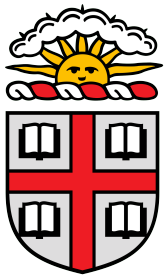
# Bridging Discrete and Continuous Protein Folding through Hydrophobicity and Molecular Dynamics

Colin Baker, Pranav Mahableshwarkar, Ritambhara Singh, and Sorin Istrail

Advisor: Sorin Istrail, Reader: Ritambhara Singh

*A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science with Honors in Computer Science at Brown University*



Brown University
Providence, RI
May 2025

**Bridging Discrete and Continuous Protein Folding through Hydrophobicity and Molecular Dynamics**


# Abstract

Protein structure prediction – the problem of predicting three-dimensional atomic structures from amino acid sequences – is critical to advanced drug discovery. Combinatorial models of folding, such as the HP model, have led to discrete space approximations of folds and a world of rigorous mathematics around minimizing different energy functions. Deep learning models, on the other hand, directly estimate atomic coordinates, generally leveraging evolutionary information from the input sequence in reference to some larger database. Both models miss or abstract away many of the inherent physical components and more complex energetic interactions of folding. In this work, we begin to address this issue by integrating molecular dynamics into both frameworks. For combinatorial modeling, we propose a new hydrophobicity scale *WaterFire* that uses molecular dynamics to better capture amino acid contacts. As we try to generalize our hydrophobicity scale, we run into an NP-completeness problem: reducing pairwise amino acid energy terms to the HP model is NP-complete. In deep learning models, specifically AlphaFold, we propose a new mechanism to adapt the intermediate embedding space to account for physical quantities in addition to evolutionary information, leading to our development of the *Subspace Relaxation Operator (SRO)*. We conclude that molecular dynamics has an important place in improving the physicality of existing protein structure prediction paradigms.

# Contents

# 1 Introduction

## 1.1 Roadmap

The prediction of protein structures from their amino acid sequences is of critical importance to both drug discovery and a better understanding of cellular processes. While recent advances in more implicit, deep learning approaches to protein folding have had significant success and attention in the past ten years [1], computational protein folding algorithms have historically relied on spatially discretizing the problem towards combinatorics. For example, one of the most fundamental models of protein folding is the hydrophobic-polar (HP) model, developed by Ken Dill, which borrows from biochemistry the result that protein folding is primarily driven by hydrophobic contacts [2]. In particular, the HP model treats each amino acid as hydrophobic or polar, simplifying the generally unknown objective function of protein folding into a maximization problem over adjacent hydrophobic amino acids. A number of foundational results have characterized optimal folding under this model on different lattices [3].

However, in more recent attempts to predict protein structures, groups such as John Jumper's lab at Google have utilized advances in linguistic deep learning to develop implicit predictors of three dimensional structure in continuous space. Jumper's foundational model – AlphaFold – has predicted the protein structures of millions of amino acid sequences without any direct simplifications [1]. While these deep learning models are beginning to yield results in drug discovery, their mechanism of prediction is obfuscated even to their architects. As a result, complete confidence in the predictions of such models is hard to justify.

In this work, we examine the potential of using molecular dynamics to improve both the past and present of protein structure prediction. To make combinatorial protein folding more physical, we define a new hydrophobicity scale using molecular dynamics, packaging its implementation as a tool named *WaterFire*. In the process, we note the benefits and pitfalls of generalized combinatorial algorithms to handle more complex scales. We specifically focus on the NP-completeness of reducing a contact potential matrix to the HP model. To make AlphaFold more physical, we construct a model named the *Subspace Relaxation Operator (SRO)* that uses structural molecular dynamics data to modify the learned embedding space used for structure prediction. Through these modifications, we observe that molecular dynamics has a place in both the rigorous mathematical world of the HP model as well as the magical realm of deep learning for protein structure prediction.

We proceed in the following manner. For the combinatorics section,

1. We begin by introducing the notion of context-independent, context-dependent, and protein structure-informed hydrophobicity scales mathematically.

2. We establish a new method called *WaterFire* for characterizing protein structure-informed hydrophobicity using molecular dynamics.

3. We then generalize existing combinatorial models of protein folding to handle variable hydrophobicity scales by reduction, noting certain NP-completeness challenges.

For the deep learning section,

1. We examine the AlphaFold architecture and its physical pitfalls with respect to random perturbations and homology.
2. We present a model – the *Subspace Relaxation Operator (SRO)* – which leverages molecular dynamics to improve the physicality of AlphaFold in its intermediate stages.

First, however, we build up to a brief introduction of molecular dynamics – enough to understand the contributions of this work.

## 1.2 Energy Functions

Energy functions lie at the core of the protein structure prediction problem. After all, the native conformation of a protein can be thought of as some local, maybe even global, minimum of its energy function. We begin by defining it symbolically, so that we can refer to it throughout this work.

**Definition 1**

An energy function is some function $E(\cdot) : \mathbb{R}^{3n} \to \mathbb{R}$. For example, suppose $q$ is some $n \times 3$ real-valued matrix, where each row represents the position of one atom in a molecule. Then, $E(q) = k$ would imply that the configuration $q$ has energy $k$.

We will often refer to generic sets of atomic coordinates by the variable $q$. For example, an $n$ atom protein might have atomic coordinates $q \in \mathbb{R}^{3n}$. In the protein structure prediction problem, we want to find $q$ that most closely matches observed protein structures.

Also, note that when we vaguely refer to "energy," we generally mean the potential energy of a molecule or protein, particularly when talking about physics-based energy functions. Since we also have some quite abstract, simple models, we will avoid making any such specifications in general. This function is also parameterized by the type of atom at each coordinate; however, we also generally abstract this component notationally.

Significant quantities of research in chemical physics and other disciplines seek to estimate the true energy function of any molecule, which we denote $E_t(\cdot)$. Were we to have this elusive energy function and be able to evaluate its gradient in a small amount of time, we could simulate biomolecular systems, relax them, and evaluate them much more effectively than we can now. However, the energy function is complex enough that its approximations have their own time-accuracy tradeoff under current paradigms. Quantum Monte Carlo methods take high order polynomial time but are quite accurate, while intermolecular force fields and the HP model take very little time but lose large quantities of information [4]. We will discuss these more time efficient notions in depth throughout this document.

With this notation in mind, we can briefly return to the idea that protein structure prediction is highly related to $E_t$. Often, we think of a native protein structure as one of the most stable configurations of the atoms in its amino acid sequence. Since the notion of stable is one to one with lowering energy at a high level, this strain of thought implies that native protein structures are global, or almost global, minima on $E_t$. However, since we do not know $E_t$ and can only

approximate it, we cannot easily find such candidate minima. Even worse, $E_t$ is nonconvex and exists in high dimensional space, such that finding many local minima does not say anything about other local minima in distant subspaces of $\mathbb{R}^{3n}$. So, we look to models like AlphaFold and more coarse approximations of $E_t$ like the HP model to predict native structures.

### 1.3 Molecular Dynamics

Energy functions have applications far beyond the realm of protein folding. Molecular dynamics uses approximate energy functions and their derivatives to approximate the actual behavior of molecules in a system. In the simplest sense, molecular dynamics is the integration of Newton's second law:

$$F = ma$$

where $F$ (force) and $a$ (acceleration) are functions of the atomic positions $q$ of the system, which $m$ (mass) remains constant. One key simplified observation from physics is that:

$$F = -\nabla E_t(q)$$

This equation makes some intuitive sense: atoms tend to move in the direction that minimizes energy. Existing in some unstable state is bound to impose strong forces on the atoms in a direction of lower energy.

We also know, by the definition of acceleration, that $a = \frac{d^2q}{dt^2}$, such that Newton's second law is a second order differential equation. Since this differential equation describes the relationship between atomic positions and time, we can integrate the equation to determine how atomic positions evolve from some initial state. Usually, we integrate by treating the second order differential equation as a system of two first order differential equations:

$$\begin{aligned}\frac{dq}{dt} &= v \\ \frac{dv}{dt} &= -\frac{1}{m}\nabla E_t(q)\end{aligned}$$

which we can integrate in discrete timesteps using, for example, the Verlet algorithm, or any other integration strategy for first order differential equations. In this work, we always use a timestep of 2fs, or $2 \cdot 10^{-15}$s – when we refer to an "MD step", this is the discrete quantity we use. We also use OpenMM to apply this integration computationally [5]. Notice that we have used an intermediate variable $v$; practically, $\frac{dq}{dt}$ is the velocity of every particle.

To integrate these equations for a particular system, we need some initial conditions on $q$ and $v$. In particular, we need a vector in what is sometimes called the "phase space" of our system, $\mathbb{R}^{6n}$. This vector fully describes the initial positions and velocities of every atom. As is increasingly true for short simulations, this initialization determines the subset of the phase space we explore. The dependence of molecular dynamics trajectories on our initialization of $q$ is what prevents it from directly predicting native protein structures. This dependence decreases the longer the simulation runs, but in the current environment, we do not yet have close to enough computational power to simulate long enough trajectories for this decrease to be meaningful on large biomolecules.

When we perform molecular dynamics throughout our forays into combinatorics and deep learning, we generally take in some prior on $q$, usually as the result of a structure prediction. We then randomly assign velocities according to the Maxwell-Boltzmann distribution. As we will see in both

chapters, molecular dynamics can help improve predictions from existing methods, even though it requires this prior.

We also need some approximation of $E_t$. For our purposes, we generally use force fields.

**Definition 2**

For some differentiable approximation of an energy function $E(t)$, a force field $F(t)$ is a callable function that returns $-\nabla E(t)$. In most protein-related applications, to maintain computational efficiency, $E(t)$ is a sum of simple differentiable terms (e.g. harmonic oscillators, Lennard-Jones potentials, etc.).

Defining the force function directly allows integration strategies to call it like an oracle whenever necessary. Throughout this work, we use the Amber 99SB force field [6] – one of the most used approximations in biomolecular simulations.

We can now begin looking at combinatorial algorithms for structure prediction, eventually looking to integrate our high-level knowledge of molecular dynamics.

# 2 Combinatorics

In this chapter, we examine the notion of a hydrophobicity scale to better understand existing folding techniques in the HP model, formally defined below in Definition 6. We then discuss the implications of using a real-valued hydrophobicity scale with HP model approaches to improve their practical value. Our goal will be to characterize some mathematical and practical results of trying to extend HP model folding into the space of real structures using molecular dynamics.

Combinatorial algorithms, like those that operate on the HP model, seek to use the fact that hydrophobicity is the primary determinant of protein kinetics (and to some extent, their thermodynamics) [2]. These algorithms give hydrophobicity scales their value. In numerically determining the hydrophobicity of a given amino acid, we can start to come up with simplified energy functions that allow us to make more and more reasonable guesses at native protein structures. The promise of such algorithms drove much of the research presented in our first foray into hydrophobicity scales.

## 2.1 Hydrophobicity Scales

### 2.1.1 Definitions

To abstract away other components of folding, computer scientists have often defined some context-independent hydrophobicity function $f_H$ that takes in a given amino acid and outputs a specific value associated with its hydrophobicity. For example, in the HP model, such a function is used to provide a threshold for whether an amino acid is hydrophobic or polar, a distinction which directly produces a fold according to several foundational lattice algorithms. This function is invariant to context; i.e., the domain of $f_H$ is strictly over the set of amino acids, which we define as $\Sigma_A$ throughout this work. We can formally define this notion.

**Definition 3**

Let a **context-independent hydrophobicity function** be defined as $f_H : \Sigma_A \to \mathbb{R}$ that maps each amino acid to some real-valued quantity associated with its hydrophobicity.

This construction is a significant simplification – reducing the interaction of an amino acid and its surrounding solvent to a number ignores numerous chemical considerations. However, their simplicity drove the field of protein structure prediction prior to the emergence of deep learning. A significant amount of research was produced on such functions [7] as a result of the foundational observation that protein folding kinetics and thermodynamics are largely driven, at a high level, by hydrophobic interactions – an observation at the center of this chapter. Below, we briefly explore experimental and computational efforts to define such a function. Each of these functions relies on fundamentally different notions of hydrophobicity, although they share a similar general character.

From here, we can define a context-dependent hydrophobicity function – an abstract idea that helps us bridge old and new notions of hydrophobicity.

**Definition 4**

Let a **context-dependent hydrophobicity function** be defined as $f_H : \Sigma_A^* \to \mathbb{R}^*$ that maps each individual amino acid **in a given protein sequence** of arbitrary length to some real-valued quantity associated with its hydrophobicity. For some sequence $S$, $f_H(S)_i$ is the hydrophobicity of the $i$th amino acid.

This is a highly abstract notion that, without major implicit learning, requires full protein structure prediction embedded as a subprocess – not a particularly realistic idea. Instead, we define a notion that has the structure as a prior, which we explore in more detail:

**Definition 5**

Let a **protein structure-informed hydrophobicity function** be defined as $f_H : \Sigma_A^* \times \mathbb{R}^* \to \mathbb{R}^*$ that maps each individual amino acid **in a given protein structure** of an arbitrary number of atoms and arbitrary sequence length to some real-valued quantity associated with its hydrophobicity. For a given sequence $S$ and structure coordinates $q$, $f_H(S, q)_i$ is the hydrophobicity of the $i$th amino acid.

This idea of a scale evaluates each individual amino acid in the context of some conformation. Such a concept allows us to determine which amino acids contribute to high energy in a conformation via hydrophobic character, accounting for more than just behavior intrinsic to residue type. This scale is practically achievable, unlike Definition 4, but contains more contextual information than Definition 3. For example, if some hydrophobic amino acid serves to stabilize the water dynamics around it via interaction with nearby amino acids, this notion would not impose as harsh an energy penalty for its solvent exposure. This extra information comes at the cost of requiring a structural prior, however, which we will later address with a self-consistency algorithm (Algorithm 5).

To motivate our look into hydrophobicity scales, we also define the HP model. As alluded to in our introduction to molecular dynamics and energy functions, the HP model can be thought of as an energy function. In most standard applications of hydrophobicity scales to protein folding, mathematicians have tended to use this model [2], which simplifies matters even a bit beyond a single scale. In this work, we focus specifically on the "side-chain HP model" to align with our implementation, but the mathematical ideas all extend nicely to even earlier models. So, we will generally omit the "side chain" part for simplicity.

Consider a non-unique (a symptom of simplifications) transformation $T$ that takes in a set of atomic positions $q$ and returns the corresponding list of enumerated amino acids $S$ and their backbone and side chain positions, $\{x_i\}_{i\in S}$ and $\{y_i\}_{i\in S}$ respectively, on an arbitrary lattice. Note that when we refer to $s_i \in S$, we refer to the $i$th amino acid in $S$, and when we refer to $i \in S$, we refer to the enumeration of $S$ such that $i \in \{1, ...|S|\}$. In order to be a valid fold, all $x_i$ must form a self-avoiding walk when considered sequentially, and each $y_i$ must a neighbor of $x_i$ on the lattice for all fixed $i$ – these are exactly the geometric constraints enforced by known backbone-backbone and backbone-side chain bonds. Define an adjacency function $\Delta(y_i, y_j)$ such that $\Delta$ is 1 when $y_i$ is adjacent to $y_j$ on the lattice and 0 otherwise; when $\Delta(y_i, y_j) = 1$, we say $s_i, s_j$ form a "contact". Then, we define the HP model as follows, dividing by two to avoid double-counting contacts.

**Definition 6**

The HP model, when considered with a particular lattice and transformation $T$, is fully defined by an energy function on $q$. In particular,

$$E(q) = \frac{1}{2} \sum_{i,j \in S \times S} \Delta(y_i, y_j) C(i, j)$$

where $C$ is $-k$ if $i, j \in H$ for some selected $H \subseteq S$ and 0 otherwise. In other words,

$$C_{ij} = -k \mathbb{1}_{i \in H} \mathbb{1}_{j \in H}$$

We sometimes call the function $C$ a *contact potential matrix*, given that for a sequence $S$, it contains some energy of attraction for each pair of amino acids in $S$. It can also be thought of as a function from $S \times S \to \mathbb{R}$. $C$ can be generalized beyond the specific case in the above definition to account for arbitrarily many concerns with the HP model. For example, the adjacency of residues $s_i$ and $s_j$ might induce an energy penalty proportional to the distance between $i$ and $j$ in $S$. Even more concretely, $C$ might be better suited as a real-valued, rather than effectively binary, matrix, given the complex nature of residue interactions. Additionally, it is a strong assumption to let amino acids outside some $H \subseteq S$ have zero contribution to the overall energy of a conformation.

The theory behind this formulation comes from the observation we have previously noted: hydrophobic interactions are, on average, a key descriptor of protein kinetics and thermodynamics. If we let $H$ be the set of all hydrophobic amino acids, then the definition of the HP model above reads as "the energy of a fold, in the HP model, is proportional to the number of adjacent hydrophobic residues." This more intuitive definition indicates that any optimal fold under the HP model energy function places as many hydrophobic residues as possible adjacent to each other on the lattice. This model, appearing to simplify protein folding to its main "principal component", drove a significant quantity of research, from NP-completeness on the 3D cubic lattice to 86% optimal folds in the

face-centered cubic (FCC) lattice [8, 9].

In the standard HP model, $C$ is determined from a given hydrophobicity scale and some threshold to determine when a residue should be in $H \subseteq S$ or in $H^c$ (typically referred to as $P$). Using any notion of hydrophobicity scale, where each amino acid in a sequence gets a singular number, we can construct $C$. The particular construction of $C$ in Definition 6 encourages any atoms over a given threshold in the scale to be close to one another in the context of minimizing HP model energy.

With these definitions and models in mind, we can begin by examining existing context-independent hydrophobicity functions.

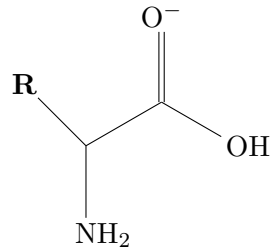
### 2.1.2 Existing Context-Independent Scales

In addition to giving us some direct notion of amino acid properties, context-independent hydrophobicity functions provide the framework for simplified models to give predictions or intriguing priors for more complex prediction methods. However, these benefits come at the cost of oversimplification.

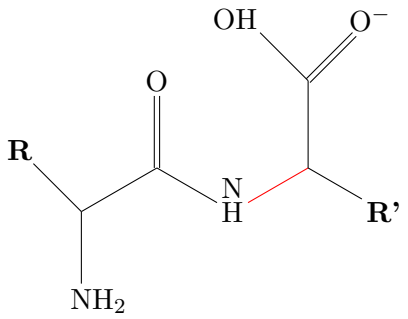
The most widely used scales that we examine will generally fall within two themes. Each scale will either determine some chemical relationship of an isolated variant of each amino acid with water and nonpolar substances (which we call "experimental), or it will observe aggregate properties of the amino acid across selected experimental protein structures ("empirical"). Experimental scales lose any hydrophobic, polar, or other attractive properties that might arise from amino acid interactions. Empirical scales rely on a finite sample of data to uncover some aggregate quantity, which inherently ignores the possibility for variance (as we begin to capture in context-dependent and structure-informed hydrophobicity functions).

In the following sections, we examine the chemical foundations of each scale, with close attention to where they fall short of a more complex reality. We will begin by understanding experimental scales, as they came first chronologically.

**Nozaki-Tanford** The Nozaki-Tanford scale laid the framework for generating experimental hydrophobicity values, although it did so for only nine amino acids [10]. Some of the equations here are repetitive across other scales that leverage this method, so we will often refer back to their pioneering work. Before we begin, recall that an amino acid with side chain $R$ has the following functional form:




Where two amino acids link together at the carboxyl and amino ends, as in the following example:




Nozaki and Tanford generated hydrophobicity values for a given amino acid $X$ with side chain $R$ by using exactly the lone structure of $X$ as in our first drawing above. From here, Nozaki and Tanford dissolved as much $X$ as possible in water, 100% ethanol, and dioxane. The two scientists had in mind that 100% ethanol is a good proxy for the environment at the core of a protein – a major assumption. Nozaki and Tanford used the resulting solutions to determine how much $X$ dissolved in each different substance. After this experimental procedure, they began to calculate free energy differences of moving $X$ between solutions, with the idea that the free energy change between water and ethanol would say something about $X$'s preference for water interaction.

By observing that $X$ has the same chemical potential at saturation in all solvents, Nozaki and Tanford computed the desired free energy change by setting

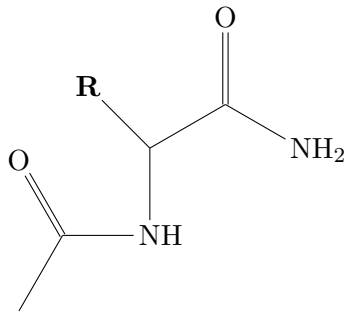
$$\mu_{X,s} + RT \ln N_{X,s} + RT \ln \gamma_{X,s} = \mu_{X,s'} + RT \ln N_{X,s'} + RT \ln \gamma_{X,s'}$$

Where $\mu, N, \gamma$ are the chemical potentials, mole fractions at saturation, and activities (representing self-interaction in the solvent), respectively. By noticing that the free energy change $\Delta G_{s,s'}$ of one mole of $X$ transferring from a solvent $s$ to a solvent $s'$ is $\mu_{X,s'} - \mu_{X,s}$:

$$\begin{aligned}
\mu_{X,s'} - \mu_{X,s} &= (\frac{\partial G}{\partial n_{X,s'}})_{T,P}(1) + (\frac{\partial G}{\partial n_{X,s}})_{T,P}(-1) \\
&= (\frac{\partial G}{\partial n_{X,s'}})_{T,P} dn_{X,s'} + (\frac{\partial G}{\partial n_{X,s}})_{T,P} dn_{X,s} \text{ (for one mole transfer)} \\
&= \Delta G_{s,s'}
\end{aligned}$$

We can get measurements for free energy of transfer by $RT \ln \frac{N_{X,s}}{N_{X,s'}} + RT \ln \frac{\gamma_{X,s'}}{\gamma_{X,s'}} = RT \ln \frac{N_{X,s}\gamma_{X,s'}}{N_{X,s'}\gamma_{X,s'}}$. Given the unavailability of $\gamma_{X,s}$ for many solvents, Nozaki and Tanford chose to neglect self-interaction, correcting in an absolute sense by subtracting the computed free energy change of transfer of glycine from all results. Nozaki and Tanford, pioneers along the dimension of hydrophobicity scales, presented these free energy changes as their first scale.

**Fauchére-Pliśka** Fauchére and Pliśka followed Nozaki and Tanford by observing a problem with the free carboxylate (conjugate carboxylic acid) on the carbon terminus of their individual amino acids [11]. To combat this issue, Fauchére and Pliśka instead investigated derivatives of the standard form of each amino acid $X$ – N-acetyl-amino acid amides (as below).




In particular, Fauchére and Pliśka postulated that the Nozaki-Tanford procedure would fail to be consistent at varying pH values, considering the relatively strong base on the carboxyl end of a standard amino acid. Additionally, the basic behavior at this end would make amino acids significantly less soluble in organic solvents like ethanol, making consistent results even harder to reproduce.

Fauchére and Pliśka chose to sacrifice the exactness of the Nozaki-Tanford procedure in favor of remedying these issues by decreasing the resonance potential on the carboxyl end with an amide and decreasing the basic behavior of the amino end with an attached acetyl group. In doing so, Fauchére and Pliśka got more consistent results than Nozaki-Tanford. They additionally adopted a secondary strategy for measuring the free energy change by placing these amides in a mixture of water and an organic solvent. By separating the mixtures, Fauchére and Pliśka were able to determine the partition coefficient of each amino acid derivative, allowing free energy change to be calculated as proportional to its logarithm (a similar notion to the logarithm of the saturated mole fractions in either species, as in Nozaki-Tanford). Mathematically, the partition coefficient is another way of measuring a quantity parallel to $\frac{N_{X,s}}{N_{X,s'}}$, which produces free energy values by an identical procedure to Nozaki and Tanford.

**Kyte-Doolittle** One of the most used scales to date was produced by Kyte and Doolittle [12], regarding work done by Wolfenden [13] and Chothia [14]. Kyte and Doolittle followed a similar approach to Nozaki and Tanford, but instead measured hydrophobicity values by determining the water-vapor partition of each amino acid. In particular, by observing that hydrogen bonding and other interactions that define water as a solvent no longer occur in the gas phase, Kyte and Doolittle were able to use the partition coefficient of each amino acid in water/vapor as a proxy for hydrophobicity. In particular, they computed the free energy change as

$$-RT \ln \frac{N_w V_g}{N_g \phi}$$

Where $\phi$ is the apparent molar volume of some amino acid $X$ in the liquid phase and $V_g$ is the same in the gas phase, such that $\ln \frac{N_w V_g}{N_g \phi}$ is like a concentration-based partition coefficient. In particular, this is similar to approximating the activity ratio as a ratio of concentrations, yielding a rough form of transfer free energy. This is again a similar notion to the expression we derived for $\Delta G_{s,s'}$ in Nozaki-Tanford; i.e. $\frac{N_{X,s}\gamma_{X,s'}}{N_{X,s'}\gamma_{X,s'}}$

Kyte and Doolittle, in finalizing their scale, additionally incorporated values from an empirical scale developed by Chothia. Chothia, examining all existing protein structures established by

1976, computed context-independent hydrophobicity in a much simpler way than the experimental methods we have looked at so far. Specifically, he determined the fraction of each amino acid type that occurred 95% buried across all structures, by a simple count. He computed an alternate scale that executed the same procedure, but instead counted the number of 100% burial occurrences. As Kyte and Doolittle observe, Chothia's results correlate highly with the experimental methods we have seen thus far; for example, the correlation between Chothia's function and Fauchére and Pliśka's function is 0.78 [7]. Throughout this work, we primarily use Kyte and Doolittle's final values when using a context-independent scale as a prior for some more complex notion; we include the values in Table 1.

| **Amino Acid** | **Hydrophobicity** | **Amino Acid** | **Hydrophobicity** |
|---|---|---|---|
| A (Alanine) | 1.8 | N (Asparagine) | -3.5 |
| C (Cysteine) | 2.5 | P (Proline) | -1.6 |
| D (Aspartic Acid) | -3.5 | Q (Glutamine) | -3.5 |
| E (Glutamic Acid) | -3.5 | R (Arginine) | -4.5 |
| F (Phenylalanine) | 2.8 | S (Serine) | -0.8 |
| G (Glycine) | -0.4 | T (Threonine) | -0.7 |
| H (Histidine) | -3.2 | V (Valine) | 4.2 |
| I (Isoleucine) | 4.5 | W (Tryptophan) | -0.9 |
| K (Lysine) | -3.9 | Y (Tyrosine) | -1.3 |
| L (Leucine) | 3.8 | M (Methionine) | 1.9 |

Table 1: Kyte-Doolittle hydrophobicity scale for amino acids. Positive values indicate hydrophobicity, negative values indicate hydrophilicity.

**Wertz and Scheraga** As an example of a more complex empirical scale, we examine the work of Wertz and Scheraga [15]. Wertz and Scheraga, in particular, examined a database of protein structures by counting, for each residue type, the frequency at which it appeared on the inside of a protein. As a result, they determined a value in $[0, 1]$ for each residue type pertaining to its average affinity for burial into the core.

However, there is no definitive method for determining whether a particular instance of a residue is in the interior or exterior of a structure. At a high level, Wertz and Scheraga proposed dividing up $\mathbb{R}^3$ into $1\text{Å}^3$ voxels by a set of lines in the $x = [1, 0, 0]$, $y = [0, 1, 0]$, and $z = [0, 0, 1]$ directions. In particular, these lines span the smallest rectangular prism that contains the Cartesian coordinates for the protein and 1.7Å padding in each direction, where each edge of the prism is in the direction of a standard unit vector. From here, they computed the intersection of each line with the van der Waals radius of each atom in the protein. Each atom was scored on its exposure by the number of times it occurred as the final intersection of a line in the constructed set, such that each residue was scored on the aggregation (average) of its atomic scores. Wertz and Scheraga then binarized these scores and averaged over the entire database per residue type.

As a small caveat, Wertz and Scheraga also tested for pockets of exposure by moving a 2Å spacer down the length of each line; if the spacer intersected no atoms, that particular interval in real space was determined to be empty. As a result, the nearest atoms to each end of the spacer were considered to be exposed, in some capacity. Empirical scales like this one, as seen with Chothia and Fauchére-Pliśka, are well correlated with experimental scales. As another example, the correlation

between the Wertz-Scheraga scale and the Kyte-Doolittle scale is 0.71 [7].

### 2.1.3 Promises and Shortcomings of Context-Independent Scales

Context-independent hydrophobicity scales are invaluable as a prior for determining which amino acids in a sequence will generally localize to the core. However, both empirical and experimental functions alike sacrifice information content for simplicity. Since the action and behavior of an amino acid is reduced, under these functions, to a single number, we cannot hope to capture the nuanced dynamics of each isolated amino acid with water. Even if we are ready to make such a simplification, both experimental and empirical context-independent scales have significant shortcomings. In all of the above scales, we face the critical issue of failing to capture behavioral variance, as alluded to at the beginning of this section.

Experimental scales, such as Nozaki-Tanford and Fauchére-Pliśka, measure the properties of each amino acid in isolation. Such an approach fails to capture the effect of a given amino acid on the surrounding solvent when placed in some particular context. For example, if some *on-average* hydrophobic amino acid is oriented in such a way that it supports two *on-average* hydrophilic amino acids on the surface of a protein, it might not have a particularly hydrophobic effect. However, under experimental scales, we would still treat that hydrophobic amino acid as its average, such that when positioned on the surface, we would assume it to be relatively "unstable".

Empirical scales, such as Wertz-Scheraga, take the average properties of each amino acid across many proteins. However, such empirical averages inherently have variance. This type of variance corresponds directly to the chemical variance we just described in the experimental context.

This ignorance of variance is a critical shortcoming to the use of these scales for downstream folding that seeks to maximize hydrophobic contacts. Even if we were willing to explain the affinity (i.e., energetic stability of their adjacency) of two amino acids entirely by hydrophobicity, using a context-independent scale would always enforce affinity between amino acids that are on-average hydrophobic – an unrealistically uniform assumption. By augmenting context-independent scales with information from molecular dynamics, we can begin to form better, sequence-specific priors for which amino acids will localize to the core.

## 2.2 Implementation of a Water Density Fluctuation Scale

### 2.2.1 Existing Approaches

Little research has been done into modifying hydrophobicity scales as in 2.1.2 using molecular dynamics, partially because hydrophobicity is better thought of as a field than applied to each amino acid. However, some recent research has explored the use of simulation to quantify hydrophobicity more generally. As a result, we can now start to return to molecular dynamics.

Initial attempts looked to describe the hydrophobicity of a surface by its surrounding water density [16], the hypothesis being that hydrophobic surfaces would "repel" surrounding water molecules. However, this notion was eventually determined to only weakly correlate with more robust experimental measures of hydrophobicity [17]. More recent attempts have instead used, for example, water density fluctuations, water orientation fluctuations, and hydrogen bonding statistics to measure hydrophobicity [18]. These newer ideas all rest on the loose hypothesis that the stability of the hydrogen bonding lattice around a given solute determines its surface hydrophobicity.

We seek to marry these ideas with existing context-independent hydrophobicity scales to produce structure-informed hydrophobicity scales. In particular, by constructing a framework to eval-

**Algorithm 1** WaterFire

**Require:** $q$ (initial structure), $S$ (amino acid sequence), $T$ (number of MD steps), $\beta$ (coefficient on correction), $p$ (desired pH)

Protonate $q$ using PDB2PQR with PROPKA to pH $p$
Initialize a molecular dynamics trajectory with $q$ and integrate for $T_i = 100$ steps with an external force encouraging it to remain close to its original content.
Explicitly solvate $q$ with water at pH $p$ and run a new molecular dynamics trajectory for $T$ steps
Initialize a density matrix $D$ of shape $|S| \times T$
At every step $t \in 1, ...T$, compute the number of water molecules within radius $r$ of any atom in each residue $i$; store this value in $D_{it}$
$F \leftarrow$ the standard deviation of each row in $D$
**for** $1 \leq i \leq |S|$ **do**
    **if** $\exists t, D_{it} = 0$ **then**
        $F_i \leftarrow 0$
**for** $1 \leq i \leq |S|$ **do**
    **if** $F_i \neq 0$ **then**
        $H_i \leftarrow \beta \ln \frac{F_i(\sum_{j=1}^{|S|} \mathbb{1}_{F_j \neq 0})}{\sum_{j=1}^{|S|} F_j}$
    **else**
        $H_i \leftarrow 0$
    Add the Kyte-Doolittle hydrophobicity number for residue $i$ to $H_i$
**return** $H$

uate water density fluctuations, we have implemented a new tool to modify context-independent hydrophobicity to fit structural context.

### 2.2.2 Implementation of WaterFire

In combining water density fluctuations and context-independent hydrophobicity, we developed a tool named WaterFire to generate structure-informed hydrophobicity scales. As can be seen in Algorithm 1, WaterFire is a function from atomic coordinates $q$ and an amino acid sequence $S$ to a set of $|S|$ real values – exactly the type of structure-informed scale we defined in Definition 5. We can look at each step in Algorithm 1 individually to get a sense for the theory behind this scale.

WaterFire begins by adding protons to the input protein structure. To mimic the conditions under which we want to evaluate hydrophobicity, this step is necessary – it ensures that each amino acid is in a realistic state. For example, if we arbitrarily allowed some specific amino acid to lack all donating protons, it might have unrealistically basic behavior, leading to unrealistic water interactions. WaterFire defaults to pH 7, which is standard in humans; however, its generality to arbitrary pH values is a nice side effect. To implement this component, we use external packages PDB2PQR and PROPKA [19, 20]. PDB2PQR measures the charge and radius of each atom in $q$ using the Amber force field we use for molecular dynamics, while PROPKA uses that information to estimate the pKa of each amino acid and protonate accordingly (relative to the input pH). These packages handle the geometric constraints of adding hydrogens themselves.

From here, WaterFire runs molecular dynamics over our newly augmented $q$ (now with new

positions for each added hydrogen) with the same Amber force field from our introduction to molecular dynamics. It runs for only a small number of steps ($T_i$, default 100) to reach a relatively stable configuration. To the force field output, we additionally add an energy term $k((x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2)$ for each atom in the protein, which applies a force rerouting each atom to its original position. These restraints ensure that the relaxation does not alter the initial structure too drastically. The purpose of this relaxation is to work out any structural kinks introduced by the protonation pipeline.

Next, WaterFire adds water molecules to $q$ until there is a 0.5nm buffer in all directions, using GROMACS [21]. Adding this set of water molecules will allow us to track water density in simulation.

From here, we run a longer molecular dynamics trajectory of $T$ steps (up to the user's preference and computational power) and compute the water density as described in Algorithm 1.

The final calculation, after taking the standard deviation, is the crucial observation of WaterFire. In particular, we assume that for large enough proteins, the mean water density fluctuation is a reasonable reference for a neutral amino acid, in terms of hydrophobicity. As such, we compute something analogous to the free energy of substitution of each amino acid, where we instead treat any energy terms with water density fluctuations. This calculation allows us to adjust the relative hydrophobicity of each amino acid based on how it behaved in the molecular dynamics trajectory. Were we to calculate the adjustment solely under these assumptions, it would have the form

$$\ln \frac{F_i}{(\frac{\sum_{j=1}^{|S|} F_j}{n})} = \ln \frac{F_i \cdot n}{\sum_{j=1}^{|S|} F_j}$$

One complication is that any residues in the core of the prior, $q$, will not be exposed to water and thus have no water density fluctuation. We exclude any such residues $s_i$ – those with no water density at some step $t$, or those with no overall fluctuation – from the mean $\frac{1}{n}\sum_{j=1}^{|S|} F_j$ by setting $F_i = 0$ if it is not already. Functionally, we leave them with their initial hydrophobicity value according to the prior scale and calculate the denominator of the reference as $\sum_{j=1}^{|S|} \mathbb{1}_{F_j \neq 0}$. These changes give our final adjustment calculation:

$$\ln \frac{F_i}{(\frac{\sum_{j=1}^{|S|} F_j}{\sum_{j=1}^{|S|} \mathbb{1}_{F_j \neq 0}})} = \ln \frac{F_i(\sum_{j=1}^{|S|} \mathbb{1}_{F_j \neq 0})}{\sum_{j=1}^{|S|} F_j}$$

Even though we therefore do not make adjustments to core hydrophobics, the scores provided by WaterFire still give us some way to decide what other amino acids might be sensible to try in the core in some future HP fold because of the instability they induce in surrounding water interactions.

By adding this augmentation to any existing hydrophobicity scale, WaterFire begins to explain the variance that context-independent scales miss. As we will formalize below, we can initialize hydrophobicity values using a standard scale, run an HP model approximation, apply WaterFire, and iterate. This iteration allows us to incorporate contextual hydrophobicity into HP folds. In other words, WaterFire represents an opportunity to make the HP model and other hydrophobicity-reliant folds slightly more physical.

An example of the fluctuation and scale outputs can be seen in Figure 1, where we visualized all of our structures using a visualization package we wrote with an NGL backend [23]. Notice

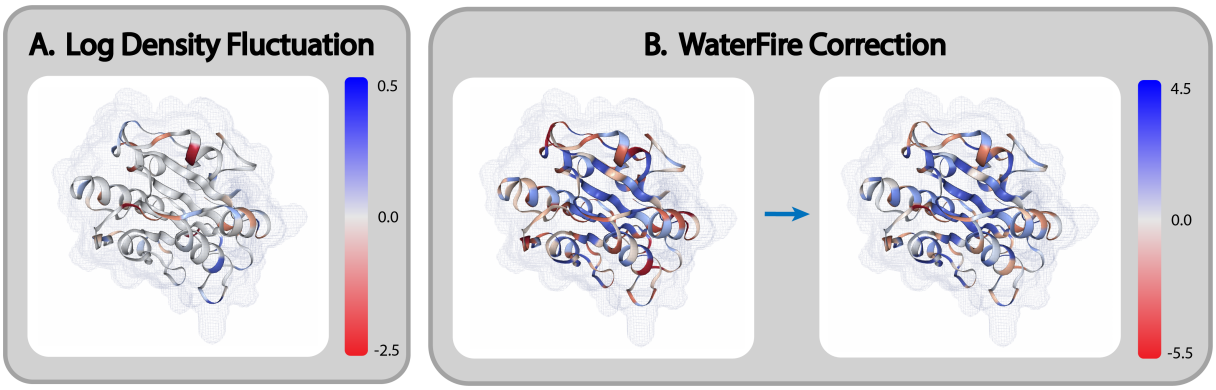



**Figure 1:** Example of WaterFire output after 10000 molecular dynamics iterations on PDB identifier 1ATZ, the von Willebrand Factor [22]. **A** shows the log of the demeaned water density fluctuation superimposed onto the protein's structure. Notice that the core section of the protein has a value of 0 as a result of our filtering of core residues described in Algorithm 1. Water density at a residue is determined by the number of water atoms within $r = 3$ Å of any atom in that residue. **B** shows the change from standard Kyte-Doolittle hydrophobicity (left) when we add the WaterFire correction with $\beta = 3$ (right). Kyte-Doolittle hydrophobicity numbers are in the interval $[-4.5, 4.5]$, which induces the color scale we use.

that water density fluctuation is highly variable across the surface. This implies that the molecular dynamics trajectories do accrue information about the relative stability of each amino acid on the surface, such that we can induce a meaningful change from the original Kyte-Doolittle scale. Note also that some external loops that are labelled strongly hydrophilic in the Kyte-Doolittle scaling have relatively high water density fluctuation, such that the conformation might benefit from orienting them more towards the core. An example of such a case is the bottom-most loop in the images of panel B.

## 2.3 Reduction to Lattice Models

Our implementation of a water density fluctuation hydrophobicity scale leaves us with a set of real values for each amino acid in a sequence. Now, recalling the definition of the HP model, we can apply this hydrophobicity scale to get a fold.

First, however, we examine an NP-completeness proof. We originally set out to construct the contact potential matrix $C$ directly from molecular dynamics, using the pioneering work of Miyazawa and Jernigan [24] as a prior; however, we quickly ran into issues determining how to reduce our work to the HP model or otherwise determine a simple energy function. To understand this proof, we first examine existing algorithms in the HP model.

### 2.3.1 Existing Approximations in the HP model

Having already characterized the HP model as the energy function

$$E_{HP}(q) = -\frac{1}{2}k \sum_{i,j \in S \times S} \Delta(y_i, y_j) \mathbb{1}_{i \in H} \mathbb{1}_{j \in H},$$

we can think of what it might look like to minimize it. An algorithm that globally minimizes this coarse $E_{HP}(q)$ would provide a consistent method to pack all hydrophobics in an amino acid

sequence into the core – a powerful notion for producing priors. Istrail and Lam characterize approximations of such minima on a variety of different lattices, including standard 2D and 3D cubics as well as more complex side-chain models [3].

Note that, as it is, the minimum of $E_{HP}(q)$ is not very well defined, given the transformation $T$ described in Definition 6 is many-to-one. As a result, the typical problem in HP model folding is to instead minimize $E_{HP}(S, H, x, y)$ with respect to $x$ and $y$, where $S$ is a list of amino acids, $H \subseteq S$ is the set of hydrophobic residues, $x = \{x_i\}_{i \in S}$ are the lattice coordinates of each side chain, and $y = \{y_i\}_{i \in S}$ are the coordinates of each backbone. This minimization is subject to the constraints noted earlier; i.e. $x$ must be a self-avoiding walk and $y_i$ must be adjacent to $x_i$ for all $i$. From here, we can choose any $q$ in the space $T^{-1}(S, H, x, y)$ as a candidate fold. The typical algorithm seeking to approximately minimize $E_{HP}$ thus takes in $S, H$ and finds $\mathsf{argmin}_{x,y} E_{HP}(S, H, x, y)$.

Since $y$ are lattice coordinates, each $y_i$ has a finite number of neighbors. On the 2D square lattice, for example, each coordinate has at most four neighbors. As such, $\Delta(y_i, y_j)$ for fixed $y_i$ would be positive for at most three $y_j$ – one of its neighbors must be $x_i$. This puts a lower bound on the minimum of $E_{HP}(S, H, x, y)$ for fixed $S$ and $H$ – namely, $-k(\frac{3}{2}|H|)$, as each amino acid in $H$ can have at most three side chain contacts with other amino acids in $H$. We generally work in the face-centered cubic (FCC) lattice, where this lower bound is $-k(\frac{11}{2}|H|)$ [3]. Given geometric constraints of the backbone, Hart and Istrail show that the lower bound can be increased to $-k(\frac{9}{2}|H|)$ on FCC. However, in all reasonable lattices, the number of adjacent coordinates on a given lattice is often finite. This implies that, as we observe in the square and FCC lattices, the optimal energy is at worst linear in $|H|$. This result is corroborated by more extensive results compiled by Istrail and Lam [3].

Approximation algorithms for the HP model on the FCC lattice take in $S, H$ and return $x, y$ in some way that approximately minimizes $E_{HP}(S, H, x, y)$. We can define this notion more generally as follows.

**Definition 7**

An approximation algorithm in some variant of the HP model with optimal energy $-\ell^*(|H|)$ (optimal contacts $\ell^*(|H|)$) is $\ell$-optimal if it guarantees energy $-\mathsf{max}(\ell(|H|), 0)$ (contacts $\mathsf{max}(\ell(|H|), 0)$) for some function $\ell : \mathbb{N} \to \mathbb{R}$. A similar algorithm is "exact" $\alpha$-optimal if it guarantees energy $-\alpha\ell^*(|H|)$ for some scalar $\alpha \in \mathbb{R}$ for all $|H|$. It is instead "asymptotically" $\alpha$-optimal if it guarantees energy $-\alpha\ell^*(|H|)$ for some scalar $\alpha \in \mathbb{R}$ as $|H| \to \infty$.

So, in the FCC side chain model, an algorithm is $\alpha$-optimal if it guarantees energy $-\alpha k(\frac{9}{2}|H|)$, where the nature of the guarantee depends on whether the approximation is exact or asymptotic. Notice that an exact $\alpha$-optimal algorithm is $\ell$ optimal for $\ell(x) = \alpha k(\frac{9}{2}|H|)$; in other words, exact $\alpha$-optimality is achieved when $\ell(x) = \alpha\ell^*(x)$.

Typically, algorithms on the HP model are asymptotically $\alpha$-optimal when they have to sacrifice a fixed number of hydrophobic contacts to account for geometric constraints: for example, in the Hart-Istrail 86% optimal algorithm on the FCC lattice [9]. In these cases, we can rely on the associated guarantee function $\ell$, which provides more information than the limiting value of $\alpha$. In fact, $\alpha$-optimality is generally only a simplified way of looking at $\ell$-optimality; however, because of its ease of interpretation, it is often highlighted more prominently [9].

In order to test WaterFire on real proteins as in Figure 2, we implemented the Hart-Istrail algorithm as an example of one possible $\alpha$-optimal approximation of the HP model energy function [9]. At a high level, the Hart-Istrail algorithm stacks hydrophobic side chains in a $2 \times 4 \times n$ prism on an FCC lattice. It then weaves connecting hydrophilic and backbone sections as loops protruding out from each column in the prism. For example, if a subsequence of a protein has two hydrophobics separated by some number of only hydrophilics, the algorithm stacks the two hydrophobic residues in the $z$ direction in the prism, extending the loop of hydrophilics out into the $xy$ plane. This algorithm occasionally needs to sacrifice a hydrophobic to connect each of the eight columns of the prism; however, the number of such cases is bounded. In this particular case, Hart and Istrail show that the number of guaranteed contacts is $\frac{31}{8}|H| - 69$. Their algorithm is thus asymptotically 0.86-optimal compared against $\ell^*(x) = \frac{9}{2}x$; i.e. $\ell$-optimal for $\ell(x) = \frac{31}{8}x - 69$.

One underlying assumption of this algorithm is the partition of the protein sequence $S$ into the set of hydrophobic residues $H$ and the set of hydrophilic residues $P$. This is done by thresholding a hydrophobicity scale. The set $H$ is then used to construct $C$, the contact potential matrix, as $k\mathbb{1}_{i\in H}\mathbb{1}_{j\in H}$ for all amino acid pairs $i, j$.

### 2.3.2 NP-Completeness of Selecting $H$ For HP Approximations

However, we might want to generalize the HP model to be flexible to $C$ to handle some of our aforementioned concerns with the standard HP model energy function – namely, that hydophilics should contribute something to energy, and so should geometric distortions. Suppose we instead computed $C$ in the absence of a hydrophobicity scale; in other words, we compute contact potentials for each pair of amino acids first, rather than computing them from some auxiliary, information-poor vector. In principle, $C$ might be any arbitrary symmetric real-valued matrix. We initially considered constructing WaterFire to this end.

However, we can show that the problem of choosing $k$ specific amino acids we want to be maximally adjacent during folding is NP-complete when $C$ is arbitrary. By NP-complete, we mean that our problem is in the *nondeterministic polynomial* time complexity class, and any problem in this class can be reduced in polynomial time to our problem. In other words, if we want to reduce the problem of maximizing the energy of a fold according to an arbitrary $C$ to the HP model by selecting a set of hydrophobic residues $H$, we cannot do so in polynomial time (provided P $\neq$ NP). This result holds for any approximation algorithm with $\ell$-optimality in the HP model, provided $\ell(\cdot)$ has an asymptotic slope greater than 1. We add this result to the extensive list of NP-completeness results compiled by Istrail and Lam, such as the NP-completeness of minimizing energy in the 3D cubic HP model, although this particular result is the first of its kind [3].

**Theorem 1**

Reducing a real-valued contact potential matrix to any $\ell$-optimal approximation of HP model folding is NP-complete. In other words, the decision problem of determining whether there exists some subset of amino acids $H$ with guaranteed energy less than or equal to $E$ under some $\ell$-optimal algorithm given a real-valued contact potential matrix is NP-complete.

**Setup** In order to go about this proof, we prove the following lemma.

**Lemma 1**

The generalized symmetric $\ell$-densest submatrix problem is NP-complete for any increasing function $\ell(\cdot) : \mathbb{N} \to \mathbb{R}$ such that for some $k \in \mathbb{N}$, for all $k' > k$, $\ell(k') \geq \ell(k' - 1) + 1 > 0$. In other words, $\mathsf{inf}\{k | (\ell(k) > 0) \wedge (\forall k' > k, \ell(k') \geq 1 + \ell(k' - 1))\}$ exists.

We denote the set of such functions $\ell$ where this infimum exists by $\mathcal{L}$; intuitively, they are the set of functions from $\mathbb{N} \to \mathbb{R}$ that have an asymptotic slope of at least 1. Once we have a proof of Lemma 1, we then equate this problem to HP model folding. We begin by defining the problem itself.

**Definition 8**

Fix a function $\ell(\cdot) \in \mathcal{L}$. Suppose we have a symmetric matrix $M$ and $t \in \mathbb{R}$. The generalized symmetric $\ell$-densest submatrix problem is the problem of determining whether $M$ has some submatrix of any size $k$ such that its smallest $\mathsf{max}(\lceil \ell(k) \rceil, 0)$ off-diagonal elements have sum greater than or equal to $t$. We say that a pair $M, t$ is in the language $\ell$-DS if there does exist such a submatrix; otherwise, it is not.

To prove Lemma 1, we will adhere to the following steps. We begin by defining $k$-clique and a modified version of $k$-clique, which we call threshold $k$-clique. We then show that threshold $k$-clique is NP-complete. Finally, we use this to show Lemma 1 by providing a polynomial-time reduction from threshold $k$-clique to $\ell$-DS.

**Clique Problems** We begin by defining $k$-clique.

**Definition 9**

Suppose we have an undirected graph $G = (V, E)$ and a positive integer $k$. The $k$-clique problem is the problem of determining whether there exists $V' \subseteq V$ such that $|V'| = k$ and for all $u, v \in V' \times V', (u, v) \in E$. We say that a pair $G, k$ is in the language $k$-clique if such a $V'$ exists; otherwise, it is not.

In simpler terms, a pair $G, k$ is in $k$-clique if $G$ has a cluster of $k$ vertices that are fully connected by edges.

We define threshold $k$-clique similarly.

**Definition 10**

Suppose we have an undirected graph $G = (V, E)$ and a positive integer $\mathbf{k \geq c}$ for some fixed non-negative integer $c$. The threshold $k$-clique problem is the problem of determining whether there exists $V' \subseteq V$ such that $|V'| = k$ and for all $u, v \in V' \times V', (u, v) \in E$. We say that a pair $G, k$ is in the language threshold $k$-clique if such a $V'$ exists; otherwise, it is not.

This problem is similar to $k$-clique, but instead only checks for fully connected subgraphs of size at least $c$ for some fixed $c$.

Now, we want to show the following lemma to help our proof of Lemma 1:

**Lemma 2**

The threshold $k$-clique problem is NP-complete for any non-negative threshold $c$.

$k$-clique is well known to be NP-complete [25], so we can prove threshold $k$-clique is NP-hard by finding a polynomial-time reduction $g$ mapping instances of $k$-clique to instances of threshold $k$-clique. Then, we show threshold $k$-clique is also in NP to complete our proof of Lemma 2. Consider the procedure for $g$ laid out in Algorithm 2, which is parameterized by the fixed threshold $c$.

**Algorithm 2** Computation of $g$

**Require:** $(G(V, E), k, c)$

1: Construct a set of new vertices $V^*$ of size $c$
2: Initialize a new graph $G'(V', E')$ with $V' = V$, $E' = E$
3: $V' \leftarrow V \cup V^*$.
4: $E' \leftarrow E' \cup (V^* \times V^*)$
5: $E' \leftarrow E' \cup (V \times V^*)$
6: **return** $G'(V', E'), k + c$

We now prove the three statements necessary for $g$ to be a valid polynomial-time reduction: $G, k \in k$-clique $\rightarrow g(G, k, c) \in$ threshold $k$-clique for arbitrary $c$, $G, k \notin k$-clique $\rightarrow g(G, k, c) \notin$ threshold $k$-clique for the same $c$, and $g$ is polytime computable. Since this is a rather simple proof compared to Lemma 3, we do it all in one step.

The procedure $g$ connects every vertex in $G$ with a fully connected graph of size $c$ in $G'$, such that any clique on $G$ of size $c'$ is a clique on $G'$ of size $c' + c$. As a result, if $G$ does have a $k$-clique, $G'$ will have a $(k+c)$-clique, such that $(G', k+c)$ will be in threshold $k$-clique $(k+c \geq c)$. However, if $G$ does not have such a clique, $G'$ will not have a $(k + c)$-clique – we only increase the size of each clique in $G$ by $c$, adding no new edges among vertices originally in $G$. This procedure is also polynomial time computable, as $c$ is fixed – we only do one operation linear in $|V|$ in line 5.

We have thus proven that $g(G, k, c)$ is in threshold $k$-clique for threshold $c$ if and only if $G, k$ is in $k$-clique. Since $g$ is polytime computable, threshold $k$-clique is NP-hard for any threshold $c$.

Given a possible instance of threshold $k$-clique, $(G = (V, E), k)$, and a certificate $c \subseteq V$, we can check whether $G, k$ is in threshold $k$-clique in polynomial time. In particular, we verify that the cardinality of $c$ is equal to $k$ and greater than the threshold in linear time $O(|V|)$, and then check whether $c \times c \subseteq E$ in linear time – $O(|E|)$. Therefore, threshold $k$-clique is in NP, given we can verify an instance in polynomial time.

Since threshold $k$-clique is NP-hard and in NP, it is NP-complete.

**Proof of Lemma 1** We can now prove NP-hardness of $\ell$-DS by finding a polynomial-time reduction, $f_\ell$, from threshold $k$-clique to $\ell$-DS for any $\ell \in \mathcal{L}$. Then, we can show it is in NP, concluding that Lemma 1 is correct. We start by stating this idea formally.

**Lemma 3**

There exists a polynomial-time reduction $f_\ell$ from threshold $k$-clique with some threshold $c$ to $\ell$-DS, for all $\ell \in \mathcal{L}$.

$f_\ell$ must take in $G, k$ and return $M, t$ such that $G, k \in$ threshold $k$-clique $\leftrightarrow f_\ell(G, k) \in \ell$-DS for some threshold $c$. Note that $f$ is parameterized by $\ell$, given that we are showing NP-completeness for any choice of these parameters. We get to choose any fixed $c$ for each $\ell$, given threshold $k$-clique is NP-complete for all thresholds; notationally, we include $c$ as an extra argument to $f_\ell$. Consider the following construction for $f_\ell$:

**Algorithm 3** Computation of $f_\ell$

**Require:** $(G(V, E), k, c)$
1: $M_{ij} \leftarrow \mathbb{1}_{(i,j)\in E} - 2|E|\mathbb{1}_{(i,j)\notin E'}$ for $i, j \in V$
2: **if** $k \geq c$ **then**
3: $\quad t \leftarrow \lceil \ell(k) \rceil$
4: **else**
5: $\quad t \leftarrow 2|E| + 1$
6: **return** $M, t$

Note that the diagonal of $M$ will be filled with $-2|E|$, assuming our undirected graph has no self-edges. However, since self-edges do not impact the presence of a clique and we do not keep track of diagonal elements in $\ell$-DS, this decision will not pose any issue.

We can now prove the necessary three statements for $f_\ell$ to be a valid polynomial-time reduction, starting with the following lemma. Recall that we can parameterize our clique threshold using $\ell$, given we fix it in advance; in other words, we are showing NP-completeness for all $\ell \in \mathcal{L}$ in parallel.

**Claim 1**

If $G, k \in$ threshold $k$-clique, $f_\ell(G, k) \in \ell$-DS for threshold $c = \mathsf{inf}\{k | (\ell(k) > 0) \wedge (\forall k' > k, \ell(k') \geq 1 + \ell(k' - 1))\}$.

Note that with $c = \mathsf{inf}\{k | (\ell(k) > 0) \wedge (\forall k' > k, \ell(k') \geq 1 + \ell(k' - 1))\}$, we know that for any instance $G, k$ of threshold $k$-clique (i.e., $k \geq c$), $t = \lceil \ell(k) \rceil$ is positive. Also note that the infimum exists, given $\ell \in \mathcal{L}$, and we defined $\mathcal{L}$ to be the set of all functions where this infimum exists.

If $G, k \in$ threshold $k$-clique, then $G$ has a clique of size $k > c$. Call the vertices in this clique $V' \subseteq V$; then, for $i, j \in V'$, $(i, j) \in E$. By construction of $f_\ell$, notice that $M$ is symmetric, as $G$ is undirected. The construction also implies that the corresponding $M_{ij} = 1$ for all $i, j \in V'$, such that $M$ has a submatrix of size $k$ populated entirely by 1s off the diagonal, given $|V'| = k$. Therefore, there exists a submatrix of size $k' = k$ where the sum of its smallest $\lceil \ell(k') \rceil$ (not 0, by the positivity of $t$) off-diagonal elements is $\lceil \ell(k) \rceil$. However, since $t = \lceil \ell(k) \rceil$ for $f_\ell(G, k)$, this means that $f_\ell(G, k) \in \ell$-DS.

Now, we prove the other direction.

**Claim 2**

If $G, k \notin k$-clique, $f_\ell(G, k) \notin \ell$-DS for the same threshold $c = \mathsf{inf}\{k | (\ell(k) > 0) \wedge (\forall k' > k, \ell(k') \geq 1 + \ell(k' - 1))\}$.

If $G, k \notin k$-clique, either $k < c$ *or* there is not a set $V'$ as in our proof of Lemma 3.

We begin by eliminating the former case. If $k < c$, the associated $t$ through $f_\ell$ is $2|E| + 1$. But, by construction, $M$ has at most $2|E|$ 1s and is otherwise all negative, such that no submatrix, even with all off-diagonal elements included, can sum to $t$. Therefore, $f_\ell(G, k) \notin$ threshold $k$-clique.

For the latter case, we want to show that the associated $M$ has no submatrix of size $k'$ with sum of its smallest $\mathsf{max}(\lceil \ell(k') \rceil, 0)$ off-diagonal elements greater than or equal to $t$. Note that now, since $k \geq c$ by elimination of the former case, $t = \lceil \ell(k) \rceil > 0$. We proceed by cases over $k'$.

First, let $k' < k$. Each submatrix of size $k'$ corresponds to a set of vertices $V' \subseteq V$, $|V'| = k'$. In the worst case, $V'$ is a clique of size greater than $c$, and the associated submatrix is all 1s off the diagonal; otherwise, the submatrix would have a large negative term that would definitively be in the set of $\mathsf{max}(\lceil \ell(k') \rceil, 0) = \lceil \ell(k') \rceil$ smallest off-diagonal elements (or, $\mathsf{max}(\lceil \ell(k') \rceil, 0) = 0$ and the sum is $0 < t$). However, even when $V'$ is a clique, the sum of its $\lceil \ell(k') \rceil$ smallest off-diagonal elements is $\lceil \ell(k') \rceil$, given each element is of size 1. This is necessarily strictly less than $t = \mathsf{max}(\lceil \ell(k) \rceil, 0)$, since $\ell$ is positive and monotonically increasing by at least 1 per unit increase in its input for $k \geq c$. We get this property by definition of $c$.

Next, let $k' \geq k$. By our assumption that $G$ has no clique of size $k$ (and thus, size $\geq k$), there is no submatrix in $M$ of size $\geq k$ consisting of all 1s off-diagonal – otherwise, there would be a set of $k$ fully connected vertices in $G$. As a result, each submatrix of size $k'$ must have at least one off-diagonal entry with value $-2|E|$, by our construction of $f_\ell$ and the fact that $k' \geq k \geq c$. Also by construction, there are at most $2|E|$ positive entries in $M$; only cells $M_{ij}, (i, j) \in E$ are positive. Therefore, for any submatrix of size $k'$, it has sum at most 0, which is an upper bound for the sum of its smallest $\lceil \ell(k') \rceil$ off-diagonal elements. As such, it has sum less than $t = \lceil \ell(k) \rceil > 0$, given $\ell$ is positive for $k \geq c$.

We have thus proven that all submatrices of $M$ of any integer size $k'$ have sum of their smallest $\lceil \ell(k') \rceil$ off-diagonal elements less than $t = \lceil \ell(k) \rceil$, meaning $M, t \notin \ell$-DS.

Now, we know our reduction is correct – all we need to do is prove that it is polynomial time.

**Claim 3**

$f_\ell(G, k)$ is computable in polynomial time for any undirected graph $G$ and positive integer $k$.

Constructing $M$ requires one iteration over $E$ to fill in all cells with $(i, j) \in E$ and compute the size of $E$. It then requires $|V|^2 - |E|$ steps to fill in the rest of $M$. Constructing $t$ requires a constant time computation of $\lceil \ell(k) \rceil$ or $2|E| + 1$, both of which we know or can compute in less than quadratic time in $|V|$. Therefore, the runtime of $f_\ell$ is $\mathcal{O}(|V|^2)$, which is polynomial in the size of $G$.

By Claim 1, Claim 2, and Claim 3, we have proven Lemma 3. In conjunction with Lemma 2, this result implies that the $\ell$-DS problem is NP-hard.

Additionally, $\ell$-DS is in NP. Given a candidate instance $M, t$ and a certificate $c$ that is a submatrix of size $|c|$ of $M$, we can verify that the smallest $\lceil \ell(|c|) \rceil$ off-diagonal elements of $c$ sum to greater than or equal to $t$ by computing $\lceil \ell(|c|) \rceil$ in constant time and summing in at worst linear time – $\mathcal{O}(M)$. Since membership verification is polynomial time, $\ell$-DS is in NP.

Since $\ell$-DS is NP-hard and in NP, it is NP-complete, proving Lemma 1.

**Equivalence of Lemma 1 and Theorem 1** Suppose we have some contact matrix $C$ and want to approximate a fold in the HP model; i.e., we want to solve

$$\mathsf{argmin}_{x,y} E^*_{HP}(S, C, x, y)$$

We define $E^*_{HP}$ as our generalized energy function, where we replace $-k\mathbb{1}_{i\in H}\mathbb{1}_{j\in H}$ with $C_{ij}$ in our earlier definition of $E_{HP}$ from Section 2.3.1:

$$E^*_{HP}(S, C, x, y) = \frac{1}{2} \sum_{i,j \in S \times S} \Delta(y_i, y_j) C_{ij}$$

Remember that we divide by $\frac{1}{2}$ to avoid double-counting contacts. Since $C$ is symmetric, each off-diagonal component will be duplicated, such that the guaranteed energy of a folding algorithm given a function $\ell(\cdot)$ is the *half* sum of the largest $\lceil 2\ell(k) \rceil$ off-diagonal elements for a submatrix of size $k$. Note that diagonal elements always contribute zero energy, as the HP-model correctly avoids any statements or optimizations related to self-interaction.

Most importantly, rather than taking in $H$ according to a hydrophobicity scale, this more general energy function takes in a contact potential matrix $C_{ij}$. For example, observe that

$$E^*_{HP}(S, -k\mathbb{1}_{i\in H}\mathbb{1}_{j\in H}, x, y) = E_{HP}(S, H, x, y)$$

by our original definition of $E_{HP}$.

We can approach this minimization problem by doing some sort of global optimization over each pair in $C$ to minimize energy. However, we might instead want to reduce to the standard HP model by constructing a set $H$ to minimize guaranteed energy by way of some $\ell$-optimal approximation algorithm, like Hart-Istrail [9].

To choose this set $H$, we want to find the subset of $S$ with the minimum amount of energy guaranteed by an $\ell$-optimal approximation algorithm of our choice with $\ell \in \mathcal{L}$, a restriction we will discuss qualitatively below. Recall that an $\ell$-optimal approximation guarantees $\lceil \ell(k) \rceil$ contacts for some function $\ell \in \mathcal{L}$ and $H$ such that $|H| = k$. Therefore, our desired subset is exactly the submatrix of $C$ with the smallest sum across its largest $\lceil 2\ell(k) \rceil$ off-diagonal elements, for any size $k$. We know this is the case, as we get the smallest guaranteed energy by this choice of $H$, given the factor of $\frac{1}{2}$ is applied uniformly to all selections. At the very least, we guarantee that the $\lceil \ell(k) \rceil$

highest energy contacts in our chosen set $H$ have a minimal energetic sum compared to any other set $H$.

However, Lemma 1 says that the problem of choosing this subset is NP-complete. The problem of choosing the submatrix of $C$ with the smallest sum of its largest $\lceil 2\ell(k) \rceil$ off-diagonal elements is exactly the same as choosing the submatrix of $-C$ with the largest sum of its smallest $\lceil 2\ell(k) \rceil$ off-diagonal elements. Lemma 1 shows that this problem is NP-complete when $2\ell$ is in $\mathcal{L}$, as $-C$ is some arbitrary symmetric matrix.

On the other hand, we have no guarantee that $2\ell \in \mathcal{L}$. In particular, an approximation algorithm only has its associated guarantee $2\ell \in \mathcal{L}$ if, for some sequence length, each incremental hydrophobic guarantees at least one extra contact in the structure. To some extent, this is a reasonable assumption – most reasonable lattices have sites with at least 3 neighbors, such that $2\ell^*(|H|) \geq 3|H|$, as in the 2D square lattice case. Any algorithm that is at least exactly or asymptotically $\frac{1}{3}$-optimal on this "worst-case" lattice would thus satisfy $2\ell \in \mathcal{L}$. As the slope of $\ell^*$ increases for more and more realistic 3D lattices, an approximation need only satisfy $\frac{1}{9}$ (in the FCC case) or less $\alpha$-optimality to be $\ell$-optimal for some $\ell \in \mathcal{L}$. As shown by Hart and Istrail's 86% optimal algorithm, $\frac{1}{9}$ is a relatively low bar.

Regardless, we conjecture that with increasing $\ell \notin \mathcal{L}$, $\ell$-DS is still NP-complete. Suppose we accordingly relaxed the restriction that $\ell$ increases by at least one for each increase by one of its input. Then, the reduction from threshold $k$-clique above would fall through, as in the $k' < k$ case of our proof for Claim 2, we would run into the problem that $\lceil \ell(k') \rceil$ could be equal to $\lceil \ell(k) \rceil = t$.

However, we could instead reduce from a variant of threshold $k$-clique where $G, k$ is an instance of this new problem if it contains a clique on any $k'$ such that $\ell(k') = \ell(k) > 0$. Provided the number of such $k'$ is finite for all possible $k$, this problem is NP-complete – it essentially just puts all $k'$ greater than the selected threshold in threshold $k$-clique into finite buckets. As long as we have some way to find the start of the next bucket $k'$ for some input $G, k$ in polynomial time, we can connect a clique of size $k' - k$ to $G$. The new instance $(G', k')$ would only be in our new problem if $G$ had a clique of size $k$, given $k'$ is the smallest clique size in its bucket; if there is no $k'$-clique in $G'$, there certainly is no larger clique.

Reducing from this new problem would allow us to cover any reasonable $\ell$. We know this is the case, as if there were ever an exponential set of inputs $\mathcal{X}$ such that $\ell(x) = \ell(x')$ for all $x, x' \in \mathcal{X} \times \mathcal{X}$, $\ell$ would asymptotically fall away from $\ell^*$ towards 0, as $\ell^*$ is generally linear given finite lattice sites. However, this more complex reduction is a needlessly complex notion, given most useful approximations have a guarantee $\ell$ such that $2\ell \in \mathcal{L}$.

Additionally, even our choice to reduce from threshold $k$-clique weakens this result a little; for small enough $k$, asymptotic algorithms give no guarantee ($\ell(k) \leq 0$), such that the result is only truly NP-complete for long enough sequences. This doesn't decrease the strength of the result practically, however, as asymptotic algorithms are useless on these problematic shorter sequences. If we weaken $2\ell$ to additionally be outside of $\mathcal{L}$ such that we need a more complex reduction, we would also only have to search a limited set of possible $k$ (those that are minimal in their bucket). For the weakest of approximations, this means that NP-completeness only comes into play for long sequences such that there are many buckets over which we need to search.

We can take a moment to appreciate this result. In particular, we have learned that it is not feasible to optimize even lattice folding by choosing a set of amino acids whose contacts we prioritize. Additionally, it becomes increasingly infeasible for stronger approximations, as the NP-completeness challenges arise even on shorter sequences. We conclude that there is no polynomial-time way to

begin to move folding with an arbitrary contact potential matrix from contact-focused optimization to amino-acid-focused optimization unless there are simplifications in $C$. This holds for all amino-acid-focused approximation algorithms such that $\ell \in \mathcal{L}$, and likely for an even larger set. In some way, we can claim that protein folding is infeasible to attack by focusing on a specific subset of amino acids, because we cannot choose the best subset itself efficiently. The most energetically important amino acids, according to a lattice approximation, are infeasible to isolate from arbitrary pairwise energies.

### 2.3.3 Simplifications Towards Polynomial Time

However, the problem of choosing a set $H$ from a contact potential matrix $C$ is not completely hopeless. In particular, recall $C_{ij}$ is the contact potential of $s_i, s_j \in S$ for some amino acid sequence $S$. Consider $C$ instead in its functional form, from $S \times S \to \mathbb{R}$. Then, if $C$ is non-increasing in some ordering of $S$, there exists a polynomial-time algorithm for selecting $H$. First, we define what this notion of non-increasing means.

**Definition 11**

We call $C$ "non-increasing" for an ordering $O$ of $S$, $O : S \to \{1, ...|S|\}$, if $O(s_i) \leq O(s_k)$ and $O(s_j) \leq O(s_\ell) \to C(s_i, s_j) \geq C(s_k, s_\ell)$ for $i \neq j, k \neq \ell$.

For our purposes, recall $C$ is also symmetric; i.e. $C(s_i, s_j) = C(s_j, s_i)$ – the energy of attraction is a property of the two unordered amino acids.

**Algorithm 4** Selection of $H$ from Non-increasing $C$ Given an $\ell$-Optimal Algorithm

**Require:** $C$, $S$, $\ell$

In $S'$, store the ascending sorting of $s \in S$ by $\sum_{s' \in S \setminus \{s\}} C(s, s')$
$\mathsf{best} \leftarrow (0, \infty)$
**for** $k = 1 : |S|$ **do**
    $\mathsf{current} \leftarrow \frac{1}{2}$ the sum over the largest $\lceil 2\ell(k) \rceil$ elements in the submatrix of $C$ defined by $S'_{1:k}$
    **if** $\mathsf{best}[1] > \mathsf{current}$ **then**
        $\mathsf{best} \leftarrow (k, \mathsf{current})$
**return** $H = S'_{1:\mathsf{best}[1]}$

We can begin by proving that Algorithm 4 correctly chooses $H$ such that the submatrix associated with $H$ in $C$ has the best (minimal) guaranteed energy; more formally, that this submatrix of size $k$ has the minimal sum across all $\lceil 2\ell(k) \rceil$ of its largest off-diagonal elements. Finding this submatrix gives us the best performance guarantee for some $\ell$-optimal algorithm.

First, notice that a non-increasing $C$ implies the existence of some $O$ as in Definition 11. Then, observe that $S'$ is exactly sorted such that $O(s'_i) \geq O(s'_j)$ for all $i < j \in \{1, ...|S|\}$ for some valid $O$. We know this is the case by contradiction. Suppose that for all valid $O$, $O(s'_i) < O(s'_j)$ for some $i < j$. Then, for all $s \in S \setminus \{s'_i, s'_j\}$, $C(s'_i, s) \geq C(s'_j, s)$, such that $\sum_{s \in S \setminus \{s'_i\}} C(s'_i, s) \geq \sum_{s \in S \setminus \{s'_j\}} C(s'_j, s)$. Note we add the cross term $C(s'_i, s'_j)$ on both sides of the inequality. However, we sorted $S'$ such that if $i < j$, $\sum_{s \in S \setminus \{s'_i\}} C(s'_i, s) \leq \sum_{s \in S \setminus \{s'_j\}} C(s'_j, s)$. The only possible case

where $O(s'_i) < O(s'_j)$ and $i < j$ is thus when $\sum_{s\in S\setminus\{s'_i\}} C(s'_i, s) = \sum_{s\in S\setminus\{s'_j\}} C(s'_j, s)$; but, then there exists a valid $O$ such that $O(s'_j) \geq O(s'_i)$, given we can flip the ordering of two points in $O$ if they have the same $C(\cdot, s)$ for all $s \in S$.

From here, notice that the submatrix of size $k$ in $C$ with the smallest sum of its largest $\lceil 2\ell(k)\rceil$ off-diagonal elements is $S'_{1:k}$. Let $H \subseteq S$ with $|H| = k$. If $H \neq S'_{1:k}$ then there exist indices $i \leq k < j$ such that $s'_j \in H$ and $s'_i \notin H$. Since $O(s'_i) \geq O(s'_j)$ and $C$ is non-increasing, we have for all $s \in S\setminus\{s'_i, s'_j\}$, $O(s'_j) \leq O(s'_i) \rightarrow C(s'_j, s) \geq C(s'_i, s)$ and, by symmetry, $C(s, s'_j) \geq C(s, s'_i)$. Replacing $s'_j$ by $s'_i$ in $H$ can only decrease each entry in the off-diagonal of the $k \times k$ submatrix, so the sum of its $\lceil 2\ell(k)\rceil$ largest off-diagonal entries does not increase. Repeating this exchange eventually shows that $S'_{1:k}$ therefore attains the minimum desired sum across all submatrices of size $k$.

Since we directly compare this minimal sum for all possible submatrix sizes $k \in \{1, ...|S|\}$, Algorithm 4 achieves the minimum possible guaranteed energy across all possible $H \subseteq S$.

This algorithm is additionally polynomial time in $S$. Computing the sums on which we sort takes $\mathcal{O}(|S|)$, and sorting then takes $\mathcal{O}(|S| \log |S|)$. In the worst case, the following loop sums $|S|^2 - |S|$ elements for each submatrix size upper bounded by $|S|$, such that this step is necessarily $\mathcal{O}(|S|^3)$.

We have therefore shown that, when $C$ is non-increasing as in Definition 11, we can distinguish the language $\ell$-DS in polynomial time. Given $M, t$ and $\ell$, we can make up some list $S$ associated with the margins on $M$, run Algorithm 4, and compare best to $t$.

The assumption that $C$ is non-increasing is not, in general, a physical one. One amino acid will not dominate another completely in terms of contact energy, much less a full ordering of domination as in Definition 11. However, it does allow us to reduce any hydrophobicity scale to the HP model, if we let $C(s_i, s_j)$ be non-increasing in terms of the hydrophobicity of $i$ and $j$.

For example, if we let $C = -\sum_{i,j\in S} E_i E_j \mathbb{1}_{E_i>0}\mathbb{1}_{E_j>0}$ where $E$ is a scale generated using WaterFire, $C$ is non-increasing with respect to the ordering $O(s_i) > O(s_j) \leftrightarrow E_i > E_j$. Notice that this $C$ is exactly the outer product of the hydrophobicity scale $E$, zeroing out any energetic terms that include a residue with positive energy, following prior notions of generalized hydrophobicity [26]. More generally, with our current implementation of WaterFire and some increasing function like this modified outer product, we can efficiently find a candidate set $H$.

The results of these induced simplifications in $C$ are mixed. We defined the contact potential matrix $C$ to be non-increasing when there exists an order of amino acids such that the contact energy of a higher order amino acid is always greater than the contact energy of a lower order amino acid, all else equal. Then, we showed that this assumption implies the optimal $H$ is the highest order $k$ amino acids for some $k$, such that we can iterate over $k$ linearly to find the exact optimal $H$. However, we concluded that this dominance assumption isn't necessarily a good one physically. On the other hand, any increasing function of a hydrophobicity scale with two arguments allows us to use this algorithm, and thus reduce to HP model folding. So, under highly simplifying assumptions, we can reduce from a contact potential matrix $C$ to a set $H \subseteq S$ that allows us to use $\ell$-optimal approximations.

### 2.3.4 Integrating Contextual Information

For this section, we construct the contact potential matrix $C$ by our example in the previous section, similar to previous theories of generalized hydrophobicity [26]:

$$-\sum_{i,j\in S} E_i E_j \mathbb{1}_{E_i>0}\mathbb{1}_{E_j>0}$$

which allows us to choose $H$ efficiently given a hydrophobicity scale with values $E_s$ for all $s \in S$. This construction of $C$ is very simple with a context-independent scale, and will yield $H \subseteq \{s|E_s \geq 0\}$. It is also the standard way of selecting $H$ under the assumption that an algorithm can make every possible contact in $H \times H$; in this case, $H = \{s|E_s \geq 0\}$, such that we just threshold the hydrophobicity scale $E$ at 0. Since most hydrophobicity scales imply hydrophobicity for any residue with $E_s \geq 0$, this idea is canonical.

However, we reach yet another challenge when we instead use structure-informed hydrophobicity. Namely, we need to determine how to produce a structure for each calculation of the hydrophobicity scale – WaterFire needs a structure to initialize its molecular dynamics! To combat this, we propose a new algorithm for integrating structure-informed hydrophobicity scales with the HP model.

**Algorithm 5** Structure-Informed HP Folding

**Require:** $S$ (amino acid sequence), $E_0$ (initial hydrophobicity scale), $f$ ($\ell$-optimal HP folding algorithm), $g$ (structure-informed hydrophobicity scale), $T$ (number of iterations)

1: **for** $1 \leq t \leq T$ **do**
2: Run our selection algorithm to find $H_t \subseteq S$ from $E_{t-1}$
3: $x_t, y_t \leftarrow f(S, H_t)$
4: Extrapolate atomic coordinates $q_t$ from lattice coordinates $x_t, y_t$
5: $E_t \leftarrow g(q_t, S)$
6: **return** $q_T$

In Algorithm 5, we initialize a structure prediction using some standard HP model approach (i.e., $E_0$ is some context-independent hydrophobicity scale) and then iteratively update our hydrophobicity and structure estimates. One critical external tool we use is PULCHRA, which allows us to extrapolate backbone and side chain centroids to fully-fledged atomic coordinates [27]. While the atomic coordinates are by no means perfect or even fully physical in some cases, PULCHRA still gives us a prior for WaterFire and our molecular dynamics pipeline.

In essence, Algorithm 5 seeks self-consistency: a structure for which the structure-informed hydrophobicity scale would produce a similar structure. This notion encourages structures that are consistent with the logic behind the structure-informed hydrophobicity scale. For example, in the case of WaterFire, this algorithm will equilibrate at structures that have relatively constant water density fluctuations around the surface, such that the hydrophobicity scale changes minimally from iteration to iteration. Other structure-informed hydrophobicity scales might encourage other properties; for example, leading to consistent water orientation distributions around the structure. In all such cases, this iterative algorithm allows for the refinement of HP model folds, up to consistency with some physical notion.

We evaluate such folds using two metrics: potential energy according to our usual Amber force field and root mean square deviation (RMSD). The calculation of RMSD begins with the

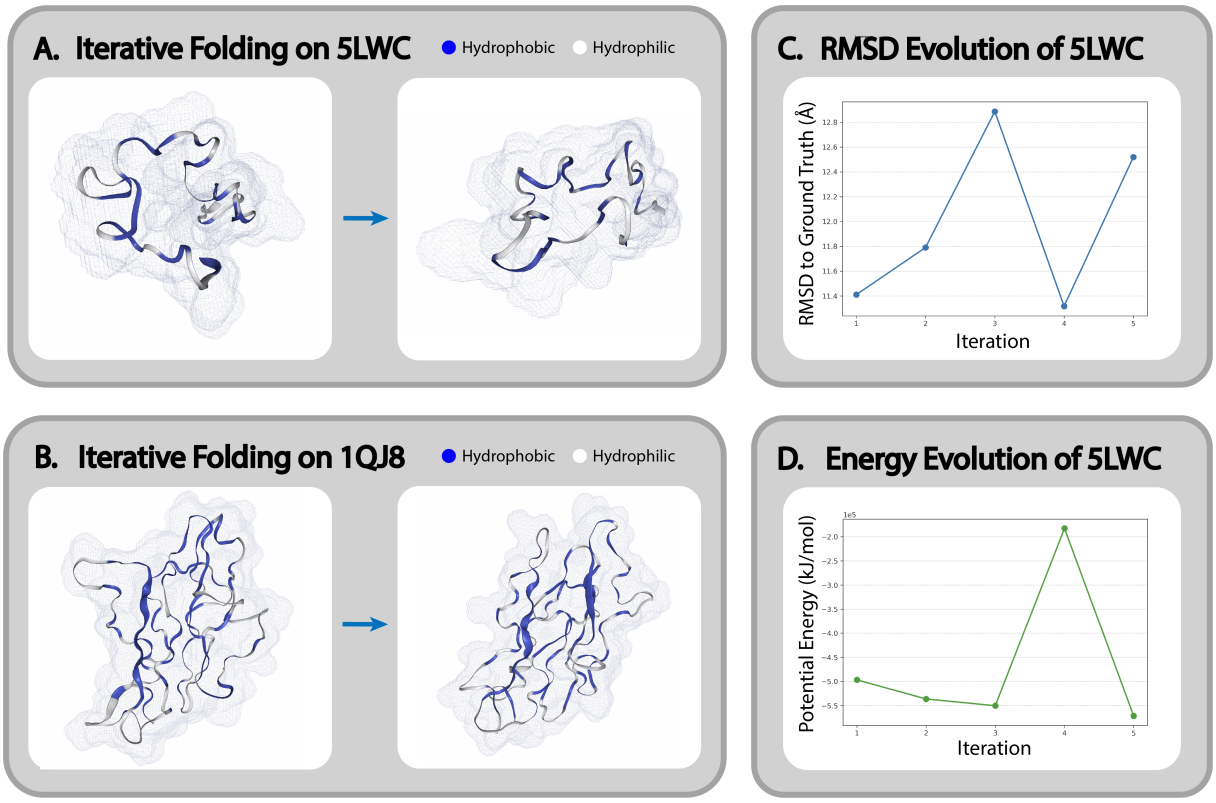



**Figure 2:** Example of Algorithm 5 on two proteins, PDB identifiers 5LWC and 1QJ8. Both **A** and **B** show the result of running 10000 MD steps of 2fs on a structure extrapolated directly from the Hart-Istrail approximation using raw Kyte-Doolittle scores on the left-hand side. On the right hand side, they show the result of running an equivalent MD simulation using WaterFire adjusted scores using density fluctuations from the prior overall iteration (of which we ran 5 for 5LWC and 2 for 1QJ8). **C** shows the change in RMSD to an experimental structure [29] over each overall iteration for 5LWC, while **D** shows the same plot for potential energy of the final MD state under our usual Amber force field, explicit water molecules included.

famous exact result for optimally superimposing protein structures [28]. Using these superimposed structures $q, q'$, where each $q_i, q_i'$ are atomic coordinates in $\mathbb{R}^3$, RMSD is defined to be

$$\mathsf{RMSD}(q, q') = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|q_i - q_i'\|^2}$$

such that it measures the average square error of each atomic position and takes the square root.

Using our implementation of both WaterFire and the Hart-Istrail HP approximation, we were able to test Algorithm 5 on a real protein sequence, as in Figure 2, evaluating according to these metrics. While the results are mixed, we found it an interesting experiment to see whether incorporating physical context into a single scale was sufficient to yield more physical results, whether folding on lattices is fundamentally nonphysical, or whether the truth lies somewhere in between. From Figure 2, we were able to determine that the secondary structure is influenced heavily by the lattice representation, as is obvious by the large and relatively invariant RMSD values against experimental structures. Additionally, there are no secondary structure elements (alpha helices, beta

sheets) detected by our visualization tool in any prediction, WaterFire scales included. However, the incorporation of structure information allows for some reasonable alternate sampling among such structures, given that the potential energies tend to be in a similar range despite highly variable initializations. We believe that by continuing to relax lattice constraints in HP model folding or incorporating secondary structure information, this style of algorithm might allow for more and more reasonable folds.

One other observation we made from running these experiments is that the potential energy of the final fold inversely correlates closely with the mean density fluctuation around the protein. This result indicates a potential flaw in the WaterFire approach. With WaterFire, Algorithm 5 will converge to a structure that has a relatively uniform mean density fluctuation by self-consistency; however, this fluctuation may be large, but still uniform. In this case, the equilibrium reached by Algorithm 5 will be suboptimal and potentially high energy. As a result, one future direction for this type of scale would be to reference each fluctuation in WaterFire to some empirical value independent of the mean fluctuation around the protein, so that the equilibrium is instead guided by an absolute rather than relative standard.

## 2.4 Conclusion

In this chapter, we have covered a wide variety of topics in combinatorial protein folding. We began by defining the classical hydrophobicity scale as context-independent, which set the stage to introduce our new definition of structure-informed hydrophobicity scales. We then examined historic hydrophobicity scales, noting the challenge of distilling rich physical information into a single invariant function.

From here, we proposed our own structure-informed scale, WaterFire, which incorporates slightly more physical information in its still simple representation. WaterFire takes in some structural prior and corrects the hydrophobicity of each amino acid by how it behaved in simulation. In this work, we measure the behavioral hydrophobicity of each amino acid by the surrounding water density fluctuations it induces.

We then delved into the primary use of such scales: HP model folding. We defined the HP model and $\ell$-optimal approximations, noting their geometric limitations. We then considered whether we could incorporate even more structural information than in WaterFire, such as constructing a fully-fledged contact potential matrix as some extension of a structure-informed scale. However, we observed that applying HP model approximations to more general scales requires the solution to an NP-complete problem – in particular, the selection of which residues to deem hydrophobic in the HP model.

On the other hand, we then observe that there does exist a polynomial-time reduction to the HP model for specific contact potential matrices. Specifically, if the contact potential matrix is non-increasing with respect to some ordering of the amino acids, taking the highest order amino acids yields the lowest guaranteed energy selection of $H$, as those amino acids "dominate" the rest in terms of energy contribution. While this type of contact potential matrix does not make the most physical sense, we note that any non-increasing function of two variables of any hydrophobicity scale does satisfy this non-increasing property. For example, we use a modification of the negative outer product as a function to construct a contact potential matrix from WaterFire.

Finally, we address the problem of generating structural priors for structure-informed hydrophobicity scales by introducing an iterative algorithm for HP model folding. We test this algorithm using WaterFire and the Hart-Istrail [9] 0.86-optimal approximation algorithm on the FCC lattice.

We find that even with structural context, the inherent space of possible structures extrapolated from a lattice approximation is likely too small to get close to experimental structures. However, we do show that we can sample from the particular structural style of a given approximation algorithm.

Incorporating physical context into HP model folding is incredibly challenging. By nature, the HP model is highly discrete and loses significant quantities of physical information. However, we were able to characterize both its limitations and its benefits. We were able to show that singular, rather than pairwise, amino acid optimization poses problems for incorporating physical information, but that we can nonetheless incorporate more than existing context-independent approaches.

While HP model folding, with our modifications, is still quite limited compared to the newer deep learning methods, the new advances of the past decade still leave much to be desired in terms of applying physical concepts to prediction. We now move on to highlighting these issues in AlphaFold and proposing a way to address them.

# 3 Deep Learning

Since its first appearance at CASP, a biannual protein structure prediction competition, AlphaFold and methods like it have dominated the field [1]. Methods like these use deep learning to predict atomic coordinates directly from a protein sequence. The general strategy of current deep learning paradigms for structure prediction is to abstract away the implicitly learned patterns by first finding similar sequences and then interpolating.

As a thought experiment, imagine that each protein sequence could be represented in two dimensions and that the set of all reasonable protein sequences (i.e., biologically relevant) could be placed in a subspace of $\mathbb{R}^2$. AlphaFold and its competitors are given, in training, structural information at a discrete set of points in this subspace. The problem they seek to solve for a given sequence is twofold:

1. Locate the input sequence $S$ in $\mathbb{R}^2$ and find its nearest neighbors that have structural information.

2. Interpolate the structural information given by the nearest neighbors.

This approach is highly sequence-focused, rather than incorporating any physical forces or energy functions. We call this type of approach *homology-based*. In this section, we challenge the efficacy of such an approach by examining whether homology captures physical challenges. We then propose a way to incorporate physical notions into models like AlphaFold, which we call the *Subspace Relaxation Operator* (SRO).

## 3.1 AlphaFold

We begin by examining AlphaFold in particular. We consider its architecture, as well as how its homology-based structure impacts its predictions.

### 3.1.1 Model Architecture

As was the goal in our earlier foray into combinatorics, AlphaFold's goal is to approximately find $\mathsf{argmin}_q E_t(q)$. Rather than trying to approximate the energy function and find an exact solution as in the combinatorics case, AlphaFold attempts to approximate experimental data. As a result,

one important assumption in these deep learning models arises: they assume that experimentally derived structures roughly minimize $E_t(q)$. One other viewpoint is that fitting experimentally derived structures is more important than globally minimizing $E_t(q)$, as the true global minimum might rarely be populated practically. For example, the global minimum might be surrounded by tall, high-order saddle points on the surface implied by $E_t(\cdot)$.

In either case, AlphaFold is trained to produce structures from sequences by backpropagation over a complex deep learning architecture. Specifically, for each sequence-structure pair $(S, q)$ in its dataset, AlphaFold is trained as a function $f$ seeking to minimize the difference between $f(S)$ and $q$. This difference is evaluated by the AlphaFold loss functions, which we examine briefly below, and then backpropagated to improve the performance of $f$ during training.

To understand exactly what this optimization seeks to achieve, we begin by looking at each component of AlphaFold's architecture, starting with the *Multiple Sequence Alignment* (MSA). We will then move on to the *Evoformer*, followed by a look at the *Structure Module*. These three components make up the core of AlphaFold's prediction pipeline. Usually, AlphaFold runs roughly three iterations of each of these components to produce a final structure. We then wrap up by looking at the AlphaFold loss terms.

**MSA** The MSA is the primary component of AlphaFold that relies on homology. It is also the first step in predicting structure $f(S)$ from some initial sequence $S$ of length $n$. Given some database of sequences $\mathcal{S}$, the MSA step seeks to find the alignment of all sequences in $\mathcal{S}$ to $S$ such that some probabilistic measure of alignment cost is minimized. From here, the best $c$ individual alignments with $S$ from $\mathcal{S}$ are saved and embedded in a space of dimension $e_m$. In other words, it produces a matrix of shape $c \times n \times e_m$ containing information on similar sequences to $S$.

In practice, this task is frequently done with a profile hidden Markov model. For example, HHblits, a method used by tools like AlphaFold, initializes a hidden Markov model with a linear chain of states, each with adjacent insertion and deletion states [30]. Given $S$, it then fills in likely parameters for each transition and emission according to the physicochemical properties of each amino acid; for example, the distribution at a positional state with a hydrophobic in $S$ might be concentrated on a small subset of hydrophobic amino acids. This parameter initialization takes into account the surrounding 13 residue window for each positional state in the sequence.

From here, HHblits uses this HMM to query a set of precomputed HMMs learned from clusters on the database $\mathcal{S}$. These comparisons yield similar sequences and slowly allow for the accumulation of the $c$ necessary sequences. The embedding of these sequences is then handled by AlphaFold or any other deep learning algorithm that leverages an MSA.

Once the MSA embedding has been computed, AlphaFold also computes a pair embedding of shape $n \times n \times e_p$; each cell in the $n \times n$ matrix has an embedding dimension of $e_p$. It can be encoded from some prior on the pairwise distances (a distogram) or initialized simply to a positional encoding of the residue index differences. Some feed-forward transformation of the average outer product of each row in the MSA is then added to the initial encoding within the first Evoformer block prior to any manipulation of the pair embedding.

**Evoformer** With the MSA and pair embeddings computed, AlphaFold moves onto the Evoformer stack – the beginning of the real deep learning components. The Evoformer stack is composed of 48 Evoformer blocks, where each block proceeds by Algorithm 6.

**Algorithm 6** EvoformerBlock

**Require:** $M \in \mathbb{R}^{c \times n \times e_m}$, $Z \in \mathbb{R}^{n \times n \times e_p}$

$M \leftarrow M + \mathsf{MSARowAttention}(M)$
$M \leftarrow M + \mathsf{MSAColumnAttention}(M)$
$M \leftarrow M + \mathsf{MSATransition}(M)$
$\Delta Z \leftarrow \mathsf{Linear}(\frac{1}{c}\sum_{i=1}^{c} M_i \otimes M_i)$
$Z \leftarrow Z + \Delta Z$
$Z \leftarrow Z + \mathsf{TriangleMulOutgoing}(Z)$
$Z \leftarrow Z + \mathsf{TriangleMulIncoming}(Z)$
$Z \leftarrow Z + \mathsf{TriangleAttentionStart}(Z)$
$Z \leftarrow Z + \mathsf{TriangleAttentionEnd}(Z)$
$Z \leftarrow Z + \mathsf{PairTransition}(Z)$
**return** $M, Z$

Each individual step in Algorithm 6 is a complex set of operations. In the two MSA attention steps, the Evoformer block applies a standard self-attention mechanism across each row and each column of the MSA. The self-attention across rows can, to some extent, be interpreted as tending to relationships between amino acids in the same sequence. The self-attention down the columns, in a similar respect, can be thought of as tending to evolutionary or other relationships at the same position among multiple different sequences in the MSA. The final transition on the MSA consists primarily of standard feed-forward blocks.

As mentioned in the MSA section, $Z$ is augmented with the average outer product across all embedded sequences in the MSA, which has reshaped dimension $n \times n \times e_m \times e_m$ natively. However, the MSA is generally projected into a lower-dimensional space first, followed by a linear layer to reach dimension $n \times n \times e_p$. Functionally, this flattens $e_m \times e_m$ into a single dimension of size $e_p$ for each amino acid pair.

After this augmentation of $Z$, several triangle blocks are used to update $Z$. The key observation of these blocks is to only attend to the set of amino acid pairs with at least one shared row or column index. For example, if we applied standard attention to our $n \times n \times e_p$ matrix, we would need to compute some function over every $(i, j), (k, l)$ pair of $e_p$-dimensional vectors. Instead, with triangle blocks, we only ever compute over pairs that take the form $(i, k), (k, j)$. This simplifies the computational order of each triangle block by one polynomial degree, as we avoid an extra factor of $n$ introduced by the free variable $l$ in the naive case. This architectural choice is motivated by the triangle inequality: whatever we see at $(i, j)$ should be consistent by some measure with the aggregation of $(i, k), (k, j)$.

After the triangle blocks, the final pair transition modifies $Z$ by a set of more standard feed-forward layers. At the end of Algorithm 6, we are left with a refined version of $M, Z$.

Recall that this block is repeated 48 times in a standard AlphaFold iteration. AlphaFold additionally runs three or more iterations of the MSA, Evoformer, and structure module to produce its final predictions. This means that, were we to save the $M, Z$ produced by each Evoformer block, we would end up with at least 144 such pairs. When we eventually get to the SRO later on, we

will refer to each such pair as indexed by iteration and block number. For example, $Z_{2,43}$ would be the 43rd block of the 2nd overall AlphaFold iteration.

**Structure Module** Once the embeddings of the 48th block in the current iteration have been produced, AlphaFold uses them to extrapolate a fully fledged structure with 3D coordinates. The structure module begins by dropping all but the first row of $M_{i,48}$ and running a feed-forward layer on the result. We refer to the result of this operation from $\mathbb{R}^{c \times n \times e_m} \to \mathbb{R}^{n \times e_s}$ as the single embedding, $V_{i,48}$. Then, it applies InvariantPointAttention in addition to some feed forward layers to $Z_{i,48}$ and $V_{i,48}$ to fill in a rotation-translation pair in 3D space for each amino acid, initialized at the origin with no rotation. In applying this module $N$ times, AlphaFold generates a list of rotation-translation pairs for each amino acid. It then produces a final structure by applying each pair in sequence to its associated amino acid.

We generally think of the structure module as a direct transformation $f_s(\cdot)$ from $\mathbb{R}^{n \times e_s} \times \mathbb{R}^{n \times n \times e_p}$ to $\mathbb{R}^{3n_a}$, where $n_a$ is the number of atoms implied by the set of amino acids.

The structure module additionally contains some auxiliary modules – most importantly, using $Z$, it predicts the structural error it will incur, as a sort of confidence score. This error is measured by a Local Distance Difference Test (LDDT), such that the error the module predicts is generally denoted pLDDT. At a high level, LDDT measures the difference in pairwise distances across the two structures along the backbone. For example, if one alpha carbon $i$ is 5 Å away from alpha carbon $j$ in the predicted structure but only 2 Å in the reference structure, the pair $i, j$ would incur an error of 3 Å. LDDT measures the fraction of these errors that are less than some cutoff threshold for all backbone pairs. At CASP, for example, they compute this fraction for cutoffs of 4 Å, 2 Å, 1 Å, and 0.5 Å and average across each case. Unlike RMSD, LDDT operates on atomic pairs, such that it does not require superposition.

The only other auxiliary module we leverage predicts a discrete distribution of possible inter-residue distances for each amino acid pair. For example, this distribution will be of shape $n \times n \times k$, where $k$ is the number of distance intervals on which we support the distribution. We call this distribution a predicted distogram.

Once the structure module outputs a complete structure $q \in \mathbb{R}^{3n_a}$, the prediction process is complete. One observation we will return to, however, is that if this module can be thought of as $f_s(\cdot) : \mathbb{R}^{n \times e_s} \times \mathbb{R}^{n \times n \times e_p} \to \mathbb{R}^{3n_a}$, we can in principle run it on any $(V, Z)_{i,j}$.

**Loss** The standard loss terms in AlphaFold, which we also use below for the SRO, include frame-aligned point error (FAPE), supervised $\chi$ loss, violation loss, RMSD, distogram loss, and pLDDT. We can briefly lay out what each of these terms seeks to do.

FAPE loss measures the average error of a predicted structure against a reference structure across all $n$ rotations and translations of the predicted structure that align one amino acid to its equivalent in the reference structure. At a high level, it measures the error in all other amino acids when we assume one amino acid is placed correctly, then averages across all possible assumptions of this type. Supervised $\chi$ loss measures the errors in predicted side-chain angles ($\chi$) for each amino acid. Violation loss penalizes large errors in standard energetic terms. For example, if there are large steric clashes or unusual bond lengths, violation loss increases. RMSD loss measures the same RMSD we defined in Section 2.3.4. Distogram loss measures the softmax distance between

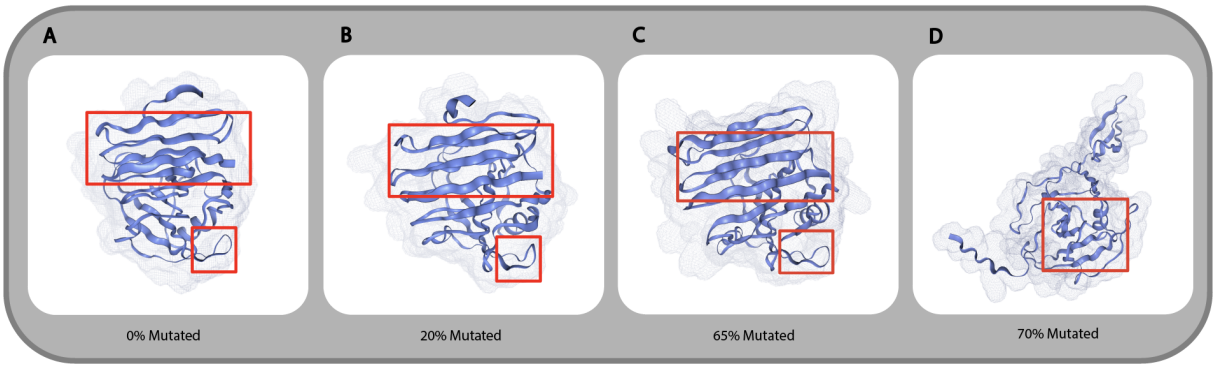



**Figure 3:** Selected structural changes of PDB identifier 6KWC [32] after successive independently random point mutations. **A** displays the structure of 6KWC with no mutation. Red highlights indicate key structural components to track through further mutation. **B** shows the structure of 6KWC with 20% of its amino acids mutated uniformly randomly. Highlighted structural elements remain intact. **C** shows the structure of 6KWC with 65% mutation. Highlighted structural elements continue to remain intact. **D** displays the structure of 6KWC with 70% mutation. Highlighted structural elements decompose, with some reminiscence in the new highlighted region.

the distogram predicted in one of the auxiliary structure module outputs and the actual set of inter-residue distances. pLDDT loss measures the difference between the predicted and true LDDT.

Note that the first four lost terms – FAPE, supervised $\chi$, and violation loss – all measure direct structural errors. Distogram and pLDDT loss, on the other hand, measure the accuracy with which AlphaFold predicts its own errors. When these types of confidence loss terms are backpropagated through the whole model, they encourage weights that yield confident, rather than accurate, predictions.

### 3.1.2 Model Pitfalls

Our look into the AlphaFold architecture highlights its general dependence on homology. The functional input to the deep learning component of the model is the MSA, which is a purely evolutionary, rather than physical, notion. We hypothesize, as a result, that AlphaFold may not learn much about the physics and chemistry behind protein folding. To begin to test such hypotheses, we installed an open-source version of AlphaFold – OpenFold – and began to run predictions [31]. One concession we had to make was using a slightly more limited dataset for running MSAs, as we did not have four terabytes of storage readily available; however, this issue does not diminish the power of our results. We often refer to AlphaFold and OpenFold interchangeably, as we use the OpenFold codebase for all implementation, but AlphaFold is the primary inspiration for their work.

To demonstrate the evolutionary issue we conjecture, we performed the experiment visualized in Figure 3. Specifically, we successively chose some unmutated position from the original sequence of a protein with PDB identifier 6KWC [32] and swapped it with a randomly selected amino acid. This procedure produced a set of $n$ amino acid sequences, each of length $n$, where $n$ is the original length of 6KWC. We then ran OpenFold to infer the structure associated with each sequence in this set.

From a physical point of view, we expect that this sequence of perturbations would lead to a relatively smooth decomposition of the original structure – each amino acid replaced would moderately perturb its surroundings until the protein is unrecognizable. Consequently, while singular

point mutations would have a minimal effect on structure, larger random sets of mutations would lead to large changes in structure.

However, AlphaFold predicts that the structure should remain almost identical until we mutate 65% of protein, as in Figure 3. Once we mutate one or two more amino acids, AlphaFold's prediction quickly changes shape into disarray. We additionally found that this momentary destruction of an otherwise continuous pattern is driven by the loss of any significant sequence alignment in the MSA.

This result implies that AlphaFold does not physically respond to point mutations – it would be highly unphysical to observe that 65% mutation in an amino acid sequence leaves its structure largely intact. However, AlphaFold's behavior is clearly explained by homology: the moment there is no longer a subsequence in a perturbed intermediate that closely matches the true sequence in the MSA's sequence database, it can no longer produce a reasonable structure. Up to that point, it predicts almost exactly the structure implied by the homology of the last remaining subsequences untouched by perturbation – a rather unreasonable assumption, given there are many other important subsequences in the new perturbed sequence.

With this result in mind, we concluded that AlphaFold likely depends too significantly on homology-based relationships, specifically to predict the structures of sequences that may be far from any existing experimental sequence but have some small biological subsequence. In other words, we observe that AlphaFold's predisposition to act like an interpolation engine, as we initially introduced, has some consequences despite its many resounding successes.

## 3.2 Operating on the Embedding Space

To help alleviate some of these concerns, we designed our SRO model to operate on the embedding space of AlphaFold to relax its dependence on the MSA. In particular, this model transforms the pair embedding $Z$ of some intermediate Evoformer block into a new embedding $\tilde{Z}$ on the same space that, in theory, will produce a more reasonable protein when run through the structure module. Such a model would help break the reliance on the MSA throughout the standard Evoformer stack. Additionally, this type of model would give credibility to the idea that we can apply physical operations through training a refinement on the embedding space; in other words, we would show that the embedding space is *operable* in parallel with some real chemical operation. We begin by discussing our generation of a dataset to train such a model.

### 3.2.1 Model Objective

Suppose we retain the notation $f_s(\cdot) : \mathbb{R}^{n\times e_s} \times \mathbb{R}^{n\times n\times e_p} \to \mathbb{R}^{3n_a}$ for denoting the application of the structure module to some $(V, Z)$ pair to produce structural coordinates $q$. We can let our model be denoted by $f_r(\cdot) : \mathbb{R}^{n\times n\times e_p} \to \mathbb{R}^{n\times n\times e_p}$. Then, our goal is to train $f_r(\cdot)$ on some input embeddings $(V, Z)$ such that $f_s(V, f_r(Z))$ is some physical transformation of $f_s(V, Z)$. We chose not to modify $V$, as there is no clear way to reincorporate what we would learn about $V$ into $M$, which would be necessary to continue running the result of our model through successive Evoformer blocks.

To train such a model, we need pairs of embeddings $(V, Z)$ and physically operated structures $\tilde{q}$ that result from some transformation of $q = f_s(V, Z)$. The particular scheme we use to generate such a pair is visualized by the section of Figure 4 below the diagonal. In particular, we run AlphaFold on a given input sequence, save some intermediate embedding $(V, Z)$ from intermediate Evoformer blocks, produce $f_s(V, Z)$ using the structure module, and then generate $\tilde{q}$ using molecular dynamics after solvation.

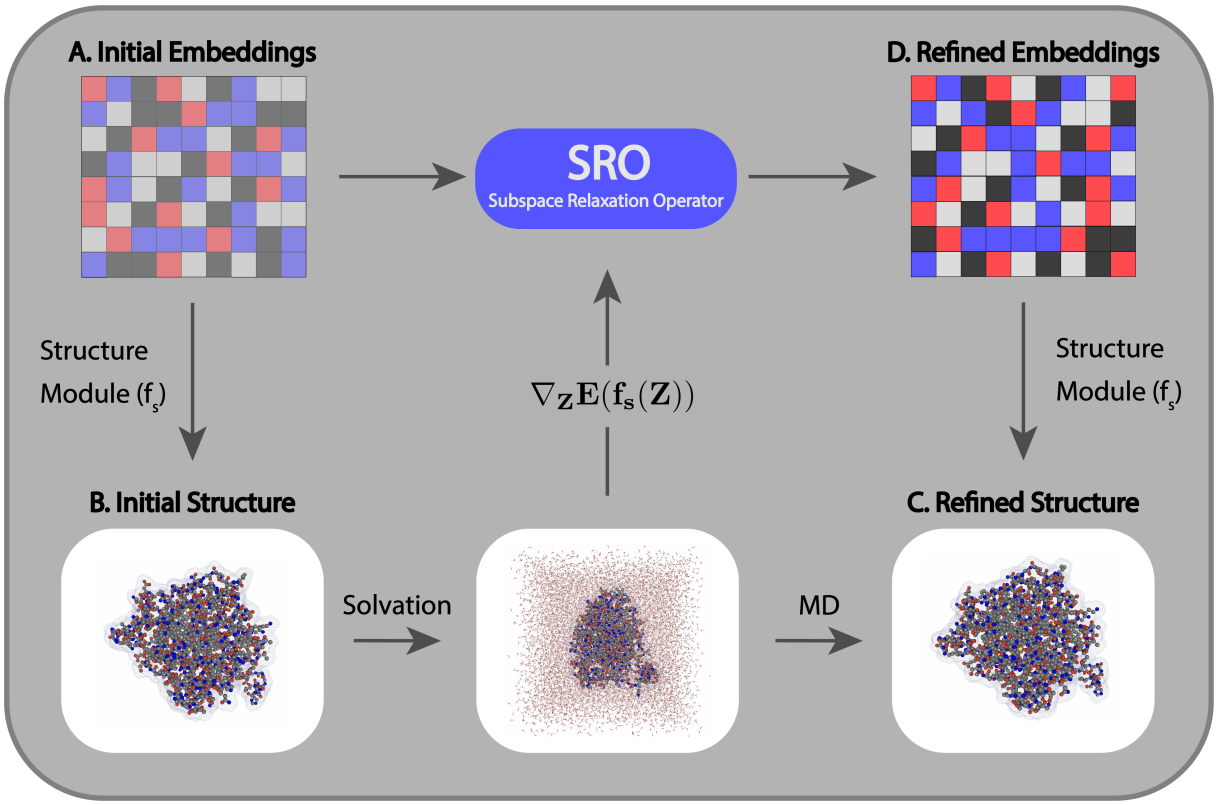



**Figure 4:** Model objective schematic for the Subspace Relaxation Operator (SRO). **A** represents the original pair embedding $Z$ that SRO seeks to refine. **B** shows $f_s(V, Z)$ – the structure associated with the original embedding. **C** shows a molecular dynamics structure generated from $f_s(V, Z)$ that is lower in energy than the original prediction according to our usual Amber force field. **D** shows the final embedding $\tilde{Z}$ produced by our SRO model, trained with the goal of reproducing the lower energy structure by way of $f_s(V, \tilde{Z})$.

To train our model on data of this variety, we run the SRO component of our model to produce $(V, \tilde{Z})$. We then compute $f_s(V, \tilde{Z})$ and measure loss with weights similar to those used by OpenFold during finetuning. We then backpropagate this loss through the structure module $f_s$ to $(V, \tilde{Z})$, finally using these gradients to update the weights of the SRO. Figure 4 illustrates this process above the main diagonal – we produce data by moving from initial embeddings to predicted structures to simulated structures, and train this process in the embedding space by moving from initial to refined embeddings.

### 3.2.2 Dataset Generation

The first major step in training our SRO towards this objective was generating the dataset. We began by taking 5000 random monomers from the PDB70 dataset and running OpenFold inference on each one [33]. From there, we randomly sampled 5 intermediate embeddings $(V_i, Z_i)$ for each monomer and saved $q_i = f_s(V_i, Z_i)$. We then protonated each $q_i$ to pH 5 using PDB2PQR and PROPKA [19, 20] as in our combinatorial work, followed by explicit solvation using GROMACS [21]. Next, we ran molecular dynamics using our usual Amber force field in 2fs time increments for 30000 steps at 300K, amounting to a short 60ps simulation. We extracted the resulting structures $\tilde{q}_i$, which left us with $5000 \cdot 5 = 25000$ embedding-structure pairs.

The two main highlights of this procedure are our choices to simulate for such a short time of only 60 picoseconds and to protonate the protein to pH 5 rather than the standard condition of pH 7.

We decided to limit simulation time primarily to examine the capacity of our model under small physical operations before jumping to more complex approximations. The longer we run MD, the less correlated our final structure will be with our original input. This could yield a noisier transformation in the embedding space, potentially making it harder to train a generalizable model. Before making a more complex dataset, we wanted to first see whether the embedding space was operable in a cleaner setting.

The decision to protonate to pH 5 was to showcase one potential application for the SRO: to modify embeddings to adhere to some particular set of solvent conditions. In other words, by training on pH 5 data, we hoped that the SRO would learn to convert some standard prediction to a more energetically favorable state with respect to a pH 5 solution. The same principle could be extended to arbitrary cases, such as examining what happens to pepsin, a critical digestive protein, at pH 2 in the human gastrointestinal system – all we would need to do is generate a new pH 2 dataset! This idea is one of the particular benefits of the SRO. Because we use MD to generate the ground truth structures, we can produce a set of ground truth structures to fit any solvent conditions. These new structures would produce a trained SRO particularly attuned to these desired solvent conditions.

### 3.2.3 Model Architecture and Training

With our 25000 generated $(V, Z) - \tilde{q}$ pairs, we began training our model, which proceeds according to Algorithm 7.

**Algorithm 7** Subspace Relaxation Operator (SRO)

**Require:** $V \in \mathbb{R}^{n \times e_s}, Z \in \mathbb{R}^{n \times n \times e_p}$

$q \leftarrow f_s(V, Z)$

Compute the forces $\nabla_q E(q)$ under the desired solvent conditions according to our usual Amber force field

Compute $\nabla_Z E(q) = \nabla_q E(q) \cdot \nabla_Z q = \nabla_q E(q) \cdot \nabla_Z f_s(Z)$

$\beta \leftarrow \mathsf{Lin}(\mathsf{LayerNorm}(\nabla_Z E(q))), \gamma \leftarrow \mathsf{Lin}(\mathsf{LayerNorm}(\nabla_Z E(q)))$

$\Delta Z \leftarrow \gamma \odot Z + \beta$

$\Delta Z \leftarrow \Delta Z + \mathsf{TriangleMulOutgoing}(\Delta Z)$

$\Delta Z \leftarrow \Delta Z + \mathsf{TriangleMulIncoming}(\Delta Z)$

$\Delta Z \leftarrow \Delta Z + \mathsf{TriangleAttentionStart}(\Delta Z)$

$\Delta Z \leftarrow \Delta Z + \mathsf{TriangleAttentionEnd}(\Delta Z)$

$\Delta Z \leftarrow \mathsf{PairTransition}(\Delta Z)$

**return** $\tilde{Z} = Z + \Delta Z$

At a high level, our model begins by initializing a residual on $Z$ by applying standard FiLM conditioning [34] to the gradients of some energy function with respect to $Z$, as propagated through the structure module. Notice that since $\nabla_q E(q) = f_q$ where $f_q$ are the three dimensional forces on each atom in $q$, we can compute these easily using a force field. We can then backpropagate them through the structure module by setting the gradients of its final output to be $f_q$. We often call this step "force conditioning," as we condition our predicted residual on actual atomic forces. The

model can be trained without force conditioning, as well, by starting with $\Delta Z = Z$ in training and inference.

From here, the model applies the logic of a standard Evoformer block to this initialized residual to predict the change we need to make to $Z$ to produce $\tilde{q}$. By initializing the Evoformer-like components to the same weights as one of the Evoformer blocks in OpenFold's standard model, we seek to ensure the residual we predict will be correlated with the embedding space we are hoping to modify.

For training purposes, once we have the predicted residual, we evaluate its loss by computing $f_s(V, Z + \Delta Z)$ and measuring the standard forms of loss used in AlphaFold. To apply updates according to the loss, we backpropagate the loss through the structure module to $Z + \Delta Z$ and use these gradients to continue backpropagating through the SRO.

We implemented this model in PyTorch [35] and trained it using 18000 samples on the Brown Center for Computing and Visualization computing cluster using four NVIDIA GeForce RTX 3090 GPUs. We distributed training across all four [36], collecting gradients whenever applying a back-propagation update. We used batch size $b = 2$ with a variety of gradient accumulation strategies designed to produce larger effective batches. For example, after 30 batches, we collect all gradients across the four GPUs and average them to produce an effective batch size of $4 \cdot 30 \cdot 2 = 240$, roughly 1% of our overall dataset. We generally trained over 15 epochs, although we observed in our most updated models that performance peaks closer to two epochs, which is unsurprising for this type of finetuning task. Additional epochs, in particular, seem to result in overfitting to the training data with an increase in validation error.

One particular challenge of this implementation is the memory required to handle a batched matrix of shape $b \times n \times n \times e_p$; we initially hoped to try training a diffusion model, but the gradients on each of these matrices were too large for even a single-element batch.

### 3.2.4 Results

Thus far, we have trained the SRO under a number of different hyperparameter and training regimes. Our most notable results are presented in Figure 5.

One of our biggest challenges in training the SRO was the noisiness of the training, as seen in Figure 5. This challenge results primarily from the diversity of our dataset, over which the standard variation of OpenFold had a high variance in prediction success. We attempted to use gradient accumulation and gradient clipping to avoid this noisiness, but the inherent variance in OpenFold's predictions and embeddings across our dataset is present in all cases. Notice that the curves do, however, trend down – a good indication that our model is learning. With or without force conditioning, the model begins to fit the data.

However, when we examine whether the model generalizes to a validation set of proteins, we find that force conditioning greatly improves RMSD to our ground truth molecular dynamics structure, $\tilde{q}$. Without any information about the atomic forces on the initial structure predicted by the embedding, $q$, the SRO predicts a residual that corrupts any validation structure, reaching massive RMSD values of 19 Å. However, when it is trained with force conditioning, the SRO can achieve quite accurate changes of around 2 Å. This result is particularly fascinating – it implies that the SRO is learning something physical about the application of forces when we do condition the model, but that it is difficult to learn the information contained in these forces from only the AlphaFold embedding. A possible corollary is that the AlphaFold embeddings do not well describe physical quantities.

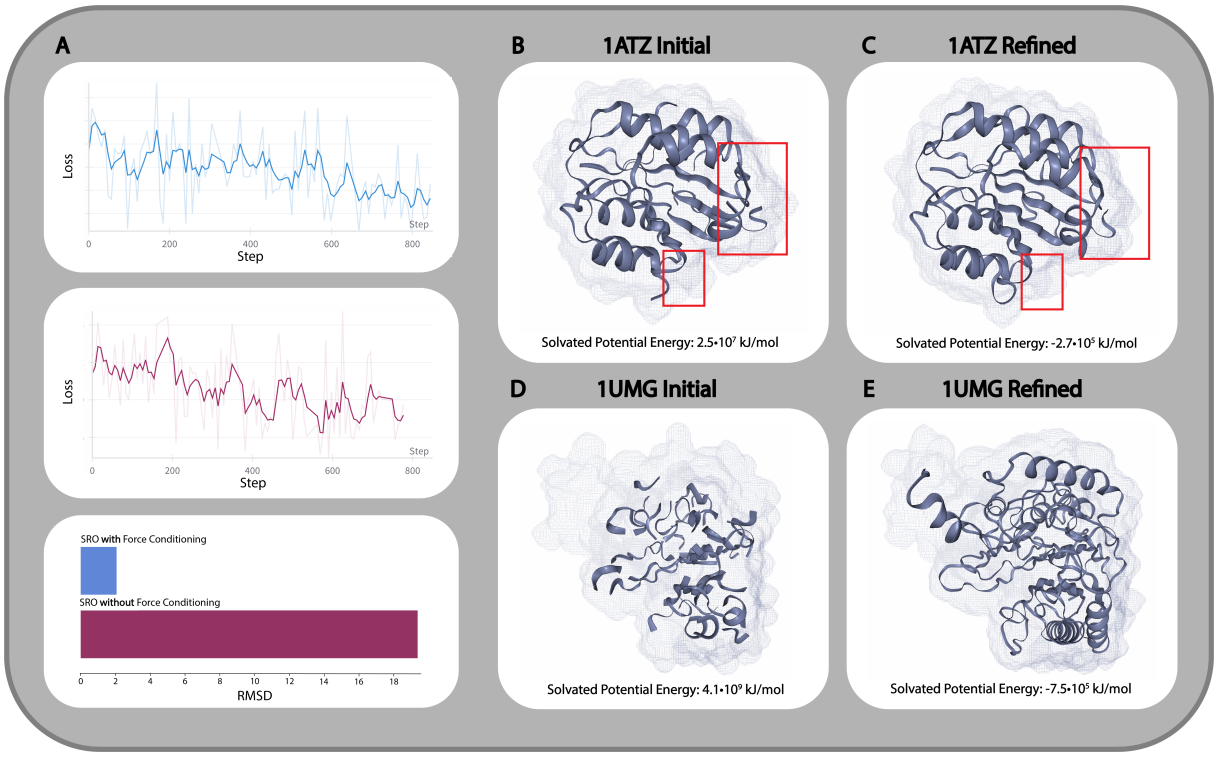



**Figure 5:** Results of training the SRO with and without force conditioning, with the respective weights initialized to the triangle components of the final Evoformer block in Openfold. **A** displays training loss curves with and without conditioning on the gradient of structure energy with respect to the input embedding, in addition to the best RMSD achieved on validation in the both cases [37]. **B** and **C** display the result of running the SRO with energy conditioning on PDB identifier 1ATZ [22], with red boxes to highlight changes in observable secondary structure. **D** and **E** show the same result for PDB identifier 1UMG. In all structural examples, the structure was relaxed using a brief energy minimization prior to visualization.

Figure 5 also presents two examples of the structure $q$ associated with an AlphaFold embedding and the structure $\hat{q}$ associated with the SRO refined embedding. In the first case (1ATZ), we can see that the relaxed structure predicted by the refined embedding contains secondary structure elements that the unrefined embedding does not yield – for example, the alpha helix in the larger highlighted region, and the loop in the smaller highlighted region [22]. In the second case (1UMG), we can see that the relaxed structure predicted by the refined embedding has a complete secondary structure; however, in the unrefined case, our visualization tool does not even recognize most of the backbone – a big improvement [38]. The potential energy of the refined structures was also substantially lower in both cases.

As a result, we conclude that the SRO is a promising model to improve the physicality of AlphaFold's Evoformer. In particular, we have shown that it is possible to operate on the Evoformer embedding space using approximate atomic forces to find corresponding lower-energy structural conformations.

## 3.3 Conclusion

In this section, we examined one approach to deep learning for protein structure prediction – AlphaFold, as open-sourced by OpenFold. We began by characterizing this style of approach as a two-step process: finding nearby sequences with structures in the training data and then implicitly interpolating. From here, we examined the specifics of the AlphaFold architecture, including the MSA, Evoformer, structure module, and loss terms. Most notably, we observed that the rather significant reliance on an input MSA is concerning, given that such a homology-based approach avoids taking into account any chemical or physical influences on folding.

We then tested this concern by examining how AlphaFold responds to mutations. We found that even when 65% of a protein sequence is mutated with randomly selected amino acids, AlphaFold predicts a highly similar structure compared to when it receives the original unmutated sequence. Such a result is highly unphysical – if more than half of a protein is completely substituted, it should have a different structure. However, it can be easily explained by a dependence on homology: only the unmutated components align to any protein sequence, such that AlphaFold is left with a similar set of inputs as if we had never mutated in the first place.

In an attempt to see if physical notions can be incorporated in the AlphaFold prediction pipeline, we devised the SRO. Our SRO refines intermediate embeddings such that they produce lower energy structures when run through the structure module. In order to refine some original embedding $Z$, the SRO begins by computing its associated structure $f_s(Z) = q$ through the structure module. Then, it computes the forces on $q$ using a force field under arbitrary solvent conditions. After backpropagating these forces to the embedding level, the SRO uses the resulting information to produce a residual $\Delta Z$ such that $Z + \Delta Z$ gives a new, refined embedding.

We set up training for the forward part of the SRO by generating a dataset of embedding-structure pairs. For some embedding $Z$, its associated structure $\tilde{q}$ was the result of running a short MD trajectory on $f_s(Z) = q$. We used the resulting dataset to train by using the SRO to compute $Z + \Delta Z$, computing $f_s(Z + \Delta Z)$, measuring loss against $\tilde{q}$, and backpropagating, keeping the structure module weights untouched. In principle, this process trains the SRO to predict embeddings that associate with some more advanced, lower energy structure for use in later Evoformer blocks.

One nice flexibility of our model is that this MD trajectory can operate with arbitrary solvent conditions. By generating a dataset according to some particular conditions and then using those conditions to compute the force gradients, the SRO can target embeddings with low energy in any specific setting.

After training the SRO under several different hyperparameter and architectural regimes, we found that we could produce lower energy, refined embeddings at pH 5. While the training was noisy, we also found that the calculation of energy gradients at the start of the SRO is a necessary step for performance – a fascinating result which implies the SRO is learning physical properties.

As we continue to develop the SRO after the conclusion of this document, we will try a number of new concepts. First, we will train under a number of different solvation regimes. For example, we may examine an array of different pH values or a set of different solvents, such as a water-ethanol mix. Additionally, we will make a larger sweep over loss weights. Finally, we will examine what happens to a full-stack AlphaFold prediction when we plug the SRO into different subsets of Evoformer blocks.

Our work with the AlphaFold embedding space highlights the potential to unify physics and homology-based deep learning approaches for protein folding. We believe that this principle may

help alleviate some of the major concerns around AlphaFold's ability to generalize in cases where it relies too much on evolutionary information.

# 4 Conclusion

In Chapter 1, we explored the possibility of using molecular dynamics to improve the results of combinatorial protein folding approaches, specifically in the HP model. We began by examining the dependence of combinatorial protein folding on hydrophobicity scales, proposing the idea of structure-informed hydrophobicity in the process. We then presented a novel structure-informed hydrophobicity scale called WaterFire – the first of its kind. In trying to generalize WaterFire to characterize amino acid pairs rather than single amino acids, we found a new NP-completeness result: the problem of reducing a contact potential matrix to the HP model is NP-complete. Continuing with our original design of WaterFire, we proposed an iterative algorithm to fold in the HP model using structure-informed scales. We then tested this algorithm on real protein sequences.

In Chapter 2, we examined AlphaFold, the pioneering deep learning model for protein structure prediction. We began by unpacking its architecture, noting its reliance on homology to produce folds. We then showed an instance where AlphaFold's reliance on homology leads to unphysical predictions. In order to remedy this issue, we proposed the SRO to modify the embedding space of AlphaFold towards more physical structures. We then generated a dataset of embeddings and structures using molecular dynamics to train this model. During training, we found that the SRO requires physical information – namely, the atomic forces of the structure associated with the input embedding – to appropriately modify the embedding space. Once we finished training the model, we found that the refined embeddings produced by the SRO correlate with lower energy structural outputs.

These two chapters highlight the potential for molecular dynamics to improve protein folding in structure prediction models that inherently rely less on molecular chemistry and physics. In the combinatorial case, we found that using molecular dynamics to modify hydrophobicity scales leads to mixed results. Incorporating more complex physical information leads to NP-completeness challenges, but even the simple physical information that we leverage efficiently in WaterFire is limited by the secondary structure imposed by lattice folding. However, we were still able to refine HP model structures by including the physical information contained in WaterFire. In the deep learning case, we found that generating data using even short molecular dynamics trajectories can improve the physicality of AlphaFold's embeddings – an observation we believe warrants further investigation.

Protein folding is of the utmost importance to drug discovery, given that it provides invaluable priors to the drug discovery process. However, it requires us to find a minimum of a nonconvex, high-dimensional function whose form we do not even know with high accuracy. By stitching together the successes in the field like HP model approximations and AlphaFold with physical information, we can get one step closer to making better approximations of these highly elusive minima.

# 5 Acknowledgments

At the outset of this section, I should note that I will focus on acknowledging those who worked with me in a research capacity during my time at Brown, but many other members of the Brown

community, including my family, have been instrumental to this work.

When I arrived in Sorin's classroom way back in 2022, I remember my incredible excitement to find a course that tied together my longest standing passions – biology and chemistry – with my newfound love for computer science. My enthusiasm could not have been more appropriately placed. After three semesters in his classroom and another three as a teaching assistant, I can easily say that my experience working with Sorin has shaped my college years. Much of this work can be attributed to multiple meetings with him per week, countless presentations at his whiteboard (well thought out or not), and many hours of hearing his advice, both scientific and general. I owe a lot to Sorin, not just because of the table I broke in his office or the many chocolates and books I have taken from him over the years, but because of the wisdom and curiosity he has imparted on me.

My early days in Sorin's classroom drove me to Ritambhara's door in the spring of 2023. The following summer was my first true research experience. Ritambhara's help allowed me to navigate not only the math underlying our optimal transport tool but also what it means to do research, inside or outside of computer science. Her excitement about using computer science to tackle biological problems was infectious and a major motivating factor for me to continue doing research over the past two years.

The one thing I felt was missing from my time at Brown after that summer was a return to chemistry – a passion of mine in the time preceding my arrival at Brown. A few words from a friend of mine in the chemical engineering department drove me to another new door – Andrew Peterson's. Andy taught me much of what I know about computational chemistry and helped me expand my comfort zone to a completely new field. By convincing me to work on GPAW, take his course in the fall, and continue learning chemistry in his research group, Andy made it effortless for me to return to one of my favorite subjects. It was in his classroom that I thought to center my thesis around molecular dynamics – a choice I'm glad I made.

The last piece of the puzzle to my academic career at Brown was meeting Pranav. Prior to meeting Andy, I was a teaching assistant with Pranav for Sorin's fall course, but only connected with him as a research collaborator last spring. We immediately started feeding off of each other and have continued to do so throughout the past year, throughout many natural ups and downs. We have learned so much together over the past year – from engineering a massive dataset on a computing cluster to developing an understanding of basic molecular physics from scratch, we have covered a significant amount of ground. Maybe most importantly, we have learned to push through many moments of doubt by relying on each other. If working on this thesis had net me nothing more than the chance to collaborate with Pranav, I would already have been as lucky as I could hope.

# References

[1] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[2] K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[3] S. Istrail and F. Lam. Combinatorial algorithms for protein folding in lattice models: A survey of mathematical results. *Communications in Information and Systems*, 9(4):303–346, 2009.

[4] J. W. Ponder and D. A. Case. Force fields for protein simulations. In *Protein Simulations*, volume 66 of *Advances in Protein Chemistry*, pages 27–85. Academic Press, 2003.

[5] P. Eastman et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, 13(7):e1005659, 2017.

[6] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling. Comparison of multiple AMBER force fields and development of improved protein backbone parameters. *Proteins: Structure, Function, and Bioinformatics*, 65(3):712–725, 2006.

[7] J. L. Cornette, K. B. Cease, H. Margalit, J. L. Spouge, J. A. Berzofsky, and C. DeLisi. Hydrophobicity scales and computational techniques for detecting amphipathic structures in proteins. *Journal of Molecular Biology*, 195(3):659–685, 1987.

[8] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. *Journal of Computational Biology*, 5(3):423–465, 1998.

[9] W. E. Hart and S. Istrail. Lattice and off-lattice side chain models of protein folding: Linear time structure prediction better than 86% of optimal. *Journal of Computational Biology*, 4(3):241–259, 1997.

[10] Y. Nozaki and C. Tanford. The solubility of amino acids and two glycine peptides in aqueous ethanol and dioxane solutions: Establishment of a hydrophobicity scale. *Journal of Biological Chemistry*, 246(7):2211–2217, 1971.

[11] J. Fauchere and V. Pliska. Hydrophobic parameters ii of amino acid side-chains from the partitioning of n-acetyl-amino acid amides. *Eur. J. Med. Chem.*, 18, 01 1983.

[12] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157(1):105–132, 1982.

[13] V. Wolfenden R., P. M. Cullis, and C. C. F. Southgate. Water, protein folding, and the genetic code. *Science*, 206(4418):575–577, 1979.

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

[15] D. H. Wertz and H. A. Scheraga. Influence of water on protein structure. an analysis of the preferences of amino acid residues for the inside or outside and for specific conformations in a protein molecule. *Macromolecules*, 11(1):9–20, 1978.

[16] D. Chandler. Interfaces and the driving force of hydrophobic assembly. *Nature*, 437(7059):640–647, 2005.

[17] R. Godawat, S. N. Jamadagni, and S. Garde. Characterizing hydrophobicity of interfaces by using cavity formation, solute binding, and water correlations. *Proceedings of the National Academy of Sciences of the United States of America*, 106(36):15119–15124, 2009. Epub 2009 Aug 25.

[18] J. Monroe, M. Barry, A. DeStefano, P. A. Gokturk, S. Jiao, D. Robinson-Brown, T. Webber, E. J. Crumlin, S. Han, and M. S. Shell. Water structure and properties at hydrophilic and hydrophobic surfaces. *Annual Review of Chemical and Biomolecular Engineering*, 11:523–557, 2020.

[19] T. J. Dolinsky, J. E. Nielsen, J. A. McCammon, and N. A. Baker. Pdb2pqr: an automated pipeline for the setup of poisson-boltzmann electrostatics calculations. *Nucleic Acids Research*, 32(suppl_2):W665–W667, 2004.

[20] M. H. M. Olsson, C. R. Søndergaard, M. Rostkowski, and J. H. Jensen. Propka3: Consistent treatment of internal and surface residues in empirical pka predictions. *Journal of Chemical Theory and Computation*, 7(6):525–537, 2011.

[21] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1–2:19–25, 2015.

[22] E. G. Huizinga, R. M. van der Plas, J. Kroon, J. J. Sixma, and P. Gros. Crystal structure of the a3 domain of human von willebrand factor: implications for collagen binding. *Structure*, 5(9):1147–1156, 1997.

[23] A. S. Rose, A. R. Bradley, Y. Valasatava, J. M. Duarte, A. Prlić, and P. W. Rose. NGL Viewer: Web-based molecular graphics for large complexes. *Bioinformatics*, 34(21):3755–3758, 2018.

[24] S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3):534–552, 03 1985.

[25] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[26] R. Agarwala, S. Batzoglou, V. Dančík, S. E. Decatur, M. Farach, S. Hannenhalli, and S. Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the hp model. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 134–143. ACM, 1997.

[27] P. Rotkiewicz and J. Skolnick. Pulchra—a fast and efficient method for reconstructing full-atom protein models from reduced representations. *BMC Structural Biology*, 8(1):80, 2008.

[28] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[29] M. Nowakowski, L. Jaremko, B. Wladyka, G. Dubin, A. Ejchart, and P. Mak. Spatial attributes of the four-helix bundle group of bacteriocins — the high-resolution structure of *bacsp222* in solution. *International Journal of Biological Macromolecules*, 107:2715–2724, 2018. PDB ID: 5LWC.

[30] M. Remmert, A. Biegert, A. Hauser, and J. Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature Methods*, 9(2):173–175, 2011.

[31] G. Ahdritz et al. Openfold: retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*, 21:1514–1524, 2024.

[32] Z. Li, X. Zhang, C. Li, A. Kovalevsky, and Q. Wan. Studying the role of a single mutation of a family 11 glycoside hydrolase using high-resolution x-ray crystallography. *The Protein Journal*, 39:671–680, 2020.

[33] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[34] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[35] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library, 2019.

[36] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

[37] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[38] H. Nishimasu, S. Fushinobu, H. Shoun, and T. Wakagi. The first crystal structure of the novel class of fructose-1, 6-bisphosphatase present in thermophilic archaea. *Structure*, 12(6):949–959, 2004.