Protein Structure Prediction from Low-Resolution Electron Density Data using Particle Belief Propagation

Roshan Rao

May 2, 2017

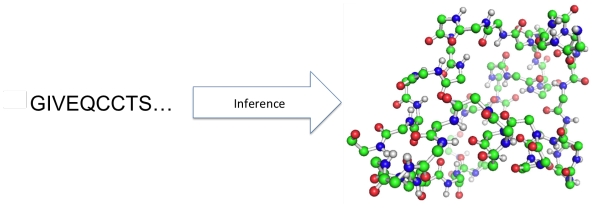
1 Introduction

3D protein structure prediction is a seminal problem in computational biology. The successful prediction of protein structure could lead to significant improvements in drug design, such as the production of enzymes targeted to perform some specific function. There are two paradigms for protein structure prediction: *ab initio* prediction, which is prediction from an amino acid sequence alone, and density-guided prediction, which is prediction from an amino acid sequence and an electron density. In this work we focus on the second paradigm, specifically looking at low-resolution density data.

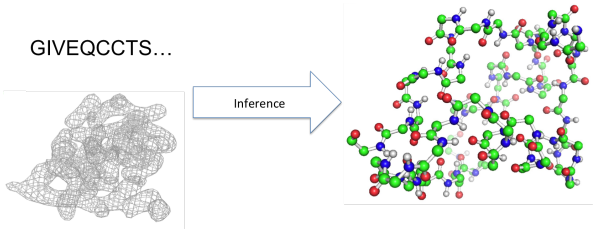
Recent advancements in a technology known as Cryo-Electron Microscopy (Cryo-EM) have allowed for the creation of near-atomic resolution electron density maps. These maps have several major advantages over the common alternative of x-ray crystallography. X-ray crystallography requires a crystallized protein; creating this is generally a slow and expensive process, and for some proteins can be impossible [4]. However, Cryo-EM maps have poorer resolution (often greater than 4 Å), which prevents current tools for automatic reconstruction on X-ray crystallographic data from converging to the correct model on Cryo-EM data [6]. Our algorithm is designed to predict protein structure from this low-resolution data, although it will work for x-ray crystallography data or any other electron density map.

The density-guided protein structure prediction problem takes as input a sequence of amino acids and a 3D electron density map and seeks to output highly accurate locations for every atom in the protein. Current methods for 3D structure reconstruction specifically from low-resolution data rely on either hand-constructed initializations [6] or extremely time consuming pre-processing [7].

This thesis proposes a fast method for determining 3D protein structure from low-



(a) *Ab initio*



(b) Density-guided

Figure 1: The two paradigms of protein structure prediction. We focus on the second paradigm - protein structure prediction from low-resolution data.

resolution denisty data based on the Diverse Particle Max Product (D-PMP) algorithm developed by Pacheco et al. [1]. This work extends the Rosetta protein folding library, a third party software produced by UW [2]. We formulate the structure prediction problem as MAP inference on a complete pairwise Markov Random Field (MRF). As D-PMP is an iterative algorithm, we note that at each iteration many of the edges in the MRF are non-informative and then propose a method for dynamically estimating a sparse subgraph on which to perform inference. Finally, we define three proposal functions for sampling new particles that help to refine the initial estimate.

When testing our algorithm on protein 4NIB, it recovers the correct structure with RMSD < 0.5 Å when provided with a 1.3 Å electron density input and a strong initialization (random walk with large variance around the correct structure). We are in the process of testing using lower resolution inputs and are exploring different methods of initialization.

1.1 Contributions

The engineering in this work was a joint effort on the part of Jason Pacheco and myself. Jason and my advisor Erik Sudderth were also instrumental in identifying problems and finding new research directions during our weekly meetings. Edward Williams worked with me on static graph structure estimation (Section 5.1) for a class project in CSCI 2420. The results and text in that section are taken from our project report.

2 Graphical Models and MAP Inference

This work frames protein structure prediction as a problem of continuous MAP inference on a graphical model. This section provides a brief introduction to graphical models, max-product belief propagation, and the extension Diverse Particle Max Product (D-PMP).

2.1 Markov Random Fields

Let $x = \{x_1, \ldots, x_N\}$ be a set of N random variables. Let $p(x)$ be a probability density function (pdf) over these variables. Graphical models are tools that represent p in a form that allows for efficient inference algorithms. Suppose p can be factored in the following manner for some set $\mathcal{C}$:

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c), \quad Z = \int_{\mathcal{X}} \prod_{c \in \mathcal{C}} \psi_c(x_c) dx_c \tag{1}$$

where each c is a subset of $\{1, \ldots, N\}$, x_c contains the variables who's indices are in c, and ψ_c is a function over these variables, commonly referred to as a *potential function*. A graph $G = (\mathcal{V}, \mathcal{E})$ is a Markov Random Field (MRF) with respect to a distribution $p(x)$ if it has nodes $\mathcal{V} = \{v_i \mid 1 \leq i \leq N\}$ and edges $\mathcal{E}$ such that each $c \in \mathcal{C}$ represents a clique in G. Graphical models are important because they encode independence relationships between variables. Algorithms can then leverage these independence relationships to perform efficient inference.

The goal in this work is to find the Maximum *a Posteriori* (MAP) estimate of a distribution p. Since we only want to maximize this distribution, we can ignore the normalization constant Z. We can also maximize the log of the distribution, since this will maximize the distribution as well. If we let $f(x) \propto \log p(x)$, then

$$x^{\text{MAP}} = \operatorname*{argmax}_x f(x) = \operatorname*{argmax}_x \sum_{c \in \mathcal{C}} \psi_c(x_c) \tag{2}$$

This work employs a specific form of MRF known as a Pairwise MRF. In Pairwise MRFs, all potential functions are functions of either one or two variables. So now f can be factorized as

$$f(x) = \sum_{s \in \mathcal{V}} \psi_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \psi_{st}(x_s, x_t) \tag{3}$$

This both simplifies inference equations and drastically increases speed.

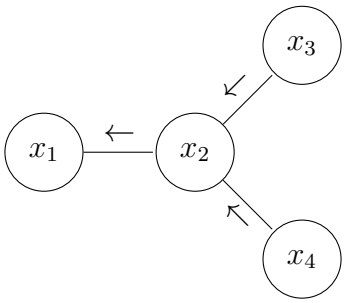


Figure 2: Graphical representation of Equation 4. Arrows represent direction of message passing.

2.2 Max Product Belief Propagation

When all variables x_i are discrete, max product belief propagation is an efficient MAP inference algorithm [12]. Consider the following pairwise MRF,

$$f(x_1, x_2, x_3, x_4) = \psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\psi_{24}(x_2, x_4) \tag{4}$$

To find the value of x that maximizes f, let

$$\max_x f(x) = \max_{x_1} \max_{x_2} \max_{x_3} \max_{x_4} \psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\psi_{24}(x_2, x_4) \tag{5}$$

By using the distributive law, this can be rearranged so that

$$\max_x f(x) = \max_{x_1} \underbrace{\max_{x_2} \psi_{12}(x_1, x_2) \underbrace{\left[\max_{x_3} \psi_{23}(x_2, x_3)\right]}_{m_{32}(x_2)} \underbrace{\left[\max_{x_4} \psi_{24}(x_2, x_4)\right]}_{m_{42}(x_2)}}_{m_{21}(x_1)} \tag{6}$$

The quantities $m_{32}(x_2)$ and $m_{42}(x_2)$ are independent of each other given x_2, so they can be maximized independently. These quantities are referred to as *messages* because they can be thought of as messages along the edges of a graphical model. The final message $m_{21}(x_1)$ is then calculated as the product of the two incoming messages and the ψ_{12} potential. The MAP estimate can also be calculated by keeping track of the values that maximize each message for each variable. In general the message update equation for a message from node t to node s for a pairwise MRF is,

$$m_{ts}(x_s) = \max_{x_t} \psi_t(x_t)\psi_{st}(x_s, x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t) \tag{7}$$

where $\Gamma(t)$ is the set of neighbors of node t in G. When G is a tree-structured (acyclic) graph, max product belief propagation calculates the exact MAP estimate after two message update passes. If G is a cyclic graph, then max product belief propagation calculates an approximation to the true function. Belief propagation is run until messages converge, but this does not guarantee a reasonable solution for all models [14] [16].

2.3 Reweighted Max-Product BP

Reweighted Max-Product (RMP) was introduced by Wainwright et al. [13] as a solution to this problem. It begins by introducing the distribution ρ, which is a distribution over spanning trees $\mathcal{T}$ of the graph G. This is then used in conjunction with Jensen's inequality to show an upper bound on the MAP log-probability

$$\max_x f(x) \leq \sum_{T \in \mathcal{T}} \rho(T) \max_x f_T(x) \tag{8}$$

where f_T includes only the potential functions in f that correspond to nodes and edges in the spanning tree T. This can then be rewritten in terms of edge appearance probabilities ρ_{st} for all $(s,t) \in \mathcal{E}$,

$$\max_x f(x) \leq \max_x \left[\sum_{s \in \mathcal{V}} \psi_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \rho_{st} \psi_{st}(x_s, x_t) \right] \tag{9}$$

which lead to the message update equation,

$$m_{ts}(x_s) = \max_{x_t} \psi_t(x_t) \psi_{st}(x_s, x_t)^{\frac{1}{\rho_{st}}} \frac{\prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t)^{\rho_{ut}}}{m_{st}(x_t)^{1-\rho_{st}}} \tag{10}$$

Notice that if $\rho_{st} = 1$ for all edges $(s,t) \in \mathcal{E}$, then the traditional Max-Product BP algorithm is recovered. However, RMP solves the MAP inference problem for a much larger class of models than traditional Max-Product BP.

To see why, we must explore the connection between MAP inference and Linear Programming (LP). Exact MAP inference is equivalent to solving an LP over the marginal polytope. The marginal polytope can be thought of as the space of valid distributions. Max-Product BP solves a relaxation of this LP by allowing potentially invalid distributions. This simplifies the problem, but is only guaranteed to converge to the true MAP result if the underlying distribution is tree structured. RMP instead maximizes a bound by approximating the true distribution with a convex combination of tree structured distributions [13].

RMP provides two guarantees. First, RMP is guaranteed to converge to a fixed point. Second, if there exists a consistent set of maximizers across all spanning trees, then an RMP fixed point is guaranteed to solve the MAP LP relaxation. It does not guarantee that the fixed point that RMP converges to will necessarily be the one that solves the MAP LP relaxation.

2.4 Diverse Particle Max Product

Both of the Max Product algorithms discussed work only on discrete random variables. The computed messages $m_{st}(x_t)$ are functions of x_t; when x_t is a discrete random variable, this function can be stored as a vector. If x_t is continuous, then it is unclear how

to compute and store the messages. Particle Max Product (PMP) extends Max Product algorithms into the continuous domain [15]. Traditional approaches to inference on graphs in the continuous domain either calculate messages directly with integrals (e.g. Kalman Filters) or discretize the state space and convert the problem into one of discrete inference. Unfortunately, the first method is impossible for most continuous distributions, while the second is only possible when the number of dimensions is very small. In protein structure prediction, each variable represents a position and orientation in 3D space, and there can be hundreds or thousands of variables, which makes discretization computationally infeasible.

For a continuous random variable, the standard RMP message update (Eq. 10) is a continuous maximization, which may be impossible to calculate. Instead, if $\mathcal{X}$ is the continuous domain, then at each iteration, PMP considers a finite discrete subset $\mathbb{X}$, and optimizes over this subset. The elements of this subset are known as *particles*. Since $\mathbb{X} \subset \mathcal{X}$, PMP optimizes a lower bound on the true function,

$$\max_{x \in \mathbb{X}} f(x) \leq \max_{x \in \mathcal{X}} f(x) \tag{11}$$

Each iteration of PMP is carried out in three stages. In the first stage, a series of stochastic proposals are used to augment the current particle set. This allows exploration of the state space to find higher-likelihood configurations. These proposals can be relatively simple, such as a random walk with gaussian noise. This type of proposal perturbs particles in the current set, allowing exploration of nearby configurations. In the second stage, RMP is run to approximate the max-marginal distribution of every particle (the approximate max-marginal distribution for a given node is a reweighted product of incoming messages and the unary potential). In the third stage, some subset of the particles are selected based on their max-marginal distributions.

There are three ways of selecting particles from the current set. The simplest method is *greedy* PMP (G-PMP), which only selects the argmax at each iteration. This makes inference highly efficient, but tends to fall into local optima very easily, and does a poor job of exploring the state space. *Top-N* PMP (T-PMP) selects the N particles at each node with the highest approximate max-marginal beliefs. This improves on G-PMP by having the potential to capture multiple local optima. However, many of the selected particles are essentially duplicates, with only slight alterations. This duplication in particles is inefficient computationally, and retains the problem of falling into local optima too quickly. The final selection method is *diverse* PMP (D-PMP), which encourages diversity in particle selection [1]. It does so by selecting a subset of particles that minimize distortion to the RMP messages (Eq. 10). Duplicate particles have the same value in the RMP message update; near-duplicate particles have extremely similar values. By minimizing message distortion, D-PMP provides a way to select particles at multiple local optima in every iteration. Additionally, the selection method operates entirely on the RMP update, so is not application-specific.

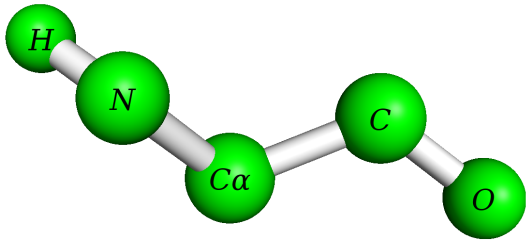


Figure 3: Single amino acid with atoms labeled using shorthand notation. Side chain is not represented in this (or any other) diagram.

3 Protein Structure Prediction

Here we introduce the problem of protein structure prediction with Cryo-EM data. This will explain protein and amino acid terminology, provide an introduction to Cryo-EM electron densities, and cover previous work in this field.

3.1 Protein and Amino Acid Terminology

A protein is a large molecule made up of a series of amino acids. A single amino acid is made up of two carbons, one oxygen, one nitrogen, and one hydrogen atom (see Figure 3). In addition, each amino acid has a side chain; different amino acid types correspond to different side chains. In this work, we do not represent the side chain explicitly, instead modeling it simply as a large atom. The side chain is not shown in any diagrams, and no energies refer explicitly to it. For brevity, each atom in the amino acid is referred to by its capitalized first letter, with the exception of one of the carbon atoms, which is known as the alpha carbon and is abbreviated Cα. When referring to a particular atom from amino acid i, the atom is indexed like so: N(i), Cα(i), etc. In a protein, every amino acid (except the first and last) is bonded only to the preceding and following amino acid.

Protein 'fragments' are used in many structure prediction applications [2]. A fragment is a section of a protein, usually 3 or 9 amino acids in length. Possible fragments are computed by comparing the input amino acid sequence to sequences of proteins with known structure, then generating a lookup table of protein fragments beginning at each amino acid in the sequence.

3.2 Electron Density

An electron density map is essentially an image of a protein. It provides an idea of where each amino acid in the protein is. High resolution electron density maps ($\sim$ 1.3

Å) allow a biochemist to identify the precise positions of most atoms in the protein. Lower resolution density maps will require weeks or months of work from a biochemist to identify these positions, and beyond a certain resolution threshold will prevent manual identification of atom locations [4].

The most common experimental method for obtaining an electron density map is X-Ray crystallography. This is a time consuming process in which a large quantity of the protein must be produced, purified, and crystallized. Crystallization is usually difficult, and requires a trial-and-error process with experimental conditions. After the crystal is created, a beam of X-Rays captures an image of the protein, and a Fourier transform then gives the electron density map [4].

A different approach to obtaining an electron density map is cryo-electron microscopy (Cryo-EM). Here, the chosen molecule is frozen using liquid nitrogen or liquid helium. The frozen molecule then undergoes electron microscopy, where electrons are fired at the specimen. The electron radiation, however, also damages the specimen. Freezing lessens the damage only at lower resolutions (4-20 Å), what are known as 'near-atomic' resolutions [3]. Although Cryo-EM is a much faster process than X-Ray crystallography, the resulting maps are usually too low resolution for manual identification of atomic locations. They are also too low resolution for current automatic methods of structure prediction that are designed for X-Ray crystallographic data, as these tend to fail when the resolution is below 3 Å [6].

3.3 Related Work

We use the Rosetta protein folding package (`https://www.rosettacommons.org/`) extensively. The package implements tools for protein representation and manipulation, as well as for calculation of potential functions. Standard Rosetta protein folding inference is designed for *ab initio* structure prediction (i.e. without an electron density map), but Rosetta also implements features for handling Cryo-EM density, including for scoring amino acids [2].

A series of papers by DiMaio provide a variety of methods for protein structure determination with Cryo-EM maps. The Automatic Crystallographic Map Interpreter (ACMI) is an algorithm that attempts to automatically find the location of every alpha carbon in the protein structure [4]. It has two steps: a *local matching* step and a *global constraint* step. The local matching step iterates over every amino acid and finds length 5 protein fragments centered around that amino acid. It then performs a 6D search over position and orientation to best fit the fragment into the Cryo-EM density map. This is a very computationally expensive search. The global constraint step considers these best fit locations as a discretization of possible states for each amino acid, thereby avoiding the issue of continuous optimization. It then represents the protein as a complete pairwise MRF, with edge potentials scoring distance between adjacent amino acids and prevent-

ing amino acids from occupying the same space. As the authors note, operating on a complete graph is inefficient; additionally, since the algorithm uses traditional belief propagation as opposed to reweighted belief propagation, the computed results can often be inaccurate.

A follow-up paper by DiMaio uses a *particle filter* approach to refine the placement of amino acids in the Cryo-EM density map performed by ACMI [5]. Particle Filtering is similar to to PMP, but cannot operate on graphs with cycles. Therefore, this model only considers potentials on edges between adjacent amino acids. Clashes between amino acids (where amino acids occupy the same space) cannot be scored, so are ignored. This works well only in the context of refinement, where the initialization is very strong.

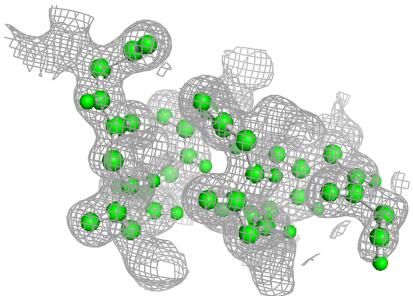
Wang et al. use a similar approach, first finding length 9 protein fragments centered around each amino acid, searching the electron density for the best position and orientation, then stitching together these best fit results with a follow up algorithm [7]. Here, they use Monte Carlo simulated annealing with a score function that does score clashes between amino acids. Simulated annealing avoids the problem of performing inference on a complete graph, but this still requires an exhaustive 6D search for every fragment for every amino acid.

4 D-PMP Inference for Structure Prediction

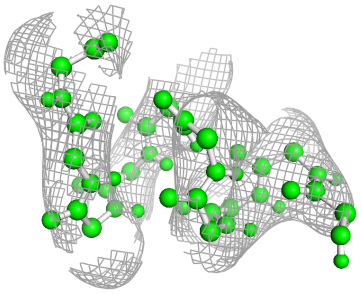
D-PMP requires three inputs from the user; a graphical model on which to perform inference, a function to optimize, and proposal functions to find higher likelihood configurations. This section covers these three topics, with the function broken into separate potential functions that are evaluated on nodes or edges. Additionally, it examines several methods of providing an initial particle set to the algorithm.

4.1 Graphical Model

Let $y_1, y_2, \ldots, y_N$ denote the input amino acid sequence. We then initially represent a protein as a complete MRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{v_i \mid 1 \leq i \leq N\}, \mathcal{E} = \{e_{ij} \mid \forall v_i, v_j \in \mathcal{V}\}$. D-PMP is a particle based optimization algorithm, so each state of each node in the MRF is a particle. In this case, a particle is an eight dimensional vector that represents a potential 3D position and orientation of the amino acid, as well as the angle of the C-O and N-H bonds. If we have N amino acids and M particles for each amino acid, then this is an MRF with N nodes and M states at each node. Let $\mathbb{X}$ represent the set of all particles, $\mathbb{X}_i$ represent the set of particles at the i-th node corresponding to the i-th amino acid in the input sequence, and $x_i^{(p)}$ represent the p-th particle in this set. A protein x then is a set containing one particle from each set $\mathbb{X}_i$, i.e. $x = \{x_i^{(p_i)} \mid 1 \leq i \leq N\}$.



(a) 1.3Å resolution energy map



(b) 4.0Å resolution energy map

Figure 4: Alpha helix from protein 4NIB inside two electron density maps at different resolutions. Traditional x-ray crystallography methods fail when resolution is worse than 3 Å.

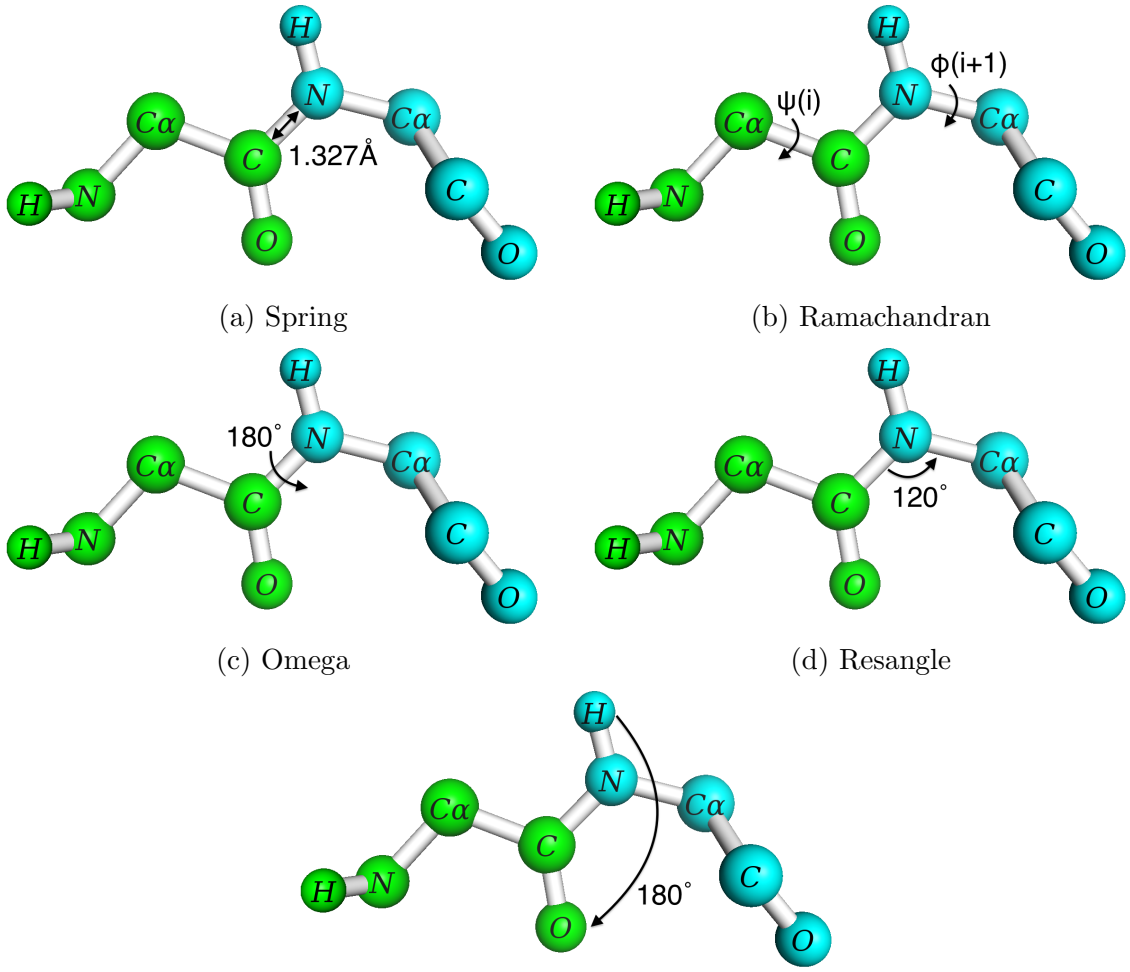
4.2 Potential Functions

Our algorithm maximizes a function f over this graph, where f takes in a protein and outputs a value in $\mathbb{R}$. To allow for efficient inference, we restrict f to be a function over a pairwise MRF, which makes inference quadratic in M. This is not a large restriction, since most protein potential functions are derived from energies in chemistry, which naturally decompose in this pairwise form. We use three types of potential functions in this model: an electron density potential, covalent bonding potentials, and a Van der Waals (repulsive) potential.

Electron Density Potential This potential is based on the input Cryo-EM map. The input map is discretized and the resulting discretization is interpolated to find the potential when provided with an input particle $x_i^{(p)}$. The term is implemented in Rosetta and we use this version without alteration. The electron density is a unary potential term. Figure 4 shows density maps at two different resolutions. We are interested in maps where the resolution is close to or worse than in Figure 4b.

Covalent Bonding Potentials There are five covalent bonding potentials in use. Covalent bonding potentials constrain bond geometry between adjacent amino acids (amino acids i and $i+1$). All five covalent potentials are specific to our model and were implemented by us. See Figure 5 for a detailed explanation of the five potentials.

Van der Waals Potential In chemistry, Van der Waals forces are both attractive and repulsive forces between atoms. For our purposes, we model only the repulsive portion of the Van der Waals forces. This potential acts to discourage atoms from clashing (occupying the same space or being too near each other). It is optimized for the case where we do not represent the side chain explicitly, and instead model the side chain as a large atom. The term is implemented in Rosetta and described in Rohl [2]. This is the only potential term that can be nonzero on any edge in the graph.



(a) Spring

(b) Ramachandran

(c) Omega

(d) Resangle

(e) Oxyhyd

Figure 5: The five covalent bonding potentials in use. Each subfigure depicts two amino acids with a black arrow showing the part of the geometry that is constrained. The green amino acid (bottom left in each image) is the i-th amino acid, and the blue amino acid (top right in each image) is the $(i+1)$-th amino acid. Except for the Ramachandran potential, all are quadratic penalties for deviation from ideal bond geometry. **(a)** Spring potential penalizes deviations from the ideal bond length of 1.327 Å between C(i) and N(i+1). **(b)** Ramachandran potential measures joint probability of $\psi(i)$ and $\phi(i+1)$ angles. Joint probability estimated empirically from a dataset of proteins taken from the Protein Data Bank. **(c)** Omega potential constrains Cα(i), C(i), and O(i) to be coplanar with Cα(i+1), N(i+1), and H(i+1). It does so by constraining the bond angle between C(i) and N(i+1) to be 180°. **(d)** Resangle potential constrains the angle defined by C(i), N(i+1), and Cα(i+1) to be 120°. **(e)** Oxyhyd potential constrains O(i) and H(i+1) such that the angle defined by the two atoms is 180°.

4.3 Proposal Functions

As part of the stochastic search process, we utilize three proposal functions to augment the particle set. At the beginning of every D-PMP iteration, one of these functions is chosen at random.

Random Walk + Optimization This proposal adds gaussian noise to a particle in the current set by setting $x_i^{\text{new}} = x_i^{(p)} + N(0, \Sigma)$. It then performs gradient optimization (LBFGS) by treating the resulting amino acid as a rigid body and optimizing its position in the electron density map. The resulting particle is then added to the set x_i.

Neighbor Based From the covalent energy terms described in Figure 5, it is clear that the geometry between neighboring amino acids is tightly constrained. The neighbor proposal sets $x_i^{\text{new}} = q_{\text{nbr}}(x_j^{(p)})$ where $j = i \pm 1$ and p is chosen randomly. The function q_{nbr} sets all bond geometry to ideal except for the ϕ and ψ angles, since this is the only component of neighboring geometry with significant variation. It then sets the ϕ and ψ angles (see Figure 5b for explanation of these angles) by sampling from the Ramachandran distribution, which is a joint distribution over ϕ and ψ angles.

Fragment + Optimization Fragment proposals are a central feature of Rosetta *de novo* protein folding [2]. The fragment proposal function is $x_{i:i+\ell}^{\text{new}} = q_{\text{frag}}(x_i^{(p)})$, where ℓ is the length (in number of amino acids) of the fragment. The function keeps $x_i^{(p)}$ in the same position and orientation, then sets the rest of the amino acids according to a protein fragment sampled from a protein with a similar amino acid sequence. As with the random walk proposal, we follow this with gradient optimization (LBFGS) by treating the entire fragment as a rigid body and optimizing its position in the electron density map. All resulting particles are then added to the particle set.

4.4 Initialization

To run D-PMP, it is necessary to provide an initial set of particles. In the first stages of testing, we use a 'cheating' initialization. This is done by taking a large-variance random walk around the ground truth structure. We use a standard deviation of five angstroms on the location of the amino acid and 120° variance on all angles. This provides a strong initialization, which is a good baseline to perform early evaluations. If the inference algorithm has trouble recovering from this initialization, then it will certainly perform poorly on non-'cheating' initializations.

The first non-'cheating' initialization we examined is similar to the Fragment + Optimization proposal. This initialization selects an amino acid i at random, samples a fragment containing i, then randomly places it somewhere in the electron density. This is followed by rigid-body gradient optimization of the electron density potential. The sampled amino acids are added to the initial particle set, and this process is repeated until the set is full. This is a relatively weak initialization; it can do well for smaller

proteins, but for larger proteins it may not place any particles near the correct location for some amino acids.

We are also looking at alternate methods of initializing. One possibility is initializing from the output of an existing structure prediction algorithm, such as Resolve [8]. Resolve identifies probable locations of Cα atoms in the electron density, but may not find appropriate locations for all Cα atoms. Even if only a fraction of Cα atoms are properly placed, D-PMP could propagate these good solutions outwards to other amino acids using neighbor based and fragment + optimization proposals, both of which use the location of one amino acid to place nearby amino acids.

5 Graph Structure Estimation

A major problem with f is that it is defined on the complete graph. BP performs poorly on a complete graph, often failing to converge or converging to an incorrect solution [16]. Additionally, even when successful, the resulting algorithm can take many iterations to converge, causing significant performance issues. Therefore, our goal is to approximate f with a function $\hat{f}$ that is defined on only a sparse subset of the edges in the complete graph. To do this, we note that although inter-atomic energies never vanish, they do have an inverse square relationship with distance. This allows us to remove edges between amino acids that are beyond some specified interaction distance, and thus have inter-atomic energy near zero. On the other hand, removing edges between amino acids that are within that interaction distance can significantly impact energy evaluation and cause belief propagation to provide incorrect results.

Our goal then is to estimate an edge set $\hat{\mathcal{E}}$ and graph $\hat{G} = (\mathcal{V}, \hat{\mathcal{E}})$, such that,

$$\hat{f}(x) = \sum_{s \in \mathcal{V}} \psi_s(x_s) + \sum_{(s,t) \in \hat{\mathcal{E}}} \psi_{st}(x_s, x_t) \tag{12}$$

Since f is defined on the complete graph, the relationship between f and $\hat{f}$ is given by,

$$\hat{f}(x) + \sum_{(s,t) \notin \hat{\mathcal{E}}} \psi_{st}(x_s, x_t) = f(x) \tag{13}$$

5.1 Static Prediction

One approach to this problem is to predict the graph structure from the amino acid sequence beforehand. During initial testing of the structure prediction algorithm, we had been using the true graph structure, obtained using knowledge of the correct protein structure and a maximum interaction distance of 10 Å. This is generally a sparse graph containing $O(N)$ edges, rather than $O(N^2)$ edges. In many cases, this graph structure

had worked well, so we looked into methods for estimating the adjacency matrix of a protein given its amino acid sequence, a subproblem of structure prediction known as protein contact prediction.

Modern approaches to protein contact prediction typically rely on both the input amino acid sequence and on relationships between homologous protein domains, which are subdivisions of proteins that are evolutionarily related [9] [10] [11]. This requires structural information on a large number of evolutionarily related proteins, which we hoped to avoid. Additionally, these methods tend to produce many false negatives in their final predictions. False negatives would cause significant changes in the energy function, so these methods are not suitable for our purposes.

To get around this problem, we attempted to train a simple logistic regression model on amino acid sequences. Since the goal was to eliminate false positives, we hoped that this method would allow us to eliminate some edges, even if the resulting graph was more dense than strictly necessary. We used three features extracted from the observed amino acid sequence y. These were a distance feature $\phi_{\text{dist}}(y_i, y_j) = |i - j|$, a sequence length feature $\phi_{\text{seqlen}}(y) = N$ (which acts as a prior), and amino acid pair features. The amino acid pair features were defined as follows: let A, B be two types of amino acids. Then $\phi_{A,B}(y_i, y_j) = \mathbb{1}\{y_i = A, y_j = B\}$, with $\phi_{A,B} = \phi_{B,A}$. Since there are 20 total types of amino acids, this creates 210 distinct indicator variables. Let a_{ij} be an indicator variable with $a_{ij} = \mathbb{1}\{e_{ij} \in \hat{\mathcal{E}}\}$. Then the logistic regression likelihood function is then given by,

$$\mathcal{L}(y, \theta) = \exp\left\{ \sum_{i,j} a_{ij} \left[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \right] - \Phi(y, \theta) \right\} \tag{14}$$

The maximum likelihood training goal is to find the weights θ that maximize this function. $\Phi(y, \theta)$ is the log normalization constant. Additionally, we note that an edge between amino acids y_i, y_j is more likely to be present if edges between amino acids y_i, y_k and between amino acids y_j, y_k are also present - in other words, $\mathbb{P}(e_{ij} \in \hat{\mathcal{E}} \mid e_{ik} \in \hat{\mathcal{E}}, e_{jk} \in \hat{\mathcal{E}}) > \mathbb{P}(e_{ij} \in \hat{\mathcal{E}})$. These cannot be incorporated in logistic regression, so we modeled the problem as inference on a conditional random field (CRF), shown in the graphical model in Figure 6. The likelihood function for the CRF is given by,

$$\begin{aligned} \mathcal{L}(y, \theta) = \exp\Bigg\{ & \sum_{i,j} a_{ij} \left[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \right] \\ & \sum_{i,j,k} \Big(\theta_2 \mathbb{1}\{a_{ij} + a_{ik} + a_{jk} = 2\} + \theta_3 \mathbb{1}\{a_{ij} + a_{ik} + a_{jk} = 3\} \Big) - \Phi(y, \theta) \Bigg\} \end{aligned} \tag{15}$$

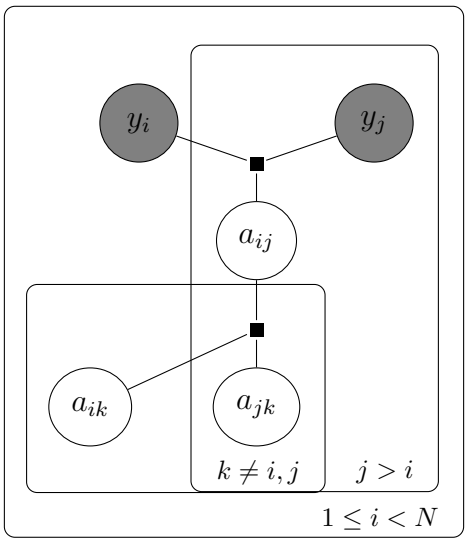


Figure 6: Graphical model of Eq. 15. A CRF is similar to an MRF, but is conditioned on observed data. In this example, the input amino acid sequence y is observed; observed variables are differentiated from non-observed variables by shading. a_{ij} has factor representing conditional probability given y_i, y_j. a_{ij}, a_{jk}, a_{ik} have factors increasing the probability of the third if the other two equal 1.

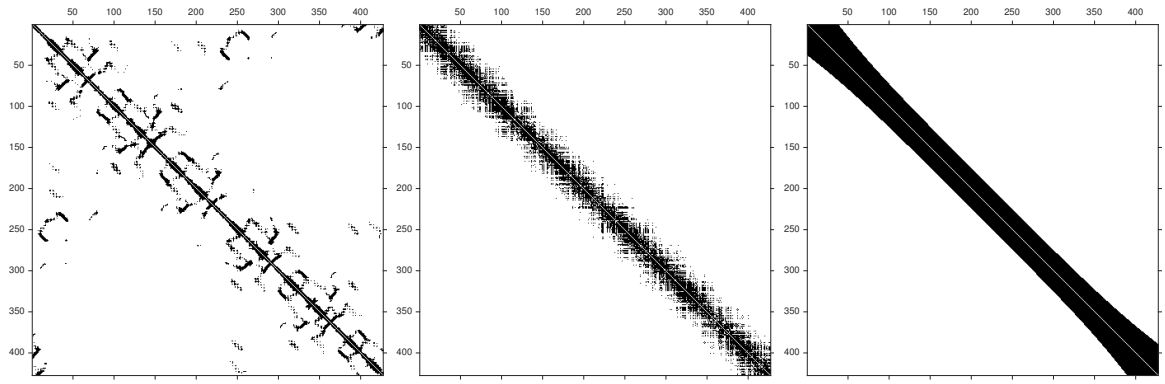
The CRF likelihood objective (Eq. 15) is difficult to compute. It requires an iteration over all triplets of amino acids, so is an $O(N^3)$ function, where N can be over 1000. Instead, we optimize the pseudo-log likelihood, which maximizes the conditional likelihood of a node a_{ij} give its neighbors $a_{\Gamma(a_{ij})}$. The pseudo-log likelihood objective function is [17],

$$PLL(y, \theta) = \sum_{ij} \log p(x_{ij} | a_{\Gamma(a_{ij})}) - \log \left[\exp(\log p(a_{ij} = 1 | x_{\Gamma(a_{ij})})) + \exp(\log p(a_{ij} = 0 | x_{\Gamma(a_{ij})})) \right] \quad (16)$$

$$\log p(a_{ij} = 1 | x_{\Gamma(a_{ij})}) = \left[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \right] + \sum_k \left[\theta_3 \mathbb{1}\{a_{ik} + a_{jk} = 2\} + \theta_2 \mathbb{1}\{a_{ij} + a_{jk} = 1\} \right] \quad (17)$$

$$\log p(a_{ij} = 0 | a_{\Gamma(a_{ij})}) = \sum_k \theta_2 \mathbb{1}\{a_{ik} + a_{jk} = 2\} \quad (18)$$

Unfortunately, we found that these features provided very little signal for determining



(a) True adjacency matrix. (b) Logistic regression. (c) Pseudo-likelihood.

Figure 7: True adjacency matrix and estimated matrices using logistic and pseudo-likelihood learning for protein 1EPS.

contacts. This can be best seen when visualizing the predicted edge sets (via adjacency matrices) for individual proteins, as in Figure 7.

Both methods used show a heavy reliance on the sequence distance feature; essentially no long range contacts are captured at all. The pseudo log-likelihood estimate in particular appears to ignore amino acid type altogether. This suggests that the current feature set is not able to provide enough information to predict long range contacts, and that only the distance feature is being used.

In addition to these problems, this method for estimating the graph structure presents other problems as well. First, any method that reduces the number of false negatives significantly will also return a large number of false positives. Second, any false positives in the initial estimate could result in compounding errors in D-PMP inference, so small mistakes could lead to large failures. Finally, and most importantly, the true graph structure is not actually the correct structure to perform inference on in every iteration. Although the true graph structure captures every significant energy term in the final protein structure, it does not do so for the intermediate structures that exist during inference. For example, suppose amino acids y_i and y_j are distant in the true structure. In this method, this implies that there is no edge between them. However, we want to be sure that if particles $x_i^{(p)}$ and $x_j^{(q)}$ are too close to each other, then the repulsion between them is scored, regardless of whether they are actually nearby in the final structure.

5.2 Dynamic Prediction

The previous approaches we considered relied on a static graph structure that would be used for all D-PMP iterations. However, noting that the positions of amino acids can change significantly over the course of inference, it makes little sense to use a single

sparse graph structure for every iteration. Taking this into account, we explored methods for dynamically altering the graph structure over the course of D-PMP inference.

Unlike in the previous section, where we estimate a graph structure based on the input amino acid sequence, here we estimate a graph structure that will be most useful for inference on the current set of particles. First, the potential functions that we use to model a protein have a special structure (see Equation 19). The electron density potential is a unary term, and so does not add edges to the graph. All covalent bonding potentials apply only to adjacent amino acids, so form a chain graph. Only the repulsive Van der Waals potential applies over all edges in the graph.

$$f(x) = \underbrace{\sum_{i=1}^{N} \psi_{\text{ElecDens}}(x_i) + \sum_{i=1}^{N-1} \psi_{\text{Covalent}}(x_i, x_{i+1})}_{\text{Chain-structured potentials}} + \underbrace{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \psi_{\text{VDW}}(x_i, x_j)}_{\text{Non-chain-structured potential}} \tag{19}$$

Moreover, the Van der Waals energy is repulsive, and so only applies to amino acids that are nearby. Beyond a certain distance this potential is zero, so we can always drop edges where the corresponding particles are far enough away that there is no repulsive energy between them. This gives us a new way to estimate a graph structure. Before running belief propagation, we can check each edge to determine if the Van der Waals potential is non-zero for any particle in the current particle set. To construct the edge set, let

$$|i - j| = 1 \implies e_{ij} \in \hat{\mathcal{E}} \tag{20}$$

$$\exists x_i^{(p)}, x_j^{(q)} \in \mathbb{X} \text{ s.t. } \psi_{\text{VDW}}(x_i^{(p)}, x_j^{(q)}) \neq 0 \implies e_{ij} \in \hat{\mathcal{E}} \tag{21}$$

Theorem 1. *Suppose $\hat{f}$ is defined according to Eq. 12, and $\hat{\mathcal{E}}$ is defined according to Eqs. 20 and 21. Then $\hat{f}(x) = f(x) \forall x \subset \mathbb{X}$.*

Proof. Combining Eq. 13 and Eq. 19,

$$\hat{f}(x) + \sum_{e_{ij} \notin \hat{\mathcal{E}}} \psi_{\text{Covalent}}(x_i, x_j) \mathbb{1}\{|i - j| = 1\} + \psi_{\text{VDW}}(x_i, x_j) = f(x) \tag{22}$$

By Eq. 20, $|i - j| = 1 \implies e_{ij} \in \hat{\mathcal{E}}$, so we can ignore ψ_{Covalent}. By the contrapositive of Eq. 21, $e_{ij} \notin \hat{\mathcal{E}} \implies \psi_{\text{VDW}}(x_i^{(p)}, x_j^{(q)}) = 0 \forall x_i^{(p)}, x_j^{(q)} \in \mathbb{X}$. Therefore $\hat{f}(x) = f(x) \forall x \subset \mathbb{X}$. □

This method has the major advantage that it only eliminates non-informative edges; by eliminating only edges where all potentials are zero, the resulting function is exactly equal to the original function, but is on a sparser graph. Unfortunately, the resulting

graph often is not sparse enough. Since we let $e_{ij} \in \hat{\mathcal{E}}$ if any pair of particles $x_i^{(p)}, x_j^{(q)}$ are close enough to interact, $\mathbb{P}(e_{ij} \in \hat{\mathcal{E}})$ increases as M increases.

An alternate goal to keeping the estimated $\hat{f}$ exactly the same as the true f is to find a function $\hat{f}$ such that the MAP estimate given the current particle set is the same under $\hat{f}$ as it is under f. To find this function, we initialize a chain graph, which can evaluate the electron density and covalent bonding energies, and iteratively add edges to it. Edges are added via the following method: run belief propagation on the chain graph and find the MAP estimate. Determine any clashes in the MAP estimate - amino acids that are close enough to repel each other - and add edges between clashing amino acids to the graph. Run belief propagation on the resulting graph, find clashes, add edges between clashing amino acids to the graph. Repeat this process until the MAP estimate has no clashing amino acids that are not already accounted for in the graph structure. This is formally codified in Algorithm 1.

Algorithm 1 Iterative estimation of a graph $\hat{G}$ such that $\text{argmax}_{x \subset \mathbb{X}} \hat{f}(x) = \text{argmax}_{x \subset \mathbb{X}} f(x)$, where $\hat{f}$ uses the same potential functions as f, but only over edges and nodes in $\hat{G}$.

function GETESTIMATEDEDGESET($\mathbb{X}$)
  Initialize $\hat{\mathcal{E}} = \{e_{ij} \text{ s.t. } |i-j| = 1\}$, $\hat{G} = (\mathcal{V}, \hat{\mathcal{E}})$
  Let $x^{\text{MAP}} = \text{argmax}_{x \subset \mathbb{X}} \hat{f}(x)$ ▷ argmax estimated via RMP
  Let $\mathcal{E}_{\text{clashing}} = \{e_{ij} \mid \psi_{\text{VDW}}(x_i^{\text{MAP}}, x_j^{\text{MAP}}) < 0, e_{ij} \notin \hat{\mathcal{E}}\}$
  while $|\mathcal{E}_{\text{clashing}}| > 0$ **do**
    $\hat{\mathcal{E}} = \hat{\mathcal{E}} \cup \mathcal{E}_{\text{clashing}}$
    $x^{\text{MAP}} = \text{argmax}_{x \subset \mathbb{X}} \hat{f}(x)$
    $\mathcal{E}_{\text{clashing}} = \{e_{ij} \mid \psi_{\text{VDW}}(x_i^{\text{MAP}}, x_j^{\text{MAP}}) < 0, e_{ij} \notin \hat{\mathcal{E}}\}$
  end while
  return $\hat{\mathcal{E}}$
end function

Theorem 2. *Assuming RMP returns the true MAP estimate each time,* $\text{argmax}_{x \subset \mathbb{X}} \hat{f}(x) = \text{argmax}_{x \subset \mathbb{X}} f(x)$ *where* $\hat{f}$ *uses the same potentials as* f *but only on the edges present in* $\hat{\mathcal{E}}$ *after Algorithm 1.*

Proof. By Eq. 22, and ignoring ψ_{COVALENT} since $\hat{\mathcal{E}}$ is initialized with all edges between adjacent amino acids,

$$\hat{f}(x) + \sum_{e_{ij} \notin \hat{\mathcal{E}}} \psi_{\text{VDW}}(x_i, x_j) = f(x) \; \forall x \subset \mathbb{X} \tag{23}$$

Since Van der Waals is a penalty only, we can also note that $\hat{f}(x) \geq f(x) \forall x$. Finally, at the end of the iterative process of adding edges, there are no clashing edges in

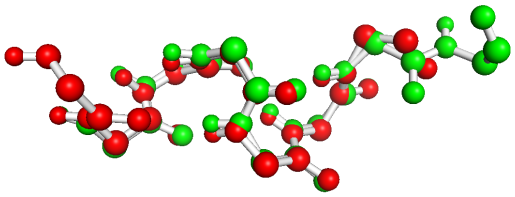


Figure 8: Shifted section of MAP estimate for one run from cheating initialization. Green shows ground truth position of amino acids, red shows estimated position of amino acids. In this section, amino acid i is placed where amino acid $i-1$ should be.

the MAP estimate that are not already included in the graph structure. Therefore $e_{ij} \notin \hat{\mathcal{E}} \implies \psi_{\text{VDW}}(x_i^{\text{MAP}}, x_j^{\text{MAP}}) = 0$, where x^{MAP} is the MAP estimate of the current particle set under $\hat{f}$. So,

$$f(x^{\text{MAP}}) = \hat{f}(x^{\text{MAP}}) \geq \hat{f}(x) \geq f(x) \; \forall x \subset \mathbb{X} \tag{24}$$

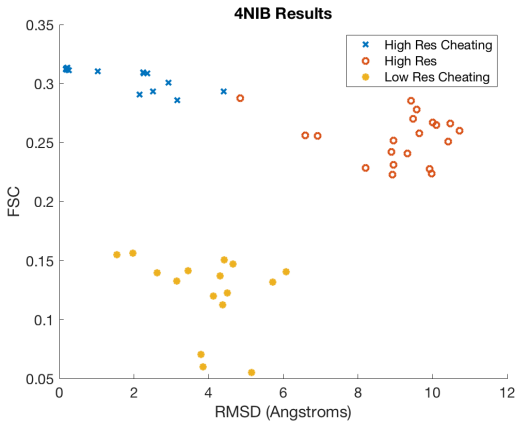
Which implies that x^{MAP} is the MAP estimate under f as well. □

This process allows us to create a sparse graph on which to perform belief propagation in each iteration of D-PMP. It guarantees that the MAP estimate in each iteration is the true MAP estimate, under the assumption that belief propagation converges to the correct solution. Since the graph is generally very sparse, this is a reasonable assumption. It does not guarantee that alternate modes will be correctly scored, but we believe that it provides a good approximation.

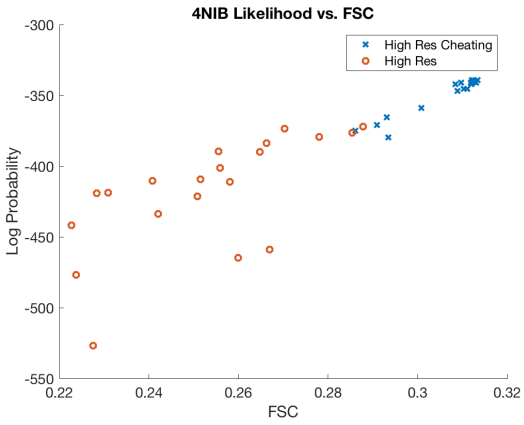
6 Experimental Results

In initial testing we evaluate on one protein, 4NIB. This is a small 51 amino acid protein. We test with a high resolution using both 'cheating' random walk and non-cheating fragment optimization initializations. We also test on low resolution electron density, but only using the 'cheating' random walk initialization.

We use two metrics to evaluate the success of an individual run. The first is the RMSD between predicted Cα positions and ground truth Cα positions. This provides a reasonable metric for success, but can be overly sensitive to particular types of errors. One common error that our model is sensitive to is a 'shift'. This is where the algorithm places amino acids in a shifted location; e.g. placing amino acid i where amino acid $i+1$ should go (see Figure 8). If this occurs for many i then the resulting structure has likelihood *almost* as high as the correct structure under our current model.



(a) High and low resolution results.



(b) Comparison of likelihood and FSC.

Figure 9: **(a)** Very low RMSD results for high resolution cheating initialization suggest that D-PMP obtains the correct structure the majority of the time. High FSC results show that when the correct structure is not obtained, a shifted version of the structure is obtained. Non-cheating initialization results are considerably worse. Trials with FSC above 0.2 generally find some portion of the structure. Trials with FSC above 0.24 find around half the protein structure. Trials with FSC above 0.27 generally find most of the protein structure. Every trial finds at least some portion of the structure, with several trials finding almost all of the structure. In low resolution, improved RMSD is likely an artifact of the cheating initialization. **(b)** Comparison suggests model is reasonable as trials with higher FSC tend to have higher likelihood. Low resolution results not shown because likelihood of low resolution and high resolution results cannot be directly compared.

This type of error could potentially be easily fixed in a post-processing step, since inference places some amino acid where there should be one, it just may not place the correct amino acid there. RMSD punishes these errors strongly, despite the fact that they are actually fairly close to the correct structure. The most striking example of this is when inference places the structure 'backwards'; i.e. when the first amino acid is placed where the last should be, and in general the i-th amino acid is place at location $N - i$. This has very high RMSD, but actually fits the electron density almost as well as the ground truth structure and could be fixed very easily by re-assigning amino acid types.

An alternative metric is Fourier Shell Correlation (FSC) [6]. This is a measure of how well the predicted structure explains the electron density. A high FSC implies that wherever an amino acid exists in the true structure, inference places some amino acid (but not necessarily the correct amino acid). A low FSC indicates that the predicted structure explains the electron density poorly, so is not just a shifted version of the ground truth structure.

We compare RMSD and FSC in each run using a scatterplot. There are three regions of interest in each scatterplot. The first contains runs with low RMSD and high FSC.

These explain the density well and place the correct amino acids in the correct locations, so are close to the ground truth structure. The second contains runs with high RMSD and high FSC. These explain the density well but place amino acids in incorrect locations, so probably get large parts of the structure correct but shifted. The third region contains runs with high RMSD and low FSC, which are generally failures where inference does not find a structure close to ground truth. There are no runs with low RMSD and high FSC because any run with low RMSD should explain the density well.

Theoretical runtime analysis of this algorithm is challenging, due to the dynamic graph structure estimation code. The worst case runtime here would involve adding only one edge each time, until the graph was complete. The average case runtime in real experiments is much better. When using a cheating initialization the entire algorithm ran in 20 minutes end-to-end. When using the non-cheating initialization, the algorithm runs in around one hour. The speed boost comes mostly in later iterations; as the MAP estimate becomes closer to ground truth, the graph structure becomes more and more tree-like. If the MAP estimate is very close to ground truth then the dynamically estimated graph structure is a tree, so the dynamic estimation, energy evaluation, and MAP estimation all run extremely quickly. These times are on the high resolution density, on low resolution density the algorithm can have a much harder time finding a tree-like structure. Additionally, the unary potential signal from the electron density is coarser and weaker, so inference finds it more difficult to resolve the MAP structure.

7 Discussion

Our results are highly dependent on both initialization and density resolution. To improve performance, we can examine three failure cases. One major failure case occurs when using the non-cheating initialization and low resolution density. In this case, inference fails to advance the MAP estimate significantly after the initialization. We believe that this is an error due to the coarse-grained nature of the low resolution density. Without a strong unary potential it is more difficult to approximate the MAP estimate in a given particle set. One option is to interpolate or otherwise create a finer-grained density from the low resolution density that could provide a better outcome.

A second common failure case occurs when large parts of the protein structure are found, possibly with shifts, but D-PMP misses other large sections. We believe that this is likely an issue with proposal functions. One issue with our current proposal set is that every proposal is a local change to a current particle. This allows for some exploration, but if there are no current particles anywhere near some section of the electron density, then no particles will be proposed in that area. There are various proposal functions that would be helpful here, but one strong possibility is to find the portion of the electron density that is not explained by the MAP estimate, and to propose particles in this area. This could have the benefit of 'filling in' unexplained

portions of the electron density, which could help maximize FSC.

The other common failure, shifts, has virtually identical probability to the true structure, only differing in electron density. We do not plan to focus on this failure case unless all other cases are resolved. The shifted structure is generally so close to the true structure that the error could be fixed by hand or by some post-processing step.

Ultimately, both the proposal functions and initializations we use are relatively simple and fast. Despite this, we can still achieve reasonable results on high resolution data a significant percentage of the time. Utilizing a stronger initialization or smarter proposals could lead to significant improvements in predicted structure.

References

[1] Pacheco, J., Sudderth, E., Proteins, Particles, and Pseudo-Max-Marginals: A Submodular Approach. *International Conference on Machine Learning*, 2015.

[2] Rohl, C. A., Strauss, C. EM., Misura, K. MS., Baker, D., Protein structure prediction using Rosetta. *Methods in enzymology*, 2004. 383: p. 66-93.

[3] Milne, J. L. S. *et al.* Cryo-electron microscopy: A primer for the non-microscopist. *The Federation of European Biochemical Societies (FEBS) Journal*, 2013. 280(1): p. 28-45.

[4] DiMaio, F., Shavlik, J., Philips, G. N., A probabilistic approach to protein backbone tracing in electron density maps. *Bioinformatics*, 2006. 22(14): p. e81-e89.

[5] DiMaio, F. *et al.*, Creating protein models from electron-density maps using particle-filtering methods. *Bioinformatics*, 2007. 23(21): p. 2851-2858.

[6] DiMaio, F. *et al.*, Atomic-accuracy models from 4.5-Å cryo-electron microscopy data with density-guided iterative local refinement. *Nature Methods*, 2015. 12(4): p. 361-365.

[7] Wang, R. Y., *et al.*, *De novo* protein structure determination from near-atomic-resolution cryo-EM maps. *Nature Methods*, 2015. 12(4): p. 335-340.

[8] Terwilliger, T. C., Automated main-chain model building by template matching and iterative fragment extension. *Acta Crystallographica*, 2013. D59, p. 38-44.

[9] Ekeberg, M. *et al.*, Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models. 2013. `arXiv:1211.1281 [q-bio.QM]`

[10] Ma J., Wang S., Wang Z., Xu J., Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning. *Bioinformatics*, 2015. 2015:btv472

[11] Golkov, V. *et al.*, Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images. *NIPS*, 2016.

[12] Pearl, J., *Probabilistic Reasoning in Intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[13] Wainwright, M. J., Jaakkola, T. S., Willsky, A. S., Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 2005. 51(11): p. 3697-3717.

[14] Wainwright, M. J., Jordan, M., Graphical models, exponential families, and variational inference. Technical report, UC Berkeley, Dept. of Statistics, 2003.

[15] Ihler, A., McAllester, D., Particle belief propagation. *AISTATS*, 2009. p. 256-263.

[16] Ihler, A., Fisher, J. W., Willsky, A. S., Loopy belief propagation: convergence and effects of message errors. *JMLR*, 2005. p. 905-936.

[17] Sutton, C., McCallum, A., An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 2011. 4(4): p. 266-373.