solution, known as the *likelihood*. Since the output space for each variable in my work is an entire density map, there are on the order of one-hundred thousand possible locations for one amino acid, making most of the probabilities very small. Therefore, I report the *log-likelihood* of the true solution, with higher scores indicating better confidence in the correct answer.

To calculate the log-likelihood probability for an *entire protein*, I average the log-likelihoods of each amino acid. That is, given a set of posterior probabilities, $\mathbf{P} = \{\hat{p}(\vec{x}_i)\}$ for each amino acid, $i = 1, \ldots, N$ and the coordinates of the deposited PDB solution for the structure, $\mathbf{T} = \left\{\vec{t}_i\right\}$, the average log-likelihood score is

$$AvgLLK = \frac{\sum_{i=1}^{N} log(\hat{p}_i(\vec{t}_i))}{N}. \tag{4.6}$$

Recall from Section 2.2.1 that a probability distribution sums to 1. An algorithm producing a posterior probability, $\hat{p}_i$ must choose how to spread this mass over the entire density map. If the probability estimate for amino acid i places all mass at the true solution, $\vec{t}_i$, then the term $log\left(\hat{p}_i\left(\vec{t}_i\right)\right)$ will take it's maximum value of 0. If, however, the value for $\hat{p}_i\left(\vec{t}_i\right)$ is low, the numerator takes on a negative value. Without using a log-transform, the probability estimates can take on minuscule probabilities (e.g., 10^{-6} for a uniform distribution), which would skew the average score. The log-transform, however, reduces the impact of strong skews in likelihood values, providing more meaningful results. In some results, I may report the *negative log-likelihood* which creates visually better results, but flips the meaning of the solution so that lower values indicate high confidence. As with correctness and completeness, I ignore amino acids without coordinates in the PDB solution and $\vec{t}_i$ specifies the Cα coordinates for amino acid i.

[1] Here, a point-estimate refers to a probability distribution where all of the probability mass is at one location.

4.3.2 Percentile Rank

While the log-likelihood provides an indication of the confidence in the true solution, it may not provide the best indication of Acmi's ability to produce the correct solution in Phase 3. Often, I am more interested in the *relative* chance of selecting the true solution. That is, how does the true solution compare to the other locations in the map? When calculating likelihood, the size of the output space varies from protein to protein, so the background (or uniform) chance of selecting the correct solution also varies significantly.

To provide an indication of weight in the true solution relative to other locations in the density map, I calculate the *percentile rank*. The percentile rank represents how highly ranked the correct solution (i.e., location from the deposited structure in the PDB) is in the posterior marginal probabilities. To calculate the percentile for a given amino acid, I sort all of the probabilities in the posterior from highest to lowest and then calculate the percent of locations lower than the true location. The desired score of 100 means the true location had the highest probability value in the map. This is akin to standardized test scores, such as the ACT or SAT, which report how well your score is relative to the general test population.

Given a set of posterior probabilities, $\mathbf{P} = \{\hat{p}(\vec{x}_i)\}$ for each amino acid, $i = 1, \ldots, N$ and the coordinates of the deposited PDB solution for the structure, $\mathbf{T} = \left\{\vec{t}_i\right\}$, the average percentile rank is

$$AvgPercRank = \frac{\sum_{i=1}^{N} rank(\hat{p}_i, \hat{p}_i(\vec{t}_i))}{N} \times 100 \tag{4.7}$$

where the rank score of an individual amino acid is

$$rank\left(\hat{p}, threshold\right) = \frac{\sum_{\vec{x} \in \hat{p}} threshold > \hat{p}\left(\vec{x}\right)}{|EDM|}. \tag{4.8}$$

Here, the numerator is a comparison function that returns 1 if a $threshold$ value (e.g., true solution probability) is greater than some location on the "list" and 0 otherwise. $\vec{x}$ is a 3D coordinate in the electron-density map, and the denominator serves as normalization constant where $|EDM|$ is the number of grid locations in the electron-density map.

5 Guiding Belief Propagation using Domain Knowledge for Protein-Structure Determination

In Section 3.2.2, I described Acmi's use of a fully connected, pairwise Markov random field to model the 3D location of each non-hydrogen atom in a protein. Since exact inference in this model is intractable, Acmi uses loopy belief propagation (BP) to calculate marginal probability distributions of each amino acid's location in an electron-density map. While previous work established that Phase 2 of Acmi does well at performing inference on difficult proteins [21], BP is only an approximate method in Acmi, suggesting there is room for improvement.

This chapter explores a critical design choice in performing inference: loopy belief propagation's message-passing protocol. I propose a novel, general framework for using *domain knowledge* as a criterion for prioritizing messages in BP. Specifically, I show that using predictions of protein-disorder regions effectively guides belief propagation in Acmi, producing more accurate, complete protein-structure models. Referring to the roadmap in Section 3.3, the contributions of this chapter apply to Phase 2 of Acmi. Much of the work in this chapter appears in Soni et al. [78].

5.1 Introduction

As outlined in Section 2.2.2.2, belief propagation is an iterative, local message-passing algorithm which distributes evidence between nodes in a graphical model. Figure 5.1 gives a high-level view of belief propagation on a simple MRF. A message is sent between two random variables (i.e, nodes), conveying the sender's belief in the recipient's state, with probabilities. At each iteration, BP must choose a message to be delivered, calculate that message, and update the recipient's current marginal probability estimate. In graphs with loops (such as Acmi's), the use of BP is not guaranteed to arrive at the exact solution, or even converge to any solution. While empirically successful in Acmi, BP is often abandoned in large, complex tasks due to the inability or slowness to converge to a solution.

Elidan et al. [27] demonstrated that the manner in which messages are chosen to be processed (or *message-passing schedule*) can dramatically affect the success of BP. They show that a simple method (e.g., process messages in a round-robin fashion) is suboptimal, slowing inference in the best case and preventing convergence in the

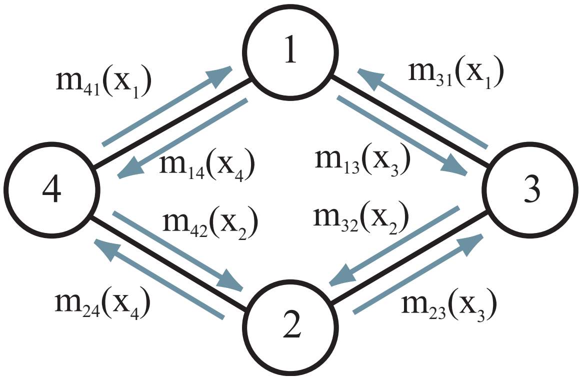


Figure 5.1: Message-passing on a simple Markov random field model. Each node represents a random variable, x_i. During belief propagation, messages, m_{ij}, are sent across edges to convey x_i's belief in x_j's current state. A *message schedule* defines the order these messages are sent.

worst case. An efficient scheduler would require fewer message calculations as well as produce better approximations to the true marginal probability distributions. In Figure 5.1, we see the possible messages that could be sent for a simple graph. The question is: *in which order should BP send these messages?*

Elidan et al. suggest a residual-based method, where each message is prioritized based on how much new information it contains relative to the previous time it was sent. Intuitively, this reduces redundancies and allows BP to identify areas of the graph furthest from convergence. However, as my results will show, this technique is ineffective in the fully connected graph used for Acmi, where it produces flat probability distributions that are insufficient for sampling protein structures. I propose, instead, a new method for scheduling messages in belief propagation using domain knowledge. I apply this general framework to the task of performing inference in Acmi, where a prediction of protein disorder [26] can serve as a priority function for message passing. My results show, across a data set of difficult protein structures, using such a function to prioritize messages in BP improves approximate-inference performance in Acmi relative to a naive scheduling protocol. Additionally, informed scheduling results in more complete and accurate protein structures.

Algorithm 5.1: Round-Robin BP in Phase 2

```
input  : amino-acid sequence $Seq$ of length $N$,
         vertex potentials $\psi_i(\vec{u}_i)$ for $i = 1 \dots N$
output : marginal probabilities $\hat{p}_i(\vec{u}_i)$ for $i = 1 \dots N$

$iter \leftarrow 1$
while Stop Criteria Not Met do
    if isOdd(iter) then
        $startRes \leftarrow 1; endRes \leftarrow N$
    else
        $startRes \leftarrow N; endRes \leftarrow 1$
    end

    for $i \leftarrow startRes$ to $endRes$ do
        // Accept messages from all neighbors
        $\hat{p}_i(\vec{u}_i) \leftarrow \psi_i(\vec{u}_i) \times \prod_{j \in \Gamma(i)} m_{j \to i}(\vec{u}_i)$

        // Calculate messages to all neighbors
        foreach $j \in \Gamma(i)$ do
            $m_{i \to j}(\vec{u}_j) = \int_{\text{EDM}} \psi_{i,j}(\vec{u}_i, \vec{u}_j) \times \frac{\hat{p}_i(\vec{u}_i)}{m_{j \to i}(\vec{u}_i)} \, d\vec{u}_i$
        end
    end
end
```

5.2 Message Scheduling in Phase 2 of Acmi (Belief Propagation)

As discussed in Elidan et al. [27], an important design decision for belief propagation is to define a protocol for ordering the passing and receiving of messages. In exact inference, the ordering of messages only affects the rate of convergence, not the final solution. In graph models with loops, however, the message-passing protocol can affect the speed and accuracy of inference [27, 80]. Message-passing protocols fall into two categories: *synchronous* or *asynchronous*. Synchronous message passing is where all outgoing messages are calculated at the same time followed by a synchronous reception of messages by all nodes in the graph. Asynchronous message passing, instead, prioritizes messages, sending and updating one at a time. Elidan et al. [27] showed that asynchronous message passing demonstrates faster convergence tendencies and lower error bounds than synchronous message passing, in general.

Algorithm 5.1 shows the original implementation of Phase 2 from Section 3.2.3, which I refer to as *Round Robin* Phase 2. In this formulation, Phase 2 of Acmi utilizes an asynchronous message-passing protocol. In particular, nodes are treated in a simple round-robin fashion where a pass begins at one end of the protein's primary sequence and works toward the other. On the first iteration, *isOdd(iter)* returns true since 1 is odd, causing amino acid 1 to be the first residue to be processed. At each step along the way, amino acid i first updates its belief based on Equation 3.6 and then updates its outgoing messages using Equation 3.7; this sequence is then repeated by amino acid $i+1$, and so on. Unlike a traditional round-robin schedule, once all amino acids are processed in one pass of Phase 2, the next pass works in the *reverse* direction. This back-and-forth process continues until all beliefs reach convergence, or some predefined maximum number of passes has been reached.

One disadvantage to this protocol is that it does not prioritize nodes based on any metric of evidence or information gain. Even in the best case, this leads to a waste of resources on passing low-information messages along the chain. A more worrying problem arises when the ordering of nodes gives high priority to nodes with poor prior information – that is, false positives in the match search from Phase 1 of Acmi. This leads to poor evidence being passed around and thus smoothing away more accurate information further down the protein sequence. The next two sections explore alternative scheduling approaches which attempt to identify important messages during the inference process.

5.3 Guiding Phase 2 using Domain Knowledge

The primary motivation for this work is that well-structured regions of the protein sequence are likely to contain better information in their local-match probabilities than disordered regions of the protein. In fact, crystallographers often use such heuristics to decide which portions of the protein molecule to begin manually placing in the density map [George Phillips and Craig Bingman, personal communication, 2009]. Portions of the protein structure that are well-structured (or *ordered*) have a unique or nearly unique conformation [26]. *Disordered* regions of structure, conversely, adopt many different conformations under the conditions of the experiment. This often results in smeared density in the protein image since an electron-density map is an average over millions of copies of the protein, each of which takes one of many possible conformations. Thus, the local-match search (i.e., Phase 1) is unlikely to produce accurate results for disordered amino acids since there is little evidence in the map. To capture this intuition, belief propagation should guide messages based on some domain-knowledge based priority function – that is, some expert-determined

Algorithm 5.2: Domain-Knowledge Guided BP in Phase 2

```
input : amino-acid sequence $Seq$ of length $N$,
        vertex potentials $\psi_i(\vec{u}_i)$ for $i = 1 \dots N$,
        priority function $p_{ord}(i)$ for $i = 1 \dots N$,
        decay factor $\Delta > 0$
output: marginal probabilities $\hat{p}_i(\vec{u}_i)$ for $i = 1 \dots N$

// Initialize priority queue based on function value, paired with node
foreach residue $i$ do
    PQ.push($< p_{ord}(i), i >$)
end

while Stop Criteria Not Met do
    // Pop top value and identify target node, $i$
    $< val, i > \leftarrow$ PQ.pop()

    // Accept messages from all neighbors
    $\hat{p}_i(\vec{u}_i) \leftarrow \psi_i(\vec{u}_i) \times \prod_{j \in \Gamma(i)} m_{j \to i}(\vec{u}_i)$

    // Calculate messages to all neighbors
    foreach $j \in \Gamma(i)$ do
        $m_{i \to j}(\vec{u}_j) = \int_{\mathrm{EDM}} \psi_{i,j}(\vec{u}_i, \vec{u}_j) \times \frac{\hat{p}_i(\vec{u}_i)}{m_{j \to i}(\vec{u}_i)} \, d\vec{u}_i$
    end

    // Add node $i$ back to queue with decayed priority
    PQ.push($< val - \Delta, i >$)
end
```

function of a message's relevance. If this function is accurate, random variables deemed more influential or *a priori* more accurate should push belief propagation toward quicker convergence and/or more accurate approximations.

Algorithm 5.2 shows an overview of my proposed inference algorithm for guiding belief propagation in Phase 2 of Acmi. The main difference from the description in Algorithm 5.1 is the introduction of a priority measure, p_{ord}. This probability function is given to Phase 2 by a user. Guided BP then places the values, $p_{ord}(i)$, into a priority queue paired with the variable it describes. Each iteration of BP takes the highest-priority variable, updates its belief, and updates messages to its neighbors. Finally, the priority value is decayed before the variable is placed back in to the priority queue.

While specifically built for my task, this formulation applies to any instance of belief propagation. The function p_{ord} should describe the relative importance of node i in influencing the network. For my task, it measures the probability that amino acid i in a protein's primary sequence will be well-structured in the final 3D solution. Guided BP in Phase 2 uses this information to decide, for a given iteration, which residue to next perform inference on. Intuitively, Phase 2 now focuses the initial iterations on passing information along regions of the sequence likely to produce stable structure. This probability measure is steadily decayed to allow other amino acids to move to the top of the queue. This will allow the (predicted) ordered regions to refine their probabilities, essentially locking in their locations. When less reliable amino acids finally work up the queue, the ordered amino acids should contain more confident messages and thus have a larger influence on the final distributions.

5.4 Related Work on Guided Belief Propagation

Several methods exist that attempt to improve BP performance by prioritizing nodes based on the amount of *new* information they expect to receive from their neighbors. As previously mentioned, Elidan et al. [27] formulated *residual belief propagation* (RBP), a scheduling function based on the intuition that messages which differ significantly from their previous value are more likely to push BP toward convergence. Conversely, a message whose new value is similar to the value the last time it was sent is contributing relatively little new information to the recipient node and should have low priority.

RBP calculates a residual factor, r_i, for each node[1]. When a neighbor of i is updated, the residual factor is calculated, capturing the amount of new information available. If that value is larger than the current value for r_i, it is updated. Parameter r_i is defined:

$$r_i = \sup_{j \in \Gamma(i)} \|m_{j \to i}^{\text{new}} - m_{j \to i}^{\text{old}}\|_1 \tag{5.1}$$

where $\Gamma(i)$ is the set of neighbors for node i and $\sup$ is the supremum of the set of residuals for messages to node i. At each step of message passing, the node with highest priority is popped off the queue. The popped node updates its belief and outgoing messages before before its residual value (and priority) is set to 0. In addition, all neighbors of the node update their priority values if applicable.

Further work by Sutton and McCallum [80] showed that residual BP still wastes resources, since calculated outgoing messages may never be sent before they are updated again. This occurs if a node's neighbor updates twice before it itself is able

[1] While the original formulation maintains a residual for each message, the symmetrical nature of ACMI's MRF allows us to generalize the method to prioritize nodes instead.

Algorithm 5.3: Residual BP in Phase 2

```
input  : amino-acid sequence $Seq$ of length $N$,
         vertex potentials $\psi_i(\vec{u}_i)$ for $i = 1 \dots N$
output : marginal probabilities $\hat{p}_i(\vec{u}_i)$ for $i = 1 \dots N$

// Initialize priority queue
foreach residue $i$ do
    PQ.push($< 0, i >$)
end

while Stop Criteria Not Met do
    // Pop top value and identify target node, $i$
    $< val, i > \leftarrow$ PQ.pop()

    // Accept messages from all neighbors
    $\hat{p}_i(\vec{u}_i) \leftarrow \psi_i(\vec{u}_i) \times \prod_{j \in \Gamma(i)} m_{j \to i}(\vec{u}_i)$

    // Calculate messages to all neighbors and update their residual values
    foreach $j \in \Gamma(i)$ do
        $m_{old} \leftarrow m_{i \to j}(\vec{u}_j)$
        $m_{i \to j}(\vec{u}_j) \leftarrow \int_{\text{EDM}} \psi_{i,j}(\vec{u}_i, \vec{u}_j) \times \frac{\hat{p}_i(\vec{u}_i)}{m_{j \to i}(\vec{u}_i)} \, d\vec{u}_i$
        $r \leftarrow \|m_{i \to j} - m_{old}\|_1$
        if $r >$ $PQ.getVal(j)$ then
            PQ.remove($j$)
            PQ.push($< r, j >$)
        end
    end

    // Add node $i$ back to queue
    PQ.push($< 0, i >$)
end
```

to update. Sutton and McCallum, instead, estimate the residual values to calculate priority and only update messages when they are actually sent.

In Section 5.6, I show how residual belief propagation affects performance in Phase 2; specifically, how well the marginal probability distributions approximate the true location of an amino acid in an electron-density map relative to the original round-robin protocol and domain-knowledge based priority function. My design of residual belief propagation for Acmi is shown in Algorithm 5.3.

5.5 Experimental Methodology

In Section 5.6, I compare several variations of belief propagation to determine the effect of a message-passing protocol on Acmi's ability to produce accurate protein structures. Table 5.1 provides a brief summary of the methods and the abbreviation used in the results. First is the original version of Phase 2 [21], which uses the round-robin scheduling algorithm in Algorithm 5.1. In the results in Section 5.6, this method is designated as ORIG. I also consider a scheduler based on residual belief propagation [27] from Section 5.4. Algorithm 5.3 shows the details for applying residual belief propagation to Acmi, where I prioritize nodes according to Equation 5.1. Phase 2 scheduled with this function is designated RESID in the results below. Last, I consider a method based on Algorithm 5.2, utilizing domain knowledge to guide Phase 2. In the results below, I denote this heuristic as GUIDED.

Not specified in Algorithm 5.2 is the source for the input $p_{ord}(i)$. This function would ideally measure the amount of order for amino acid i in the final protein-structure solution. Since I do not know this *a priori*, I approximate the concept using *protein-disorder prediction* [26]. Specifically, I use DisEMBL [56], a computational method for disorder prediction using a concept known as "hot loops" – residues without secondary structure (i.e., not a helix or strand) and with high temperature factors. Temperature factors are a term in PDB entries describing the variance of the atom's location. A high temperature factor indicates either low confidence by the crystallographer or the existence of multiple conformations. DisEMBL provides reasonable predictions, identifying 60-70% of disordered residues while predicting about 80% of ordered residues [30]. DisEMBL prediction scores are probabilities measuring the likelihood that an amino acid is in a "hot loop" region. In my experiments, I use the additive complement of this score to formulate $p_{ord}(i)$; i.e., $p_{ord}(i) = 1 - p_{disordered}(i)$.

For each of the tests in Section 5.6, all methods use the same Acmi pipeline with the only differences coming in the second phase. First, Phase 1 is run for each map in the test set to produce the observation potentials, as conceptually presented in Section 3.3. More specifically, I used the protocol in DiMaio et al. [23], which will be discussed in Chapter 7. Then, Phase 2 is run with each of the message-protocols above (i.e., ORIG, RESID, GUIDED) *using the same vertex potentials* as an input. Each algorithm was run for the equivalent of forty passes across the sequence (i.e., $40 \times N$ nodes were processed). While Phase 2 could run until convergence, previous results found 40 passes to be a good stopping point beyond which performance usually

Abbreviation	Scheduling Protocol	Definition
ORIG	Original, round-robin	Algorithm 5.1
RESID	Residual priority	Algorithm 5.3
GUIDED	Domain knowledge guided, using disorder prediction	Algorithm 5.2

Table 5.1: Summary of methods evaluated in Section 5.6. The first column specifies the abbreviation I use to represent the protocol in the text and figures. The second column describes the protocol, with the final column referencing the relevant pseudocode. The main contribution of this work is GUIDED, which guides BP using domain knowledge. RESID is an implementation of residual belief propagation for the task of Acmi, while ORIG is the original implementation of BP in Acmi.

did not improve, and sometimes degraded [21]. For results in Section 5.6.2, Phase 3 sampled protein structures from the final marginal probability distributions of the respective Phase 2 methods using the protocol in DiMaio et al. [19] (highlighted in Section 3.3 and to be explored in Chapter 8). Each map was run ten times producing a set of ten unique structures, of which the average solution is reported.

5.6 Results and Discussion

To evaluate the different methods for scheduling message passing in belief propagation, I ran each (i.e., ORIG, RESID, GUIDED) in the Acmi framework as described in Section 5.5. See Table 5.1 for a summary of the three methods. Note that the method GUIDED represents the main contribution of this work and presents a novel algorithm for belief propagation. RESID, based on work by Elidan et al. [27], is an implementation of residual belief propagation for Acmi. I compare the results across the set of ten *experimentally phased* electron-density maps from Section 4.1.2. Since this work modifies Phase 2 of Acmi, Section 5.6.1 first details the quality of approximate marginal probability distributions produced using each of the different message-passing protocols. While not directly modifying Phase 3, the probabilities produced in Phase 2 are inputs for Phase 3 sampling of protein structures, and thus the proposed methods each have an indirect effect on the final protein structure produced by Phase 3. Section 5.6.2 reports the results of using the marginals from Section 5.6.1 in Phase 3 to produce all-atom protein structures.

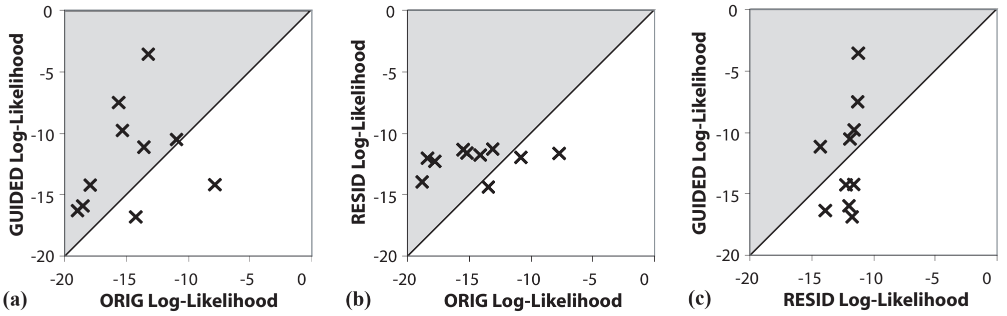


Figure 5.2: Log-likelihoods of the true (i.e., PDB) solution, averaged over all residues in a protein. Each point represents one protein in my experimentally phased test set. The shaded region indicates better performance than the original BP protocol (ORIG) for a) using protein-disorder prediction (GUIDED) and b) using residual belief propagation (RESID) to guide Phase 2. The shaded region of the plot in c) indicates better performance for GUIDED relative to RESID.

5.6.1 Approximate Marginal Probabilities

As described in Section 3.2, Phase 2 produces a marginal probability distribution for each amino acid, describing the probability of that amino acid's location at each point in the electron-density map. Figure 5.2 shows the log-likelihood of the true solution for each message protocol's results. Detailed in Section 4.3.1, the log-likelihood is the log of the probability for the true (x,y,z) coordinates for each residue, according to the Phase 2 posterior marginals. The "true" solution is the solution provided in the deposited PDB file. The higher the value, the more likely a final trace will place the residue in its correct location. Each plot in Figure 5.2 is a *versus* plot, comparing the performance of two different algorithms on the same test-set protein. Each point in this figure represents one protein structure, and is an average of log-likelihoods over all residues in that structure. Figure 5.2a compares ORIG to GUIDED, with the diagonal line designating equal performance. All points *above* the line represent maps where GUIDED produced higher average-log-likelihoods than ORIG. In all but two maps, GUIDED improved the accuracy of Phase 2's marginal probabilities. In Figure 5.2b we see a similar comparison with RESID on the y-axis and ORIG on x-axis. Here, RESID outperforms ORIG in 7 of the maps. Figure 5.2c shows a mixed picture with RESID and GUIDED splitting on the performance across the test set. The overall average-log-likelihood across all maps was -14.5 for ORIG, -12.0 for GUIDED and -12.2 for RESID. Since these numbers are in log-scale, one should read

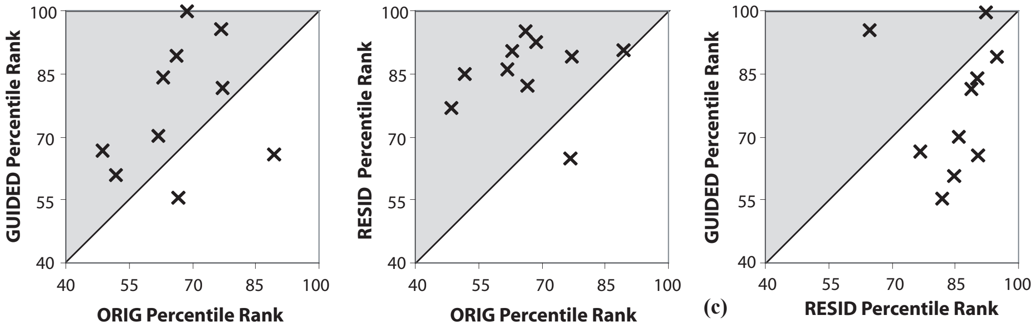


Figure 5.3: Percentile rank of the true solution's probability, averaged over all residues in a protein (see Section 4.3.2 for a definition of *percentile rank*). A higher percentile implies the algorithm puts the true locations closer to the most probable location to be sampled in Phase 3. The shaded area represents better ranks for the true solution relative to ORIG for a) GUIDED and b) RESID. The shaded region in c) shows better ranks for GUIDED than RESID. Each point represents one experimentally-phased protein-density map.

these as orders of magnitude; that is, GUIDED performs 2.5 orders of magnitude better than ORIG. In terms of likelihood of the true solution, on average, Phase 2 benefits from using either of the informed message-passing protocols.

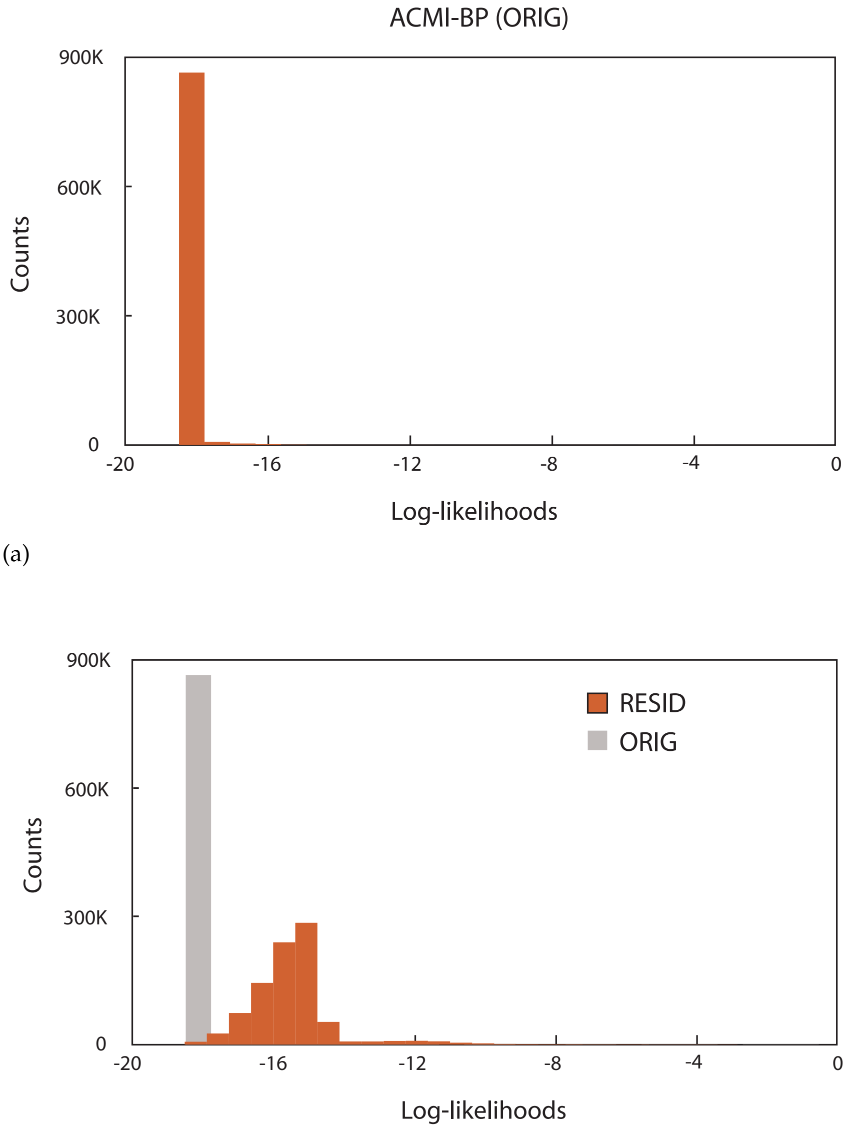
One difficulty in comparing average log-likelihood values among different proteins comes from the fact that the size of the probability space for each protein varies. That is, an amino acid from a protein in a small unit cell has fewer possible outcomes than a protein in a large unit cell. Instead of log-likelihood, we can look at the *percentile rank* of the true solution. This is a normalized metric, allowing me to compare results between maps. Figure 5.3 examines the percentile rank for the true solution of a residue, averaged over all residues in the protein. Defined in Section 4.3.2, the percentile rank of the true solution of one residue is the percentage of points in that residue's marginal probability *below* the probability of the true solution's location. Higher ranks indicate estimates closer to the true solution, with 100 being the optimal score. The plot in Figure 5.3 compares ORIG to GUIDED in a), ORIG to RESID in b), and RESID to GUIDED in c). Again, both RESID and GUIDED perform better than ORIG with a relative decrease in rank by 18 and 10 percentage points respectively. That is, ORIG on average ranks the true residue solution at the 33% mark across all maps while GUIDED ranks at the 23% level and RESID at 15%. According to this metric, RESID tends to produce better ranks than GUIDED, and both outperform ORIG.

From these previous results, we can see an improvement in marginal probability accuracy by the RESID and GUIDED message-passing schemes. Biologists, however, are more interested in seeing if this translates into better protein structures at the end of the Acmi pipeline. The final phase of Acmi, Phase 3 [19], samples all-atom structures from the marginal probabilities produced by Phase 2 (details are found in Chapter 8). This phase of Acmi revealed a shortcoming of RESID. While ORIG (and GUIDED) tend to concentrate probabilities in a few peaks, RESID produced smoother distributions with smaller and more numerous peaks. This is seen in the entropy levels, where the average entropy for an RESID produced marginal was 28.5, over five times higher than the 5.16 and 5.31 averages for GUIDED and ORIG marginals, respectively. The prime culprit is that RESID is susceptible to non-convergent oscillations. That is, a small group of nodes cannot arrive at a stable probability state after a series of messages are passed within this cluster. In this case, the residual stays high in this cluster without resolution, thus choking resources for the other nodes. In fact, for each protein in my set, the median value for the number of times a message was popped off the queue and updated in RESID was either 5 or 6, while the mean was 40. GUIDED, on the other hand, processed *all* nodes between 30 and 40 times.

Figure 5.4a and 5.4b show, for ORIG and RESID respectively, a typical distribution of probability values *within* one amino acid's marginal probability. These figures are not an indication of accuracy, but rather a way of understanding the entropy of the probability values. GUIDED produces a very similar histogram as ORIG, so it is not shown here. For reference, there are 878,952 points in one map, meaning a uniform distribution of probability would assign a log-likelihood of -13.7 to each point. For ORIG, almost all points (98.4%) have the minimum value[2] of -18.2, allowing for a small number of locations to take up most of the probability mass. RESID, on the other hand, distributes the mass much more evenly, with very few points having the minimum probability value (0.8% of all points). In fact, ORIG, has only 280 locations with a probability greater than uniform probability (i.e. greater than -13.7) while RESID has 46,396 such locations. If we thought of Phase 2 as a filter to limit locations considered in Phase 3, RESID eliminates much fewer locations.

This is problematic for Phase 3. To find a good solution, Phase 3's sequential sampling must adequately explore the conformation space of an amino acid. With limited samples, this requires a restricted space of non-negligible locations to search, which is what ORIG and GUIDED provide. RESID, however, contains more locations

[2] Acmi distributes a small amount of probability to each point to prevent probabilities of 0.



(a)

(b)

Figure 5.4: The distribution of log-likelihoods for an example amino acid's marginal probability produced by a) ORIG and b) RESID. For comparison, I have also placed the distribution for ORIG behind RESID in gray. The y-axis measures the number of points in the probability map that have a certain log-likelihood value. The maximum log-likelihood value for any point is 0 (corresponding to a probability of 1). I produced this distribution for an amino acid in 2NXF from Section 4.1.2.

of non-negligible probability than Phase 3 can sample in an efficient manner, causing the algorithm to fail. In fact, across all ten proteins, nine failed to produce any

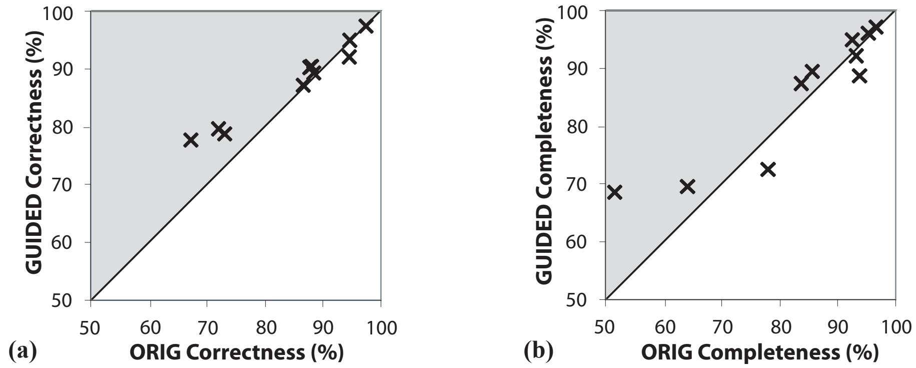


Figure 5.5: Accuracy of predicted protein structures from the test set in Section 4.1.2. Plot a) shows the correctness of predictions – the percent of predicted amino acids within 2 Å of a corresponding residue in the true solution. Plot b) shows the completeness of the predictions – the percent of residues from the true solution with a corresponding prediction within 2 Å. The shaded region indicates better performance for GUIDED.

portion of the protein structure when using RESID marginals, and the tenth only extended 5% of the total protein. The results in Figure 5.3 and the histograms in Figure 5.4 reflect that RESID excelled at preventing the true solution from having neglible probability (i.e., a percentile rank close to 0) and thus looked better on average. RESID, however, did not eliminate enough portions of the density map from consideration for Phase 3 to succeed. This explains why the rank was better for RESID, but the log-likelihoods were slightly better for GUIDED.

5.6.2 Protein Structures

After Phase 2 produces a set of marginal probabilities, Phase 3 is run to sample physically feasible protein structures. I compare how Phase 3 performs with GUIDED produced marginals relative to ORIG produced marginals. As mentioned, RESID did not produce the sharp distributions needed to sample protein structures and thus the results for RESID are not discussed here.

Figure 5.5 shows the results of my experiments on a versus plot, with the original Acmi protocol being shown on the x-axis (ORIG) and the method using domain knowledge for guidance as in Algorithm 5.2 on the y-axis (GUIDED). Each point in the plot refers to one of the test-set proteins. I assess the quality of structures using the correctness and completeness metrics detailed in Section 4.2.1. Figure 5.5a shows the correctness of the structures – the percentage of residues predicted that were within 2 Å of their corresponding true solution location. Conversely, Figure 5.5b shows the completeness of the predictions – the percent of residues available in the true (i.e, PDB) solution that were accurately predicted (within 2 Å). Anything above the diagonal indicates GUIDED produced better structures. In general, GUIDED produced more complete and correct protein structures, particularly in the hardest maps. GUIDED did worse on two proteins in terms of completeness and once in terms of correctness. The underperformance in correctness occurs on a structure Acmi was already performing well on; in fact, most of the proteins with high correctness did not change one way or the other based on the different marginals. Of the three hardest proteins, however, the correctness was dramatically higher when using GUIDED, and in two of these the completeness also improved.

5.7 Summary

Acmi was previously shown to outperform other methods in the literature in building all-atom protein structures in low quality electron-density maps [19]. The success of Acmi is due to its three-phase probabilistic framework. In this chapter's work, I improved the middle phase, Phase 2, which combines local-match information from the first phase (Phase 1) with global constraints to produce a marginal probability of each amino acid's location in the density map. While Acmi is a successful method, Phase 2's results are only approximations, leaving room to improve the resulting marginal probabilities. The accuracy of these probabilities are crucial for Acmi's ultimate success as they define the sampling search space for Phase 3 (the last phase). Results from Elidan et al. [27] indicate that Phase 2's original message-passing protocol was suboptimal and an intelligent protocol could improve BP's convergence properties.

In this chapter, I introduce a general message-passing protocol utilizing domain knowledge to guide belief propagation. I apply this to Phase 2 by using protein-disorder prediction [26] to favor message passing between amino acids predicted to be well-structured, particularly in the early iterations of BP. My results indicate that guiding Phase 2 using this function improves Acmi's overall performance. Across most maps, the rank and log-likelihood of the true locations of each residue improve. In addition, Acmi is able to build protein structures with improved completeness and correctness from these more accurate approximate marginal probabilities, with the greatest improvement coming in the most difficult test cases. The method proposed by Elidan et al., residual belief propagation [27], fails to produce adequate marginal probabilities for use in Phase 3, primarily due to its inability to sufficiently refine the large state space for each amino acid.

One avenue of future work is to apply a similar domain knowledge function to Phase 3, which utilizes particle filtering – an approximate-inference algorithm in which each iteration also requires a choice of what amino acid to next sample in the electron-density map. In addition, I would like to investigate the use of domain-specific heuristics to guide loopy belief propagation when applied to other tasks, particularly other large-state space problems in the field of computer vision. The protocol developed in Section 5.3 and Algorithm 5.2 is a general approach for guiding BP. That is, the use of a priority function is agnostic to the application at hand. Many of the tasks using BP could benefit from my domain-knowledge message-passing protocol, where rule-of-thumb heuristics can be encoded into a priority function to guide BP. The only requirement is that the domain knowledge must be mapped into a priority function over the random variables in the probabilistic model.

6 Probabilistic Ensembles for Improved Inference in Protein-Structure Determination

In Section 3.2.2, I described Acmi's use of a fully connected, pairwise Markov random field to model the 3D location of each non-hydrogen atom in a protein. Of particular importance is the model's need for approximate-inference techniques (e.g., loopy belief propagation) to calculate the best protein structure to explain a density map. In Chapter 5, I briefly introduced the shortcomings of approximate inference in Acmi, and the need for advanced methods to improve the quality of predicted protein structures. In that chapter, I proposed guiding message passing in belief propagation to improve approximate-inference results. This chapter explores a different solution to the shortcomings of approximate inference: ensemble methods for probabilistic models.

In this chapter, I develop Probabilistic Ensembles in Acmi (Pea), a framework for leveraging multiple, independent runs of approximate inference to produce estimates of protein structures. My results show statistically significant improvements in the accuracy of inference resulting in more complete and accurate protein structures. In addition, Pea provides a general framework for advanced approximate-inference methods in complex problem domains. With respect to the roadmap from Section 3.3, the contributions of this chapter apply to both Phase 2 and Phase 3 of Acmi. Much of the work in this chapter appears in Soni and Shavlik [79].

6.1 Introduction

Over the past decade, the field of machine learning has seen a large increase in the use and study of probabilistic graphical models due to their ability to provide a compact representation of complex, multidimensional problems [50]. Graphical models have applications in many areas, including natural language processing, computer vision, gene regulatory network modeling, and medical diagnosis. Recently, the complexity of problems posed in many areas of data analysis, such as statistical relational learning [34], has stressed the ability of algorithms to reason in graphical models. New techniques for *inference* are essential to meet the demands of these problems in an efficient and accurate manner.

As discussed previously, one such application is our group's work on Acmi. Acmi employs *approximate* inference techniques to produce a probability distribution for each amino acid's location in the density map. While Acmi has shown promising results, its inference is not guaranteed to arrive at the correct solution and, more importantly, is computationally expensive. This limits Acmi's ability to produce sufficiently accurate solutions in some cases.

To overcome these limitations I propose Probabilistic Ensembles in Acmi (Pea), a general framework for performing approximate inference in complex domains. Pea borrows the concept of ensemble-learning methods from the supervised machine learning literature [5, 17]. Section 2.2.3 describes ensemble methods in detail. While traditional methods build a single model to estimate the underlying true model to a problem, ensembles leverage a *collection* of estimates to produce a more accurate solution. In classification, this is useful when the complexity of the underlying model is too difficult to approximate accurately or the search for the true model is non-convex (i.e., susceptible to many local minimums). A single approximate model may only describe part of the true model accurately, but multiple, diverse models may provide a cohesive estimate in aggregate.

I extend this idea to my task by performing multiple, independent runs of approximate inference in Acmi to produce multiple probability estimates of the protein's locations. Each estimate may only be partially correct due to the loopy nature of the graph, but the estimates in aggregate may provide a better idea of the true probability distribution of an amino acid's location. My results show Pea dramatically outperforms Acmi in both the quality of inference and accuracy of protein structures produced. Pea presents a novel, general approach to improved approximate inference. While previous approaches used a collection of simplified, exact inference solutions to estimate a difficult problem [84, 86], Pea instead considers a collection of complex, approximate-inference solutions with each being built from a different view of the problem.

6.2 Probabilistic Ensembles in ACMI

With the well-documented success of ensemble methods in classification tasks, I seek to extend the idea of aggregating multiple estimates to probabilistic graphical models. As discussed in Section 2.2.3 and in the previous section, current efforts by other researchers in the area rely on simplifying the structure of an intractable graph to create a collection of tractable problems [84, 86]. This is done by removing edges in the graph until the sparseness of the graph allows for exact-inference methods. This usually requires removing all cycles in the graph creating a tree graph. This

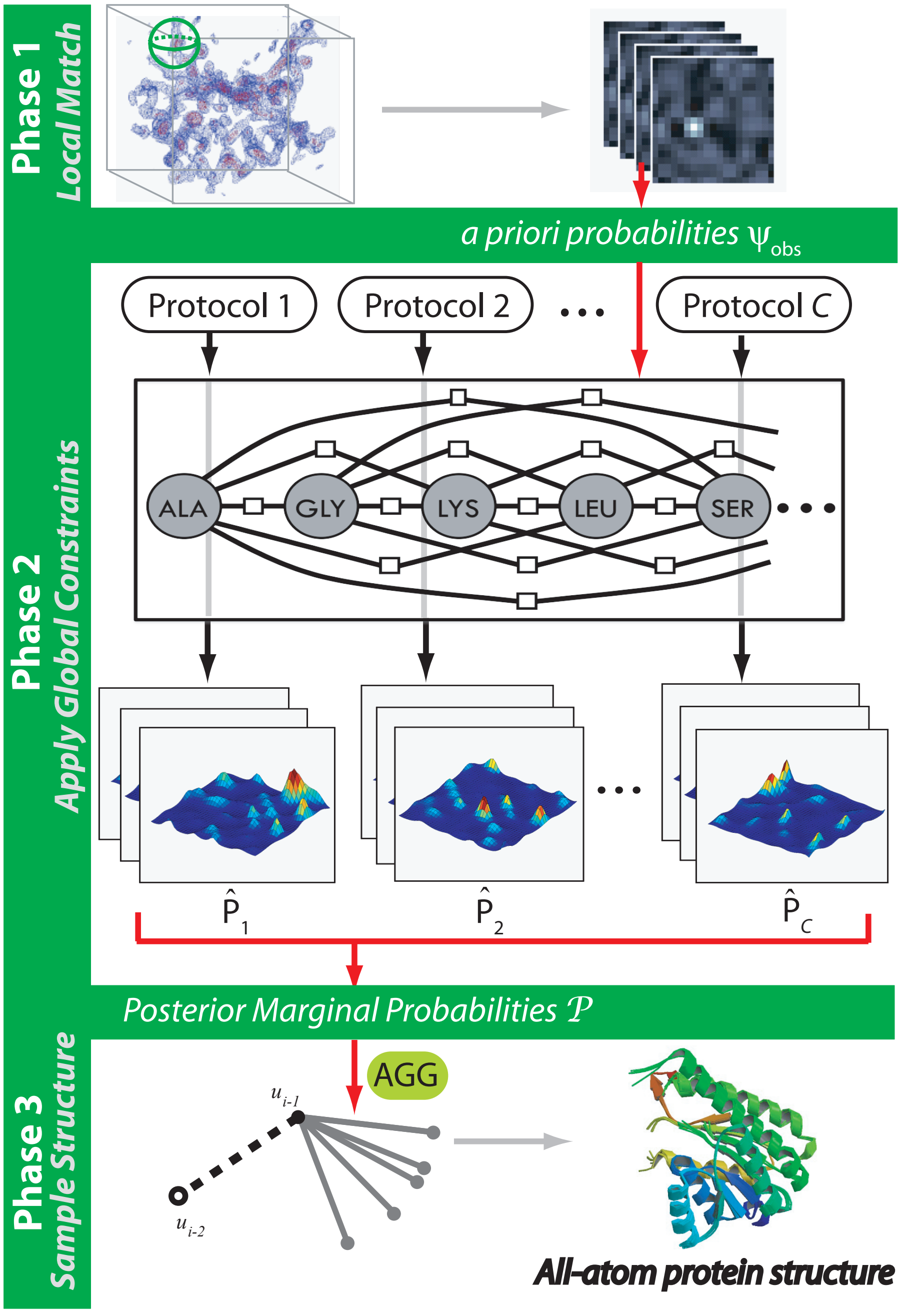


Figure 6.1: Probabilistic Ensembles in Acmi (Pea). Phase 1 is the same as in the Acmi framework (Figure 3.7). Phase 2 performs C independent inference runs, each with a unique protocol. This results in a set of C marginal probabilities for each amino acid's location. Phase 3 aggregates the set of marginal probabilities to produce a protein structure.

approach, however, does not readily extend to the graph in Acmi, which is fully connected and thus difficult to convert to the necessary number of tree-structured graphs[1]. In addition, previous work in Acmi introduced an approximation that exploited redundancies in messages passing to dramatically reduce the complexity of inference [20]. Described in Section 3.2.3, this approximation aggregates occupancy messages into one message, reducing $O(N^2)$ messages to $O(N)$ in one pass of BP without removing constraints. Converting Acmi to a tree-structured graph loses the gains from this approximation as well as important information encoded in edges. Thus, unlike previous approaches, I am interested in an ensemble solution that boosts the *accuracy* of inference, not the *tractability*.

I propose Probabilistic Ensembles in Acmi (Pea), shown in Figure 6.1 and Algorithm 6.1. Pea is a framework for generating and combining multiple approximate-inference solutions to create more accurate protein structures. As with ensemble-learning methods in classification, there are two major design components to address. In Section 6.2.1, I discuss how Pea generates a *diverse* set of solutions (i.e., ensemble components). In Section 6.2.2, I describe my framework for *aggregating* these solutions to produce a protein structure. Both of these components modify the existing Acmi framework (compare Pea in Figure 6.1 with Acmi in Figure 3.7). Specifically, generating diverse solutions amends the inference process in Phase 2 and combining these solutions takes place in structure sampling of Phase 3.

6.2.1 Generating Ensemble Components

From Section 3.2.2, Equation 3.3 calculates $P(U \mid M)$ – the probability distribution over all possible protein structures given the density map. Since this calculation is intractable, Phase 2 of Acmi produces $\hat{p}_i$, the approximate marginal probability of amino acid i's location for each amino acid in the protein sequence. Rather than performing inference once, my proposed framework, Pea, performs several independent runs of inference. As shown in Figure 6.1, each run (C in total) uses a unique protocol and outputs its own marginal probability distribution for each amino acid's location. Phase 2, in total, produces a matrix of probability distributions $\mathcal{P} = (\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_C)$ where each ensemble component c produces $\hat{P}_c = (\hat{p}_{1_c}, \hat{p}_{2_c}, \ldots, \hat{p}_{i_c})$. Here, $\hat{p}_{i_c}$ represents the probability of amino acid i's location in the density map according to component c of the ensemble.

[1] In Acmi, a fully connected graph with N vertices will have $\frac{N^2-N}{2}$ edges. To convert this MRF to a tree-structured graph, at most $N-1$ edges may exist, meaning $\frac{N^2-3N}{2} - 1$ constraints must be removed.

Algorithm 6.1: Probabilistic Ensembles in Acmi (Pea)

```
input  : amino-acid sequence $Seq$ of length $N$,
         electron-density map of protein $\mathbf{M}$,
         number of ensemble components $C$,
         inference protocol for each component $protocol_c$
output : protein structure $\mathbf{U} = (\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_N)$
         where $\vec{u}_i$ is the coordinates of amino acid $i$

// Calculate vector of observation potentials, $\Psi(U)$
for $i = 1 \ldots N$ do
    $\psi_i(\vec{u}_i) \leftarrow$ Phase1($Seq, \mathbf{M}$)
end

// Generate ensemble of posterior probabilities, $\mathcal{P}$
for $c = 1 \ldots C$ do
    $\hat{P}_c \leftarrow$ Phase2($Seq, \Psi(U), protocol_c$)
end

// Sample all-atom protein structure
$\mathbf{U} \leftarrow$ Phase3($Seq, \mathcal{P}$)
```

As mentioned in Section 6.2, a desired property of an ensemble is that the individual components are diverse. Fortunately, Elidan et al. [27] showed the choice of a message-passing protocol (i.e., what order to send and receive messages between nodes) has a large effect on the outcome of belief propagation, particularly in complex, loopy graphs such as Acmi's. Chapter 5 examined the effect of using various message-passing schedules on Acmi's performance and found that this decision choice has a significant impact on the final solution of Phase 2 [78]. Section 6.3 provides example message-schedules for generating ensemble components in Pea.

6.2.2 Aggregating Ensemble Components

In DiMaio et al. [19], we developed Phase 3 of Acmi which utilizes *particle filtering* [4], a sampling algorithm, to generate all-atom protein structures given the posterior marginal probabilities from Phase 2. Details are found in Chapter 8. Briefly, Phase 3 is an iterative process which sequentially grows a protein structure one amino acid at a time[2]. Figure 6.2 shows how, at a given iteration, Phase 3 samples the location, $\vec{u}_i$, of amino acid i.

[2] Phase 3 maintains multiple estimates (or particles) during the sampling process and uses separate steps to sample backbone and side-chain atoms. For simplicity, I only consider one particle's backbone placement in this description.

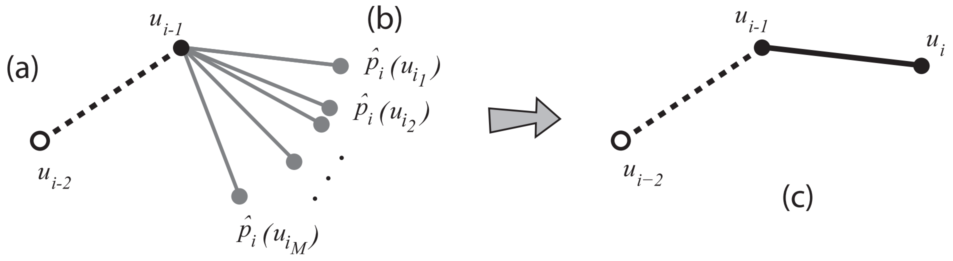


Figure 6.2: Acmi's Phase 3 backbone sampling step for amino acid i. In a) Phase 3 samples M possible new locations, $\vec{u}_{i_m}$. In b) these locations are weighted by their agreement with the Phase 2 probability, $\hat{p}_i$. In c) one location is chosen from the weighted distribution to be amino acid i's location, $\vec{u}_i$.

Phase 3 first samples M potential locations for the new amino acid based on the location of already placed amino acids in the sequence and a distribution of known angles and distances between neighboring amino acids (i.e., the adjacency potential function $\psi_{adj}(\vec{u}_{i-1}, \vec{u}_i)$ from Section 3.2.2). Next, Phase 3 assigns a weight, w_m, to each sampled estimate, $\vec{u}_{i_m}$, correlated with the likelihood of amino acid i being in that location. This is calculated from the Phase 2 posterior marginal probabilities:

$$w_m \propto \hat{p}_i(\vec{u}_{i_m}). \tag{6.1}$$

Lastly, Phase 3 chooses one of the M samples as its prediction for amino acid i's location in the structure. The sample is chosen with probability proportional to w_m.

In Pea, I must adjust Phase 3 to deal with the existence of multiple Phase 2 posterior probabilities for each amino acid. I propose several functions for combining these probabilities into a single weight measurement. First, I look at the average score over all ensemble components using a mixture model:

$$w_m \propto \sum_c \pi_c \cdot \hat{p}_{i_c}(\vec{u}_{i_m}) \tag{6.2}$$

where π_c is the mixture weight, representing the confidence that component c provides the correct distribution[3]. The average score should perform well if the true location tends to have a high score across all runs of inference while false positives are uncorrelated between runs. False positives would be smoothed out and consistent peaks would maintain high probabilities. Conversely, strong divergences

[3] π_c can be set by various measures, such as entropy or prior knowledge. I use uniform weights in my experiments.

between models would smooth away most information.

Another proposed weight function is to instead take the maximum score for a given location across all components:

$$w_m \propto \max_c \hat{p}_{i_c}(\vec{u}_{i_m}). \tag{6.3}$$

In difficult portions of a protein, it is likely that most models will miss the correct location since there is very little evidence. Given multiple estimates, it is more likely that one model found the correct answer. Using the maximum score allows Phase 3 to capture the correct location in this situation, although it may also over-emphasize false positives.

Lastly, I consider using a subsampling approach where Phase 3 randomly selects one of the ensemble components to score the location:

$$\begin{aligned} c &\sim U[1, C] \\ w_m &\propto \hat{p}_{i_c}(\vec{u}_{i_m}) \end{aligned} \tag{6.4}$$

where $U[1, C]$ returns an integer between 1 and C with uniform weight. Alternatively, c could be drawn from a weighted distribution based on π_c in Equation 6.2. This technique fits intuitively into the sampling framework of particle filtering where multiple structure estimates exist to explore several different paths to the end state.

6.3 Experimental Methodology

In Section 6.4, I compare the performance of our original ACMI framework from DiMaio et al. [19] (i.e., the work from Section 3.2 with Phase 3 implemented in Chapter 8) to my proposed algorithm, PEA. I use the set of ten *experimentally phased* electron-density maps described in Section 4.1.2 for validation.

Phase 1 (performing an independent search for local features) is the same for both algorithms, meaning Phase 2 for both the original ACMI and proposed PEA algorithms begin with the same input. To clarify, ACMI herein refers to a single-inference version of my method as shown in Figure 3.7, while PEA refers to the ensemble-inference version of ACMI from Figure 6.1.

Pea Ensemble Components

Protocol ID	No. of Iterations	Starting AA
1	40	1
2	40	middle AA
3	20	1
4	40	$\arg\max_i p_{ord}(i)$

Table 6.1: Summary of protocols for ensemble components in Pea used in Sections 6.4.1 and 6.4.2. Each protocol varies in duration (number of times each node sends a message) and starting point. AA stands for amino acid and $p_{ord}(i)$ is the protein-disorder priority function discussed in Chapter 5.

For the experiments in Sections 6.4.1 and 6.4.2, Pea utilizes an ensemble of size four with each component having its own protocol:

- Protocol 1 is the original protocol of Acmi, which is run for 40 iterations per amino acid in a round-robin fashion starting with amino acid 1, proceeding left-to-right, and then reversing at the end of each pass.
- Protocol 2 is similar to Protocol 1, but starts halfway through the sequence.
- Protocol 3 is similar to Protocol 1, but runs for only 20 iterations.
- Protocol 4 employs guided belief propagation discussed in Chapter 5 and shown in Algorithm 5.2.

I summarize these protocols in Table 6.1. Each of these protocols corresponds to an independent run, c, of inference in Pea as shown in Figure 6.1.

I compare my ensemble approach to single runs of inference in Acmi. I consider three variations of Acmi's belief-propagation protocol for Phase 2:

- ORIG, same as Protocol 1. This is the original protocol for Acmi shown in Algorithm 5.1.
- EXT, an extended version of the original protocol going for 160 iterations. EXT uses as much CPU resources as all four protocols in Pea combined.
- BEST, the top-performing individual version of Acmi from the four protocols considered for Pea.

Variations of Acmi

Abbreviation	No. of Iterations	Starting AA
ORIG	40	1
EXT	160	1
BEST	—	—

Table 6.2: Summary of three variations of Acmi considered as controls in Sections 6.4.1 and 6.4.2. Each protocol varies in duration (number of times each node sends a message) and starting point. BEST is picked from the top-performing protocol in Table 6.1.

Note that these are *independent* assessments of Acmi and are not combined. Table 6.2 summarizes these three variations. The BEST protocol provides an overly optimistic estimate of Acmi to see how Pea performs as an ensemble relative to its individual components.

For the learning curve in Section 6.4.3, 50 protocols were created. All were based on the standard, round-robin schedule and executed for 40 iterations. Each varies in the starting location and the direction of the first iteration. Twenty-five starting locations were chosen, evenly spaced out in the sequence. Each starting location spawned two protocols – one starting left-to-right, and the other right-to-left, generating 50 components in all.

Phase 3 experiments in Section 6.4.2 were run with 100 particles (see Chapter 8 and DiMaio et al. [19] for details). For the aggregators in Section 6.2.2, my shown results reflect a uniform mixture weight (i.e., $\pi = 1/C$ if there are C components to the model). I considered a weight based on the entropy of the distributions, but the results were statistically equivalent to using a uniform weight. I determine statistical significance using a paired t-test.

6.4 Results and Discussion

Using the methodology described in Section 6.3, I compare the performance of my new approach of using Probabilistic Ensembles in Acmi (Pea) against the original Acmi algorithm [19]. I compare the results across the set of ten difficult experimentally phased density maps from Section 4.1.2. Since Phase 1 remains unchanged between the two algorithms, I do not report the results of generating the observation potentials here. Section 6.4.1 first assesses the quality of approximate inference

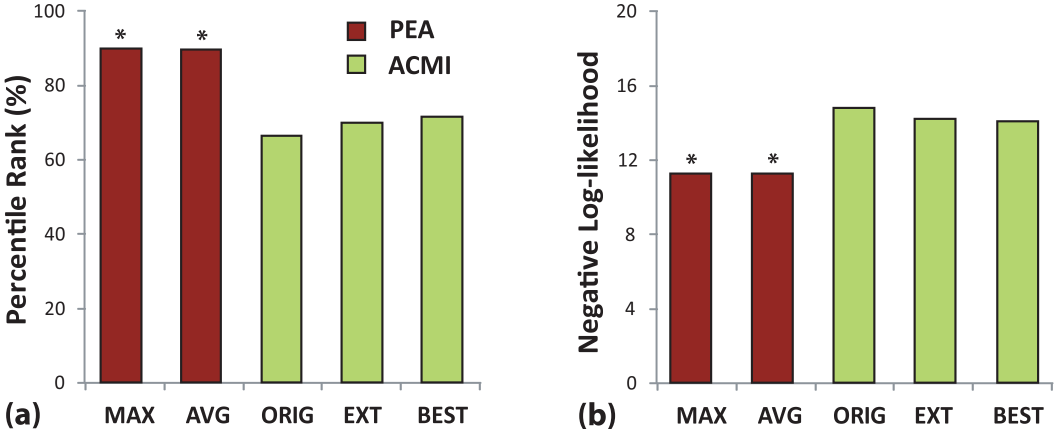


Figure 6.3: Accuracy of inference solutions across the set of ten experimentally phased electron-density maps in Section 4.1.2. In a) percentile rank of the true solution's probability. A higher percentile means the algorithm puts the true location of an amino acid closer to the top of a list of sorted probabilities. In b) the negative log-likelihood of the true solution. Lower scores mean a higher probability value for the correct answer. In both, columns are the average score over all amino acids in all test-set proteins. Dark (red) bars represent variations of PEA, while light (green) bars represent variations of ACMI. An * denotes a statistically significant difference with ORIG at $p < 0.01$.

by comparing the accuracy of the Phase 2 outputs by the two approaches (single-inference Acmi and ensemble method Pea). In Section 6.4.2, I feed these Phase 2 probabilities into Phase 3 to measure the accuracy of the all-atom protein-structure models produced by Pea and Acmi. Lastly, Section 6.4.3 shows how the accuracy of Pea changes as the number of ensemble components increases.

6.4.1 Approximate Inference

My first experiment assesses the quality of approximate-inference solutions produced in Phase 2 for both Acmi and Pea by examining the accuracy of posterior marginal probabilities. In this experiment, Acmi and Pea use the same Phase 1 outputs to run their respective Phase 2 algorithms and halt before executing Phase 3. Pea runs Phase 2 with the four ensemble components specified in Table 6.1. Given these ensemble components, I measure the accuracy of the maximum score aggregator from Equation 6.3 and the average score aggregator from Equation 6.2 (MAX and AVG, respectively in the results). The sampling algorithm from Equation 6.4 performs aggregation as a step in Phase 3 and cannot be compared here.

To compare Pea against Acmi, I consider three controls detailed in Table 6.2: the original, round-robin protocol (ORIG), an extended run of inference (i.e., 160 iterations) in Acmi (EXT), and the best-performing *individual* component of Pea (BEST). BEST is not a realistic scenario for running Acmi since I identify the top performer using the true solution, but it provides an overly optimistic, experimental control against the ensemble method to see if any gains obtained are from combining many weak solutions or from generating one strong solution.

Figures 6.3a and 6.3b show the results of running these techniques on a set of difficult protein images. Figure 6.3a shows the percentile rank which represents how highly ranked the correct solution (i.e., location from the deposited structure in the PDB) is in the posterior marginal probabilities. The percentile rank, defined in Section 4.3.2, measures the probability score for the true amino-acid location relative to all other locations in the map. The optimal score of 100 means the true location had the highest probability value in the map. In Figure 6.3b, the negative log-likelihood is the probability value for the true location, transformed as a negative-log score (see Section 4.3.1 for details). Here, I desire lower values as they indicate higher probabilities. In both figures, one column represents the average score across all 10 test-set proteins and across all amino acids in those proteins. Columns are color coded based on whether they are instance of Acmi (light green) or Pea (dark red).

Both figures show that the ensemble method, Pea, drastically outperforms the existing, single inference version of Acmi across all protocols. Both the maximum and average aggregators obtain scores in the 89th percentile compared to the original Acmi protocol which averages scores in the 66th percentile. This implies that, on average, there are three times as many false positives in Acmi versus Pea. The negative log-likelihoods tell a similar story; the probability scores improve by over three orders of magnitude by using ensembles.

One explanation of the results is that by generating four different solutions, Pea is simply utilizing one good model that is itself better than the ORIG protocol. The results for the best *individual* component of Pea (BEST), however, are only slightly better than standard Acmi, showing that Pea benefits from combining multiple, good models rather than from generating one very good model. Another alternate explanation is that Pea is receiving more CPU resources (i.e., the sum of resources for each ensemble component) and thus has an unfair advantage over ORIG. The extended run of standard Acmi (EXT) does show minor improvements over ORIG, but comes nowhere near the performance of Pea, showing that the gains of my ensemble method cannot be explained away by the increased CPU resources. In

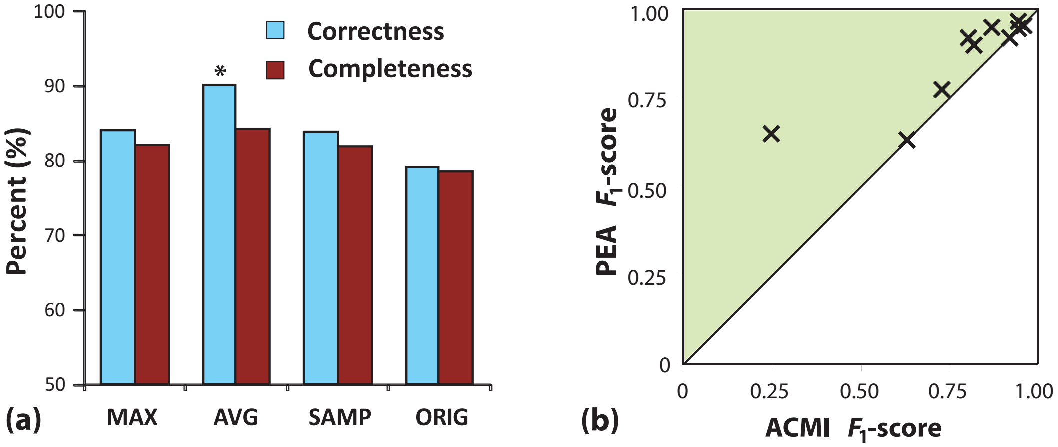


Figure 6.4: Protein-structure prediction accuracy. In a) correctness (light blue) specifies the percent of predicted structure within 2 Å of the true answer, averaged over all proteins from the experimentally phased map test set. Completeness (dark red) is the percent of the true structure a method predicted within 2 Å. ORIG is the standard Acmi algorithm and MAX, AVG, and SAMP are variations of PEA. An * indicates statistically significant difference compared to ORIG at $p < 0.05$. In b) I show a detailed comparison of F_1-scores for ACMI (x-axis) and PEA using the averaging aggregator (y-axis). F_1 is the harmonic mean between the correctness (precision) and completeness (recall) metrics. Each point represents one protein and the shaded region indicates better scores for PEA.

addition, Pea's use of resources are trivially parallelizable. The results of all pairwise differences between the Pea variations and the three Acmi variations are statistically significantly at scores of $p < 0.01$ for both metrics in Figure 6.3 based on a paired t-test.

6.4.2 Protein Structures

While the previous results indicate my ensemble technique improves the accuracy of approximate-inference probabilities, biochemists are more interested in the actual protein structures produced. As a follow-up experiment, I used the marginal probabilities from Section 6.4.1 as the input for Phase 3 of the Acmi and Pea algorithms, respectively, to produce all-atom protein structures for all 10 of my test-set proteins. I use the completeness and correctness (see Section 4.2.1 for details) of the resulting protein structures to compare my proposed aggregators for Pea against Acmi.

Figure 6.4a shows the averaged results of my experiments. The first three pairs of columns represent, respectively, the maximum (MAX), average (AVG), and sampling

(SAMP) aggregators for Pea presented in Section 6.2.2. The fourth pair of columns represent the original Acmi protocol (ORIG). Within each pair, the first column represents the *correctness* of the predicted protein structure (akin to *precision* in information retrieval). The second column represents the *completeness* of the predictions (akin to a *recall* metric). Each column represents an average over all ten test proteins. The top performer across both metrics was Pea using the averaging function to aggregate ensemble components. On average, 90.3% of its predicted amino-acid locations were correct (compared to 79.3% for the original Acmi algorithm), while completing 84.3% of the real structure (78.6% for Acmi). Importantly, all three Pea methods outperform Acmi in both correctness and completeness measures.

Figure 6.4b provides a closer comparison of Pea versus Acmi. Here, each datapoint represents the results of one protein in my test set. The x-axis value is the accuracy of the original Acmi algorithm and the y-axis is the accuracy of Pea using the average aggregator. To assess accuracy, I use an F-measure to combine the correctness and completeness metrics from Figure 6.4a. The F-measure is commonly used in the information retrieval community to balance *both* the need for high precision and high recall, rather than looking at the metric individually. Here, I use the traditional F_1 metric, which is the harmonic mean of correctness and completeness:

$$F_1 = 2 \cdot \frac{correctness \times completeness}{correctness + completeness}. \tag{6.5}$$

The line represents equivalent performance, and the shaded region represents values where Pea outperforms ACMI. In every test case, Pea performs better than or equal to Acmi in the F_1 metric, affirming the results from Figure 6.4a. The largest improvement comes in the most difficult test case, with the F_1-score improving from 0.25 to 0.66. This corresponds to an extra 41 percentage points of the true structure being built and 42 percentage points of extra predictions being correct. Overall, Pea shows substantial improvement in 6 of the 10 proteins with equal performance in the other 4, although these values are not statistically significant. The variance in overall performance is not correlated with either the size of the protein or image, but indicative of the range in image quality in my test set.

Figure 6.4b only considers the average aggregator for Pea since it performed better than the alternative options. As hypothesized in Section 6.2.2, the averaging aggregator's main advantage is that it can smooth away "noisy" probabilities. That is, as seen in Figure 6.3, Acmi's individual inference runs contain many incorrect locations (i.e., false positives) with higher probabilities than the correct solution.

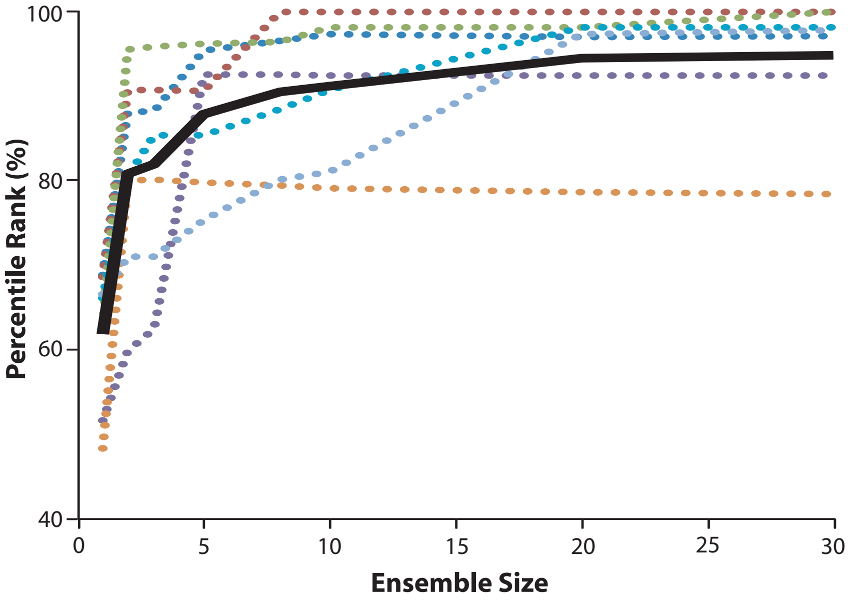


Figure 6.5: Learning curves for ensemble inference. Each dashed line represents one protein's percentile scores for Phase 2 posteriors as the number of ensemble components increases. Proteins were taken from the experimentally phased test set. The solid black line represents the average learning curve.

This makes it more difficult for Phase 3 to find the correct location for the structure during sampling. If an ensemble produces diverse solutions, however, than the "noisy" locations should be independent between runs. When averaged together, these incorrect peaks are smoothed away since they are low in probability in most components, and the signal from the true location will be boosted since its signal is detected by multiple inference runs. Thus, while each individual run produces moderate results, the average together reveals only a handful of possible solutions. The maximum aggregator and sampling aggregator also produced improved inference probabilities, but did not translate into the same level of improvement in structure quality as the averaging aggregator. It is difficult to pinpoint the exact reason, but the areas of major difference happened to be in regions of the map with the least amount of signal, implying the averaging aggregator handles noise the best.

6.4.3 Ensemble Learning Curve

As a last experiment, I consider how the size of an ensemble effects the accuracy of inference in Pea. Due to resource limitations, I could not run larger ensembles sizes for the previous experiments. Instead, for the seven smallest test-set proteins, I generated ensembles with various number of components, ranging from 1 to 50. I assessed each using percentile rank scores as described for Figure 6.3a. Figure 6.5 shows the learning curve for seven of my test-set proteins as the number of ensemble components increases (the values past 30 are not shown since no change occurred).

Pea, here, uses the mixture-model average aggregator to combine posteriors. The light, dashed lines represent the inference results for one test-set protein while the thick, black line represents the average performance across the seven proteins. As the figure shows, Pea gains accuracy from adding more components, making its largest leap in performance with the first 10 ensemble components before seeing very little improvement after 20 component ensembles. As such, I would expect even more accuracy in producing protein structures by increasing the sample size in Section 6.4.2's experiments.

6.5 Summary

While Acmi was previously shown to outperform other automated density-map interpretation methods in building all-atom protein structures in low quality electron-density maps [19], performing approximate inference in Acmi's model is an expensive process in need of advanced inference methods. In this work, I developed a new approximate-inference method based on the concept of ensemble-learning methods from the supervised machine learning community. My new framework, Probabilistic Ensembles in Acmi (Pea), executes several independent runs of inference to provide multiple, diverse solutions to the problem. I suggest several protocols for generating unique solutions for each component of the ensemble, as well as different techniques for aggregating these models to produce a single, accurate prediction of the protein structure.

My results show Pea provides improved performance on a test-set of 10 difficult protein images. This improvement is seen in the accuracy of the inference process, where the probability distributions from Pea were statistically significantly better in terms of both percentile rank and probability value assigned to the correct location of each amino acid. The results show that this improvement could not be explained by either the extra CPU resources utilized or by using the single-best component of Pea. More importantly, Pea's improved inference translates into more complete and correct protein structures.

In the future, I seek to test Acmi on a larger set of proteins, including membrane proteins which present many difficulties for crystallographers [9]. Also, the learning curve in Section 6.4.3 indicates Pea can demonstrate even better performance with an increase in ensemble size. Further work is needed to manage the computational resources required to accomplish this task.

While I presented ensembles of approximate-inference solutions for the task of protein-structure determination, my method can generally be applied to difficult inference problems where the complexity of probabilistic graphical models limits the accuracy of current methods. In future work, I look to find such applications, and to provide an in-depth comparison to related inference techniques that rely on simplifying the graph structure [84].

7 Spherical-Harmonic Decomposition for Molecular Recognition in Electron-Density Maps

In Section 3.2.1, I introduced our group's prior work on Acmi-FF, a 3D shape-matching algorithm for computing the density correlation between a protein fragment from the Protein Data Bank and a local area of the electron-density map. Generalized as Phase 1, this process requires a six-dimensional search (three in translations, three in rotations) to yield an observation potentials (or local match score) for each amino acid. Acmi-FF uses fast Fourier transforms (FFTs) to reduce the complexity of this computation over the three *translational* dimensions.

In this chapter, I report work done jointly with Frank DiMaio which uses *spherical-harmonic decompositions* instead of FFTs to reduce the complexity over the *rotational* dimensions. Spherical harmonics are analogous to a Fourier series, but mapped into spherical coordinates, allowing an efficient comparison of 3D two objects at different rotations. Our new method, Acmi-SH, is more efficient than Acmi-FF. It also produces more accurate observation potentials and more accurate protein structures. In addition, Section 7.3 introduces my work on using machine-learned classifiers to *a priori* filter the search space, thus also reducing the complexity over translational dimensions. I propose several classification methods which successfully reduce Phase 1's search space by four fold while also improving Acmi's accuracy.

While the previous two methods apply to Phase 1 in the roadmap, I also propose Shed, a method which extends the use of spherical-harmonic decompositions in Phase 1 to a different problem – identifying homologous structures in an unsolved electron-density map. Shed is a useful exploratory tool, recalling more homologous structures from the PDB than sequence-based methods alone, such as Blast. The work in this chapter appears in DiMaio et al. [23], an extension of our initial work [22].

7.1 Introduction

As outlined in Section 3.3, one important subprocess of Acmi is Phase 1 – a local, feature recognition algorithm for detecting putative locations for each amino acid in the electron-density map. This task is generally one of *shape matching* – computing the correlation between two three-dimensional objects in a manner that is robust to changes in rotation, translation, and/or scale[1]. As described in Section 3.2.1, this search process requires comparing a set of template fragments from the Protein Data Bank (PDB) against every location in the density map (i.e., three translation dimensions), over all rotations (i.e., three rotation dimensions), for each amino acid in the sequence. This 6D search is computationally expensive, requiring efficient methods for exploring the search space. In prior work, DiMaio et al. approached this task using the algorithm Acmi-FF (for **F**ast **F**ourier feature recognition) [21]. This method utilizes fast Fourier transforms (FFTs) [12] to quickly perform a comparison between two "signals" across all three-dimensional translations. Here, the signals are the density map and the simulated density of a protein fragment[2], and the comparison is the squared-density difference equation in Equation 3.1. While efficient over translations, FFTs do not reduce the complexity of the rotational search.

In this chapter, I introduce a series of algorithms utilizing *spherical-harmonic decompositions* [49] to perform shape-matching. Spherical harmonics are analogous to a Fourier series, but mapped into spherical coordinates. That is, spherical harmonics represent shapes as a set of spherical basis functions with coefficients, as shown in Figure 2.10. Importantly, spherical-harmonic decompositions (detailed in Section 2.2.4) lends to methods that match two signals efficiently over *rotations*.

In Section 7.2, I present joint work done with Frank DiMaio on a new algorithm for Phase 1, Acmi-SH (for **S**pherical **H**armonic feature recognition), which uses spherical-harmonic decompositions to score matches between template fragments and local regions of the electron-density map. This decomposition lets us efficiently match all rotations of the template fragment at a single location. This technique produces more accurate observation potentials and, consequentially, more accurate protein structures than Acmi-FF.

Second, Section 7.3 details methods for further reducing the Phase 1 search space by developing classifiers that, *a priori*, filter out translational search locations (i.e., eliminate certain (x, y, z) coordinates from consideration). Acmi-FF reduced the 6D search of Phase 1 by improving performance across all possible translations, but only obtains this speed up by using the entire density map. Conversely, a "convolution" over rotations allows me to mask – that is, to eliminate from consideration – some (x, y, z) locations in the density map. I propose and evaluate a set of "first-pass filters"

[1] Scale invariance is not an issue in electron-density images because all maps are measured using an absolute scale (e.g., Å) and the objects (i.e., atoms) are, for the most part, fixed in size from one image to another.

[2] Protein structures are stored as point estimates; programs, like Sfall [11], model the expected electron density from this point estimate.

that eliminate points that are not likely to match any template. A beneficial filter will speedup Acmi, but also increase accuracy by removing possible false positives from the density map and allowing greater granularity in the local template-match search. One such filter achieves a four-fold reduction in search locations by utilizing spherical harmonics to generate a set of rotation-invariant features for use with supervised learning methods.

Lastly, in Section 7.4, I describe an extension of the fast rotation function, from a) finding small-fragment matches to a density map, to b) searching for whole-protein matches to a density map. This whole-protein search, Shed (**S**tructural **H**omology using **E**lectron **D**ensity), detects structural homologs *without* requiring the actual protein structure of the target protein. Results show that Shed detects several structural homologs and, while requiring a raw electron-density map, outperforms Blast [2] – a popular sequence-only, homology-detection algorithm – at finding structurally similar proteins.

7.2 Local Template Matching with Acmi-SH

With Equation 3.3, I stated that Acmi uses a probabilistic model which ties local density features with global constraints. A key component in this equation is the first term, the observation potential, $\psi_i(\vec{u}_i)$, associated with each amino acid. In my roadmap for Acmi, this potential is calculated by Phase 1. In prior work, Acmi-FF (Section 3.2.1) performed this local template-matching procedure Acmi-FF. Phase 1, generally, scores pentapeptide representations of each amino acid, independently, against all locations and orientations in the density map. This "amino-acid finder" algorithm is a computationally intensive calculation. More importantly, as the only source of evidence from the density map in Acmi's Markov random field model, Phase 1's accuracy is critical to producing a quality solution. This section describes Acmi-SH [22], a method for calculating observation potentials using spherical-harmonic decompositions and the fast rotation function listed in Section 2.2.4. An overview of our local match procedure appears in Algorithm 7.1, and is illustrated in Figure 7.1.

7.2.1 Methods

As in Acmi-FF, we use pentapeptide fragments from the PDB as a proxy for the amino acid of interest in our search. Acmi-SH retrieves fragments from the PDB using the same PAM-120 similarity metric as Acmi-FF for detecting sequence similarity. In contrast to Acmi-FF, however, Acmi-SH does not cluster these fragments. Rather, the

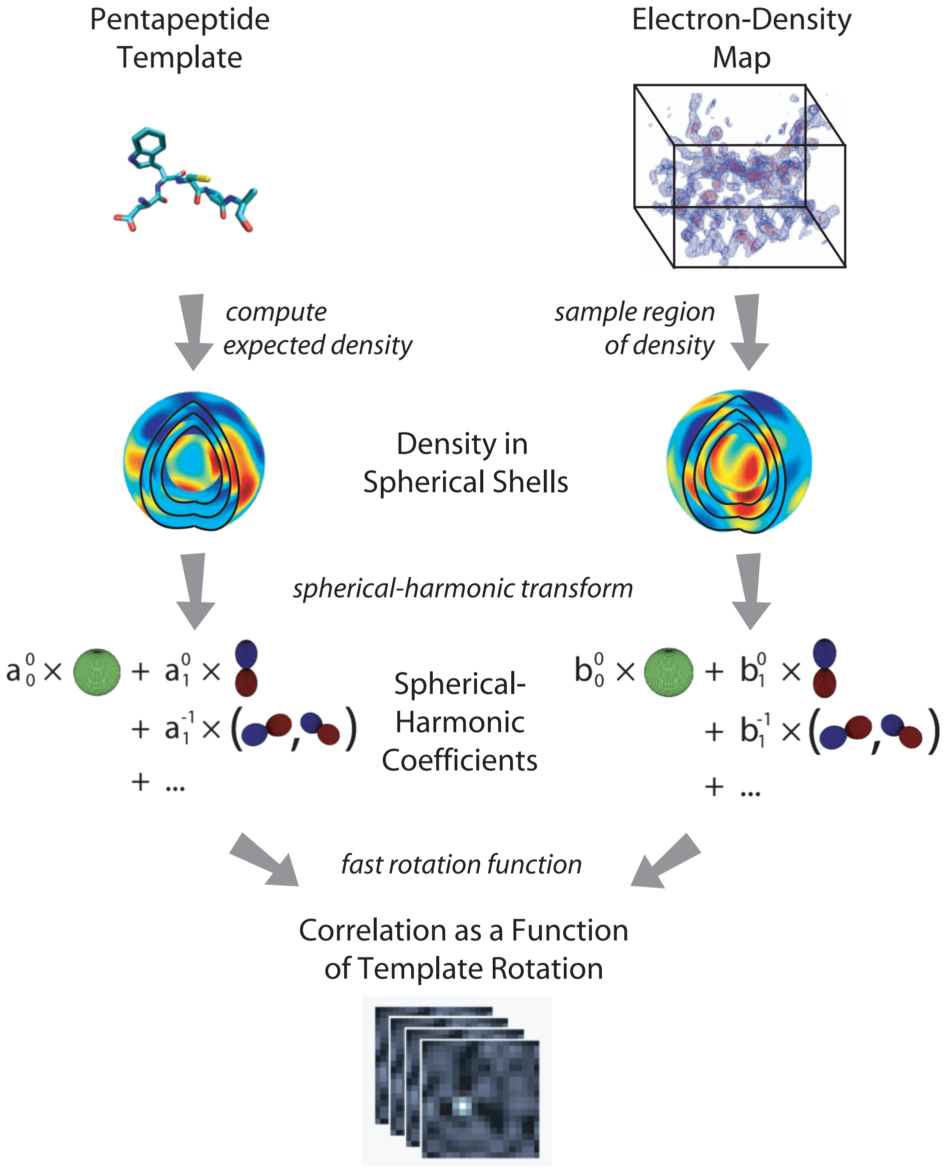

Pentapeptide Template
Electron-Density Map
compute expected density
sample region of density
Density in Spherical Shells
spherical-harmonic transform
$a_0^0 \times$ + $a_1^0 \times$ + $a_1^{-1} \times$ (,) + ...
Spherical-Harmonic Coefficients
$b_0^0 \times$ + $b_1^0 \times$ + $b_1^{-1} \times$ (,) + ...
fast rotation function
Correlation as a Function of Template Rotation

Figure 7.1: Acmi Phase 1's improved template-matching algorithm, Acmi-SH. Given some pentapeptide template (left) Acmi-SH first calculates the template's expected electron density. On the right, Acmi-SH samples a spherical region from the electron-density map. Then, it calculates spherical-harmonic coefficients for both objects, and uses the fast rotation function to compute cross correlation as a function of template rotation.

time savings described later allow us to consider each fragment itself as a search template.

Algorithm 7.1: Local Template Matching with Acmi-SH

```
input  : amino-acid sequence $Seq$,
         density map $\mathbf{M}$
output : vertex potentials $\psi_i(\vec{u}, \vec{r})$ for $i = 1 \ldots N$

// Calculate distribution of random correlations
$(\mu_{CC}, \sigma_{CC}) \leftarrow$ learn-from-tuneset()

foreach residue $i$ do
    // Find fragments with a similar 5mer sequence
    $\mathbf{PDBfrags}_i \leftarrow$ lookup-in-PDB($Seq_{i-2:i+2}$)

    foreach $frag \in \mathbf{PDBfrags}_i$ do
        // Calculate SH coefficients for fragment
        $template \leftarrow$ compute-dens($frag$)
        $templCoef \leftarrow$ SH-transform($template$)

        foreach point $\vec{u}_j \in \mathbf{M}$ do
            if is-filtered-out($u_j$) then next $u_j$

            // Calculate SH coefficients for the local region of the map
            $signal \leftarrow$ sample-dens-around($u_j$)
            $sigCoef \leftarrow$ SH-transform($signal$)

            // Calculate cross-correlation between fragment and map; convert to
                probability
            $\mathbf{CC} \leftarrow$fast-rotate($templCoef, sigCoef$)
            foreach rotation $\vec{r}_k \in \mathbf{R}$ do
                $z_k \leftarrow (\mu_{CC} - CC_k)/\sigma_{CC}$
                $p_{null} \leftarrow$ normCDF $(z_k)$
                $\psi_i(\vec{u}_j, \vec{r}_k) \leftarrow (1 - p_{null})/p_{null}$
            end
        end
    end
end
```

When calculating the match for some pentapeptide against the density map, Acmi-SH begins by computing the density we would expect to see given the pentapeptide using Sfall [11]. We then interpolate this calculated density in concentric spherical shells extending out to 5 or 6 Å in 1 Å steps. This is done to cover most of the density in an average pentapeptide. A fast spherical-harmonic transform computes spherical-harmonic coefficients corresponding to each spherical shell using a recursion similar to that used in fast Fourier transforms [37]. In parallel, we produce a similar set of coefficients for the density around some grid point in the electron-density map. Given these two sets of spherical-harmonic coefficients – $\vec{a}$ corresponding to the template and $\vec{b}$ corresponding to some location in the density map – a fast implementation of Equation 2.12 computes the *cross correlation* over all rotations of the template pentapeptide. Acmi-SH uses the implementation of Kostelec and Rockmore [52].

After computing the cross correlation (CC), we compute the vertex potential $\psi_i(\vec{u_i})$ as the probability that a particular cross correlation value is *not* generated by chance. That is, we assume that the distribution of the cross correlation between some template's density and some random location in the density map is normally distributed with mean μ and variance σ^2:

$$CC^* \sim \mathcal{N}(x; \mu, \sigma^2). \tag{7.1}$$

We estimate these parameters, μ and σ^2, by computing cross correlations between the template and random locations in the map. Given some cross correlation score, cc_k for some rotation index k, we compute the expected probability that we would see score cc_k or higher by random chance,

$$p_{null}(cc_k) = P(CC \geq cc_k; \mu, \sigma^2) = 1 - \Phi((cc_k - \mu)/\sigma). \tag{7.2}$$

Here, $\Phi(z)$ is the normal cumulative distribution function. Each amino acid's potential is then $(1 - p_{null})/p_{null}$.

7.2.2 Results and Discussion

In DiMaio et al. [22], we evaluate Acmi-SH by running the method against a battery of three tests. Herein, I detail only the results from the one experiment that is part of my contribution. The two additional tests can be found in the original work or in Frank DiMaio's thesis [18]. Specifically, the first of the two experiments measures the error rate of using a spherical-harmonic representation. In other words, how much information is lost when considering some finite set of spherical-basis functions. This data helps set the bandwidth parameter, B, for Acmi-SH between 8 and 12 basis functions, and shows spherical-harmonic decompositions represent protein structure with high fidelity. Second, DiMaio et al. compares the accuracy of Acmi-SH against Acmi-FF using the negative log-likelihood metric from Section 4.3.1. The results show Acmi-SH, on average, produces more accurate observation potentials by two orders of magnitude across the model-phased test set in Section 4.1.1, and in

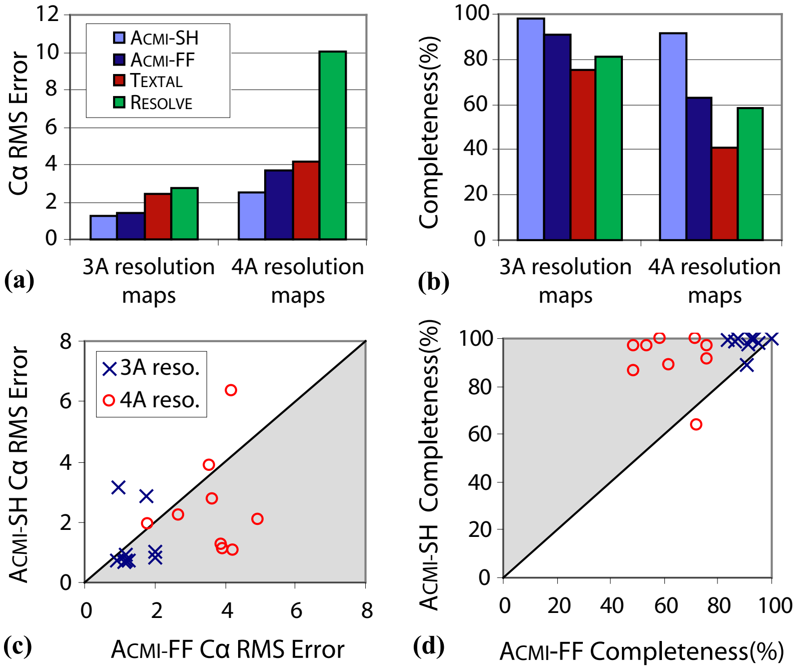


Figure 7.2: Comparing Acmi-SH's protein models with three other methods. The average scores across all test-set proteins are shown in (a) for Cα RMS error and (b) for completeness – the percentage of amino acids from the true solution that are predicted (within 2Å) by the respective method. The scatter plots compare Acmi-FF's performance with Acmi-SH's individually for each test-set protein on (c) RMS error and (d) completeness percentage. For (c) and (d) the shaded regions indicate superior performance by Acmi-SH. The proteins in this experiment come from the model-phased test set in Section 4.1.1.

less computational run time.

Here, I detail the results of our final experiment, which compares the performance of Acmi with both Phase 1 methods (i.e., Acmi-FF and Acmi-SH) against two other automated techniques specialized to low-resolution maps: Ioerger's Textal [42] and Terwilliger's Resolve [82], both described in Section 2.1.3. For the Acmi methods, Phase 1 is either the prior method (Section 3.2.1) utilizing Acmi-FF for local matching, or Acmi-SH. Following Phase 1, we run the original implemention of Phase 2 (i.e., Acmi-BP), specified in Section 3.2.3. Finally, Phase 3 produces a Cα backbone trace using Equation 3.8[3]. In figures and discussion below, Acmi-FF represents this pipeline with Acmi-FF being used in Phase 1; similarly Acmi with

[3] This work precedes Chapter 8 in the development of Acmi, so Acmi only produces backbone traces and not all-atom protein models for our results in this chapter.

our new proposal technique represents Acmi-SH.

We utilize the model-phased test set from Section 4.1.1 where we truncate the resolutions of ten density maps to 3 Å and 4 Å. To compare the accuracy of the models, we utilize two metrics. First is the Cα RMS error, defined in Section 4.2.2, which finds the best alignment between the predicted backbone trace and true solution and measures the average root-mean squared distance between the respective Cα atoms. Lower scores imply a closer alignment and thus higher accuracy. Second, we report the completeness of predictions, discussed in Section 4.2.1, which is the percent of amino acids from the true solution properly predicted. Higher scores indicate better predictions. Figures 7.2a and 7.2b show the average Cα RMS error and percentage completeness over the ten structures for each method. A more detailed versus plot is shown in Figures 7.2c and 7.2d , where each point is a test-set protein at either 3 or 4 Å simulated resolution. The x-axis indicates Acmi-FF's score while the y-axis shows the value for Acmi-SH. The shaded regions indicate points with better results for Acmi-SH (i.e., lower Cα RMS error and higher completeness percentages, respectively).

Acmi-SH shows a clear improvement over all other approaches. For 3 Å maps, Acmi-SH shows slightly lower RMS error than Acmi, but roughly 1.0 Å and 1.3 Å lower error than Textal and Resolve, respectively. With the 4 Å maps, however, Acmi-SH's advantage increases dramatically, producing structures with significantly less error than all other methods. When comparing Acmi-FF to Acmi-SH, Acmi-SH rarely performs poorly, with 8 of 10 structures having less than 1 Å of error in the 3 Å maps and 7 of 10 maps performing better than Acmi-FF in 4 Å maps. One potential problem with the RMS-error metric is that a method can improve its score by being conservative with its predictions since the error does not penalize for amino acids not predicted (e.g., predicting 5 amino acids with no RMS error is better than predicting 500 with 1 Å of error). Figures 7.2b and d, however, show that Acmi-SH's lower Cα RMS error occurs in conjunction with *more* complete predictions. In 3 Å maps, Acmi-SH achieves almost 100% completeness and stays above 90% for 4 Å maps as well. This is substantially higher than other methods, particularly at 4 Å where Acmi-FF comes in second place with 62% completion. The scatter plot shows this advantage occurs consistently, with only one protein at 4 Å performing better under Acmi-FF.

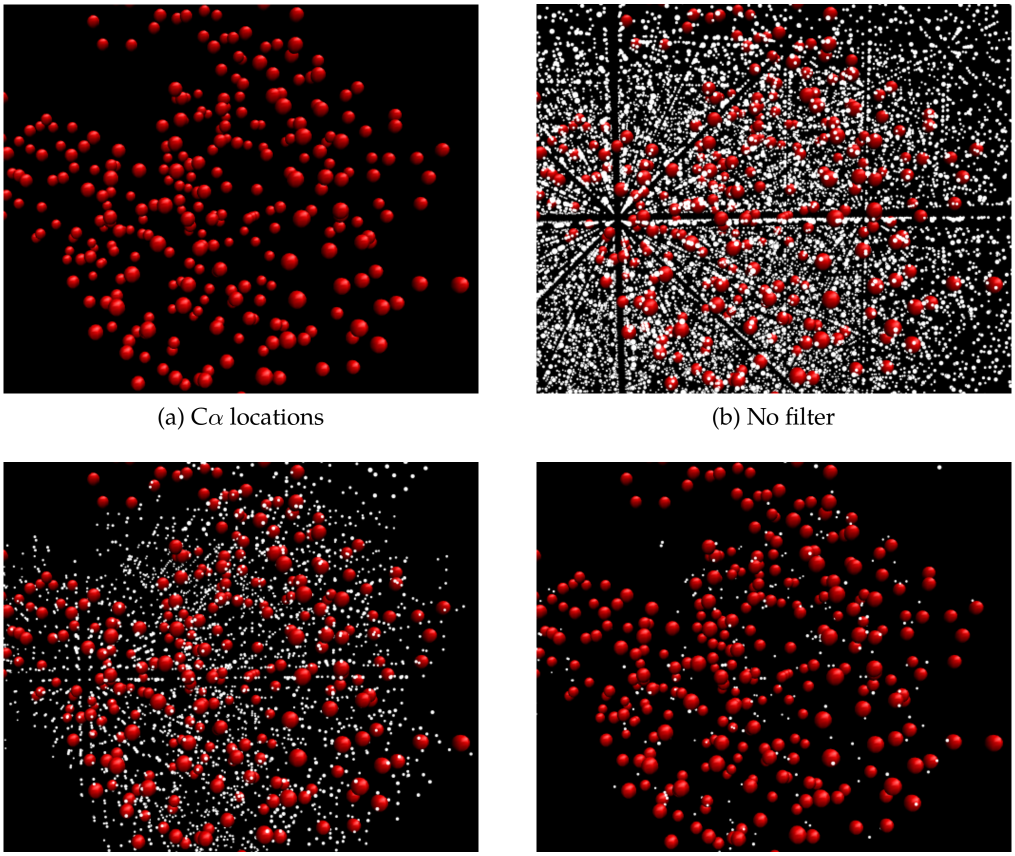
The accuracy increase in using spherical harmonics likely comes from several different places. The increased efficiency allows a finer angular sampling: the bandwidth limit $B = 12$ in Acmi-SH is analogous to a 15° angular spacing for

Acmi-FF. This increased efficiency also lets us search for each individual template – without clustering – which may also help accuracy since Acmi-SH can explore a larger set of conformations for each amino acid. Finally, band-limiting the signal, which throws out the highest-frequency components, may help eliminate noise from the density map allowing preventing the match search from overfitting to noise. Even with this improved accuracy, the running time of Acmi-SH is about 60% of that of Acmi-FF using the settings above and the point-density filter in Section 7.3.1 for eliminating 80% of search locations.

7.3 Filtering Methods to Prune Acmi-SH Search Space

Acmi-FF (Section 3.2.1) performs Fourier convolutions over the entire electron-density map to efficiently match a template to a map. One disadvantage to this approach is that the entire map must be considered in every calculation. In protein-structure determination, however, the number of locations containing a template match is small compared to the size of the map; that is, there are on the order of 1 Cα's per 1000 grid points. Fortunately, Acmi-SH does not require a search at all locations since rotational alignments are done independently at each point in the map. A significant reduction in computation could be achieved if I can efficiently eliminate the areas of the map not containing templates before performing a fast rotation alignment to each template.

Figure 7.3 illustrates how such a "first-pass filter" would effect the search space for Acmi-SH. In Figure 7.3a, I show the locations of the known Cα atom locations for a sample protein in red. These values are taken from the final PDB file, but is not available to Acmi in practice. Figure 7.3b shows, in white, the search locations for a fully exhaustive template-match search in Acmi-SH – a spherical-harmonic decomposition and fast-rotation match is performed (Figure 7.1) at every (x, y, z) grid point. A good filter will substantially reduce the search space as in Figure 7.3c by eliminating locations unlikely to contain a Cα. Here, the hypothetical filter removed a majority of the grid space from the template-match search. Filtering, however, can lead to a reduction in performance if it is too aggressive and eliminates true positives. Figure 7.3d shows that such an instance; Acmi-SH misses a significant number of actual Cα locations in its search.



(a) Cα locations

(b) No filter

(c) Moderate filter

(d) Aggressive filter

Figure 7.3: Illustration of a first-pass filter in ACMI-SH. In a), I show (in red) the true Cα locations, unknown to the filter. Without a filter, ACMI-SH must do a full-rotation match at every grid location, shown in b) as white dots. A filter can dramatically reduce this search space, as shown with a moderate filter in c) and a more aggressive filter in d).

7.3.1 Methods

In DiMaio et al. [22], Frank DiMaio and I compared several simple "first-pass filters" that use information from the density map to estimate the likelihood that a template is centered at some location in the map. Three of these filters are based upon the observation that in density maps, especially poor-resolution maps, Cα locations correspond to the highest-density points in the map [55]. We consider filtering points based on the grid point's density, as well as the average density in a 2 or 3 Å radius around each point. The example in Figure 7.3 uses such a filter.

We also consider a filter based on the *skeletonization* of the density map [35].

Skeletonization, similar to the medial axis transformation [7] in computer vision, gradually "erodes" the density map until it is a narrow ribbon approximately tracing the protein's backbone and (in high-resolution maps) side chains. We consider filtering each point based upon its distance to the closest skeleton point. This is also the first-pass filter used by Capra [41] to eliminate points from the density map.

Finally, in DiMaio et al. [23], I developed an additional filter based on a set of rotation-invariant descriptors derived from spherical-harmonic coefficients. While spherical-harmonic coefficients are not rotation invariant, Kondor [51] theoretically describes a transformation of these coefficients which creates a rotation-invariant set of descriptors. In extending this work to our task, I describe a region of density in a way that does not change as the region of density is rotated. Although these features are more time-consuming to compute than the previous filters, computation time is significantly less than that of a full rotational alignment of hundreds of templates at a point in the map.

Briefly, Kondor [51] generalizes the bispectrum of a Fourier series to spherical harmonics. The power spectrum of a Fourier series provides the amount of energy in each frequency band of the signal and is known to be a translation-invariant representation of a signal. However, it loses all phase information and thus cannot reconstruct the original signal. The bispectrum is a way of representing a signal in a way that is shift-invariant, yet uniquely identifies the original signal (up to translational shifts). The power spectrum can be shown to be the Fourier transform of the *autocorrelation function*:

$$corr(x) = \sum_{y=0}^{n-1} f(y+x) \cdot f^*(y) \tag{7.3}$$

where $f(x)$ is the signal (e.g., density values), $f^*(x)$ is the complex conjugate of the signal, and y is an offset.

The central idea behind the bispectrum is to instead use the Fourier transform of the *triple correlation function*:

$$a(x_1, x_2) = \sum_{y=0}^{n-1} f^*(y-x_1) \cdot f^*(y-x_2) \cdot f(y). \tag{7.4}$$

Kondor [51] provides an in-depth proof generalizing this equation to functions on a sphere. The bispectrum representation – given a function band-limited to bandwidth B – produces O(B^3) descriptors that are invariant to rotations of the original signal, yet are able to uniquely reconstruct the signal (up to rotations).

Using these O(B^3) features, I consider training a support vector machine (SVM) [14] to recognize whether a region will match any template in my data set. Non-linear SVMS are a supervised learning method that learns a set of weights, α_i, for each example (corresponding to some distance – or *kernel* – function $K(x_i, x_j)$); when a new example x is encountered, the weighted sum of distances $\sum_i \alpha_i K(x, x_i)$ is used to classify the example. Thresholding this sum at different values allows me to trade off the precision and recall of the classifier.

7.3.2 Results and Discussion

I divide my experiments into two sections; the first compares the simple density filters against one another. "Simple" here refers to the fact that these methods do not require any training set to learn parameter weights. The second experiment compares the top-performing simple filter against my proposed method of using the bispectrum of spherical harmonics to train a support vector machine. Each method discussed evaluates each grid point in the density map and decides if the location is likely to contain a Cα (and thus should be part of the search in ACMI-SH) or not.

7.3.2.1 Simple Density Filters

As specified in Section 7.3.1, there are four simple filters which rely on only the map in question: point density, skeletonization, average over 2 Å sphere, and average over 3 Å sphere. Each of these filters outputs a real value score, normalized between 0 and 1 with higher scores indicating a great chance that a grid location contains a Cα. Jointly with Frank DiMaio, I compared the performance of these four simple filters using the model-phased maps from Section 4.1.1. The results at both 3 Å and 4 Å resolution are shown in Figure 7.4. To create these plots, all grid-points were first sorted according to the filter value, with higher values going closer to the top of the list. These plots show, on the x-axis, the portion of the entire map we consider as we move the threshold further down the sorted list. The y-axis shows the fraction of true Cα locations above the threshold. For example, a point at coordinates $(0.2, 0.9)$ means a filter for which – at *some* threshold value – we look at only 20% of the density map and still find 90% of the true Cα locations. This is similar to an ROC curve, except the x-axis measures all points considered (i.e., false positives plus true positives). An optimal method places all true Cα locations at the top of the sorted list, yielding a value of 1 on the y-axis with an x-axis value close to 0.

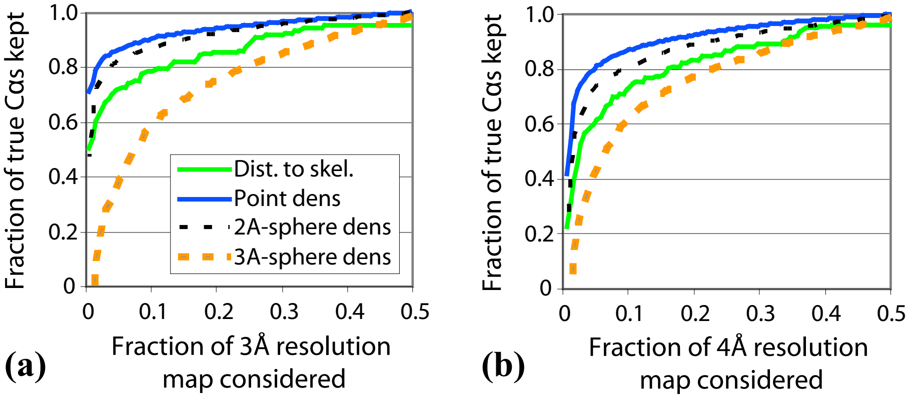


Figure 7.4: A comparison of four different filters for quickly eliminating some portion of points in the density map. Filter performance is compared on (a) 3 Å and (b) 4 Å resolution density maps from the model-phased test set.

Somewhat surprisingly, the simplest filter, the point density, performs the best at both resolutions at all thresholds. This is surprising considering the point-density filter ignores features in the region surrounding a grid point and would seem more susceptible to noise. The other methods, however, are likely smoothing out distinctive features by averaging the density and thus losing important information. The experiments in Section 7.2 used the point-density as a first-pass filter in the Acmi-SH method, eliminating a conservative 80% of the density map from rotational search.

7.3.2.2 Svm Filter Using Bispectrum Features

As a followup experiment, I compare the point-density first-pass filter above to a filter using a trained Svm model, as motivated in Section 7.3.1. For each point in the density map, I extract Kondor's real-valued numeric features, similar to Equation 7.4, from the spherical-harmonic decomposition of $R = 5$ concentric spheres of density centered at that point. I use a bandwidth of $B = 8$ and shell width of 1 Å, producing a set of features of size RB^3 for each point in the grid, plus the density value of the point. These values were chosen based on previous results for Acmi-SH and also to prevent an explosion in the feature set size. If the grid point lies within a short distance of a true Cα ($\leq \frac{\sqrt{3}}{2}$ grid units, the maximum distance from a Cα to its closest grid point), it is labeled as positive for being a place to search for a template, otherwise it is considered negative when I evaluate ground truth. Features are normalized per map.

An Svm model, unlike the four previous simple filters, requires training to learn a decision boundary. To properly evaluate my Svm filter, I employ the commonly used 10-fold cross validation procedure. As is typical, 10-fold cross validation divides my initial data set of examples into 10 subsets of examples. Of these 10 subsets, 9 are pooled together to train the Svm model – the algorithm takes these examples along with their ground truth labels and builds a model that learns how to separate the positive examples from the negative examples. I use the last subset for validation (i.e., testing). The ground truth labels are held aside while the examples are given to the model to predict a label. I refer to this subset as the *test set*. I can then compare the predicted labels against the ground truth labels to evaluate the accuracy of the model. This is repeated 10 times such that each subset is used exactly once for validation. In my experiment, each map constitutes one subset of examples since I have 10 maps (e.g., I train on 9 maps and use one as the test set). Since there are much fewer Cα's than points in the grid, I have a large skew in the ratio of negative to positive examples in the training partition. This can cause problems in training a model, so I modify the procedure by removing enough negative examples in the training partition to create an equal balance. This is neither necessary nor desired for the test set.

Classification is done using an RBF kernel in an Svm using the SVM^{light} [43] package. To set the complexity parameter C as well as the kernel width γ, the training examples in each fold are divided into two sets, 80% for actual training (called *training* set) and 20% for evaluating the parameters (called *tuning* set). The set of values considered for C are 1, 10, 100, and 1000 while γ ranges over 0.0001, 0.001, 0.01, and 0.1. The pair of values for C and γ that produce the largest area under the curve of the function described in Figures 7.4 and 7.5 for the tuning set is chosen for evaluation of the current fold's test set.

Figure 7.5a plots the averaged results for both the Svm filter and point-density filter on the 3 Å maps in my data set. The Svm filter outperforms the density filter over the entire graph in 3 Å maps. This result is consistent across all maps. To analyze the relative speedup of Svms over a simple filter, I look at the fraction of the map needed to acquire 95% of the correct Cα locations (0.95 on the y-axis of Figure 7.5). Using this metric, Svms provide a 31% reduction in number of examples needed to be analyzed by Acmi-SH, relative to a simple point-density filter alone. In fact, the difference in area under the curve and number of examples evaluated is statistically significantly better for the Svm filter with a p-value less than 0.001 according to a two-tailed t-test.

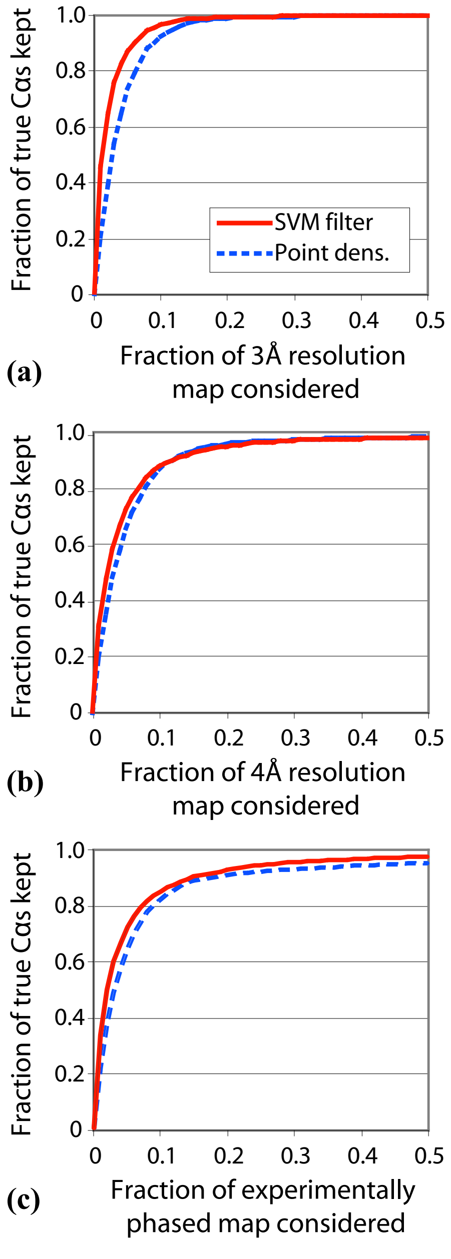


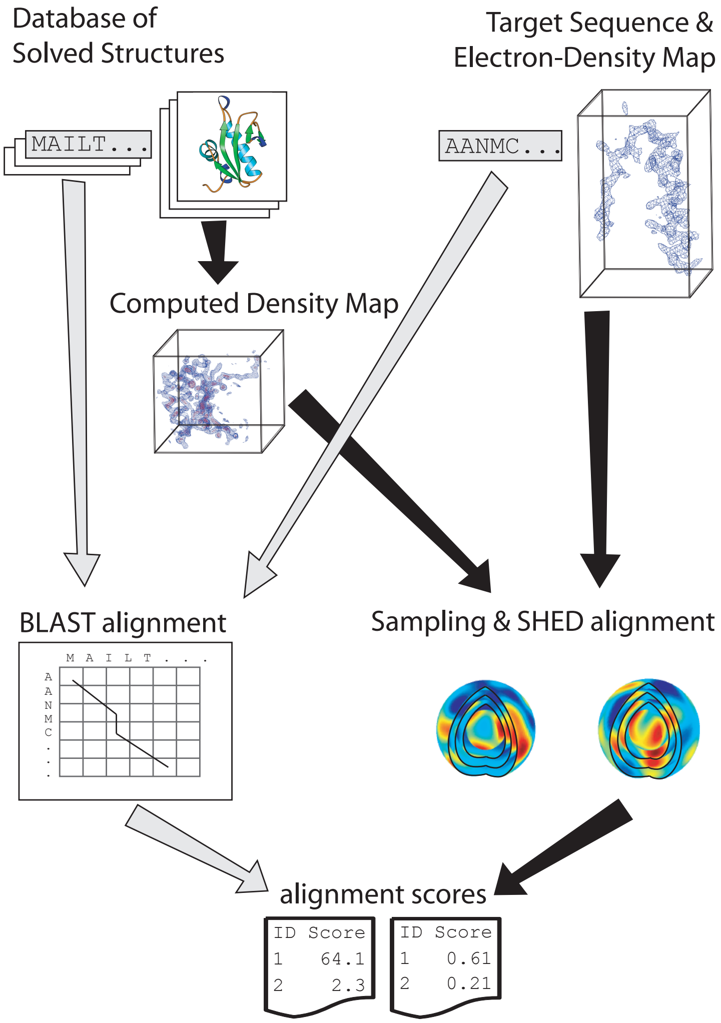
Figure 7.5: A comparison of two different filters: the best simple filter from Figure 7.4 – the point-density filter – and a filter based on a support vector machine (Svm). Filter performance is compared on model-phased maps at (a) 3 Å and (b) 4 Å resolution, as well as (c) varied resolution, experimentally phased density maps.

The results for 4 Å maps, shown in Figure 7.5b, are not as convincing. While the Svm filter does better at most levels of Cα's kept, the performance only matches that of the density filter above the 90% level. The area under the curve is slightly better for the Svm filter, but the percentage of examples evaluated when 95% of Cα's are kept actually goes up slightly, although the values are statistically insignificant. It seems likely that in lower resolution maps – where few fine details are visible – the bispectrum features provide little additional information over the density values alone. In some cases, using these additional features may even hurt performance by overfitting the data.

So far in this chapter, the maps in my data set are all well-phased density maps with poor resolution. Section 4.1.2 describes in detail a set of ten *experimentally phased* (as opposed to model-phased) density maps from the Center for Eukaryotic Genomics at the University of Wisconsin–Madison. Figure 7.5c shows the results of running the point-density filter and Svm filter on this data set. Both filters show decreased performance relative to the well-phased maps. The Svm filter, however, shows a similar improvement in performance, relative to a point-density filter, as in the 3 Å data set (Figure 7.5a). It produces a 31% relative reduction in number of examples needed to be analyzed, filtering more than 75% of grid points while keeping 95% of Cα's. This implies that Acmi-SH would run in 31% less time with an Svm filter compared to a point-density filter, and in roughly 75% of the time for a full search, all while eliminating false positives at a similar rate. The filter does come at a cost of throwing out false negatives, but a conservative filter can maintain 95% of correct locations while capturing these significant gains.

7.4 Structural Homology Search in Electron-Density Maps

In this section, I explore extending the local template-matching algorithm described in Algorithm 7.1 to handle matching large macromolecules, such as entire proteins, against an electron-density map. Rather than score a probability distribution over likely centers of a template match, I use the fast rotational-alignment function [52, 83] to quickly compare a database of protein structures against an electron-density map of the target protein to find the best matching *whole* structures. Such a tool would be useful in finding structural homologs to the target protein, even when no solved structure exists. In particular, such an algorithm may be able to detect *remote homologs* – proteins with similar structure but low sequence similarity. Sequence-only methods, such as Blast [2] fail in these cases since they rely solely on sequence-



ID	Score
1	64.1
2	2.3

ID	Score
1	0.61
2	0.21

Figure 7.6: A comparison of two homology-search algorithms. Shed (right) compares a target protein's electron-density map against a database of solved protein structures. This algorithm is similar to that in Figure 7.1, except I compare the whole protein structure against the density map instead of just small fragments. Blast (left) considers the sequences of the solved structures and target protein, using a dynamic programming model to measure similarity between two sequences. Black arrows show the movement of density information, while grey arrows indicate the use of sequence information from the inputs.

Algorithm 7.2: Structural Homology using Electron Density (SHED)

```
input  : directory of structures $\mathbf{PDB}$,
         (masked) density map $\mathbf{M}$,
         number of centers of mass to consider $K$
output : correlation coefficient $CC_i$ between $\mathbf{M}$
         and each structure in directory $i = 1 \ldots |\mathbf{PDB}|$

// Calculate SH coefficients for the density map
$COMmap \leftarrow$ center-of-mass($\mathbf{M}$)
$signal \leftarrow$ sample-sphere-around($COMmap$)
$sigCoef \leftarrow$ SH-transform($signal$)

foreach structure $PDB_i \in \mathbf{PDB}$ do
    $CC_i \leftarrow 0$
    // Consider several possible centers of mass
    for $k = 1 \ldots K$ do
        $\mathbf{C^k} \leftarrow$ multiple-COMs($PDB_i, k$)

        // For each center of mass and offsets, calculate SH coefficients for
        //  the PDB structure and score the maximum match across rotations
        foreach $COMtemplate \in \mathbf{C^k}$ do
            foreach offset $o \in \{-1, 0, 1\}^3$ do
                $template \leftarrow$ compute-density($PDB_i, COMtemplate + o$)
                $templCoef \leftarrow$ SH-transform($template$)
                $\mathbf{templCC} \leftarrow$fast-rotate($templCoef, sigCoef$)
                $maxCC \leftarrow \max_{rot} \mathbf{templCC}$
                $CC_i \leftarrow \max\{CC_i, maxCC\}$
            end
        end
    end
end
```

based information. Having such structural homologs available may greatly aid a crystallographer in map interpretation. Finally, determining structural homologs may give key insights into a protein's function even if the density map is too poor of quality to produce an atomic model. This work appears in DiMaio et al. [23].

7.4.1 Methods

My approach considers the spherical-harmonic decomposition of a set of concentric spheres of density that cover the majority of each solved density map (i.e., a template) in a database. This database may contain experimental as well as computed density data. Assuming I know the translational correspondence between each template and the target density map, I may compute similarity between the two. I use our fast rotational alignment to quickly match an entire protein to the density map, as demonstrated in Figure 7.6. I formalize the problem as follows:

> Given an electron-density map and a set of previously solved protein structures, find the solved structures that match the density map best and are thus candidates for structural homologies to the target protein.

Algorithm 7.2 provides the details of my structure-database search procedure, which I refer to as SHED (**S**tructural **H**omology using **E**lectron **D**ensity). Figure 7.6 contrasts my method with BLAST [2], which compares the sequence of a target protein against a database of known proteins and their sequence. When no structure is available for a target protein, sequence homologies can be used to imply structural homologies. SHED, on the other hand, uses the target protein's non-interpreted density map to detect potential structural homologies. Both return alignment scores indicating the degree of similarity between the (nonstructured) target protein and each protein in the solved-structure database.

SHED begins with an unsolved target protein's electron-density map and a database of previously solved protein structures (e.g., the PDB). I convert this structure database to a solved density-map database by simulating the expected electron-density of the protein's atomic coordinates. Ideally, each solved structure in the PDB would come with its original electron-density map. Unfortunately, this data is not always readily available, so – as with pentapeptide matching methods – I calculate the density I would expect to see given an atomic model. As done by the CCP4 program SFALL [11], I model the scattering of each atom using a five-term Gaussian approximation. A complication with density maps is that most unit cells contain many copies of the protein structure. For my evaluation, I assume that a crystallographer masks the target density map such that only the density values of one copy of the protein remain. Ideally, this would be done automatically.

Given these inputs, SHED performs a spherical-harmonic decomposition on a sphere centered at the center of mass of the (masked) target density map, sampling density in 16 concentric spherical shells extending to 32 Å from the center. For each solved density map in the database, SHED grabs a similar set of coefficients. The inner-most loop in Algorithm 7.2 uses the fast rotation function to quickly computer the correlation coefficients between the two density maps across all rotations. SHED takes the maximum score across these rotations as the homology alignment score.

To account for a potential mismatch in mass distribution between the two objects, I perform a small gridsearch around the center of mass of the structure from the database. For each offset, the template map obtains a new set of coefficients and calculates its maximum correlation with the target density map across rotations.

Additionally, in homology detection, a match structure may be much larger than than the target protein, but share a domain of its structure (e.g., the active site in a reaction). To account for these structures with multiple domains, I use *k*-means clustering [60] to choose the best set of centers in the solved structures assuming each input structure contains 1, 2, or 3 domains. Briefly, *k*-means clustering divides the density map into *k* partitions, each represented by its center of mass. Each grid point joins a partition based on which of the *k* center of masses (initialized randomly) it is closest to. The *k* centers are updated and the process is repeated until the values converge.

Overall, SHED's optimal alignment score between a target density map and a template density map is the maximum correlation coefficient over all center of masses, over all offsets in the grid search, and across all rotations of the two objects.

7.4.2 Results and Discussion

To test the algorithm, I download a list of 6529 protein chains from the PISCES protein-sequence culling server [85]. The database contains all PDB entries with resolution $\leq$ 3.0 Å, percentage amino-acid identity cutoff of 30%, and R-factor cutoff of 1.0. Each of our ten 3 Å resolution maps is part of the test set and aligned against each chain in the culled PDB data set, using the search method from Algorithm 7.2 and the methodology above. As mentioned, one difficulty for handling the test-set maps is that the density map may contain many molecules in the asymmetric unit. To overcome this issue, I assume that a human isolated the area of the map where one molecule exists and masks the rest of the map out. There are automated methods, such as FINDMOL [61], which attempt to do this, but results on our low resolution maps were not reliable. For our purposes, I manually masked the density map by keeping density values for all grid points within 7 Å of a Cα in one monomer of the protein. An additional problem is that noise in the density map may skew the center of mass. To account for this, I searched a 2 Å×2 Å×2 Å grid around each center of mass.

As a comparison, I use BLAST [2] to see if the performance of SHED is better than a sequence-only method. While structural-homology detection is not BLAST's

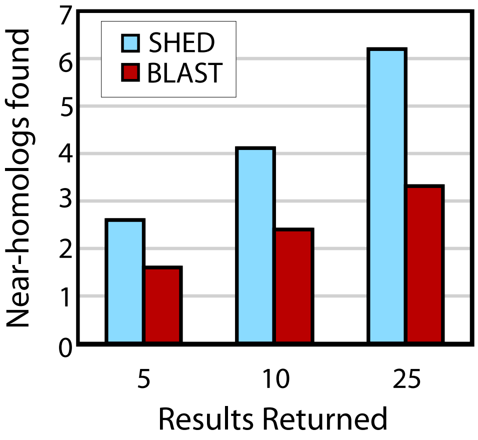


Figure 7.7: The average number of homologous structures found by Shed and Blast when considering the top 5, 10, and 25 results returned. A near-homolog is a returned result that is also returned in the top 5, 10, or 25 results of DaliLite. Test-set proteins came from Section 4.1.2's experimentally phased density maps.

original intent, it does return detected *sequence* homologies. Sequence homology is a reasonable proxy to structural homology when structural coordinates do not exist. For ground truth, the held-aside solved PDB structure for our test maps is aligned to each query chain using DaliLite [39], a dynamic-programming method which finds the optimal alignment between two PDB files taking into account sequence and structure.

Results are shown in Figure 7.7 and Table 7.1. For each map, I sort the alignment scores from all PDB structures for both Shed and Blast. Figure 7.7 displays, for each method, the average number of results in the top 5/10/25 results for that method that are also ranked in DaliLite's top 5/10/25 results. In other words, how many of true top 5/10/25 results are found in Shed's and Blast's top 5/10/25 results? This can be thought of as a measure of *recall*. Table 7.1 shows the same metric, but for each map individually. For example, Shed's top 5 scores for alignments on Map 1 were respectively ranked 2,3,4,6,1 in the DaliLite ground-truth calculation, meaning that 4 of Shed's top 5 results were in DaliLite's top 5.

On average, my Shed method finds more structural homologs than Blast. If you consider only the top 5 results returned by each method, Shed returns one extra correct structure on average. This advantage grows larger as more results are returned, although both methods begin to return many more false positives. Looking at the specific results, with the exception of Map 5, my method does better than or

	Top 5		**Top 10**		**Top 25**	
Map	Shed	Blast	Shed	Blast	Shed	Blast
1	4	4	**10**	9	**14**	11
2	**2**	1	**2**	1	**3**	2
3	1	1	1	1	1	1
4	**2**	1	**2**	1	**2**	1
5	3	3	4	**5**	4	**10**
6	**3**	1	**3**	1	**3**	1
7	**3**	1	**7**	1	**17**	1
8	1	1	**2**	1	**2**	1
9	**3**	2	3	3	3	**4**
10	**4**	1	**7**	1	**13**	1
Wins	**6**	**0**	**7**	**1**	**7**	**2**

Table 7.1: Recall results broken down by map and size of overlap between a method's top 5/10/25 results and the true top homologous alignments. The "Wins" are the number of maps over which one method outperforms the other, shown in bold.

equal to Blast, demonstrating that a density map does provide clues to the three-dimensional structure of a protein that can aid in detecting homologous structures. The bottom row in the table shows the number of "wins" each method has over the other. When 5 results are returned, Shed identifies more correct structures 6 times but never fewer. This advantage stays relatively the same, winning 7 to 1 when 10 results are returned and 7 to 2 when 25 results are returned.

Several maps, however, did not give great results for either algorithm. While both methods consistently found the single best match, maps 2, 3, 4, and 8 did not find many other matches. Looking at the DaliLite results, the alignment scores drop significantly after the top match indicating there were no more significant results to find. Another shortcoming is that the my method did not seem to find many domain, or substructure, matches. That is, most results detected only global similarity in structure. This could be addressed by isolating smaller spheres around the various center of masses. Also, the density map should also be broken into many small domains to match against possible domains in the solved structure. My algorithm does run slower than Blast, but can perform a large database search in a few hours on one CPU.

7.5 Summary

In this chapter, I described three contributions that utilize spherical-harmonic decompositions to provide an efficient representation of three-dimensional protein fragments. Spherical harmonics describe a three-dimensional object with a set of spherical basis functions and coefficients. This representation proves advantageous in Acmi, primarily for the ability to use a fast rotation function [52, 83] to efficiently match two objects across all rotations.

In Section 7.2, I described a new method for Phase 1 of Acmi's pipeline – scoring the probability of an amino acid being at a certain location in the density map using proxy fragments (or templates) from the PDB. Previous work used Fourier convolutions to quickly search over all (x, y, z) coordinates for some rotation of a template. While efficient over translations, Acmi-FF still requires an exhaustive search over rotations. Instead, we propose the use of a spherical-harmonic decomposition of a template to rapidly search all rotations of some fragment at a single (x, y, z) location. Results show that our proposed method, Acmi-SH, produces more accurate protein-backbone solutions than Acmi-FF, Textal, and Resolve in terms of both Cα RMS error and completeness of protein structure.

This new method flips the search efficiency of FFTs by being efficient over rotations but not translations. This, however, turns to be an advantage as Acmi-SH can use an initial filtering algorithm to "mask out" locations in the density map unlikely to contain any Cα locations. In Section 7.3, I showed that a simple filter using local density values is able to eliminate large portions of the density map from the Acmi-SH search with very few false negatives. I also propose generalizing spherical-harmonic decompositions to a set of rotation-invariant features, which I use in training an Svm classifier for improved filtering of density points. My proposed filter offers both improved efficiency and accuracy, compared to previous work, finding substantially better models while reducing running time by about 75%.

Finally, I extend Acmi's template-matching method to handle large protein alignments to a density map. My framework, Shed, demonstrates that electron-density maps can be used in structural-homology search. In the absence of a solved structure, Shed produces more structural homologs than methods that only use protein sequences alone, such as Blast. Shed can be useful in the early stages of structure determination and can provide important information from maps which prove too difficult to solve. Future work would need to address the limitations of needing to isolate a single monomer in the density map before the search is performed. One possible solution involves using our k-means procedure to isolate the center of each monomer and then varying the radius of our sphere of sampling.

An interesting future direction involves template searching and Acmi's probabilistic inference. Acmi-SH makes it possible to efficiently search for a fragment at a single location. This suggests an approach where Acmi-SH initially searches very few locations. As Phase 2 performs inference, new locations that appear to be promising Cα locations may emerge. Acmi-SH runs a search on these new proposals, and the process iterates with Phase 1 and Phase 2 providing constant feedback. In essence, this is using the first few iterations of the Phase 2 inference algorithm as a first-pass filter. Another area of further work is to extend my Svm first-pass filter to quickly estimate the observation potentials for Phase 1. If the rotation-invariant features maintain a high-level of fidelity, classification should be able to estimate the match score between a template and a location in the map. This would completely eliminate the need for a rotational search, reducing Phase 1 to a three-dimensional, translational-search problem.

8 Statistical-Sampling Methods to Produce All-Atom Protein Models

In Section 3.2, I described the state of Acmi prior to the contributions of this thesis. In that section, Acmi's last step takes the Phase 2 posterior marginal probabilities and creates a Cα *backbone* structure of a protein (Equation 3.8). While useful as an intermediate product, biochemists are interested in the location of *all* of the (non-hydrogen)[1] atoms in the structure including the side chain and non-Cα backbone atoms. In addition, Equation 3.8 independently calculates the most probable location for each amino acid's Cα on a 3D grid, resulting in a trace that is not guaranteed to be physically feasible.

In this chapter, I describe a new method for producing all-atom, physically feasible protein structures using a sampling technique known as *particle filtering*. This new framework, titled Acmi-PF (for **P**article **F**iltering), replaces the previous version of Phase 3 in the Acmi roadmap. Results show that the resulting structures from this technique are substantially higher in quality relative to other methods in the field on a set of difficult protein structures. Furthermore, I present an additional extension to Acmi-PF that incorporates domain knowledge into the sampling algorithm. While the results of this later work failed to meet expectations, the framework introduced suggests many areas of further exploration to improve the model. The first part of this work was done jointly with Frank DiMaio and appears in DiMaio et al. [19], while the later work on incorporating domain knowledge into Acmi-PF is independent, unpublished work.

8.1 Introduction

In prior work, DiMaio et al. [21] developed the foundation of Acmi, summarized in Section 3.2. In that work, Acmi was a two-phase process: first, a local template-matching algorithm (Acmi-FF in Section 3.2.1) and, second, a graphical model to enforce global constraints on the results from the first phase (Acmi-BP in Section 3.2.3). The results of experiments show that Acmi outperforms other methods in the field, including Textal, Resolve, and ARP/wARP [21], in tracing backbone structure in low-resolution electron-density maps.

[1] In this chapter, "all-atom" actually means "all non-hydrogen atom". X-ray crystallography, in most protein crystals, cannot resolve hydrogen atoms. Thus, most PDB-deposited structures produced via X-ray crystallography do not model hydrogen atoms in the protein structure.

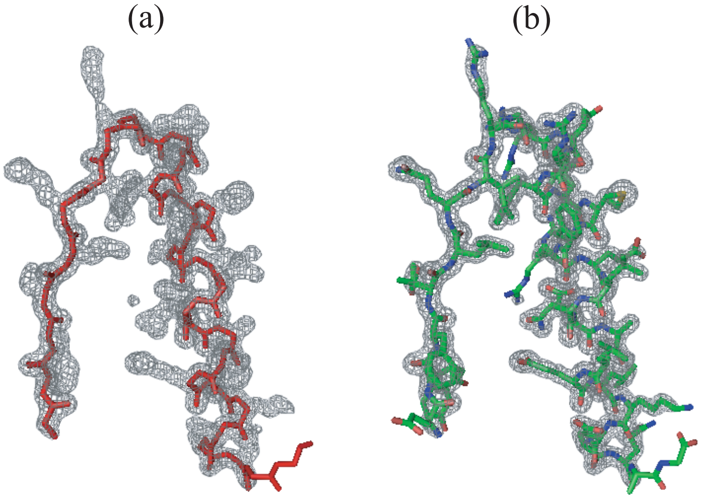


Figure 8.1: Interpretation of an electron-density map with a) only a backbone trace and b) all-atom trace of both backbone and side-chain atoms.

However, Acmi makes several simplifications that inhibit the usefulness of these results. For one, Acmi's Markov random field (MRF) model reduces each amino acid to a single atom in its representation – the Cα atom. The other methods tested are able to not only trace Cα chains through a map, but construct all-atom protein models including side-chain atoms. Figure 8.1 demonstrates, for a given electron-density map, the difference between a backbone trace and an all-atom protein model. Second, Equation 3.8 assigns the maximum probability location *independently* for each amino acid, meaning that the location of two adjacent amino acids are not necessarily biochemically consistent. Furthermore, Acmi restricts the locations of each Cα atom to a coarse 3D grid, further reducing the physical feasibility of the model.

In DiMaio et al. [19], I helped introduce Acmi-PF (for Particle Filtering), a statistical-sampling method for sequentially constructing an all-atom protein structure. This chapter will provide a brief overview of our method, which utilizes *particle filtering* [25], a sequential analog to Markov Chain Monte Carlo (MCMC) methods. Specifically, we employ statistical importance resampling (SIR), detailed in Section 2.2.5, which extends a partial protein model in a stepwise fashion by drawing from the posterior marginal probabilities produced by Phase 2 of Acmi. This new method forms Phase 3 of the Acmi pipeline from the roadmap in Section 3.3. The results of experiments on this new method show that Acmi's Phase 3 produces physically feasible structures containing both backbone and side-chain atoms. When we compare Acmi to competing methods in the field on a test set of difficult protein structures, Acmi excels in both completing the protein structure accurately and yielding structures that explain the original electron-density map according to the R_{free} metric.

In Section 8.5, I describe my work that attempts to improve Acmi-PF by incorporating biochemical domain knowledge into the sampling framework. Specifically, I describe the use of secondary-structure prediction to provide more detailed and informed sampling distributions for each sequential extention of the protein structure. This work utilizes the wealth of data available in previously solved structures in the PDB to show that secondary structure has a large influence on the bond angles and torsion angles a protein backbone takes. While incorporating this information yields mediocre results, my framework provides the building blocks for representing established biochemical knowledge into a probabilistic sampling algorithm, suggesting several avenues of future work.

8.2 Limitations of Acmi-BP

As described in Section 3.2, Acmi-BP's probabilistic inference returns, for each amino acid i in the protein's primary sequence, the marginal probability distribution, $\hat{p}_i(\vec{u_i})$, of amino acid i's Cα location. The distribution is over all grid locations in the electron-density map, $\vec{u_i} = (x_i, y_i, z_i)$. As outlined in Equation 3.8, the backbone trace is than taken as the position of each Cα that maximizes Acmi-BP's belief,

$$\vec{u_i}^* = \arg\max_{\vec{u_i}} \hat{p}_i(\vec{u_i}). \tag{8.1}$$

While results were favorable relative to other methods for producing backbone traces [21], Acmi-BP has several shortcomings. First, biologists are interested in not just the position $\vec{u_i}^*$ of each C_α, but in the location of every atom in the protein. Figure 8.1 demonstrates, for a sample electron-density map, the difference between a backbone trace and an all-atom protein model.

Second, Acmi-BP represents the probability of a Cα location over a 3D grid, meaning the maximum marginal location is assigned to grid points. By forcing locations to be at discrete grid locations, Acmi-BP often produces out-of-range bond-lengths (i.e., too short or too long).

The last issue arises from the independent nature of Equation 8.1. While Acmi-BP produces probability distributions by enforcing global constraints, Equation 8.1

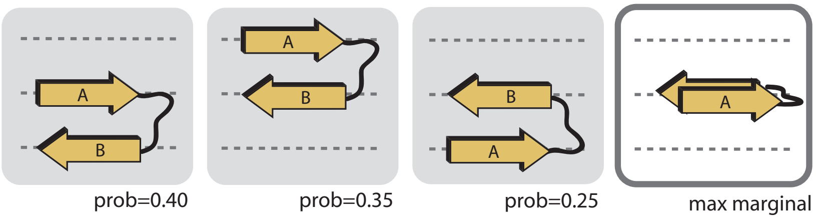


Figure 8.2: Illustration of an infeasible backbone trace produced using Equation 3.8. In the first three panels, Phase 2 of Acmi infers that there are three likely conformations for amino acid A and B. By choosing the maximum likelihood location independently for each, however, the backbone trace places both amino acids in the same location as shown in the last panel.

chooses the maximum location independently for each residue. Figure 8.2 shows one possible resulting error, where inference places high probability in three feasible traces, but the maximum marginal path results in an overlap in the chain trace. Equally problematic, two adjacent residues may be placed on opposite sides of the map because there are multiple copies of the protein chain in the sequence.

8.3 Producing All-Atom Protein Structures using Acmi-PF

Section 2.2.5 provides an overview of particle filters. In particular, I describe the general theory behind *statistical importance resampling* (SIR), a sequential Monte Carlo method for sampling variable states in a probabilistic model. Extending SIR to the task of protein-structure tracing requires us to modify the notation.

Recall that particle filters represent a probability distribution with a set of (weighted) point estimates, as in Equation 2.13. To restate:

$$p(x_{1:K} \mid y_{1:K}) \approx \sum_{i=1}^{N} w^{(i)} \delta(x_{1:K} - x_{1:K}^{(i)}) \tag{8.2}$$

where k is an index into the amino-acid sequence and i is the particle index. Each particle represents one estimate of the protein structure with K amino acids, $x_{1:K}$. An individual variable in this structure, x_k, represents the location of *all* atoms for amino acid k – side chain and backbone. The evidence variable, y_k, represents the electron-density map. Since the density map is the same for all amino acids, we can drop the subscript and use $\mathbf{y}$.

To simplify, we parametrize x_k as a Cα location b_k (the same as $\vec{u_i}$ in Equation 3.8,

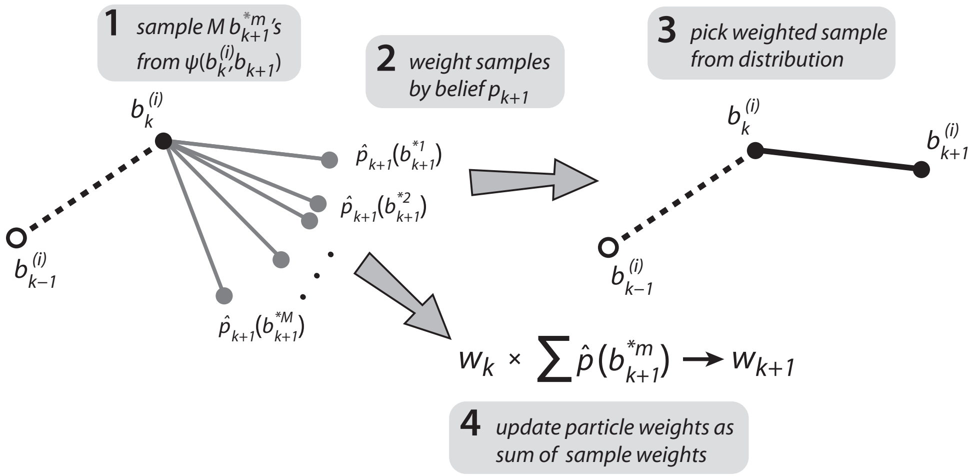

1 sample M b^{*m}_{k+1}'s from $\psi(b^{(i)}_k, b_{k+1})$
2 weight samples by belief p_{k+1}
3 pick weighted sample from distribution
$b^{(i)}_k$
$b^{(i)}_{k-1}$
$\hat{p}_{k+1}(b^{*1}_{k+1})$
$\hat{p}_{k+1}(b^{*2}_{k+1})$
$\hat{p}_{k+1}(b^{*M}_{k+1})$
$b^{(i)}_k$
$b^{(i)}_{k+1}$
$b^{(i)}_{k-1}$
$w_k \times \sum \hat{p}(b^{*m}_{k+1}) \rightarrow w_{k+1}$
4 update particle weights as sum of sample weights

Figure 8.3: An overview of the backbone forward-sampling step. Given positions b_{k-1} and b_k, we sample M positions for b_{k+1} using the empirically derived distribution of Cα–Cα–Cα pseudoangles. Each potential b_{k+1} is weighted by the belief $\hat{p}(b^{*m}_{k+1})$ – Phase 2's marginal posterior probability. We choose a single location from this distribution of weights; the particle weight is multiplied by the sum of sample weights.

where our vector notation has been dropped), and a side-chain placement s_k; that is, $x_k = \{b_k; s_k\}$. The side-chain placement identifies the 3D location of every *non-hydrogen side-chain atom* in amino acid k, as well as the position of backbone atoms C, N, and O.

Given this parametrization, the Markov process, as outlined in Algorithm 8.1, alternates between placing: (a) a Cα position and (b) a set of side-chain atoms Accompanying each step is an update to the particle's weight, reflecting how well the produced sample matches the evidence. The following two subsections describe the backbone and side-chain steps, respectively, for a single particle given a partially sampled structure $x_{j:k}$. Without loss of generality, I'll describe the sampling step as moving forward (i.e., sample x_{k+1} given $x_{j:k}$). The algorithm can also sample backwards (i.e., sample x_{j-1}), requiring only a change in indices.

8.3.1 Sampling Cα's Using Phase 2 Marginal Probabilities

In our algorithm's backbone step, our goal is to sample the Cα position $b^{(i)}_{k+1}$, given our growing trace $b^{(i)}_{j:k}$, for each particle i. From Section 2.2.5, a particle filter needs a proposal distribution from which to draw new variable states (e.g., backbone location). The *optimal* sampling function is the transition probability, $p(b_{k+1} \mid b^{(i)}_k, \mathbf{y})$

Algorithm 8.1: All-Atom Structure Sampling with Acmi-PF

```
input  : density map $\mathbf{y}$,
         amino-acid marginals $\hat{p}_k(b_k)$
output : set of protein models $x_{1:K}^{(i)}$ and weights $w_K^{(i)}$

// start at amino acid with highest certainty
choose $k$ such that $\hat{p}_k(b_k)$ has minimum entropy
foreach particle $i = 1 \ldots N$ do
    choose $b_k^{(i)}$ at random from $\hat{p}_k(b_k^{(i)})$
    $w_k^{(i)} \leftarrow 1/N$
end
foreach residue $k$ do
    foreach particle $i = 1 \ldots N$ do
        // choose $b_{k+1}$ given $b_k^{(i)}$
        $\{b_{k+1}^{*m}\}$ $\leftarrow$ choose $M$ particles from $\psi_{adj}(b_k^{(i)}, b_{k+1})$
        $w^{*m}$          $\leftarrow \hat{p}_i(b_{k+1}^{*m})$
        $b_{k+1}^{(i)}$    $\leftarrow$ choose $b_{k+1}^{*m}$ with probability $\propto w^{*m}$
        $w_{k+1}^{(i)}$    $\leftarrow w_k^{(i)} \cdot \sum_{m=1}^{M} w^{*m}$

        // choose $s_k$ given $b_{k-1:k+1}$
        $\{s_k^{*l}\}$     $\leftarrow$ all side-chain conformations for amino acid $k$
        $p_{null}^{*l}$    $\leftarrow$ prob $cc(s_k^{*l}, EDM[b_k])$ occurred by chance
        $s_k$             $\leftarrow$ choose $s_k^{*l}$ with probability $\propto 1/p_{null}^{*l} - 1$
        $w_{k+1}^{(i)}$    $\leftarrow w_k^{(i)} \cdot \sum_{l=1}^{L} 1/p_{null}^{*l} - 1$

    end
end
```

from Equation 2.16. Particle filtering, however, is based on the assumption that this distribution is too difficult to sample from directly. Fortunately, it is straightforward to estimate using Acmi-BP's produced marginals.

First, by definition of conditional probabilities, our sampling function becomes:

$$p(b_{k+1} \mid b_k^{(i)}, \mathbf{y}) = \frac{p(b_k^{(i)}, b_{k+1} \mid \mathbf{y})}{p(b_k^{(i)} \mid \mathbf{y})}. \tag{8.3}$$

That is, the distribution of locations for Cα_{k+1} given Cα_k and the density map, y, is equal to the joint distribution of both Cαs' locations given the map divided by the probability of Cα_k's already sampled location. Fortunately, we can approximate both quantities using the posterior marginal probabilities from Phase 2. Recall that the belief for amino acid k, $\hat{p}_k(b_k)$, is an approximation to the quantity $p(b_k \,|\, \mathbf{y})$. Thus, our sampling function becomes:

$$\begin{aligned} p(b_{k+1} \mid b_k^{(i)}, \mathbf{y}) &\approx \frac{\hat{p}_k(b_k^{(i)}) \cdot \hat{p}_{k+1}(b_{k+1}) \cdot \psi_{adj}(b_k^{(i)}, b_{k+1})}{\hat{p}_k(b_k^{(i)})} \\ &\approx \hat{p}_{k+1}(b_{k+1}) \cdot \psi_{adj}(b_k^{(i)}, b_{k+1}). \end{aligned} \tag{8.4}$$

Here, $\hat{p}_{k+1}(b_{k+1})$ is the result of Phase 2 – the probability of amino acid $k+1$'s Cα-atom location given the density map as evidence. The adjacency potential function, $\psi_{adj}(b_k^{(i)}, b_{k+1})$, is used here to represent the joint distribution between two adjacent Cα atoms. In other words, we sample the location of Cα_{k+1} from the product of (a) amino acid $k+1$'s marginal distribution and (b) the adjacency potential between Cα_k and Cα_{k+1}.

The optimal weight update for each particle is also intractable to compute, but can be approximated using Acmi's marginals,

$$w_{k+1}^i \propto w_k^i \times \int \hat{p}_{k+1}(b_{k+1}) \cdot \psi_{adj}(b_k^{(i)}, b_{k+1}) \, db_k. \tag{8.5}$$

Equations 8.4 and 8.5 suggest a sampling approach to the problem of choosing location of Cα_{k+1} and reweighting each particle. This sampling approach is illustrated pictorially in Figure 8.3 for sampling the location of b_{k+1} for one particle.

Briefly, given the locations of the Cα atoms for amino acids k and $k-1$ for some particle i, we sample M putative locations for b_{k+1} according to the adjacency potential function, $\psi_{adj}(b_k^{(i)}, b_{k+1})$[2]. Each of these M locations is then weighted by the Phase 2 marginal probability score, $\hat{p}(b_{k+1})$. The sum of all of these weights and locations approximates the integral in Equation 8.5, thus our weight-update function is

$$w_{k+1}^i = w_k^{(i)} \cdot \sum_{m=1}^{M} w^{*m}. \tag{8.6}$$

Finally, this procedure must return a single estimate for $b_{k+1}^{(i)}$. The collection of M estimates and their weights approximates our defined sampling distribution from Equation 8.4, so we sample and return one of the M estimates with probability proportional to each estimate's weight.

[2] This function encodes the distribution of Cα-Cα bond lengths and Cα-Cα-Cα angles as seen in solved structures in the PDB.

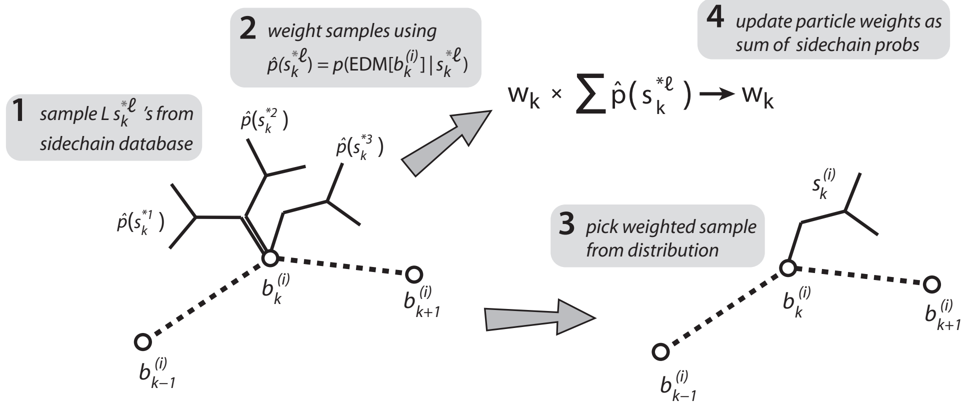

1 sample L $s_k^{*\ell}$'s from sidechain database
2 weight samples using $\hat{p}(s_k^{*\ell}) = p(\mathrm{EDM}[b_k^{(i)}] \mid s_k^{*\ell})$
4 update particle weights as sum of sidechain probs
$w_k \times \sum \hat{p}(s_k^{*\ell}) \rightarrow w_k$
$\hat{p}(s_k^{*2})$
$\hat{p}(s_k^{*3})$
$\hat{p}(s_k^{*1})$
$b_k^{(i)}$
$b_{k+1}^{(i)}$
$b_{k-1}^{(i)}$
3 pick weighted sample from distribution
$s_k^{(i)}$
$b_k^{(i)}$
$b_{k+1}^{(i)}$
$b_{k-1}^{(i)}$

Figure 8.4: An overview of the side-chain sampling step. Given positions $b_{k-1:k+1}$, we consider L side-chain conformations s_k^l. Each potential conformation is weighted by the probability of the map given the side-chain conformation. We choose a side chain from this distribution; the particle weight is multiplied by the sum of these weights.

8.3.2 Sampling Side-Chain Atoms Using PDB Templates

Once our particle filter has placed $C\alpha_{k+1}$ at the location b_{k+1}, it is ready to place all the side-chain atoms in amino acid k. We denote the position of these side-chain atoms s_k. This lag in placing a side chain after the next backbone atom is located allows us to easily place a template in the map. By aligning a template with b_{k-1} through b_{k+1}, we avoid performing an expensive search around the backbone. Given the amino-acid sequence around k, we consider all previously observed conformations of side chain k by sampling from the PDB. Details of this subprocedure can be found in Frank DiMaio's thesis [18]. Briefly, our sampling and weight update equations for placing side chains are very similar to Equations 8.4 and 8.5, with the difference being that side-chain placements do not contain a distribution analogous to the Phase 2 probabilities. Instead, we use the cross-correlation measure from Phase 1 (discussed in Section 7.2.1) to estimate the probability that a chosen side-chain placement explains the electron density in a region of the map.

As illustrated in Figure 8.4, side-chain sampling uses a method similar to the backbone sampling of the previous section. We consider extending our particle by each of the L side-chain conformations $\{s_k^{(1)}, \ldots, s_k^{(L)}\}$ sampled from our side-chain database. After computing the cross correlations between each side chain and the density map around b_k, each side-chain conformation is weighted by the probability the cross-correlation score was not produced by chance. We choose a single conformation at random from this weighted distribution of L subsamples. Lastly, we update our particle's weight by the *sum* of weights of all the side-chain conformations considered to approximate the optimal weight update function – similar to the backbone weight update in Equations 8.5 and 8.6.

8.4 Experiments and Results

We compare Acmi including our new Phase 3 algorithm (Acmi-PF) to three other automated-interpretation methods across the set of ten *experimentally phased* density maps from Section 4.1.2. These approaches, described in Section 2.1.3, are the commonly used density-map interpretation algorithms ARP/wARP, Textal, and Resolve. Refinement for all algorithms uses the same protocol, refining the predicted models for 10 iterations in Refmac5 [68][3]. Acmi uses a three-phase protocol with the implementation of Phase 1 from Section 7.2, the Phase 2 implementation from the prior work (Section 3.2.3), and the Phase 3 proposal in this section using 100 particles. In the results below, Acmi refers to this three phase pipeline.

To assess the prediction quality of each algorithm, we consider three different performance metrics: (a) backbone completeness (see Section 4.2.1), (b) side-chain identification, and (c) R_{free} factor (Section 4.2.3). The first metric measures: of all possible Cα's to predict, what percentage is within 2 Å of *any* Cα atom in the true PDB solution? The second measure counts the fraction of Cα's both correctly placed within 2 Å *and* whose amino-acid type (e.g., tryptophan) matches the PDB-deposited structure's amino-acid type. Side-chain identification is stricter than completeness – its value will always be less than or equal to completeness. Finally, the R_{free} factor, described in Section 4.2.3, is a measure of how well the predicted structure matches the electron-density map. A lower R_{free} factor indicates a better model.

Figure 8.5 compares all four methods in terms of backbone completeness and side-chain identification, averaged over all ten test-set proteins. Under both of these metrics, Acmi locates a much greater fraction of the protein than the other approaches. Acmi performs particularly well at side-chain identification, correctly identifying close to 80% of side chains over these ten poor-quality maps. In particular, in almost every instance Acmi correctly lays a backbone atom, it also identifies the correct amino-acid type of that atom. This is most likely the result of Acmi's maintenance of

[3] ARP/wARP, which integrates refinement and model-building, was not further refined.

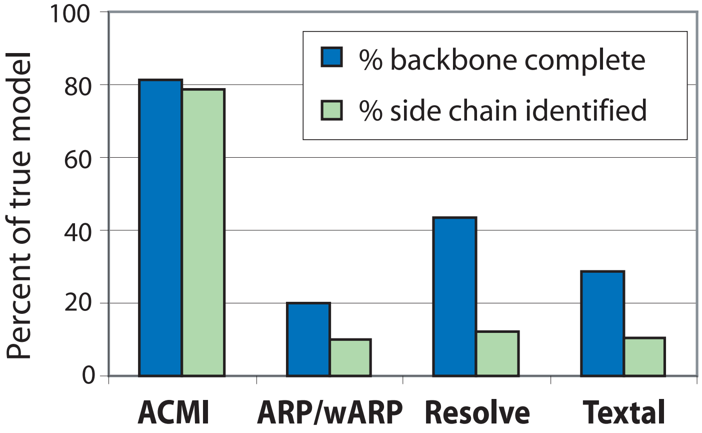


Figure 8.5: A comparison of ACMI to three other automatic-interpretation methods in terms of average backbone completeness and side-chain identification. Each bar represents the average score over our test set of ten experimentally phased density maps in Section 4.1.2.

a specific probability distribution for *each* amino acid in the sequence. Competing methods tend to identify generic protein structure first, and then register the amino-acid sequence to the structure post-hoc.

The three versus plots in Figure 8.6 compare the R_{free} of ACMI's complete model to each of the three alternative approaches, for each density map. Any point below the diagonal (shaded grey) corresponds to a map for which ACMI's solution has a lower R_{free}, and thus a better protein structure. These plots show that for all but one map, ACMI's solution has the lowest R_{free} factor. The lone exception, in Figure 8.6a, corresponds to ARP/wARP having a lower R_{free} factor on the protein 2NXF. This protein has a fairly high resolution value of 1.9 Å. It was included in our set because the map is poorly phased, making it difficult to interpret. ARP/wARP automatically traces 90%, while ACMI correctly predicts only 74%. While ACMI can handle difficult maps, ARP/wARP is able to iteratively recalculate a map given its previous predictions, thus improving the phase quality. In these experiments, ACMI does not have this capability[4], explaining its relatively poor performance in this instance.

[4] Work on recalculating phases in ACMI can be found in Section 5.5 of Frank DiMaio's thesis [18].

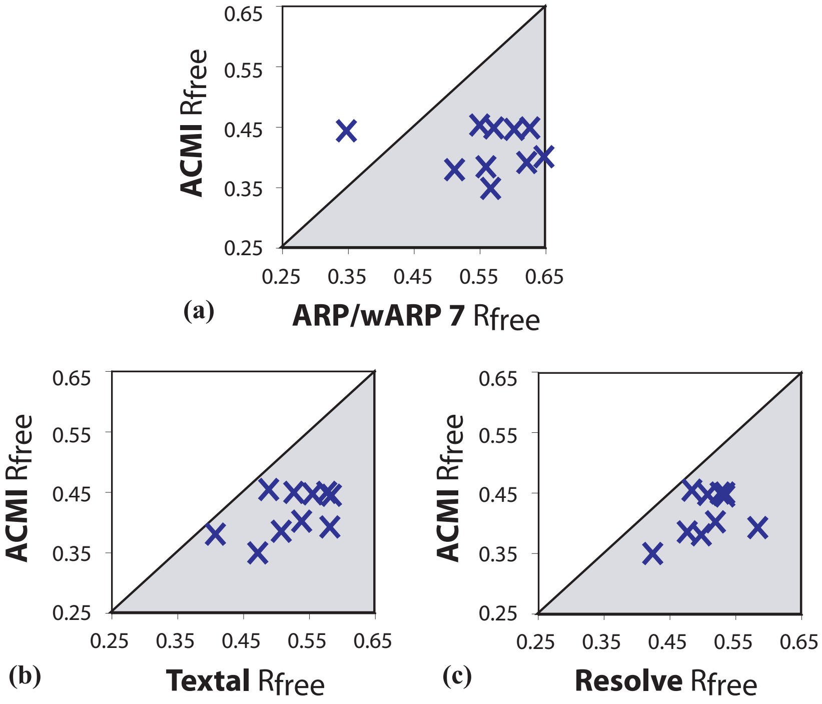


Figure 8.6: A comparison of the R_{free} factor of Acmi's interpretation for each of the ten experimentally phased density maps versus (a) ARP/wARP, (b) Textal, and (c) Resolve. The scatterplots show each interpreted map as a point, where the x-axis measures the R_{free} factor of Acmi and the y-axis the alternative approach. Points below the line indicate better quality models by Acmi.

8.5 Incorporating Biochemical Domain Knowledge into Acmi-PF

In this section, I describe my explorations into incorporating sequence-specific information into the protein-structure sampling algorithm, Acmi-PF. Specifically, I analyze the effect of using secondary-structure prediction to specify the $\psi_{adj}(\vec{b}_k, \vec{b}_{k+1})$ function of Section 8.3, which I use to sample Cα atoms.

8.5.1 Motivation

Section 8.3 introduces a method for producing all-atom protein structures using a statistical-sampling technique called particle filtering. Algorithm 8.1 describes how Acmi-PF creates a set of protein structures by extending an initial structure where, on each iteration, one backbone atom is placed followed by the side-chain atoms

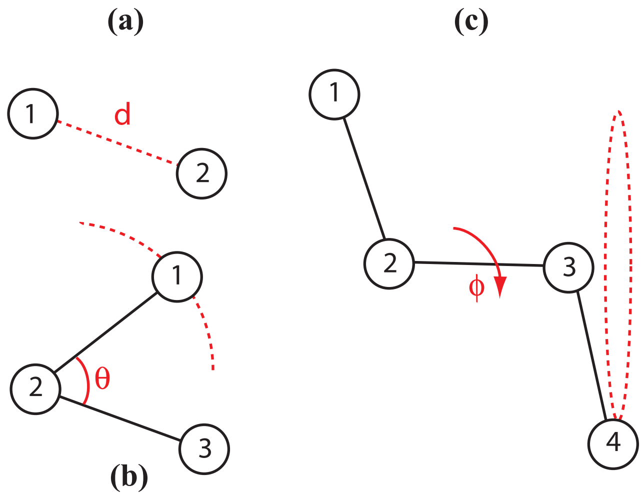


Figure 8.7: Components for sampling a new atom location. Given the distance d to the previous atom, angle θ formed with the previous two atoms, and dihedral ϕ formed with the previous three atoms, the new atom location can be exactly determined.

of the previous residue. The method for sampling the backbone atom relies on a subsampling procedure, shown in Figure 8.3. First, Acmi-PF samples potential Cα locations for amino acid $k+1$ based on the adjacency potential function $\psi_{adj}(\vec{b}_k^{(i)}, \vec{b}_{k+1})$ where $\vec{b}_k^{(i)}$ is the already sampled location of residue k's Cα in particle i. While Acmi-PF showed success in producing feasible protein models, much of the primary-sequence information – biochemically crucial to determining the protein fold – is ignored in sampling. Research in the protein-structure community has shown that a great deal of predictive information is available by considering sequence-specific information. An example is Ramachandran plots [73], which specify the distribution of dihedral angles that backbone atoms exhibit in known structures. These plots show that amino-acid type, as well as secondary structure, has a strong influence on the bond angles a protein forms in its backbone.

In this work, I analyze the value of incorporating secondary-structure information in modeling the adjacency function, ψ_{adj}, for sampling a new backbone location in Acmi-PF. Recall that this adjacency function is the same as the adjacency potential from Acmi-BP in Section 3.2.3, which enforces constraints in the Markov random field between adjacent amino acids. Specifically, the potential function between two adjacent Cα's k and $k+1$ is modeled based on three components: a pairwise *distance function* $d(x_k, x_{k+1})$, an *angular function* $\theta(x_{k-1}, x_k, x_{k+1})$ and a *dihedral*, or *torsion*

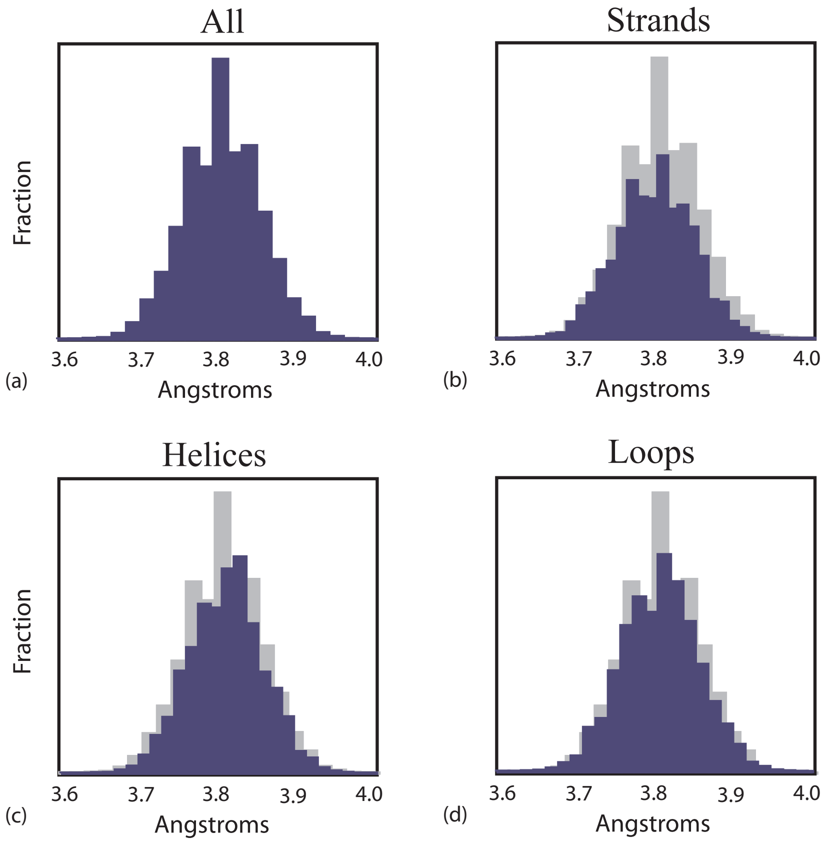


Figure 8.8: The length of Cα–Cα bonds in the PDB for a) all residues, b) residues in strands, c) residues in helices, and d) residues in loops. I created each histogram by collecting structures from the PDB, and then assigning a secondary-structure assignment to each residue using DSSP. In b), c), and d), I also show, in gray, the distribution for all residues from a) for comparison.

function $\phi(x_{k-2}, x_{k-1}, x_k, x_{k+1})$. This formulation is based on vector mathematics – given a dihedral angle, a bond angle, a bond distance, and two vectors (i.e., the previous two peptide bonds), the fourth point in the sequence can be uniquely determined. Figure 8.7 demonstrates the effect of each measurement on determining a fourth point.

ACMI, prior to this section, represents each of these as independent parametric functions, where we learn the parameter values (e.g., mean, variance, prior, distribution type) from solved structures in the PDB. ACMI ignores all residue-specific information when obtaining these values. Thus, the assumption underlying each parameter is that bond length, angle, and torsion distributions are independent of any sequence-specific traits. For example, ACMI models the backbone angles between

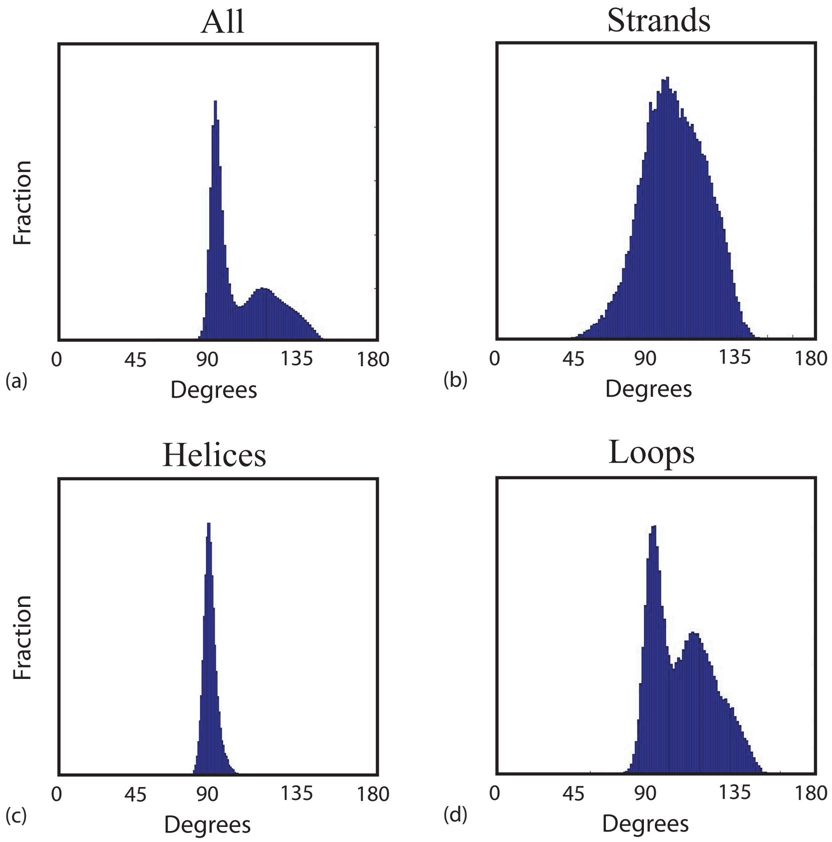


Figure 8.9: The angle of Cα–Cα–Cα bonds in the PDB for a) all residues, b) residues in strands, c) residues in helices, and d) residues in loops. Values were obtained as in Figure 8.8.

two Cα's in an α-helix the same as if they were in a β-sheet.

Figures 8.8, 8.9, and 8.10 show that this assumption, in some cases, does not hold. To produce these distributions, I determine the secondary-structure information for each protein structure in the PDB using the Dictionary of Protein Secondary Structure (DSSP) definitions [47], a standardized protocol for assigning secondary-structure labels to deposited PDB structures. DSSP assigns one of nine secondary-structure types to each residue in the PDB entry. These nine types generalize into one of three majority categories – helices, strands, or loops – which I present in these figures.

Figure 8.8 compares the distribution of bond lengths for all Cα–Cα bonds in the PDB versus the same information conditioned on secondary-structure type. As the figure shows, modeling bond lengths as invariant of secondary structure does not lose much information. The means of each distribution are roughly 3.8 Å. Helices,

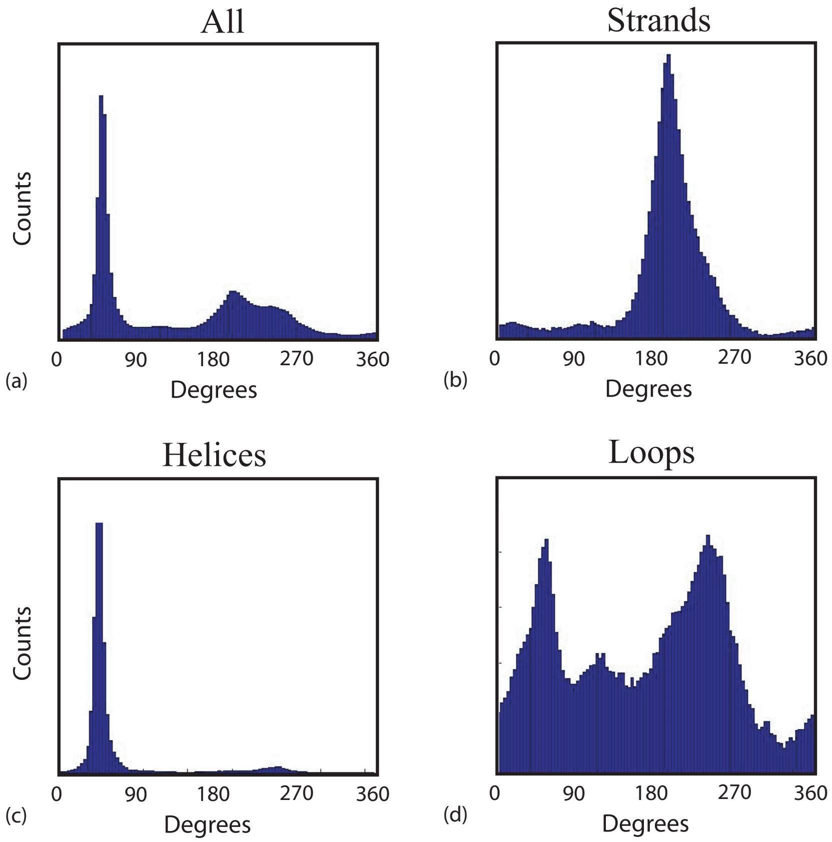


Figure 8.10: The torsion angle of Cα–Cα–Cα–Cα bonds in the PDB for a) all residues, b) residues in strands, c) residues in helices, and d) residues in loops. Values were obtained as in Figure 8.8.

however, do show a tighter distribution of lengths around this mean with a standard deviation of 0.047 Å compared with 0.078 Å for loops and 0.060 Å for strands.

While bond lengths seem invariant of secondary structure, secondary-structure type shows a significant influence on both bond-angle and dihedral-angle distributions. Figure 8.9 shows the distribution of Cα–Cα–Cα angles conditioned on secondary-structure type. When all pseudoangles are considered in one distribution, as in Figure 8.9a, bond angles show a characteristic mixture of two Gaussians. The distribution for loops shows a similar characteristic, although with different mixture parameters and variances for the two Gaussians. The key difference occurs when we consider angles occurring in helices or strands. Helices tend to group tightly around 92.2° while strands have a wider distribution centered at 123.5°. These two figures show that simply modeling all bond angles as the same ignores valuable information, particularly if a residue is in a helix or strand motif. Figure 8.10 shows a similar story with Cα–Cα–Cα–Cα dihedral angles.

These distributions justify incorporating secondary-structure information into modeling backbone traces. Intuitively, helices demonstrate tight, spiral conformations. One should expect angles to be acute and dihedral planes to be non-parallel. The angle distributions do in fact concentrate on the smaller end of feasible angles and dihedral planes lie at acute angles to one another, according to my histograms. For strands, one would expect the opposite since biochemists describe them as relatively flat and parallel chains of amino acids. Again, the figures show that angles between amino acids in sheets tend to be much larger and dihedral planes are close to parallel.

8.5.2 Methods

Acmi-PF generates a new Cα location for residue $k+1$ by sampling a value from each component of the adjacency function I described in the previous section – length d, angle θ, and torsion ϕ. These functions model statistics from deposited structures in the Protein Data Bank using parametric functions. For instance, we originally parametrize $d(x_k, x_{k-1})$ as a Gaussian distribution with mean μ_d and standard deviation σ_d. Then, to sample the distance from Cα_k to Cα_{k+1}, I randomly sample from a Gaussian distribution using these parameters:

$$d \sim \mathcal{N}(\mu_d, \sigma_d^2). \tag{8.7}$$

To generate an angle θ, Acmi-PF in Algorithm 8.1 samples from $\theta(x_{k-1}, x_k, x_{k+1})$, which models the angle between three consecutive Cα's as a mixture of two Gaussian distributions. Each Gaussian is parametrized with a prior probability (π_1, π_2), mean (μ_1, μ_2), and standard deviation (σ_1, σ_2). Acmi-PF fits these parameters to angular data from the PDB using Expectation Maximization (EM) [16, 62], a standard technique for learning parameters in Gaussian mixture models. Acmi-PF models $\phi(x_{k-2}, x_{k-1}, x_k, x_{k+1})$ as a uniform distribution, sampling values from 0 to 2π.

While this original algorithm for Acmi-PF is invariant to the actual amino-acid type being sampled, I propose instead to condition each parametrized function on the secondary structure of the amino acid, based on the discussion associated with Figures 8.8, 8.9, and 8.10. The distance function for sampling becomes $d(x_k, x_{k+1}, ss(k+1))$ where $ss(k+1)$ is the secondary-structure type (e.g., helix, strand, or loop) of residue $k+1$. This is similarly done for $\theta(x_{k-1}, x_k, x_{k+1}, ss(k+1))$

and $\phi(x_{k-2}, x_{k-1}, x_k, x_{k+1}, ss(k+1))$.

Given this framework, I could characterize each function by the same parameters as before, but with parameter values for each secondary-structure type. For example, the distance function would become:

$$d \sim \mathcal{N}(\mu_{d,ss(k+1)}, \sigma^2_{d,ss(k+1)}). \tag{8.8}$$

Instead, I consider sampling directly from the distributions in Figures 8.8, 8.9, and 8.10; that is, sampling a new backbone location utilizing a *nonparametric representation* for each function. While this requires more storage of information, this method represents the true distributions more accurately since, in many cases, they are not exactly Gaussian. When sampling for $b_{k+1}^{(i)}$, I now sample the distance, angle, and dihedral angle necessary to compute the new Cα using secondary-structure specific distributions.

8.5.3 Results and Discussion

This section presents results on the use of secondary-structure knowledge in sampling. One complication in using secondary-structure information is that secondary structure is a function of the final 3D trace of a protein. Since I do not yet have the 3D trace, I instead choose to use secondary-structure *prediction* based on a protein's primary sequence to condition backbone sampling. Specifically, I use PHDsec [76] to predict the secondary structure for each residue in the sequence. In the results, I denote the modified Acmi-PF backbone sampling using predicted secondary structure as "PHDsec." Since there may be errors in predicted secondary structure, I also consider using the ground truth secondary-structure information from the deposited structure in the PDB. While this is not a realistic scenario in practice, it allows me to gauge if performance is effected by errors in PHDsec. I use the Dssp protocol discussed in Section 8.5.1 to make these assignments. I refer to Acmi-PF backbone sampling conditioned on the true secondary structure as "DSSP" in the results below. In addition to these protocols, I consider using no secondary-structure information at all, marked "None." Note that this is the original Acmi-PF protocol from Algorithm 8.1.

For all three variations of Acmi-PF, I use the same Phase 1 and Phase 2 techniques (i.e., the same posterior marginals are given as input) and run Phase 3 with the modified backbone sampling step. I omit the side-chain sampling step from Algorithm 8.1 to simplify the comparison. I run these methods on the set of ten of experimentally

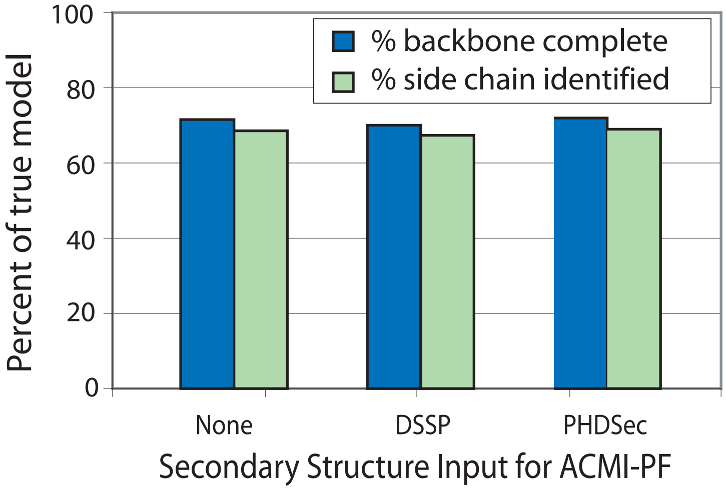


Figure 8.11: A comparison of three methods for producing sampled backbone traces in Acmi-PF across the test set of experimentally phased electron-density maps. The three methods vary in the type of knowledge added for sampling – "None" for not taking secondary structure into account (i.e., the Acmi-PF protocol in Section 8.3), "DSSP" for using the ground-truth secondary structure to determine sampling distributions, and "PHDsec" for using predicted secondary structure.

phased maps described in Section 4.1.2. To measure accuracy of the backbone trace, I measure the backbone completeness and average side-chain identification over all structures produced for each method. See the results in Section 8.4 for a definition of these terms.

The results are shown in Figure 8.11. Despite expectations, the results show no statistically significant differences between the methods. Using no information (i.e., the original algorithm) actually performs slightly better than DSSP, and the results are equivalent to PHDsec. Interestingly, using predicted secondary structure (PHDsec) outperforms using the ground truth (DSSP) in almost all proteins. This is most likely due to random chance, but could arise from secondary-structure prediction providing a smoother classification that exists in reality. For example, while the amino acid immediately following an α-helix may actually be a loop, labeling it as an α-helix may more accurately reflect it's true conformation.

These results suggest further work is needed. Briefly, one issue is that despite the modified sampling protocol, the weights of each particle are still determined by the Phase 2 posteriors, which does not take secondary-structure information into account. This means, for example, that the over sampling of large parallel dihedral angles for strands receive small weights since the posteriors put little weight in these regions

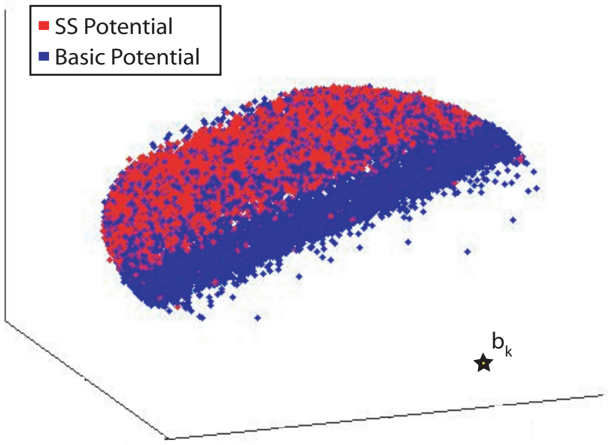


Figure 8.12: Sampled candidate Cα locations for an amino acid from one of the experimentally phased proteins. Given the previous three Cα's for an amino acid i in an alpha-helix, 1000 candidate Cα locations were sampled from (blue) a potential function that did not consider secondary-structure information and (red) one that did.

during Phase 2. Figure 8.12 shows that the secondary-structure sampling protocol does produce a different distribution of samples, and suggests the posterior weights may hinder the influence of secondary-structure information. This example shows the results of the first step of the backbone subsampling procedure in Figure 8.3 for a helix residue in one of our test-set proteins. Recall that this step samples M potential locations for backbone b_{k+1} given the previous backbone atoms in the sequence. The red dots show the sampled Cα locations for b_{k+1} when ψ_{adj} is given the secondary-structure type (i.e., the PHDsec variation of ACMI-PF). Here, b_k is located in the lower right corner. The blue dots show the sampled Cα locations under the original protocol (i.e., no secondary-structure information). As the figure shows, the distribution of samples becomes much tighter and focused when ACMI-PF considers the residue is in a helix. Using the helical information, in this case, prevents the wasteful attempts at locating the Cα at the wider angles.

8.6 Summary

In prior work, ACMI was shown to outperform other methods in automated interpretation of electron-density maps on a set of difficult protein structures [21]. ACMI, however, only produced a Cα-backbone trace of the protein structure. In these traces, atoms were independently placed leading to physically infeasible bond lengths. Crystallographers, instead, are interested in producing an all-atom model of a protein – with side chain and backbone atoms – and maintaining biochemical feasibility.

In this chapter, I reviewed joint work with Frank DiMaio on Acmi-PF, our novel Phase 3 algorithm for generating protein structures from the posterior marginal probabilities of Phase 2. Acmi-PF produces all-atom protein structures, utilizing a sampling algorithm which ensures physically feasible protein structures free of the constraints of a discrete grid. Specifically, we employ statistical importance resampling (SIR), a sequential Monte Carlo method that grows a protein structure by alternately placing a Cα atom for the next residue in the chain and then placing the side-chain and other backbone atoms for the current residue.

The results of our experiments show that Acmi with this new Phase 3 algorithm is the state-of-the-art method for determining protein structures in difficult electron-density maps. In almost every instance, Acmi produces more complete protein structures that correctly identify amino-acid type. In addition, these structures that provide a better explanation of the diffraction data underlying the electron-density map according to R_{free} values.

These results suggested exploring the idea of eliminating Phase 2, instead using the Phase 1 prior probabilities to weight samples. Initial exploration and analsis shows that the prior probabilities are too noisy/distributed to guide sampling well enough. As discussed with RBP in Chapter 5, Acmi-PF requires tight probability distributions to ensure sampling covers the entire probability space. An avenue of future work is to dedicate more resources to Phase 3 in this scenario, in combination with more domain knowledge (see below). More particles (e.g., 10000 instead of 100) increases the computational time of Acmi-PF dramatically, but should allow coverage of a more distributed probability space. The added domain knowledge will help guide the trajectories with better scoring functions.

In Section 8.5, I presented my extension of Acmi-PF which incorporates domain knowledge. First, I analyzed the influence of secondary structure on the formation of backbone bond lengths, angles, and torsions. The distribution of values demonstrated the amount of valuable information Acmi ignored by not considering secondary-structure information, particularly in angle and torsions of backbone atoms. I modified the sampling framework for backbone placement to utilize nonparametric representations of observed backbone formations in the PDB, conditioned on secondary-structure type.

The results of this later work did not improve Acmi-PF's accuracy. While backbone completeness was unaffected by the new framework, the results did demonstrate that Acmi-PF's backbone sampling procedure produced more focused distributions of estimates when given secondary-structure information. Future work could build on this development by also incorporating secondary structure into the adjacency potentials of Phase 2. Additionally, domain knowledge could play an important role in improving the assignment of weights to particles. By only using Phase 2 weights, Acmi-PF is heavily reliant on the accuracy of Phase 2, which only considers Cα relationships and ignores known influences on protein structure, such as hydrophobicity of side chains.

9 Conclusion

The increasing need for protein structures in many areas – including disease research and drug design – makes progress in structural genomics critical for the future of biomedical advancements. While *ab initio* methods [89], such as Rosetta [77], receive a lot focus in structural genomics, X-ray crystallography remains the best source for obtaining protein structures [72]. As such, developing automated techniques for determining protein structures via X-ray crystallography remains a crucial computational task.

This thesis presents my original work on the development of a set of probabilistic techniques for the automated interpretation of electron-density maps produced via X-ray crystallography. The summation of my work, building upon prior work by Frank DiMaio et al., is the Acmi package[1], which is the state-of-art technique for determining protein structure in poor-resolution maps. The automated techniques presented in this work are essential to increasing the throughput of protein-structure determination via X-ray crystallography. More importantly, by focusing on methods for the most difficult maps, my work pushes the envelope of structures that crystallographers can determine. Proteins that were previously shelved due to difficulty can now be reexamined.

As an example, Acmi played an important role in the structural determination of UCH37 (PDB: 3IHR) [8], a human de-ubiquitylating enzyme that is functionally linked with multiple protein complexes and signal-transduction pathways. Its electron-density map has poor resolution (2.95 Å) as well as large regions of protein disorder. These two qualities contributed to other automated techniques failing to obtain a structure [E. Sethe Burgie and Craig Bingman, personal communication, 2011].

In addition to my work's importance in the field of biochemistry, this thesis describes several contributions to the area of artificial intelligence, and in particular, probabilistic graphical models. The highly connected nature of protein structures necessitates complex modeling solutions. To produce structures from these models, the key computational challenge confronting my work is to reason over the large number of hidden, highly connected variables in the model. This problem is part of a general trend in probabilistic reasoning to develop advanced inference methods.

The main computational contributions of this thesis come in this area of approxi-

[1] Available at http://pages.cs.wisc.edu/∼dimaio/acmi/get_acmi.htm.

mate inference. First, in Chapter 5, I introduced a novel technique for improving the performance of loopy belief propagation through the use of domain knowledge. Chapter 6 presented further improvement in inference by using an ensemble of approximate-inference solutions, combined with novel aggregation functions, to produce multiple probabilistic perspectives for each variable. Both of these contributions are general inference methods. Lastly, Chapter 8 described my work on the use of statistical-sampling methods to produce protein-structure estimates from the approximate marginal probabilities of each amino acid's location.

A further contribution of my work is the development of techniques for incorporating biochemical domain knowledge into a probabilistic framework. Ideally, artificial-intelligence methods can be applied "off the shelf" to problems in computational biology. With most tasks, however, this is not the case as modelers require methods that can incorporate expert-level knowledge in an elegant manner. The use of disorder prediction in Chapter 5 is such an example, using a fairly simple domain-knowledge function to guide inference through a complex graphical model. In Chapter 8, I introduced further motivation for incorporating biochemical information into structure sampling. While results did not improve, the framework opens up many avenues of further exploration, such as using hydrophobocity [66] or protein-structure scoring functions [69] to further improve structure placement in the most difficult portions of protein structures.

9.1 Contributions

In this section, I recap the contributions of my thesis, grouping the discussions by the different phases of Acmi outlined in Section 3.3.

Phase 1 – 3D Shape Matching

Phase 1 estimates the observation potential function – a distribution of the probable location of each amino acid in the density map independent of information about other amino acids. This phase is conceptually an "amino-acid finder" and my contributions in this area apply generally to the subfields of **3D shape matching** and **object recognition** from the computer-vision community. My contributions include:

- In Section 7.2, I described a new method for Phase 1 of Acmi's pipeline, **Acmi-SH (Spherical Harmonics)**. Previous work [21] used Fourier convolutions to quickly search over all (x, y, z) coordinates for some rotation of a template.

Instead, in joint work with Frank DiMaio, we propose the use of a spherical-harmonic decomposition of a template to rapidly search all rotations of some fragment at a single (x, y, z) location. Results show that ACMI-SH produces more accurate protein-backbone solutions than ACMI-FF, TEXTAL, and RESOLVE in terms of both Cα RMS error and completeness of protein structure. This work appears in two publications by DiMaio et al. [22, 23].

- In Section 7.3, I proposed a series of **first-pass filters** to *a priori* eliminate large portions of the map from consideration in the expensive Phase 1 template search. Results showed that a simple filter using local density values is able to eliminate large portions of the density map from the ACMI-SH search with very few false negatives, and actually improves protein-structure solutions by eliminating potential false positives. This work appears in DiMaio et al. [22, 23].
- In developing these filters, I also proposed generalizing spherical-harmonic decompositions to a set of rotation-invariant features, which I use in training a machine-learning method called an **SVM classifier for improved filtering** of density points. My proposed filter offers improved efficiency, compared to previous work, by reducing the running time of Phase 1 by about 75%, while eliminating very few correct locations. This work appears in DiMaio et al. [23]
- In Section 7.4, I extended our template-matching method from a) amino-acid detection to b) large-protein alignments to a density map. My framework, **Structural Homology using Electron Density (SHED)**, demonstrates that electron-density maps can be used in structural-homology search. In the absence of a solved structure, SHED, produces more structural homologs than methods that use only a protein's sequences, such as BLAST [2]. This contribution lies outside the ACMI framework, but is most closely related to Phase 1. This work appears in DiMaio et al. [23].

Phase 2 – Approximate Inference in Markov Random Fields

Phase 2 of ACMI builds a pairwise Markov random field model to represent the structure of a protein, combining local features from Phase 1 with biochemical constraints. The complexity of this model makes exact inference computational infeasible, and thus the contributions of my thesis seek to build approximate-inference techniques that are accurate and efficient at estimating each amino acid's location in the density map. My work applies generally to the problem of **approximate probabilistic inference**. My contributions include:

- In Chapter 5, I introduced **guided belief propagation using domain knowledge**, a general message-passing protocol that utilizes a priority function informed by expert knowledge to guide (loopy) belief propagation. I apply this to Phase 2 of Acmi by using protein-disorder prediction [26] to favor message passing between amino acids predicted to be well-structured. My results indicate that guiding Phase 2 using this function improves Acmi's overall performance. Across most maps, the rank and log-likelihood of the true locations of each residue improve. In addition, Acmi is able to build protein structures with improved completeness and correctness from these more accurate approximate marginal probabilities, with the greatest improvement coming in the most difficult test cases. This contribution appears in Soni et al. [78].

- A secondary contribution of Chapter 5 was the implementation of Elidan et al.'s [27] **residual belief propagation in Acmi**. This function guides belief propagation using an message's *residual* value – an information-theoretic measure indicating the magnitude of change in value for a message from the previous time it was sent. The results show that this technique fails to produce adequate marginal probabilities for use in Phase 3, primarily due to its inability to sufficiently refine the large state space of Acmi. This contribution appears in Soni et al. [78].

- In Chapter 6, I developed a new approximate-inference method based on the concept of ensemble methods from the supervised machine-learning community. My framework, **Probabilistic Ensembles in Acmi (Pea)**, executes several independent runs of inference to provide multiple, diverse solutions to the problem. This involves running Phase 2 several times, varying the inference protocol each time to produce a diverse set of results. The experiments show improvement in the accuracy of the inference process, where the probability distributions from Pea are statistically significantly better in terms of both percentile rank and probability value assigned to the correct location of each amino acid. The results can not be explained by either the extra CPU resources utilized or by using the single-best component of Pea. More importantly, Pea's improved inference translates into more complete and correct protein structures. This contribution appears in Soni and Shavlik [79].

Phase 3 – Statistical Sampling

Phase 3 is the final phase of Acmi and estimates a protein-structure model. While Acmi is a probabilistic framework, biologists are interested in an actual protein structure, not the probability space of possible structures. Thus, Phase 3's output is a *point estimate* of the most likely structure given the probability model. Ideally, biologists want a physically feasible structure with all backbone and side-chain atoms located. My work on this task applies generally to the problem of **statistical sampling**. My contributions to this phase include:

- In Chapter 8, I reviewed joint work with Frank DiMaio on **Acmi-PF (Particle Filtering)**, our novel Phase 3 algorithm for generating protein structures from the posterior marginal probabilities of Phase 2. We employed statistical importance resampling (SIR) [25], a sequential Monte Carlo method that grows a protein structure by alternately placing a Cα atom for a residue in the chain and then placing the side-chain and other backbone atoms. The results of our experiments showed that Acmi with this new Phase 3 algorithm is the state-of-the-art method for determining protein structures in difficult electron-density maps, producing more complete protein structures than all other methods in the field. In addition, these structures provide a better explanation of the diffraction data underlying the electron-density map according to R_{free} values. This contribution appears in DiMaio et al. [19].

- In Section 8.5, I presented my work on **Acmi-PF incorporating domain knowledge.** First, I analyzed the influence of secondary structure on the formation of backbone bond lengths, angles, and torsions. The distribution of values demonstrated the amount of valuable information Acmi ignored by not considering secondary-structure information, particularly in angle and torsions of backbone atoms. I modified the sampling framework for backbone placement to utilize nonparametric representations of observed backbone formations in the PDB, conditioned on secondary-structure type. Unfortunately, the results of this work did not improve Acmi-PF's accuracy. Acmi-PF's backbone sampling procedure produced more focused distributions of estimates when given secondary-structure information, indicating further work is needed. This contribution is not published elsewhere.

- In addition to the contributions for Phase 2, my work on Pea in Chapter 6 included several proposed protocols for **ensemble aggregation in statistical sampling**. With Pea producing multiple Phase 2 estimates, Phase 3's algorithm needed a new weighted-sampling function for backbone-atom placement. I suggest three protocols in Section 6.2.2, including an averaging function, maximum function, and sub-sampling function. Results show that the averaging aggregator performed the best of the three, most likely due to its ability to handle the noisy artifacts of the approximate-inference process. This contribution appears in Soni and Shavlik [79].

9.2 Future Work

This chapter outlines some future directions of research building of my thesis work. For my work on shape matching (i.e., Phase 1), I believe there is a lot of potential in advancing the work on filters. One idea is to develop a feedback loop from Phase 2 as a first-pass filter. Here, I could make Phase 1's initial search very conservative, selecting only the top suggestions from a simple filter. As Phase 2 performs inference, new locations may be suggested since they "fit" the data well. Phase 1 will then perform a template-search at these locations, and repeat the process, slowly expanding the number of possible locations for amino acids to be found.

Another idea would extend the developed machine-learning filter to estimate the template-search scores directly. With the success of Kondor's bispectrum features [51] in developing an Svm model to classify Cα locations, the next step is to attempt to replace the entire Phase 1 search process using the same bispectrum of features, but with a different objective. If the rotation-invariant features maintain a high-level of fidelity, regression should be able to estimate the match score between a template and a location in the map. This would completely eliminate the need for a rotational search, reducing Phase 1 to a three-dimensional, translational-search problem. While this seems a tall order, an intermediate goal would be to train an unique first-pass filter for each amino-acid type; that is, 20 SVM classifiers trained to detect if a particular amino acid's Cα is likely to be found at a certain point. This would allow a more finely tuned Phase 1 search.

My work on Phase 2 suggests many avenues of future work. First, I believe the methods I developed in Chapters 5 and 6 are general enough to apply in other domains. As such, a good future project would be to test these methods in other applications. This would involve non-trivial design choices. Guided belief propagation, for one, requires a priority function as input that is based on domain knowledge. The criteria for this function is that there is a correlation between priority values and the accuracy of evidence (or priors) for a variable. The function, in other words, should reflect the importance of a messenger node. For example, one could utilize edge detection in image restoration [29]. If a pixel is close to an edge, it is more difficult to determine if its deviation from its neighbors is an indication of noise or an edge. Therefore, early iterations of belief propagation should concentrate on in internal regions, slowly pushing out towards the edges.

Pea, in contrast, does not require domain knowledge. In fact, Pea is general enough to apply to any approximate-inference algorithm, including MCMC methods [3] or variational methods [46]. Other methods have utilized a similar concept, but generated ensembles by simplifying the graph structure to create a collection of exact inference solutions [84, 86]. Pea does not require this, and should be able to apply to any approximate method where there is stochasticity in the model, or parameters that can be varied. Future work would also address situations where parameter learning is paired with inference, as is typical in natural language processing and statistical relational learning.

In terms of improving the accuracy of protein structures, the largest potential comes in further development on Phase 3. In particular, the scoring function for particle filtering could benefit from the use of more domain knowledge. Currently, only the marginal probabilities of Phase 2 and the side-chain density match effect the weight of a particle. In instances where Acmi fails, it is a result of poor Phase 2 probabilities for amino acids, usually in loopy regions of the structures with poor density features. In these instances, a scoring function that relies on other available evidence, including structure dynamics, could fill in the information.

For example, to score a side-chain placement for a particle[2], I could use the weight update:

$$w_k \propto w_k \cdot \sum_{m=1}^{M} p(s_k^m|\mathbf{M}) \cdot score(s_k, x_{1:k-1}) \tag{9.1}$$

where the only addition to the original update function is the value $score(s_k, x_{1:k-1})$. This function calculates the match between the new side chain, s_k, and the rest of the placed structure, $x_{1:k-1}$. This scoring function can be *independent* of the density map. In the simplest case, one could take the scoring functions used to describe protein energetics in *ab initio* methods, such as Rosetta [77]. These methods sample structures with no visual evidence and judge the sample using probabilistic tendencies seen in the PDB.

I believe a better option is to develop unique scoring functions specific to my task. For example, biochemists have described the general tendencies of amino acids to

[2] The particle index has been dropped for simplicity.

favor being on the surface of a protein [10]. This includes concepts such as accessible surface area and hydrophobicity. I could, for example, use accessible surface area prediction [64] to score a side-chain placement. If the side-chain is hydrophobic, or predicted to be in the core, than a particle that places it on the surface will receive a poor weight. Contact-map prediction [58] could be used to evaluate the pairwise placement of side chains to ensure favorable interactions receive high scores.

All of these suggestions are particularly important to Phase 3 since the Phase 2 probabilities only consider backbone dynamics, not side chains. In current work, I am attempting to use the *solvent mask* of a map – a description of where water molecules are found in the unit cell, produced in density-map modification in CCP4 [11] – to label grid points that contain water. During particle filtering, amino acids that are hydrophobic should down grade samples close to these grid points since this places them on the surface. These suggested scoring functions only scratch the surface of knowledge from biochemistry and bioinformatics. Future work should explore these different options and find an elegant scoring function for weighing the relative merits of different features.

Lastly, an important area of future work on this application is to create a larger test bed of structures. Specifically, discussions on this topic with my collaborators have centered around membrane proteins [32, 57, 88], which present many difficulties for crystallographers. Membrane proteins account for approximately 25% of all proteins, but only 150 unique structures of membrane proteins exist [9]. Membrane proteins tend to lack stability in structure, and obtaining quality crystals out of solution is difficult in most cases. If Acmi truly can push the boundaries of solvable structures, membrane proteins are a prime source for future testing.

9.3 Final Wrap-up

In this thesis, I present my contributions to the fields of artificial intelligence and structural biology, as well as suggestions for possible directions of future research. My work demonstrates that probabilistic graphical models are a good approach for protein-structure determination via electron-density maps and that, reciprocally, structural biology presents a good, challenging domain for graphical models. In particular, my work demonstrates the challenges (and potential solutions) presented by scaling graphical models to large, complex problems. In general, my work also demonstrates that graphical models are a good approach for computational biology applications, particularly those involving a high level of interactions among variables and those benefitting from the modeling of domain knowledge. The result of my work has been the advancement of the the state-of-the-art in structural biology.

REFERENCES

[1] Adams, Paul, Pavel Afonine, Gábor Bunkóczi, Vincent Chen, Ian Davis, Nathaniel Echols, Jeffrey Headd, Li-Wei Hung, Gary Kapral, Ralf Grosse-Kunstleve, Airlie McCoy, Nigel Moriarty, Robert Oeffner, Randy Read, David Richardson, Jane Richardson, Thomas Terwilliger, and Peter Zwart. 2010. PHENIX: A comprehensive Python-based system for macromolecular structure solution. *Acta Crystallographica Section D* 66(2):213–221.

[2] Altschul, Stephen, Warren Gish, Webb Miller, Eugene Myers, and David Lipman. 1990. Basic local alignment search tool. *Journal of Molecular Biology* 215: 403–410.

[3] Andrieu, Christophe, Nando de Freitas, Arnaud Doucet, and Michael Jordan. 2003. An introduction to MCMC for machine learning. *Machine Learning* 50(1-2): 5–43.

[4] Arulampalam, M. Sanjeev, Simon Maskell, Neil Gordon, and Tim Clapp. 2001. A tutorial on particle filters. *IEEE Transactions of Signal Processing* 50:174–188.

[5] Bauer, Eric, and Ron Kohavi. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning* 36(1-2): 105–139.

[6] Bishop, Charles. 2006. *Pattern Recognition and Machine Learning*. Springer.

[7] Blum, Harry. 1967. A transformation for extracting new descriptions of shape. In *Models for the Perception of Speech and Visual Form*, ed. Weiant Wathen-Dunn, 362–380. Cambridge, MA: MIT Press.

[8] Burgie, E. Sethe, Craig Bingman, S. Leigh Grundhoefer, Ameet Soni, and George Phillips. In Press. Structural characterization of UCH37 reveals the basis of its auto-inhibitory mechanism. *Protein: Structure, Function, and Bioinformatics*. PDB ID: 3IHR.

[9] Carpenter, Elisabeth, Konstantinos Beis, Alexander Cameron, and So Iwata. 2008. Overcoming the challenges of membrane protein crystallography. *Current Opinion in Structural Biology* 18(5):581–586.

[10] Chothia, Cyrus. 1976. The nature of the accessible and buried surfaces in proteins. *Journal of Molecular Biology* 105(1):1–12.

[11] Collaborative Computational Project, Number 4. 1994. The CCP4 suite: Programs for protein crystallography. *Acta Crystallographica Section D* 50:760–763.

[12] Cowtan, Kevin. 2001. Fast Fourier feature recognition. *Acta Crystallographica Section D* 57:1435–1444.

[13] ———. 2006. The Buccaneer software for automated model building. *Acta Crystallographica Section D* 62(9):1002–1011.

[14] Cristianini, Nello, and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. 1st ed. Cambridge University Press.

[15] Crowther, R. Anthony. 1972. *The Molecular Replacement Method*, vol. 13. Gordon and Breach.

[16] Dempster, Arthur, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B39(1):1–38.

[17] Dietterich, Thomas. 2000. Ensemble methods in machine learning. *Lecture Notes in Computer Science* 1857:1–15.

[18] DiMaio, Frank. 2007. Probabilistic methods for interpreting electron-density maps. Ph.D. thesis, Department of Computer Sciences, University of Wisconsin–Madison.

[19] DiMaio, Frank, Dmitry Kondrashov, Eduard Bitto, Ameet Soni, Craig Bingman, George Phillips, and Jude Shavlik. 2007. Creating protein models from electron-density maps using particle-filtering methods. *Bioinformatics* 23:2851–2858.

[20] DiMaio, Frank, and Jude Shavlik. 2006. Belief propagation in large, highly connected graphs for 3D part-based object recognition. In *Proceedings of the Sixth International Conference on Data Mining*, 845–850. Hong Kong: IEEE Computer Society.

[21] DiMaio, Frank, Jude Shavlik, and George Phillips. 2006. A probabilistic approach to protein backbone tracing in electron-density maps. *Bioinformatics* 22(14):e81–89.

[22] DiMaio, Frank, Ameet Soni, George Phillips, and Jude Shavlik. 2007. Improved methods for template matching in electron-density maps using spherical harmonics. In *Proceedings of the 2007 International Conference on Bioinformatics and Biomedicine*, 258–265. Washington, DC: IEEE Computer Society.

[23] ———. 2009. Spherical-harmonic decomposition for molecular recognition in electron-density maps. *International Journal of Data Mining and Bioinformatics* 3(2):205–227.

[24] DiMaio, Frank, Ameet Soni, and Jude Shavlik. 2008. Machine learning in structural biology: Interpreting 3D protein images. In *Introduction to Machine Learning and Bioinformatics*, ed. Sushmita Mitra, Sujay Datta, Theodore Perkins, and George Michailidis, chap. 8, 237–276. Chapman & Hall/CRC Press.

[25] Doucet, Arnaud, Simon Godsill, and Christophe Andrieu. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* 10(3):197–208.

[26] Dunker, A. Keith, Ethan Garner, Stephen Guilliot, Pedro Romero, Kurt Albrecht, John Hart, Zoran Obradovic, Charles Kissinger, and J. Ernest Villafranca. 1998. Protein disorder and the evolution of molecular recognition: Theory, predictions and observations. *Pacific Symposium on Biocomputing* 473–484.

[27] Elidan, Gal, Ian McGraw, and Daphne Koller. 2006. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*.

[28] Emsley, Paul, and Kevin Cowtan. 2004. Coot: Model-building tools for molecular graphics. *Acta Crystallographica Section D* 60(12 Part 1):2126–2132.

[29] Felzenszwalb, Pedro, and Daniel Huttenlocher. 2006. Efficient belief propagation for early vision. *International Journal of Computer Vision* 70(1).

[30] Ferron, Francois, Sonia Longhi, Bruno Canard, and David Karlin. 2006. A practical overview of protein disorder prediction methods. *Proteins: Structure, Function, and Bioinformatics* 65(1):1–14.

[31] Friedman, Nir. 2004. Inferring cellular networks using probabilistic graphical models. *Science* 303(5659):799–805.

[32] Gao, Cen, and Harry Stern. 2007. Scoring function accuracy for membrane protein structure prediction. *Proteins* 68:67–75.

[33] Geman, Stuart, and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721–741.

[34] Getoor, Lisa, and Ben Taskar. 2007. *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT Press.

[35] Greer, Jonathan. 1974. Three-dimensional pattern recognition. *Journal of Molecular Biology* 82:279–301.

[36] Healy, Dennis, Harrie Hendriks, and Peter Kim. 1993. Spherical deconvolution with application to geometric quality assurance. Technical Report, Department of Mathematics and Computer Science, Dartmouth College.

[37] Healy, Dennis, Dan Rockmore, Peter Kostelec, and Sean Moore. 2003. FFTs for the 2-sphere – improvements and variations. *Journal of Fourier Analysis and Applications* 9:341–385.

[38] Heckerman, David. 1990. Probabilistic similarity networks. *Networks* 20(5): 607–636.

[39] Holm, Liisa, and Jong Park. 2000. DaliLite workbench for protein structure comparison. *Bioinformatics* 16:566–567.

[40] Huang, Heng, Li Shen, Rong Zhang, Fillia Makedon, Bruce Hettleman, and Justin Pearlman. 2005. Surface alignment of 3D spherical-harmonic models: Application to cardiac MRI analysis. In *Proceedings of the Eighth International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 8, 67–74. Palm Springs, CA.

[41] Ioerger, Thomas, and James Sacchettini. 2002. Automatic modeling of protein backbones in electron-density maps via prediction of Cα coordinates. *Acta Crystallographica Section D* 58(12):2043–2054.

[42] ———. 2003. The TEXTAL system: Artificial intelligence techniques for automated protein model building. *Methods in Enzymology* 374:244–270.

[43] Joachims, Thorsten. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, ed. Bernhard Schlokopf, Christopher Burges, and Alexander Smola. Cambridge, MA: MIT Press.

[44] Jones, T. Alwyn, Jin-yu Zou, Sandra Cowan, and Morten Kjeldgaard. 1991. Improved methods for building protein models in electron density maps and the location of errors in these models. *Acta Crystallographica Section A* 47(2): 110–119.

[45] Joosten, Krista, Serge Cohen, Paul Emsley, Wijnand Mooij, Victor Lamzin, and Anastassis Perrakis. 2008. A knowledge-driven approach for crystallographic protein model completion. *Acta Crystallographica Section D* 64(4):416–424.

[46] Jordan, Michael, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. 1999. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, ed. Michael Jordan. Cambridge, MA: MIT Press.

[47] Kabsch, Wolfgang, and Christian Sander. 1983. Dictionary of Protein Secondary Structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22:2577–2637.

[48] Karmali, Anjum, Tom Blundell, and Nicholas Furnham. 2009. Model-building strategies for low-resolution X-ray crystallographic data. *Acta Crystallographica Section D* 65(2):121–127.

[49] Kirillov, Alexandre. 1994. *Representation Theory and Noncommutative Harmonic Analysis*, vol. 22. Springer.

[50] Koller, Daphne, and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. Cambridge, MA: MIT Press.

[51] Kondor, Risi. 2007. A complete set of rotationally and translationally invariant features for images. *CoRR* abs/cs/0701127. `http://arxiv.org/abs/cs/0701127`.

[52] Kostelec, Peter, and Dan Rockmore. 2003. FFTs on the rotation group. Technical Report 03-11-060, Santa Fe Institute's Working Paper Series.

[53] de La Fortelle, Eric, and Gérard Bricogne. 1997. Maximum-likelihood heavy-atom parameter refinement for the multiple isomorphous replacement and

multiwavelength anomalous diffraction methods. *Methods in Enzymology* 276: 472–494.

[54] Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289. San Francisco, CA: Morgan Kaufmann Publishers Inc.

[55] Leherte, Laurence, Janice Glasgow, Kim Baxter, Evan Steeg, and Suzanne Fortier. 1994. Analysis of three-dimensional protein images. *Journal of AI Research* 7: 122–159.

[56] Linding, Rune, Lars Juhl Jensen, Francesca Diella, Peer Bork, Toby Gibson, and Robert Russell. 2003. Protein disorder prediction: Implications for structural proteomics. *Structure* 11(11):1453–1459.

[57] Lundstrom, Kenneth. 2006. Structural genomics for membrane proteins. *Cellular and Molecular Life Sciences* 63:2597–2607.

[58] MacCallum, Robert. 2004. Striped sheets and protein contact prediction. *Bioinformatics* 20 Suppl 1:i224–231.

[59] Maclin, Richard, and David Opitz. 1997. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 546–551.

[60] MacQueen, James. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 281–297.

[61] McKee, Erik, Lalji Kanbi, Kevin Childs, Ralf Grosse-Kunstleve, Paul Adams, James Sacchettini, and Thomas Ioerger. 2005. FINDMOL: Automated identification of macromolecules in electron-density maps. *Acta Crystallographica Section D* 61(11):1514–1520.

[62] Mclachlan, Geoffrey, and Thriyambakam Krishnan. 1996. *The EM Algorithm and Extensions*. 1st ed. Hoboken, NJ: Wiley-Interscience.

[63] Mitchell, Tom. 1997. *Machine Learning*. McGraw-Hill.

[64] Momen-Roknabadi, Amir, Mehdi Sadeghi, Hamid Pezeshk, and Sayed-Amir Marashi. 2008. Impact of residue accessible surface area on the prediction of protein secondary structures. *BMC Bioinformatics* 9:357.

[65] Morris, Richard, Anastassis Perrakis, and Victor Lamzin. 2003. ARP/wARP and automatic interpretation of protein electron density maps. *Methods in Enzymology* 374:229–244.

[66] Muppirala, Usha, and Zhijun Li. 2006. A simple approach for protein structure discrimination based on the network pattern of conserved hydrophobic residues. *Protein Engineering Design and Selection* 19:265–275.

[67] Murphy, Kevin, Yair Weiss, and Michael Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 467–475.

[68] Murshudov, Garib, Alexei Vagin, and Eleanor Dodson. 1997. Refinement of macromolecular structures by the maximum-likelihood method. *Acta Crystallographica Section D* 53:240–255.

[69] Ngan, Shing-Chung, Michael Inouye, and Ram Samudrala. 2006. A knowledge-based scoring function based on residue triplets for protein structure prediction. *Protein Engineering Design and Selection* 19:187–193.

[70] Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufman Publishers.

[71] Perrakis, Anastassis, Richard Morris, and Victor Lamzin. 1999. Automated protein model building combined with iterative structure refinement. *Nature Structural and Molecular Biology* 6(5):458–463.

[72] Protein Data Bank (PDB). 2011. PDB current holdings breakdown. `http://www.rcsb.org/pdb/statistics/holdings.do`.

[73] Ramachandran, Gopalasamudram Narayana, Chandrasekharan Ramakrishnan, and V. Sasisekharan. 1963. Stereochemistry of polypeptide cain configurations. *Journal of Molecular Biology* 7:95–99.

[74] Rhodes, Gale. 2000. *Crystallography Made Crystal Clear: A Guide for Users of Macromolecular Models*. New York; London: Academic Press.

[75] Risbo, Torben. 1996. Fourier transform summation of Legendre series and D-functions. *Journal of Geodesy* 70:383–396.

[76] Rost, Burkhard, and Chris Sander. 1993. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology* 232:584–599.

[77] Simons, Kim, Ingo Ruczinski, Charles Kooperberg, Brian Fox, Chris Bystroff, and David Baker. 1999. Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. *Proteins* 34:82–95.

[78] Soni, Ameet, Craig Bingman, and Jude Shavlik. 2010. Guiding belief propagation using domain knowledge for protein-structure determination. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*. Niagara Falls, NY.

[79] Soni, Ameet, and Jude Shavlik. 2011. Probabilistic ensembles for improved inference in protein-structure determination. In *Proceedings of the Second ACM International Conference on Bioinformatics and Computational Biology*. Chicago, IL.

[80] Sutton, Charles, and Andrew McCallum. 2007. Improved dynamic schedules for belief propagation. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*.

[81] Terwilliger, Thomas. 2000. Structural genomics in North America. *Nature Structural Biology* 7:935–939.

[82] ———. 2003. Automated main-chain model building by template matching and iterative fragment extension. *Acta Crystallographica Section D* 59(1):38–44.

[83] Trapani, Stephano, and Jorge Navaza. 2006. Calculation of spherical harmonics and Wigner D functions by FFT: Applications to fast rotational matching in molecular replacement and implementation into AMoRe. *Acta Crystallographica Section A* 62:262–269.

[84] Wainwright, Martin, Tommi Jaakkola, and Alan Willsky. 2003. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*.

[85] Wang, Guoli, and Roland Dunbrack. 2003. PISCES: A protein sequence culling server. *Bioinformatics* 19:1589–1591.

[86] Weiss, David, Benjamin Sapp, and Ben Taskar. 2010. Sidestepping intractable inference with structured ensemble cascades. In *Advances in Neural Information Processing Systems 23*, ed. John Lafferty, Chris Williams, John Shawe-Taylor, Richard Zemel, and Aron Culotta, 2415–2423.

[87] Weiss, Yair, and William Freeman. 2001. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation* 13: 2173–2200.

[88] White, Stephen. 2009. Biophysical dissection of membrane proteins. *Nature* 459:344–346.

[89] Zhang, Yang. 2008. Progress and challenges in protein structure prediction. *Current Opinion in Structural Biology* 18(3):342 – 348.