Protofold II:
Enhanced Model and Implementation for Kinetostatic Protein Folding*

Pouya Tavousi, Morad Behandish, Horea T. Ilieş, and Kazem Kazerounian

A reliable prediction of 3D protein structures from sequence data remains a big challenge due to both theoretical and computational difficulties. We have previously shown that our kinetostatic compliance method (KCM) implemented into the Protofold package can overcome some of the key difficulties faced by other *de novo* structure prediction methods, such as the very small time steps required by the molecular dynamics (MD) approaches or the very large number of samples needed by the Monte Carlo (MC) sampling techniques. In this article, we improve the free energy formulation used in Protofold by including the typically underrated entropic effects, imparted due to differences in hydrophobicity of the chemical groups, which dominate the folding of most water-soluble proteins. In addition to the model enhancement, we revisit the numerical implementation by redesigning the algorithms and introducing efficient data structures that reduce the expected complexity from quadratic to linear. Moreover, we develop and optimize parallel implementations of the algorithms on both central and graphics processing units (CPU/GPU) achieving speed-ups up to two orders of magnitude on the GPU. Our simulations are consistent with the general behavior observed in the folding process in aqueous solvent, confirming the effectiveness of model improvements. We report on the folding process at multiple levels; namely, the formation of secondary structural elements and tertiary interactions between secondary elements or across larger domains. We also observe significant enhancements in running times that make the folding simulation tractable for large molecules.

1 Introduction

Proteins are large biomolecules that are responsible for a vast array of biological functions inside the cell, and appear in the form of enzymes, antibodies, motor proteins, transport proteins, etc. [3]. The function of a protein strongly depends on its 3D structure (i.e., 'conformation') which in turn can be directly determined from the linear sequence of amino acids (AAs) linked together to form the protein chain (i.e., 'configuration') [4].[1] Therefore, the computer-aided prediction of the folded structure of a protein from the knowledge of its sequence (referred to as 'protein folding') is the key to understanding many biological processes in the cell. This knowledge is crucial toward the ultimate goal of modeling proper function or malfunction at molecular and cellular level (e.g., deadly diseases such as cancer, Alzheimer's, Parkinson's, etc.) and is central to a variety of bioengineering applications including 'protein engineering' [5, 6].

1.1 Related Work

There are several different computational approaches for protein folding prediction, ranging from knowledge-based techniques to methods starting from physical principles [7].

Knowledge-Based Methods. The knowledge-based approaches predict the structure of a given protein using the information extracted from previously determined structures and known types of folds. They are generally more reliable than physics-based methods, but have limited applicability in predicting new types of folds. Examples of knowledge-based techniques are homology or comparative modeling [8–10] and fold recognition or threading [11–13]. We refer the reader to [7] for a comprehensive review of such methods.

Physics-Based Methods. On the other hand, methods that utilize models formulated empirically or obtained from physical principles are less reliable and more time-consuming, but apply to a wider range of folding simulations [7]. These methods range from *de novo* [14, 15] to *ab initio* [16, 17] prediction techniques. Here we briefly review some of the common *ab initio* approaches, namely sampling methods and MD simulations [7].

Sampling methods generate numerous samples in the conformation space, followed by an evaluation of their free energies. Different search algorithms are used to find the unchallenged global minimum of the free energy, assumed to be associated with the native structure according to the 'thermodynamic hypothesis' [4]. These search methods include simulated annealing [18, 19], basin hopping [20–23], evolutionary algorithms [24–26] and MC simulation with biased moves [27–29]. A review of conformation sampling methods for protein folding can be found in [30]. Sampling methods have two major limitations, namely 1) they do not provide any information about the biological pathway; and 2) finding the global minimum is not guaranteed because of the finite number of samples.

*This article builds on two shorter conference papers presented at the ASME IDETC/CIE'2013 [1, 2].

[1] In the robotics literature, the term configuration is typically used to describe the complete set of kinematic variables. However, the term conformation is typically used for that purpose in molecular biology.

On the other hand, MD approaches simulate the biological pathway using a model built upon physical principles. Standard MD techniques include Newtonian dynamics [31–34], Langevin dynamics [35–38] and Brownian dynamics [39–42]. A review of MD simulation methods for protein folding is provided in [43]. In order to keep the numerical algorithms stable, very small time steps (in the order of femtoseconds) along the simulation trajectory are required, which does not support folding simulation of typical proteins that span milliseconds except for small molecules [43].

Kinetostatic Compliance Method. The KCM was introduced in [44, 45] to overcome some of the key challenges in the aforementioned approaches. In this method, implemented in the software package Protofold [46–48], the protein chain is modeled as a kinematic linkage which complies under the kinetostatic effect of the force-field obtained form intramolecular interactions between the atoms. The key contributions of KCM were

1. modeling the constrained kinematics of the protein chain with significantly fewer degrees of freedom (DOF) than, for example, those of the 'beads and springs' model used in many MD methods; and

2. converging faster to the minimum energy state by using kinetostatic (i.e., 1st-order) variations rather than dynamic (i.e., 2nd-order) response.

In KCM, each rotatable joint, used to model the constrained motion of the chemical bonds, changes by an amount proportional to the effective torque on that joint. It was shown that KCM is a faster and more stable alternative to the traditional dynamic simulation techniques [48]. The Protofold platform has since provided a kinematic testbed for subsequent research activities. Examples are predicting hydrogen bond connectivity sub-graphs [49], its application to the design of stable peptide nano-particles [50], the analysis of protein mobility (using the 'pebble game' algorithm) [51], the development of mechanical models for secondary structural elements [52], and nano-machanism synthesis from a 'link soup' of pre-specified structural elements [53, 54].

In the earlier stages of the development, the energetics were limited to intramolecular interactions in the gas-phase of the protein—e.g., Coulombic and van der Waals forces exchanged between atoms of the protein itself, ignoring the interactions with solvent molecules. However, an important class of biologically significant proteins are water-soluble, whose folding process is predominantly driven by the interactions with the solvent, particularly the so-called 'hydrophobic effect'[2] which was missing from Protofold I [48].

[2]The hydrophobic effect is explained as the strong tendency of nonpolar sidechains to pack together to form a hydrophobic core protected from the solvent by a hydrophilic surface [3]. This effect is formulated in terms of entropic changes in the solvent molecules surrounding the protein surface.

Computing Solvation Effects. From a computational perspective, the solvation effects can be modeled in a number of different ways, broadly classified into 'explicit' and 'implicit' techniques.

The explicit methods use all-atom force field models such as SPC, SPC/E, TIP3P, TIP5P [55, 56], or coarse-grained (CG) models [57,58] which are less structured representations of the solvent obtained by mapping two or more particles onto a single interaction site [55]. A prohibitive computational cost is associated with the large number of solvent molecules required to model a bulk solution.

Alternatively, approximate schemes that include the solvent effects implicitly can provide useful quantitative estimates, yet remain computationally inexpensive [59]. The implicit models estimate the contribution of each solvent-exposed atom to the solvation free energy using empirical formulae, most commonly expressed as a linear function of the solvent accessible surface area (SASA) [60]. An exact computation of SASA requires obtaining the surface area of the envelope of overlapping spheres [61], which is computationally expensive. Alternatively, approximate formulations have been developed to efficiently predict the *expected* (i.e., probabilistic average) SASA based on simplifying assumptions on the distribution of the coordinates of atoms (or groups of atoms) in the 3D space. For instance, CHARMM [62] uses the probabilistic approach from [63], which estimates the SASA as a function of the distances between pairs of atoms or residues. A similar model with similar parametrization [64] was used in GROMOS [65], a recent improvement of which was given in [66]. AMBER [67, 68] uses a fast linear combination of pairwise overlaps (LCPO) algorithm [69], which improves the method in [63] by adding more terms to the approximation. Although being widely popular in well-known MD packages, these methods rely on simplifying assumptions that compromise accuracy. For example, the method in [63] assumes a uniform random spatial distribution of atoms or residues, which introduces bias into the simulation results.

The inclusion of the solvation free energy computed using an adequately accurate evaluation of the SASA results in a more realistic energy- and force-model for simulating the natural behavior of water-soluble proteins.

1.2 Outline & Contributions

In Section 2, we introduce an improved free energy model making use of the linear implicit model given in [60] to compute the solvation free energy- and force-field from a knowledge of the SASA and its gradient for individual atoms at a given 3D conformation. We develop a simple offset surface enumeration technique that can approximate the SASA and its gradient up to any desired accuracy. Our method is significantly more accurate than the probabilistic methods such as those given in [63, 69] yet noteably faster than the exact method given in [61], while the trade-off between speed and accuracy can be adjusted by a proper choice of the enumeration (i.e., surface sampling) density.

A second major contribution of this work is to develop significantly more efficient algorithms and data structures in Section 3 to speed up the computations, and to implement them into Protofold II:

1. We use a 3D hash table data structure based on a uniform spatial grid that supports fast queries to identify pairs of proximal atoms. This helps speeding up the computations by eliminating negligibly small interactions associated with pairs of atoms that are farther than a so-called 'cut-off distance'.

2. We use a tree-based data structure to span the protein chain efficiently for the purpose of characterizing interaction types between pairs of atoms based on their distances along the topological structure of the chain.

3. We develop sequential and parallel surface enumeration algorithms to compute the SASA and its gradient for individual atoms needed for the solvation energy and force computations, respectively, up to desired accuracy.

4. We employ prefix sums [70] to compute the joint torques on the kinematic linkage of the protein chain, given the resultant forces on the individual atoms.

As a result, the numerical complexity for each KCM cycle, including the computation of resultant forces on the atoms and the links (except those resulting from solvation effects) and their conversion to joint torques, is reduced from $O(n^2)$ in Protofold I [48] to $O(n)$ in Protofold II, where n is the number of atoms in the protein molecule.[3]

The SASA evaluations for solvation force computations in our model turns out to be the bottleneck to the entire simulation—up to several orders of magnitude slower than the electrostatic and van der Waals force computations. Fortunately, the surface enumeration algorithm lends itself well to high-throughput data parallelism. In Section 4 we first present the CPU-parallel implementation using OpenMP, leading to moderate speed-up factors (up to one order of magnitude). To leverage the massive processing power offered by the single-instruction multiple-thread (SIMT) architecture of the modern graphics hardware—onto which our data-parallel SASA enumeration algorithm maps perfectly—we present a GPU-parallel implementation and its optimization. The implementation takes advantage of the device memory hierarchy and hiding memory access latencies, in turn leading to larger speed-ups (up to two orders of magnitude).

2 Formulation

Section 2.1 starts with an overview of the underlying kinematic principles of the KCM simulation first introduced in [44–48]. The protein chain is modeled as an open kinematic linkage with reduced DOF in terms of dihedral and rotamer angles, which complies under the effect of interatomic and solvation forces. Next, the energy- and force-field formulation used in Protofold II is described in Section 2.2, with special emphasis on the newly introduced solvation effects. Lastly, the KCM optimization process is presented in Section 2.3.

2.1 Kinematic Model

Proteins are long polymeric chains made of AAs, which exist in only 20 different types (except for few rare exceptions), joined together as a linear polypeptide chain [3], structural details of which are summarized in Appendix A. Here we restrict ourselves to the kinematic representation of the chain's conformation within the scope of KCM.

Linkage Parameterization. Figure 1 schematically illustrates the repetitive sequence of $-\mathrm{N}-\mathrm{C}_\alpha-\mathrm{C}-$ atoms,[4] called the 'backbone' or the 'main chain', with 'side chains' resembling branches that extend out of it. As explained in Appendix A, the backbone conformation can be specified to an adequate accuracy by two sets of dihedral angles; namely,

- $-180^\circ \le \phi_i < +180^\circ$ (around $\mathrm{N}-\mathrm{C}_\alpha$ in AA_i); and
- $-180^\circ \le \psi_i < +180^\circ$ (around $\mathrm{C}_\alpha-\mathrm{C}$ in AA_i);

for $1 \le i \le m$, where m is the number of AA residues along the chain. The conformation of each side chain, on the other hand, can be specified by up to 4 extra angles $-180^\circ \le \chi_{i,k} < +180^\circ$ for $1 \le k \le l_i$ where $0 \le l_i \le 4$ is the number of side chain links of AA_i, and the subscript k corresponds to the bonds numbered in the obvious order along the side chain C and N atoms.

To set a reference for the angle measurements, the zero-reference position description (ZRPD) method [71] is used. The zero-position (ZP) for the protein chain is defined as the conformation of the serial linkage in which all peptide planes are coplanar (i.e., $\phi_i^0 = \psi_i^0 = -180^\circ$) and side chain dihedrals are set to default low energy values identified as 'rotamers' [45].

To unify the notations, all angular variables are denoted by $\theta_{j,k}$ $(1 \le j \le 2m, 0 \le k \le 4)$ where

$$\theta_{2i-1,0} = \phi_i + 180^\circ, \quad 1 \le i \le m, \tag{1}$$

$$\theta_{2i,0} = \psi_i + 180^\circ, \quad 1 \le i \le m, \tag{2}$$

$$\theta_{2i-1,k} = \chi_{i,k} - \chi_{i,k}^0, \quad 1 \le i \le m,\ 1 \le k \le l_i \le 4, \tag{3}$$

where $0^\circ \le \theta_{j,k} < 360^\circ$. The shifts in (1) and (2) by the intercept values of $\phi_i^0 = \psi_i^0 = -180^\circ$ and in (3) by the favorable rotamer angles $\chi_{i,k}^0$ ensure $\theta_{j,k} = 0^\circ$ at the ZP conformation.

A similar indexing scheme is used to identify the unit vectors along the rotation axes of revolute joints associated with these angles denoted by $\mathbf{u}_{j,k}$ $(1 \le j \le 2m, 0 \le k \le 4)$, i.e.,

- $\mathbf{u}_{2i-1,0}$ $(1 \le i \le m)$ is the unit vector along the bond between N of AA_i and C_α of AA_i;

[3] This is only the case under certain assumptions given in the subsequent sections, which are relatively reasonable for practical cases.

[4] Hereon, the notations C_α and C correspond to the alpha-carbon and carboxyl-carbon, respectively.

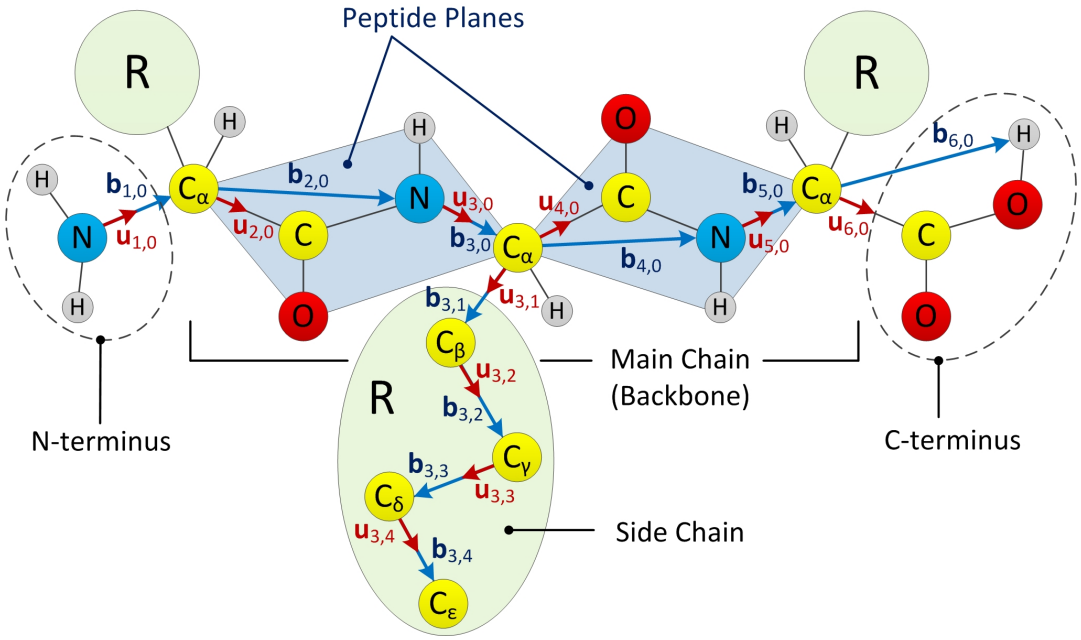

Figure 1: The polypeptide chain is modeled as a kinematic linkage, in which the peptide planes are assumed to be rigid.

- $\mathbf{u}_{2i,0}$ $(1 \leq i \leq m)$ is the unit vector along the bond between C_α of AA_i to the C of AA_i; and
- $\mathbf{u}_{2i-1,k}$ $(1 \leq i \leq m, 1 \leq k \leq 4)$ are the unit vectors along the successive side chain C and N atoms.

Thus the kinematics of the linkage—which abstracts the protein conformation—can be completely specified in terms of the rigid body rotation transformations obtained from these rotation angles and rotation axes.

The spatial orientation of the rigid peptide planes can be described conveniently with a pair of base vectors whose linear combination spans the peptide plane. The so-called 'body vectors' are denoted by $\mathbf{b}_{j,k}$ $(1 \leq j \leq 2m, 0 \leq k \leq 4)$, i.e.,

- $\mathbf{b}_{2i-1,0}$ $(1 \leq i \leq m)$ is the base vector that connects the N of AA_i to the C_α of AA_i;
- $\mathbf{b}_{2i,0}$ $(1 \leq i \leq m)$ is the base vector that connects the C_α of AA_i to the N of AA_{i+1}; and
- $\mathbf{b}_{2i-1,k}$ $(1 \leq i \leq m, 1 \leq k \leq 4)$ are the base vectors along the successive side chain C and N atoms.

The first two sets of vectors listed above are called the 'main chain body vectors'. Every vector in the peptide plane that describes the relative positions of any two atoms can be obtained as a linear combination of these base vectors as $C_1\mathbf{b}_{2i,0} + C_2\mathbf{b}_{2i+1,0}$. The coefficients C_1 and C_2, referred to as 'peptide plane constants', are invariant with respect to the rotations in the chain, thus can be precomputed prior to the KCM simulation. Different pairs of coefficients are used for vectors describing the relative positions of different pairs of atoms. Based on experimental evidence, it is a reasonable assumption that these coefficients are the same across all AAs [72], and the average values are given in Table 1.[5]

Table 1: Peptide plane constants for bond vectors [72].

BV	C_1	C_2	BV	C_1	C_2
$\overrightarrow{C_\alpha C}$	-0.2761	$+1.4488$	$\overrightarrow{CO}$	-1.3324	$+2.3401$
$\overrightarrow{CN}$	$+1.2761$	-1.4488	$\overrightarrow{NH}$	$+1.4103$	-2.5111

In addition to the main chain body vectors, the 'side chain body vectors' (the third group listed above) are defined for the relative positions of the C and N atoms along the side chains. We refer the reader to [72] for more details about the molecular model of the peptide unit.

For a protein chain with m AA residues, the number of links can be obtained as

$$l = \left(2m + \sum_{i=1}^{m} l_i\right) \leq 6m = O(m), \tag{4}$$

noting that $l_i \leq 4$. The term $2m$ counts two rigid links per each AA's backbone—one for $-$CO$-$NH$-$ and one for $-C_\alpha-$ in the peptide unit—in order to have each rigid link connected to the next with a single revolute joint along either N$-C_\alpha$ or $C_\alpha-$C, as depicted in Fig. 1. The second term accounts for the number of additional side chain links. As a result, the total DOF of the kinematic linkage is equal to the number of links. Table 2 gives a complete description of dihedral angles, unit vectors, and body vectors for the entire protein chain.

[5] Nevertheless, in Protofold II the user has the option to choose whether to use the values provided in Table 1 for all AAs, or to maintain the refined values when available—e.g., when the protein is imported from the protein data-bank (PDB).

Table 2: Kinematic variables of the polypeptide linkage.

Symbol	Description
$\theta_{2i-1,0}$	Torsion angle ϕ_i around main chain N$-$C$_\alpha$ in AA_i
$\mathbf{u}_{2i-1,0}$	Unit vector along main chain N$-$C$_\alpha$ in AA_i
$\mathbf{b}_{2i-1,0}$	Body vector from N to C$_\alpha$ in AA_i
$\theta_{2i,0}$	Torsion angle ψ_i around main chain C$_\alpha-$C in AA_i
$\mathbf{u}_{2i,0}$	Unit vector along main chain C$_\alpha-$C in AA_i
$\mathbf{b}_{2i,0}$	Body vector from C$_\alpha$ in AA_i to N in AA_{i+1}[†]
$\theta_{2i-1,k}$	Torsion angle $\chi_{i,k}$ of side chain C/Ns in AA_i
$\mathbf{u}_{2i-1,k}$	Unit vector along side chain C/Ns in AA_i
$\mathbf{b}_{2i-1,k}$	Body vector connecting side chain Cs in AA_i

[†] The exception is $\mathbf{b}_{2m,0}$ which connects C$_\alpha$ to the carboxyl H in AA_{2m}.

Kinematic Equations. The instantaneous conformation of the protein chain is related to the ZP conformation with a set of rigid body transformations. Given a particular conformation in terms of $\theta_{j,k}$ $(1 \le j \le 2m, 0 \le k \le 4)$, the unit vectors and body vectors are transformed as follows:

$$[\mathbf{u}_{j,k}] = [M_{j,k}][\mathbf{u}^0_{j,k}], \quad 1 \le j \le 2m,\ 0 \le k \le 4, \tag{5}$$

$$[\mathbf{b}_{j,k}] = [M_{j,k}][\mathbf{b}^0_{j,k}], \quad 1 \le j \le 2m,\ 0 \le k \le 4, \tag{6}$$

where the superscript 0 indicates the reference ZP conformation. $[M_{j,k}]$ is the 3×3 matrix representation of the rigid body transformation $M_{j,k} \in \mathrm{SO}(3)$ that maps the ZP unit and body vectors $\mathbf{u}^0_{j,k}$ and $\mathbf{b}^0_{j,k}$ to their transformed orientations $\mathbf{u}_{j,k}$ and $\mathbf{b}_{j,k}$, respectively. These vectors are expressed using 3×1 column matrices. The transformation matrix for the main chain vectors $(k = 0)$ can be calculated as a product of successive rotations around individual joints in the main chain:

$$[M_{j,0}] = \prod_{r=1}^{j} [R_{r,0}], \quad 1 \le j \le 2m, \tag{7}$$

while the transformation matrix for the side chain vectors $(k \ge 1)$ is defined as a product of rotations around joints in the main chain, and those around the side chain:

$$[M_{2i-1,k}] = \prod_{r=1}^{2i-1} [R_{r,0}] \prod_{s=1}^{k} [R_{2i-1,s}], \quad 1 \le i \le m,\ 1 \le k \le 4, \tag{8}$$

where $[R_{r,s}]$ is the 3×3 matrix representation of the joint rotation transformation $R_{r,s} \in \mathrm{SO}(3)$ around the ZP unit vector $\mathbf{u}^0_{r,s}$ with an angle $\theta_{r,s}$ $(1 \le r \le i, 0 \le s \le k)$ [45], using the right-hand rule to choose the direction.

Once the body vectors are obtained using (6) for a given conformation, the moved atom center positions can be computed for the individual atoms. Assuming that the N atom at the amino-terminus is fixed at the origin, the coordinates of the main chain N and C$_\alpha$ atoms are obtained as

$$[\mathbf{r}_{j,0}] = \sum_{r=1}^{j} [\mathbf{b}_{r,0}], \quad 1 \le j \le 2m-1, \tag{9}$$

where the index $j = 2i-1$ corresponds to the C$_\alpha$ atom of residue AA_i while the index $j = 2i$ corresponds to the N atom of the residue AA_{i+1} for $1 \le i \le m$. The coordinates for the other atoms in the peptide group, namely H, C and O, are obtained from those for C$_\alpha$ and N, and a linear combination $C_1\mathbf{b}_{2i,0} + C_2\mathbf{b}_{2i+1,0}$ of main chain body vectors using the coefficients C_1 and C_2 given in Table 1. For the side chain C and N atoms, the coordinates are similarly obtained as

$$[\mathbf{r}_{2i-1,k}] = \sum_{r=1}^{2i-1} [\mathbf{b}_{r,0}] + \sum_{s=1}^{k} [\mathbf{b}_{2i-1,s}], \quad 1 \le i \le m,\ 1 \le k \le 4, \tag{10}$$

where $k = 1, 2, 3$, and 4 corresponds to the successive side chain C and N atoms in the residue AA_i. The coordinates for all other side chain atoms are obtained similarly from vectors along the side chain bonds subjected to the same set of motions [44].

The atom position vectors obtained from (9) and (10) at each iteration are used in the next section to compute the energies, forces, and torques that will determine the motion for the subsequent iteration.

2.2 Force Model

The interatomic effects can be classified into covalent and noncovalent interactions. The covalent interactions need not be considered explicitly in the force-field, since they are implicitly introduced in terms of the kinematic constraints innate to the kinematic chain model adopted in Section 2.1. The noncovalent forces, which are responsible for conformational changes in the protein chain, can be derived from the free energy formulation that follows.

For a protein chain made of n atoms, we assign each atom with a unique index $1 \le i \le n$, and its center coordinates $\mathbf{r}_i \in \mathbb{R}^3$ obtained from dihedral angles using kinematic equations (9) and (10).[6] Each atom is identified by a tuple $a_i = (i, \mathbf{r}_i, R_i, q_i, \epsilon_i, \gamma_i, \cdots)$ $(1 \le i \le n)$, containing its index, position, radius, charge, well depth parameter, solvation parameter, and other atomic constants, to be introduced shortly. The set of all atoms in the molecule is denoted by $\mathbb{A} = \{a_1, a_2, \cdots, a_n\}$. The aggregated free energy of all atoms in $\mathbb{A}$ can be decomposed into the following terms:

$$G^{\mathrm{tot}}(\mathbb{A}) = G^{\mathrm{elec}}(\mathbb{A}) + G^{\mathrm{vdw}}(\mathbb{A}) + G^{\mathrm{cav}}(\mathbb{A}), \tag{11}$$

where $G^{\mathrm{elec}}(\mathbb{A})$ is the electrostatic energy, including intramolecular charge interactions, hydrogen bonding effects, and the induced polarization in the solvent when the molecule is dissolved. $G^{\mathrm{vdw}}(\mathbb{A})$ is the sum of intramolecular van der Waals energies, also called 'steric effects', resulted from induced dipoles in the molecule. The sum of the first two terms has been accounted for in Protofold I [46–48] using the AMBER force-field model [67]. $G^{\mathrm{cav}}(\mathbb{A})$ is the nonpolar solvation free energy, the free energy change resulting from transfering a molecule from vacuum to solvent, i.e., the entropic change due to the formation of the cavity occupied by the

[6] Note the slight change of notations from Section 2.1, where the subscript $j = 2i-1$ or $2i$ referred to the AA index $1 \le i \le m$, while in Section 2.2 the single subscript $1 \le i \le n$ refers to the atom index.

instantaneous 3D shape of the protein [73]. Experimental results have shown that many water-soluble protein folding reactions are predominantly driven by a favorable reduction in $\Delta G^{\text{cav}}(\mathbb{A})$ [3], hence we incorporated this term into the improved energy formulation for Protofold II.

Electrostatic Interactions. The charge interactions are formulated using the modified form of Coulomb's law [67]:

$$G^{\text{elec}}(\mathbb{A}) = \sum_{a_i \in \mathbb{A}} \sum_{a_j \in \mathbb{A}-\{a_i\}} \frac{1}{4\pi\varepsilon_{i,j}} \frac{q_i q_j}{d_{i,j}}, \tag{12}$$

where $d_{i,j} = \|\mathbf{r}_i - \mathbf{r}_j\|_2$ is the interatomic center distance, q_i and q_j are the electrostatic charges, and $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of the pair of atoms $a_i, a_j \in \mathbb{A}$, respectively. $\varepsilon_{i,j} = \kappa_{i,j}\varepsilon_0$ is the 'dielectric constant' and is generally larger than vacuum permittivity $\varepsilon_0 \approx 8.854 \times 10^{-12}$ Farads (i.e., $\kappa_{i,j} > 1$). Thus (12) can be used to calculate the interactions between charges in the solvent, if a continuum dielectric model is used [3]. The dielectric constant reflects the ability of the environment to attenuate electrostatic interactions, e.g., $\kappa_{i,j} \sim 80$ for aqueous solvent and $\kappa_{i,j} \sim 2$–4 for the interior of the protein [3], where the larger value for the former is due to the induced polarization of water molecules. It is worthwhile noting that because of the nonuniformity of the dielectric medium, the most accurate computation of the electrostatic energy requires solving Poisson-Boltzman (PB) differential equations [74]. However, solving PB for every cycle of the KCM simulation is computationally expensive, and approximate alternatives such as generalized Born (GB) model can be used [75, 76]. A simple distance-dependent dielectric constant is used here (following the convention in [48]) to mimic the polarization effect, with closer interactions weighted more heavily [67]. The resultant Coulombic force $\mathbf{F}_i^{\text{elec}} = -\nabla_{\mathbf{r}_i} G^{\text{elec}}$ applied on the atom a_i by other atoms is then obtained as

$$\mathbf{F}_i^{\text{elec}}(\mathbb{A}) = \sum_{a_j \in \mathbb{A}-\{a_i\}} \frac{1}{4\pi\varepsilon_{i,j}} \frac{q_i q_j}{d_{i,j}^2} \mathbf{e}_{i,j}, \tag{13}$$

where $\mathbf{e}_{i,j} = (\mathbf{r}_i - \mathbf{r}_j)/d_{i,j}$ is the unit vector along the line of centers of the pair of atoms $a_i, a_j \in \mathbb{A}$. Since $\mathbf{F}_i^{\text{elec}} \propto 1/d_{i,j}^2$, electrostatic interactions between atoms that are farther than a so-called cut-off distance $d_{\text{cut}}^{\text{elec}} := 9.0$ Å are usually deemed negligible in the literature [3].[7] Therefore (13) is approximated as follows to reduce the number of pairwise computations between all atoms:

$$\mathbf{F}_i^{\text{elec}}(\mathbb{A}_i^{\text{elec}}) \approx \sum_{a_j \in \mathbb{A}_i^{\text{elec}}} \frac{1}{4\pi\varepsilon_{i,j}} \frac{q_i q_j}{d_{i,j}^2} \mathbf{e}_{i,j}, \tag{14}$$

where $\mathbb{A}_i^{\text{elec}} = \{a_j \in \mathbb{A} - \{a_i\} \mid d_{i,j} \leq d_{\text{cut}}^{\text{elec}}\}$ is referred to as the neighborhood of atom a_i associated with the electrostatic force-field, and consists of all the atoms whose distance to a_i are bounded by the cut-off distance $d_{\text{cut}}^{\text{elec}}$.

[7] Our experiments with larger molecules show that 9.0 Å is not always a proper cut-off distance and larger values need to be used, as demonstrated in Section 5.3.

Van der Waals Interactions. The van der Waals interactions are formulated using the empirical Lennard-Jones 6-12 potential function formula [67]:

$$G^{\text{vdw}}(\mathbb{A}) = \sum_{a_i \in \mathbb{A}} \sum_{a_j \in \mathbb{A}-\{a_i\}} \epsilon_{i,j} \left[\left(\frac{D_{i,j}}{d_{i,j}} \right)^{12} - 2 \left(\frac{D_{i,j}}{d_{i,j}} \right)^{6} \right], \tag{15}$$

where $d_{i,j} = \|\mathbf{r}_i - \mathbf{r}_j\|_2$ is the interatomic center distance, $D_{i,j} = R_i + R_j$ is the 'van der Waals distance' in which R_i, R_j are the van der Waals radii of the atoms $a_i, a_j \in \mathbb{A}$, respectively. $\epsilon_{i,j} = \sqrt{\epsilon_i \epsilon_j}$ is the 'depth of potential well' for the particular pair of atoms. It is worthwhile noting that the van der Waals effects have the same origin as the electrostatic forces, and reflect the induced dipoles due to transient fluctuations in electron clouds of the interacting atoms [3]. The resultant van der Waals force $\mathbf{F}_i^{\text{vdw}} = -\nabla_{\mathbf{r}_i} G^{\text{vdW}}$ on the atom a_i by other atoms is then obtained as

$$\mathbf{F}_i^{\text{vdw}}(\mathbb{A}) = \sum_{a_j \in \mathbb{A}-\{a_i\}} 12\epsilon_{i,j} \left(\frac{D_{i,j}^{12}}{d_{i,j}^{13}} - \frac{D_{i,j}^{6}}{d_{i,j}^{7}} \right) \mathbf{e}_{i,j}, \tag{16}$$

where $\mathbf{e}_{i,j} = (\mathbf{r}_i - \mathbf{r}_j)/d_{i,j}$ is the unit vector along the line of centers of the pair of atoms $a_i, a_j \in \mathbb{A}$. The van der Waals forces have a much smaller effect radius and are significant only when the atoms approach each other very closely. The repulsive component becomes very large when the two atoms penetrate into each other, an effect widely known as the 'steric clash'. The attractive component known as the 'London dispersion' force, on the other hand, is dominant when the atoms are farther than the van der Waals distance $D_{i,j}$ [3]. These interactions decay much faster than Coulombic forces, hence a smaller cut-off distance of $d_{\text{cut}}^{\text{vdw}} := 5.0$ Å is sufficient [3] resulting in the following approximation:

$$\mathbf{F}_i^{\text{vdw}}(\mathbb{A}_i^{\text{vdw}}) \approx \sum_{a_j \in \mathbb{A}_i^{\text{vdw}}} 12\epsilon_{i,j} \left(\frac{D_{i,j}^{12}}{d_{i,j}^{13}} - \frac{D_{i,j}^{6}}{d_{i,j}^{7}} \right) \mathbf{e}_{i,j}, \tag{17}$$

where $\mathbb{A}_i^{\text{vdw}} = \{a_j \in \mathbb{A} - \{a_i\} \mid d_{i,j} \leq d_{\text{cut}}^{\text{vdw}}\}$ is referred to as the neighborhood of the atom a_i associated with the van der Waals force-field, and consists of all the atoms whose distance to a_i are bounded by the cut-off distance $d_{\text{cut}}^{\text{vdw}}$.

Interaction Classification. The interactions discussed so far are between the pairs of atoms that are *not* covalently bonded, thus (14) and (17) have to be modified to eliminate the terms corresponding to the pairs of bonded atoms (i.e., '1-2 interactions'). Furthermore, it is a common convention in molecular mechanics to modify the electrostatic and van der Waals interactions between the pairs of atoms bonded to a common atom, i.e., atoms that are 2 bonds apart along the chain (i.e., '1-3 interactions'), as well as the atoms that are 3 bonds apart along the chain (i.e., '1-4 interactions') [77]. Consequently, the empirical forms of (14) and (17) are

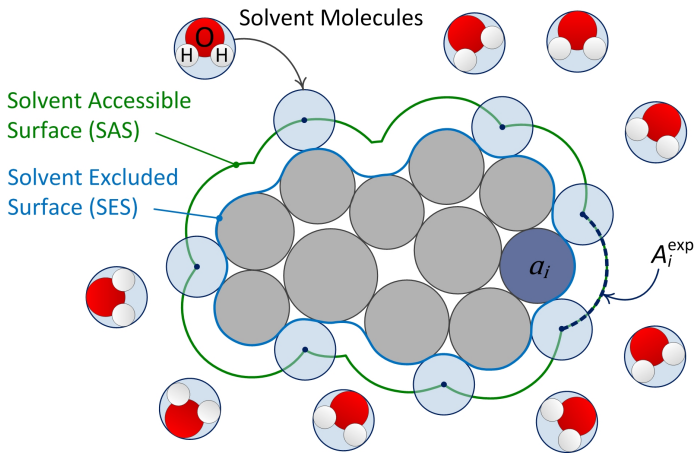

Figure 2: The solvent-accessible and -excluded surfaces.

Table 3: Atomic solvation parameters adopted from [78] for different adjustments by [79,80]. Units are in $kcal\ mol^{-1}\mathring{A}^{-2}$.

Atom type	Adjustment in [79]	Adjustment in [80]
C	$+0.004 \pm 0.003$	$+0.012 \pm 0.003$
O/N	-0.113 ± 0.014	-0.116 ± 0.013
S	-0.017 ± 0.022	-0.018 ± 0.021
O^-	-0.166 ± 0.038	-0.175 ± 0.036
N^+	-0.169 ± 0.031	-0.186 ± 0.022

modified as

$$\mathbf{F}_i^{\text{elec}}(\mathbb{A}_i^{\text{elec}}) \approx \sum_{a_j \in \mathbb{A}_i^{\text{elec}}} \frac{w_{i,j}^{\text{elec}}}{4\pi\varepsilon_{i,j}} \frac{q_i q_j}{d_{i,j}^2} \mathbf{e}_{i,j}, \tag{18}$$

$$\mathbf{F}_i^{\text{vdw}}(\mathbb{A}_i^{\text{vdw}}) \approx \sum_{a_j \in \mathbb{A}_i^{\text{vdW}}} 12 w_{i,j}^{\text{vdw}} \epsilon_{i,j} \left(\frac{D_{i,j}^{12}}{d_{i,j}^{13}} - \frac{D_{i,j}^{6}}{d_{i,j}^{7}} \right) \mathbf{e}_{i,j}, \tag{19}$$

where $w_{i,j}^{\text{elec}}$ and $w_{i,j}^{\text{vdw}}$ are the weight factors for the electrostatic and van der Waals forces, respectively, for the pair of atoms $a_i, a_j \in \mathbb{A}$ depending on their interaction type. The weights are set to $w_{i,j} = 0$ for 1-2 interactions, and $0 \leq w_{i,j} \leq 1$ for 1-3 and 1-4 interactions, whose values vary across different force models [62, 65, 67, 68]. $w_{i,j} = 1$ for all other situations. In other words, the atoms that have at least 4 bonds in between them along the graph of covalent bonds are far enough to be considered unaffected by the covalent electron clouds, as originally formulated in (14) and (17).

Nonpolar Solvation Effects. The hydrophobic free energy of solvation, which reflects the entropy changes in the solvent molecules due to cavity creation, is formulated using the linear empirical formulation in [78]:

$$G^{\text{cav}}(\mathbb{A}) = \sum_{a_j \in \mathbb{A}} \gamma_j \ A_j^{\text{exp}}, \tag{20}$$

where γ_j is the atomic solvation parameter and A_j^{exp} is the Lee and Richards SASA for the atom $a_j \in \mathbb{A}$ [81]. To obtain the atomic SASA at a given snapshot, a probe radius of $R_{\text{H}_2\text{O}} = 1.2 - 1.4\ \mathring{A}$ is used to offset the van der Waals surfaces of the individual atoms as illustrated in Fig. 2. These offset spheres are overlapped to obtain the contributions of different atoms to the total SASA. The atomic solvation parameter γ_j is an estimate of the free energy per unit area required to transfer the atom from vacuum to water, and depends on the atom type [78]. Table 3 shows the values of this parameter for different atom types (namely, C, uncharged O or N, S, O^-, and N^+) obtained in [78] based on the experimental results in [82] adjusted by [79, 80]. The hydrophobic interaction forces $\mathbf{F}_i^{\text{cav}} = -\nabla_{\mathbf{r}_i} G^{\text{cav}}$ on the atom a_i by other atoms is then obtained as

$$\mathbf{F}_i^{\text{cav}}(\mathbb{A}) = -\sum_{a_j \in \mathbb{A}} \gamma_j \nabla_{\mathbf{r}_i} A_j^{\text{exp}}, \tag{21}$$

where $\nabla_{\mathbf{r}_i} A_j^{\text{exp}}$ is the gradient of the exposed area of the atom a_j due to an infinitesimal displacement of a_i. It is important to note that, unlike the force formulae presented earlier for the electrostatic and van der Waals effects in (13) and (16), the summation in (21) for the solvation free energy gradient iterates over all $a_j \in \mathbb{A}$ *including* a_i itself.

Considering the case when $i = j$, one realizes that displacing the atom a_i in any direction has the same effect on the geometry of the protein surface as displacing all the other atoms in the opposite direction. Therefore

$$\nabla_{\mathbf{r}_i} A_i^{\text{exp}} = -\sum_{a_j \in \mathbb{A} - \{a_i\}} \nabla_{\mathbf{r}_j} A_i^{\text{exp}}. \tag{22}$$

Substituting for this term in (21) leads to the following symmetric form, whose range of summation excludes a_i itself, similar to (13) and (16):

$$\begin{aligned} \mathbf{F}_i^{\text{cav}}(\mathbb{A}) &= -\gamma_i \nabla_{\mathbf{r}_i} A_i^{\text{exp}} - \sum_{a_j \in \mathbb{A} - \{a_i\}} \gamma_j \nabla_{\mathbf{r}_i} A_j^{\text{exp}} \\ &= \sum_{a_j \in \mathbb{A} - \{a_i\}} \left(\gamma_i \nabla_{\mathbf{r}_j} A_i^{\text{exp}} - \gamma_j \nabla_{\mathbf{r}_i} A_j^{\text{exp}} \right). \end{aligned} \tag{23}$$

We show in Section 4 that (23) is computationally preferable over (21). To further simplify (23), note that for a pair of atoms a_i and a_j the infinitesimal displacement of one does not affect the overlapped solvent exposed area of the other if their offset spheres (i.e., the spheres with radii $R_i^{\text{off}} = R_i + R_{\text{H}_2\text{O}}$ and $R_j^{\text{off}} = R_j + R_{\text{H}_2\text{O}}$, respectively) do not intersect, i.e., $\nabla_{\mathbf{r}_i} A_j^{\text{exp}} = \nabla_{\mathbf{r}_j} A_i^{\text{exp}} = 0$ if $d_{i,j} > R_i + R_j + 2R_{\text{H}_2\text{O}}$. Therefore, the number of terms that contribute a nonzero value to the summation of (23) is significantly reduced:

$$\mathbf{F}_i^{\text{cav}}(\mathbb{A}_i^{\text{cav}}) = \sum_{a_j \in \mathbb{A}_i^{\text{cav}}} \left(\gamma_i \nabla_{\mathbf{r}_i} A_j^{\text{exp}} - \gamma_j \nabla_{\mathbf{r}_j} A_i^{\text{exp}} \right), \tag{24}$$

where $\mathbb{A}_i^{\text{cav}} = \{a_j \in \mathbb{A} - \{a_i\} \mid d_{i,j} \leq R_i + R_j + 2R_{\text{H}_2\text{O}}\}$ is referred to as the neighborhood of the atom a_i associated with the nonpolar solvent effects. For practical reasons that will be explained in Section 3.2, we use a larger neighborhood

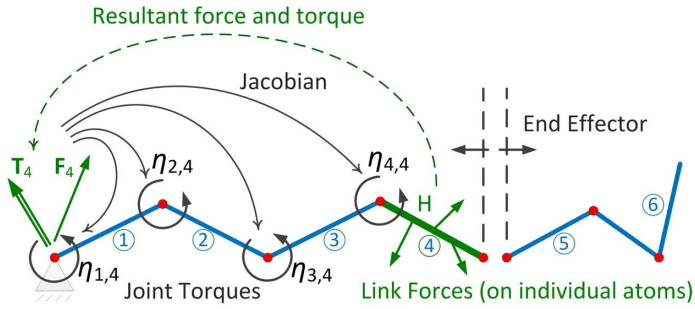

Figure 3: The forces on each link are converted into an equivalent set of joint torques on the preceding joints in the chain.

redefined as $\mathbb{A}_i^{\text{cav}} = \{a_j \in \mathbb{A} - \{a_i\} \mid d_{i,j} \leq d_{\text{cut}}^{\text{cav}}\}$ using the more conservative (but constant across all pairs of atoms) cut-off distance of $d_{\text{cut}}^{\text{cav}} := 2(R_{\max} + R_{\text{H}_2\text{O}})$, where $R_{\max} = \max_{a_i \in \mathbb{A}}\{R_i\}$. A value of $d_{\text{cut}}^{\text{cav}} := 8\mathring{A}$ is typically safe. Note that unlike the case with (13) and (16), eliminating pairwise interactions with $d_{i,j} > 8\mathring{A}$ from (24) does not impart an approximation error.

2.3 Kinetostatic Simulation

We use the KCM (presented in [44–48] for protein folding) to explicitly integrate the conformational changes of the linkage model under the kinetostatic effect of the force-field computed in Section 2.2.

Link Forces and Torques. For a protein chain with a total of $l = O(m)$ links, where m is the number of AA residues, the resultant force and torque applied to the j^{th} link $(1 \leq j \leq l)$ are computed as

$$\mathbf{F}_j^{\text{link}} = \sum_{a_i \in \mathbb{L}_j} \left(\mathbf{F}_i^{\text{elec}} + \mathbf{F}_i^{\text{vdw}} + \mathbf{F}_i^{\text{cav}}\right), \tag{25}$$

$$\mathbf{T}_j^{\text{link}} = \sum_{a_i \in \mathbb{L}_j} \mathbf{r}_i \times \left(\mathbf{F}_i^{\text{elec}} + \mathbf{F}_i^{\text{vdw}} + \mathbf{F}_i^{\text{cav}}\right), \tag{26}$$

where $\mathbf{r}_i$ is the absolute center position vector of the atom $a_i \in \mathbb{A}$ obtained from (9) and (10) in Section 2.1 (with different index notation), and $\mathbb{L}_j \subset \mathbb{A}$ is the subset of atoms that belong to the j^{th} link along the chain. Note that the origin of the absolute coordinate system (arbitrarily picked the same as the N-terminus anchor of the chain) is selected as the torque center for all links.

Equivalent Joint Torques. Since the revolute joints are assumed to be frictionless, the action of the link forces and torques can be replaced by an equivalent set of torques acting on the joints [48], as shown in Fig. 3. For a given rigid link, one can trace a unique serial chain of h successive links $(1 \leq h \leq l)$ starting from the N-terminus and ending at the link under consideration, which is equivalent to a path along the graph of the open linkage. The contribution of the force $\mathbf{F}_h^{\text{link}}$ and the torque $\mathbf{T}_h^{\text{link}}$ applied to the end-effector link (i.e., the h^{th} link along the serial chain) to the joint torque at the k^{th} joint along the chain preceding the end-effector, denoted as $\eta_{k,h}$ $(1 \leq k \leq h, 1 \leq h \leq l)$, can be computed using the conventional manipulator Jacobian matrix $[J]$ [48]:

$$[\boldsymbol{\eta}_h] = [J]^{\text{T}} \begin{bmatrix} \mathbf{T}_h^{\text{link}} \\ \mathbf{F}_h^{\text{link}} \end{bmatrix}, \tag{27}$$

where $[\boldsymbol{\eta}_h] = [\eta_{1,h}, \eta_{2,h}, \cdots, \eta_{h,h}]^{\text{T}}$ represents an $h \times 1$ array of joint torques that the end-effector force and torque will induce on the different joints preceding the end-effector along the serial chain. $[J]^{\text{T}}$ is the transpose of the $6 \times h$ Jacobian matrix $[J] = [\mathbf{J}_1, \mathbf{J}_2, \cdots, \mathbf{J}_h]$ for a given configuration of the chain [48]. The k^{th} column of the Jacobian associated with the k^{th} revolute joint is given by

$$[\mathbf{J}_k] = \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_k \times (\mathbf{p}_h - \mathbf{p}_k) \end{bmatrix}, \tag{28}$$

where $\mathbf{u}_k$ is the unit vector along the k^{th} joint and $(\mathbf{p}_h - \mathbf{p}_k)$ is a vector connecting a point $\mathbf{p}_k$ along the k^{th} joint's axis to the point $\mathbf{p}_h$ where the end-effector force $\mathbf{F}_h$ applies—i.e., the atom positions obtained from (9) and (10). This process is repeated for all links to calculate the contribution of each link on the joints preceding that link in the serial chain. The total torque for each joint is obtained from a summation of these contributions [48]:

$$\tau_k = [\mathbf{J}_k]^{\text{T}} \sum_{h=k}^{l} \begin{bmatrix} \mathbf{T}_h^{\text{link}} \\ \mathbf{F}_h^{\text{link}} \end{bmatrix}, \tag{29}$$

where the indexing $h = k, k+1, \cdots, l$ of the links is ordered along the main chain from amino-terminus to carboxyl-terminus, and can branch along the side chains. The range of the summation in (29) implies that each joint torque τ_k $(1 \leq k \leq l)$ depends on the forces $\mathbf{F}_h^{\text{link}}$ and torques $\mathbf{T}_h^{\text{link}}$ on the succeeding links $(k \leq h \leq l)$.

Kinetostatic Simulation. Making use of the assumption in [48] that the inertia forces at the atomic scale have negligible effects on the dynamics of folding compared to interatomic forces, the successive kinetostatic fold compliance relates the joint torques to the changes in the dihedral angles as follows:

$$\Delta\theta_j = \kappa \frac{\tau_j}{|\tau_{max}|}, \tag{30}$$

where $\Delta\theta_j$ and τ_j are the angular change and the joint torque of the j^{th} revolute joint $(1 \leq j \leq l)$, respectively. $|\tau_{max}|$ is the maximum joint torque throughout the entire chain, used to normalize the torques to the interval $[0, 1]$, and the coefficient κ is chosen small enough to avoid large changes in the angles, and to achieve numerical stability. One can notice that the conformational change computed in (30) is analogous to taking a step along the steepest-descent direction of the free energy gradient in the conformation landscape.

The computed changes in the joint angles are applied to the kinematic chain, and the entire process is repeated on the

updated conformation until a convergence criteria is reached, as described in more detail is Section 4.

It is worthwhile noting that once the chain conformation (i.e., optimization variables) is modeled as in Section 2.1 and the energy-field (i.e., objective function) is formulated as in Section 2.2, the search for local or global minima of the free energy in (11) can be undertaken using a variety of classical (e.g., conjugate-gradients) and stochastic (e.g., genetic algorithm) optimization methods. Since the focus of this article is mainly on force-field modeling and computing, we skip a detailed treatment of the search phase.

3 Algorithms

This section presents efficient algorithms and data structures to speed up the force field computation during kinetostatic iterations. To leverage the proximity information between the atoms, we use a 3D hash table data structure based on a uniform spatial grid in Section 3.2. To classify the interatomic interaction types based on chain topology to compute the electrostatic and van der Waals effects, we use a tree-based data structure in Section 3.3 that replaces the adjacency matrix used in Protofold I [46–48]. To compute the solvation effects, we develop an approximate (yet adequately accurate) surface enumeration technique in 3.4, efficient CPU- and GPU-parallel implementations of which will be detailed in Section 4. Finally, we compute joint torques by aggregating contributions of different links (traversed along different paths in the linkage graph) on the joints along the chain, using the well-known 'prefix computation' in Section 3.5 which can also be implemented efficiently in parallel [70]. We show that the computational complexity of all steps is decreased from $O(n^2)$ in Protofold I [46–48] to expected $O(n)$ in Protofold II for a protein chain with a total of n atoms.

3.1 Rigid Transformations

At every snapshot $t \geq 0$ of the KCM, the protein conformation is described by a set of dihedral angles $\theta^t_{j,k}$ defined in (1) through (3).

- At the first iteration ($t = 0$), all angles are initialized as $\theta^0_{j,k} = 0$ (ZP conformation).
- At subsequent iterations ($t \geq 1$), for $1 \leq j \leq 2m$ (where m is the number of AA residues) and $0 \leq k \leq l_i$ (where $0 \leq l_i \leq 4$ is the number of side chain links of the residue AA_i), the angles are obtained as $\theta^t_{j,k} = \theta^{t-1}_{j,k} + \Delta\theta^{t-1}_{j,k}$, where the increment $\Delta\theta^{t-1}_{j,k}$ is computed using (30) from the previous step's configuration and joint torques.

Once the dihedral angles are known, the transformation matrices $[M^t_{j,k}]$ are obtained from (7) and (8) using sequential matrix multiplication traversing the linkage tree from the anchored amino-terminus to the open carboxyl-terminus. Next, the unit vectors $\mathbf{u}^t_{j,k}$ and the body vectors $\mathbf{b}^t_{j,k}$ are computed from (5) and (6). Since the number of links is clearly less than the number of atoms, these computations take $O(n)$ time. The Cartesian coordinates of the individual atom center positions $\mathbf{r}_i \in \mathbb{R}^3$ ($1 \leq i \leq n$) are obtained from the body vectors using (9) and (10), which also takes $O(n)$. In the following sections, we assume that both dihedral angles and atom center positions are known for the purpose of computing the next snapshot's energies, forces, and torques.

3.2 Proximity Queries

The brute-force approach for obtaining the proximity information at each snapshot is to check center distances against the cut-off distance for all possible pairs of atoms, which takes $O(n^2)$ time. Using this method, the approximate truncated formulae given in (12), (15), and (20) would take the same asymptotic time as the exact all-pairs formulae given in (13), (16), and (23), respectively, which is $O(n^2)$. Geometric hashing provides a simple solution to speed up proximity queries.

3D Hash Table. We use a 3D hash table data structure based on a uniform Cartesian grid, which bounds the current 3D structure of the protein and arranges the atom indices into groups based on their center positions. Each 3D grid cell is associated with a so-called 'bucket' that stores the indices of the atoms whose centers are located inside the cell in a linked-list. The grid dimensions are set dynamically to adapt to the shape of the protein's bounding box at the current snapshot.

The grid cells are chosen to be cubic, i.e., with equal edge length s_c along all 3 Cartesian axes. Given the min/max corner coordinates of the bounding box of the atom centers $\mathbf{r}_{\min}, \mathbf{r}_{\max} \in \mathbb{R}^3$—which can be obtained in $O(n)$ by scanning through the n atom center coordinates—we choose s_c in such a way that it results in $\lceil \alpha n \rceil$ grid cells/buckets, where $\alpha > 0$ is an arbitrary constant. More precisely, we choose $s_c = [v_{\mathrm{BB}}(\mathbb{A})/(\alpha n)]^{\frac{1}{3}}$ where $v_{\mathrm{BB}}(\mathbb{A})$ is the protein bounding box volume. The dimensions of the grid bounding box are then chosen as $\lceil (\mathbf{r}_{\max} - \mathbf{r}_{\min})/s_c \rceil s_c$ (slightly larger than the dimensions of the protein bounding box $\mathbf{r}_{\max} - \mathbf{r}_{\min}$), where the operator $\lceil \cdot \rceil$ is applied componentwise along the 3 Cartesian axes. Before we proceed with presenting the complexity analysis, we make the following assumptions:

Assumption 1 Due to the extremely strong repulsive van der Waals forces, the atoms that are not covalently bonded cannot penetrate into each other, and those covalently bonded intersect over a small volume. Given any arbitrary subset of atoms $\mathbb{A}' \subseteq \mathbb{A}$ with $R_{\min} = \min_{a_i \in \mathbb{A}'} R_i$ and $R_{\max} = \max_{a_i \in \mathbb{A}'} R_i$, let the maximum penetration volume between any pair of covalently bonded atoms $a_i, a_j \in \mathbb{A}'$ be upper-bounded by $\epsilon \min\{v_i, v_j\}$ where $v_i = \frac{4\pi}{3} R_i^3$ is the volume of the atom a_i, and $0 \leq \epsilon < \frac{1}{4}$ is a small number. Since each atom makes at most 4 covalent bonds, the unpenetrated volume for the atom a_i is lower-bounded by $(1-4\epsilon)v_i$, hence it is safe to assume that $(1 - 4\epsilon) > 0$. Then the volume $v(\mathbb{A}')$ occupied by the union of all atoms in $\mathbb{A}'$ is bounded as $\frac{4\pi}{3}(1 - 4\epsilon)|\mathbb{A}'|R^3_{\min} \leq v(\mathbb{A}') \leq \frac{4\pi}{3}|\mathbb{A}'|R^3_{\max}$. Consequently, there exists an 'average' radius $\bar{R}(\mathbb{A}')$ bounded

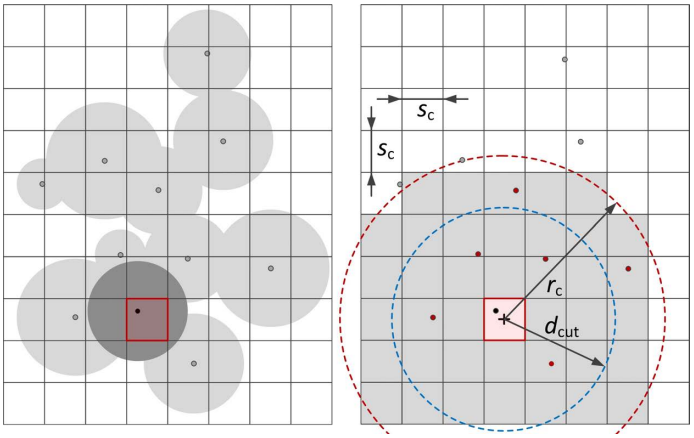

Figure 4: Atom centers are hashed into a 3D grid and the neighbors are selected within a cut-off distance.

as $(1-4\epsilon)^{\frac{1}{3}}R_{\min} \leq \bar{R}(\mathbb{A}') \leq R_{\max}$, such that $v(\mathbb{A}') = \frac{4\pi}{3}|\mathbb{A}'|\bar{R}^3(\mathbb{A}')$, where typically $\bar{R} \sim 1\mathring{A}$.

Assumption 2 It is also reasonable to assume that if the protein is either in an extended conformation aligned with one of the Cartesian axes (which is the case near the ZP conformation) or in a globular conformation (which is the case for most water-soluble proteins at their folded conformation), the empty space inside the bounding box is not extremely larger than the space occupied by the protein atoms, i.e., $v_{\text{BB}}(\mathbb{A})/v(\mathbb{A}) = O(1)$. Supported by experimentation, we assume this to be the case in the intermediate conformations as well, to simplify the analysis. However, there are possible conformations (e.g., extended along a diagonal direction in the axis-aligned bounding box) that would violate this assumption and result in slightly larger running times than predicted here, in spite of the low probability.

Letting $\mathbb{A}' := \mathbb{A}$ (hence $|\mathbb{A}'| = n$) in Assumption 1, and noting from the definition that $v_{\text{BB}}(\mathbb{A}) = (\alpha n)s_c^3$, from Assumption 2 it follows that $\frac{3\alpha}{4\pi}(s_c/\bar{R})^3 = O(1)$. Therefore, if we choose the grid cell size $s_c \sim 1\mathring{A}$ then $\alpha = O(1)$ and the number of buckets will be $\lceil \alpha n \rceil = O(n)$.

Table Construction. At each snapshot of the simulation, the algorithm scans through all the atoms with updated positions, and the deterministic hash function simply maps the atom center positions $\mathbf{r}_i \in \mathbb{R}^3$ that lie inside a grid cell into the corresponding 3D array of buckets. The 3D index $\mathbf{k} \in \mathbb{Z}^3 \cap [0, +\infty]^3$ of the bucket to which a given atom $a_i \in \mathbb{A}$ belongs is determined in $O(1)$ time as $\mathbf{k} = \lfloor (\mathbf{r}_i - \mathbf{r}_{\min})/s_c \rfloor$, where the operator $\lfloor \cdot \rfloor$ is applied componentwise along the 3 Cartesian axes. Therefore, scanning through the atoms and constructing the 3D grid data structure is expected to $O(n)$ time and to requires $O(n)$ space.

Neighbor Queries. Once the atoms are arranged into the buckets, the algorithm iterates through the grid cells and scans through the linked-lists within the buckets. For each atom a_i in a given bucket associated with the grid cell index $\mathbf{k}$, the set of 'neighbor atoms' defined as

$$\mathbb{A}_i = \{a_j \in \mathbb{A} \mid \|\mathbf{r}_i - \mathbf{r}_j\|_2 \leq d_{\text{cut}}\}, \quad d_{\text{cut}} \in \{d_{\text{cut}}^{\text{elec}}, d_{\text{cut}}^{\text{vdw}}, d_{\text{cut}}^{\text{cav}}\} \tag{31}$$

can be identified rapidly for a given cut-off distance d_{cut} associated with any of the energetic interactions explained in Section 2.2. As illustrated in Fig. 4, a spherical region of radius $r_c = d_{\text{cut}} + \sqrt{3}s_c$ is considered around the (center point of) each grid cell to look for the (center point of) 'neighbor cells', defined as the collection of cells which *completely* lie inside this spherical region. The cut-off distance d_{cut} is offset by the diagonal size of the cells $\sqrt{3}s_c$ which takes into account the worst-case difference of the distance between cell centers and the distance between atom centers. This guarantees that the set of all atoms inside this collection of covered cells (denoted as $\mathbb{A}_i'$) contains the set of all neighbor atoms, i.e., $\mathbb{A}_i' \supseteq \mathbb{A}_i$, where $\mathbb{A}_i$ is one of the neighbor sets $\mathbb{A}_i^{\text{elec}}$, $\mathbb{A}_i^{\text{vdw}}$, or $\mathbb{A}_i^{\text{cav}}$. Letting $\mathbb{A}' := \mathbb{A}_i'$ in Assumption 1, the volume occupied by this set of atoms is $v(\mathbb{A}_i') = \frac{4\pi}{3}|\mathbb{A}_i'|\bar{R}^3(\mathbb{A}_i')$. Noting that $\mathbb{A}_i'$ is contained inside the spherical region of radius r_c, $v(\mathbb{A}_i') \leq \frac{4\pi}{3}r_c^3$ hence $|\mathbb{A}_i'| \leq (r_c/\bar{R})^3 = [(d_{\text{cut}} + \sqrt{3}s_c)/\bar{R}]^3 = O(1)$, since $\bar{R}, s_c \sim 1\mathring{A}$ and $d_{\text{cut}} \sim 10\mathring{A}$. As a result, it is expected to take $O(1)$ time to scan through the atoms inside the collection of neighbor cells, and $O(n)$ total time to traverse all pairs of atoms using the 3D hash table.

For parallel implementation purposes, we construct (and dynamically maintain) a 'neighborhood matrix' composed of an array of n $O(1)-$sized linked-lists (one list per atom), where the i^{th} list $(1 \leq i \leq n)$ contains the indices of the neighbor atoms $\mathbb{A}_i'$. Constructing this data structure is expected to take $\sum_{i=1}^n |\mathbb{A}_i'| = \sum_{i=1}^n O(1) = O(n)$ time and space, and accessing each atom's neighbors is expected to take $|\mathbb{A}_i'| = O(1)$ time.

Energy and Force Computations. Once the pairs of neighbors are identified, computing their electrostatic and van der Waals forces can be done in $O(1)$ time per pair, using the analytical truncated equations given in (18) and (19), respectively. It is important to note that such interactions are *pairwise*, i.e., they depend on the relative positions of pairs of atoms $a_i \in \mathbb{A}$ and $a_j \in \mathbb{A}_i^{\text{elec}}$ or $\mathbb{A}_i^{\text{vdw}}$, and the presence of any third atom $a_k \in \mathbb{A}$ $(k \neq i, j)$ does not affect the force exchanged between a_i and a_j. Therefore, the computation algorithm is straightforward: it iterates over all atoms for $1 \leq i \leq n$ (sequentially or in parallel) and for each atom, it computes $\mathbf{F}_i^{\text{elec}}(\mathbb{A}_i')$ and $\mathbf{F}_i^{\text{vdw}}(\mathbb{A}_i')$, by sequentially aggregating the contributions of the hashed neighbor atoms $a_j \in \mathbb{A}_i'$, using (18) and (19), respectively.

Unfortunately, this is *not* the case for solvation force computation using (23) or (24), which requires computing the gradients of the atomic SASA with respect to the coordinates of the set of neighbor atoms. The SASA variations in one atom $a_i \in \mathbb{A}$ with respect to an infinitesimal change in the position of another atom $a_j \in \mathbb{A}_i^{\text{cav}}$, can be affected by the presence of a third atom $a_k \in \mathbb{A}_i^{\text{cav}}$ $(k \neq i, j)$, thus

cannot be obtained in a pairwise fashion. This is because the overlaps of the pairs of offset spheres are not mutually disjoint. A segment of the offset surface can be overlapped with more than one neighbor sphere simultaneously, thus displacing one of the overlapping spheres may or may not affect the SASA. We return to this subject in Section 3.4 where a surface sampling algorithm is proposed as a simple solution.

3.3 Bonds Tree/Graph

To decide the weight factors $w_{i,j}^{\text{elec}}$ in (18) and $w_{i,j}^{\text{vdw}}$ in (19), one needs to quickly identify the types of interactions based on the number of bonds between pairs of atoms as described in Section 2.2. An $n \times n$ look-up table was used in Protofold I [46–48] for all pairs of n atoms which required a preprocessing step with $O(n^2)$ time and space. This can be improved by constructing a tree/graph data structure that stores the combinatorial structure of the chain, in which the vertices are atoms and edges are the covalent bonds between them. By excluding a single edge from the loops associated with the rare aromatic groups in certain side chains (e.g., imidazole in His and indole in Trp), this graph can be converted to a tree whose root is arbitrarily chosen as the N atom of the amino-terminus. The interaction types are then identified by the shortest path lengths between atoms.

The algorithm starts from the root and visits all vertices using a standard tree traversal routine. For each vertex, it stores a pointer to its parent and the index of the corresponding atom's AA residue. During the force computation in each KCM iteration, the residue indices for a pair of atoms of interest are checked. If the atoms are farther than a residue apart (i.e., if AA indices are neither identical nor consecutive), the weights in (18) and (19) are simply set to 1 (i.e., 1-4 interactions or beyond). Otherwise, the algorithm checks 1) if one atom is the parent of the other (i.e., 1-2 interaction); 2) if one atom is the grand parent or sibling of the other (i.e., 1-3 interaction); or 3) if one atom is the grand grand parent or sibling of parent of the other (1-4 interactions). This requires $O(n)$ time and space for preprocessing and $O(1)$ query time during the KCM iterations.

3.4 Surface Enumeration

As mentioned in Section 1.1, several attempts have been made to approximate the SASA and its derivative by a pairwise treatment of the overlaps, including the probabilistic methods [63, 64, 66, 69], popular in many molecular simulation software such as CHARMM [62], GROMOS [65], and AMBER [67,68]. In an early attempt to add solvation effects to Protofold II, we used these pairwise approximate formulae, which made it possible to compute the solvation forces with running times comparable to those of the electrostatic and van der Waals force computations. However, a comparison with the exact method [61] showed that when the distribution of the atoms deviates from that assumed in the the probabilistic methods, prohibitively large errors can be introduced into the resultant effects. The exact method [61] takes $O(|\mathbb{A}_i'|)$ operations for computing the SASA and its gradient for $a_i \in \mathbb{A}$ by using the coordinates of all neighbor atoms $\mathbb{A}_i^{\text{cav}} \subseteq \mathbb{A}_i'$. This is asymptotically $O(1)$ time per atom (since we reasoned earlier that $|\mathbb{A}_i'| = O(1)$ under Assumption 1), but in practice it is not nearly as fast as using the pairwise formulae. Alternatively, we use an approximate method that relies on an enumeration of the surface area, in which the deviations from the exact results can be controlled to a desired precision in a trade-off with computation time.

Offset Sphere Sampling. For a given atom $a_i \in \mathbb{A}$ of van der Waals radius R_i, an offset sphere of radius $R_i^{\text{off}} = R_i + R_{\text{H}_2\text{O}}$ concentric with the atom sphere is considered. The atom's SASA is obtained by measuring the area A_i^{exp} of the portion of the offset surface that is not overlapped by the offset sphere of any neighbor atom $a_j \in \mathbb{A}_i^{\text{cav}}$ (hence exposed to the solvent). To approximate the SASA, one can generate a large but finite 'quasi-uniform' set of sample points denoted as Q_i $(1 \le i \le n)$ on the surface of the offset sphere of the atom $a_i \in \mathbb{A}$, by which we mean a sampling that allows approximating the exposed fraction of the surface by the ratio of the number of exposed sample points to the total number of sample points. In other words, if we let

$$Q_i^{\text{exp}} = \{\mathbf{q} \in Q_i \mid \nexists a_j \in \mathbb{A}_i' \ : \ \|\mathbf{q} - \mathbf{r}_j\|_2 \le R_j^{\text{off}}\} \qquad (32)$$

be the subset of the solvent-exposed sample points, i.e., the points that are outside the offset spheres of all neighbors, then $\lim_{|Q_i|\to\infty} |Q_i^{\text{exp}}|/|Q_i| = A_i^{\text{exp}}/A_i^{\text{off}}$. If we define the 'exposure ratio' as $f_i^{\text{exp}} = |Q_i^{\text{exp}}|/|Q_i|$, the SASA can be approximated as $A_i^{\text{exp}} \approx f_i^{\text{exp}} A_i^{\text{off}}$ where $A_i^{\text{off}} = 4\pi(R_i^{\text{off}})^2$, using a large enough sample size $|Q_i| \gg 1$.[8]

There are different ways to obtain a quasi-uniform deterministic sampling on a sphere with consistent incremental quality [84]. For example, one could use a triangular spherical meshing algorithm, which starts from an icosahedron approximation of the sphere and recursively creates successive triangular subdivisions projected back on the sphere. Alternatively, one could use a polar geodesic sampling algorithm, which starts from a set of orbits with uniform angular distribution and samples a number of points uniformly on each orbit proportional to the orbit's circumference. We take the latter approach whose details are presented in [2].

To improve the efficiency, one could always precompute the coordinates for a single sampling Q on a unit sphere centered at the origin, and map it into individual atoms with different offset sphere center positions $\mathbf{r}_i$ and radii R_i^{off} using the mapping $Q_i = T_i(Q)$ where $T_i(\mathbf{q}) = \mathbf{r}_i + R_i^{\text{off}}\mathbf{q}$ for $1 \le i \le n$ and $\mathbf{q} \in Q$. This implies an equal number of sample points $N = |Q|$ for all atoms, selected as $N = 4\pi(R_{\max}^{\text{off}})^2/\delta A$ where $R_{\max}^{\text{off}} = R_{\max} + R_{\text{H}_2\text{O}}$ is the maximum offset sphere radius,

[8] An alternative approach is uniform random sampling, e.g., using the simple method in [83]. Random sampling is easier to implement in parallel since every sample point in Q_i would be independent from others, and results in $A_i^{\text{exp}} = [f_i^{\text{exp}}]A_i^{\text{off}}$ where $[f_i^{\text{exp}}]$ is the *expected* ratio of the exposed sample points in probabilistic terms. However, it requires much larger sample sizes to approach the expectation and to achieve adequate accuracy in practice.

and δA is the desired characteristic area element carried by each sample point. Hence the sampling takes $O(nN)$ operations for all atoms regardless of the sampling technique. For implementation purposes, we assign an arbitrary ordering to the sample points, letting $Q = \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\}$ and denoting transformed sample points as $\mathbf{q}_{i,k} = T_i(\mathbf{q}_k)$.

Energy and Force Approximations. Substituting for $A_i^{\text{exp}} \approx f_i^{\text{exp}} A_i^{\text{off}} = 4\pi f_i^{\text{exp}} (R_i^{\text{off}})^2$ into (20), the total solvation free energy can be computed directly from

$$G^{\text{cav}}(\mathbb{A}) \approx \sum_{a_j \in \mathbb{A}} 4\pi \gamma_j f_j^{\text{exp}} (R_j^{\text{off}})^2. \tag{33}$$

To obtain the solvation force on any atom $a_i \in \mathbb{A}$, the energy must be differentiated with respect to the atom's center coordinates $\mathbf{r}_i \in \mathbb{R}^3$, giving rise to (23). It is very important to note that an infinitesimal displacement of the atom a_i can change the SASA of a_i itself, as well as that of the neighbor atoms $a_j \in \mathbb{A}_i^{\text{cav}} \subseteq \mathbb{A}_i'$. However, we showed in Section 2.2 that the two effects are geometrically dependent, yielding the symmetric form in (24). From a computational perspective, (24) is preferred over (24), because

1. it eliminates the need for computing the gradient $\nabla_{\mathbf{r}_j} A_i^{\text{exp}}$ for the cases when $i = j$, hence decreasing the number of such computations from n^2 to $n(n-1)$; and
2. its symmetric form lends itself to a data-parallel implementation that is balanced between computation and data sharing tasks, as we show in Section 4.

This follows from the fact that for a given pair of indices $i \neq j$, the first term in the formula for $\mathbf{F}_i^{\text{cav}}$ is identical to the second term in the formula for $\mathbf{F}_j^{\text{cav}}$ (both given by (24)), cutting the number of required SASA gradient computations to $n(n-1)/2$. Substituting for A_i^{exp} and A_j^{exp} in (24), the solvation forces can be approximated as

$$\begin{aligned}\mathbf{F}_i^{\text{cav}}(\mathbb{A}_i') \approx 4\pi\gamma_i (R_i^{\text{off}})^2 \sum_{a_j \in \mathbb{A}_i'} \nabla_{\mathbf{r}_j} f_i^{\text{exp}} \\ - 4\pi \sum_{a_j \in \mathbb{A}_i'} \gamma_j (R_j^{\text{off}})^2 \nabla_{\mathbf{r}_i} f_j^{\text{exp}},\end{aligned} \tag{34}$$

where $\nabla_{\mathbf{r}_j} f_i^{\text{exp}}$ and $\nabla_{\mathbf{r}_i} f_j^{\text{exp}}$ can be approximated using the forward-difference method from finite variations of f_i^{exp} and f_j^{exp} with respect to the positions of the atoms a_i and a_j, respectively:

$$\nabla_{\mathbf{r}_j} f_i^{\text{exp}} \approx \sum_{s=1,2,3} \frac{f_{i,j,s}^{\text{exp}} - f_i^{\text{exp}}}{\delta r} \mathbf{e}_s, \tag{35}$$

where $\delta r > 0$ is the finite difference, $\mathbf{e}_s$ $(s = 1, 2, 3)$ are the unit vectors along the 3 Cartesian axes, and $f_{i,j,s}^{\text{exp}}$ are the exposure ratio of the atom $a_i \in \mathbb{A}$ after changing the position of the neighbor atom $a_j \in \mathbb{A}_i'$ from the current value $\mathbf{r}_j$ to a hypothetical variant $\mathbf{r}_j + \delta r \mathbf{e}_s$.

Enumeration Algorithm. In order to compute the exposure ratio and its finite-difference gradient, we use a binary enumeration function $B : \mathbb{A} \times Q \to \{0, 1\}$ $(1 \leq i \leq n)$ such that $B(a_i, \mathbf{q}_k) = 1$ if the sample point $\mathbf{q}_{i,k} \in T_i(Q)$ on the offset sphere of the atom $a_i \in \mathbb{A}$ is overlapped by at least one neighbor offset sphere (i.e., if $\exists a_j \in \mathbb{A}_i'$ such that $\|T_i(\mathbf{q}_k) - \mathbf{r}_j\|_2 \leq R_j^{\text{off}}$) and $B(a_i, \mathbf{q}_k) = 0$ if the sample point is exposed to the solvent. The algorithm iterates over all atoms for $1 \leq i \leq n$ and all sample points for $1 \leq k \leq N$ (sequentially or in parallel). For each sample point, the indicator $B_{i,k} := B(a_i, \mathbf{q}_k)$ is initialized to 0, and each point is tested against the set of neighbors $\mathbb{A}_i'$, scanned sequentially. As soon as one overlapping neighbor is found, $b_{i,k}$ is set to 1 and there is no need to test the rest of the neighbors. The exposure ratio is then computed as

$$f_i^{\text{exp}} = 1 - \sum_{\mathbf{q}_k \in Q} \frac{B(a_i, \mathbf{q}_k)}{|Q|} = 1 - \sum_{k=1}^{N} \frac{B_{i,k}}{N}. \tag{36}$$

In the worst case, this takes $|\mathbb{A}_i'|N$ tests and N binary sums per atom where $N = |Q|$ is the sample size, which adds to $O(N)$ basic operations per atom (since we reasoned earlier that $|\mathbb{A}_i'| = O(1)$ under Assumption 1), and a total of $O(nN)$ time for all atoms.

For every sample point, the sequential inner loop of the algorithm can be repeated $3|\mathbb{A}_i'|$ times for computing the variations $f_{i,j,1}^{\text{exp}}$, $f_{i,j,2}^{\text{exp}}$, and $f_{i,j,3}^{\text{exp}}$ used in (35), after introducing the finite difference to the 3 Cartesian coordinates (one at a time) of each neighbor atom $a_j \in \mathbb{A}_i'$. This takes $3|\mathbb{A}_i'|^2 N$ more tests per atom, still asymptotically $O(N)$ but not fast enough in practice. There is a notably more efficient way to do the latter computation by ruling out the subset of sample points that cannot possibly contribute to $f_{i,j,s}^{\text{exp}} - f_i^{\text{exp}}$ $(s = 1, 2, 3)$ in (35) during the first iteration when computing $B_{i,k}$ indicators. In particular, if a sample point is overlapped by more than one neighbor, displacing any neighbor does *not* affect its exposure state (from overlapped to exposed or vice versa), hence it does not contribute to $f_{i,j,s}^{\text{exp}} - f_i^{\text{exp}}$. To leverage this property, we expand the binary definition of the state function to $C : \mathbb{A} \times Q \to \mathbb{Z} \cap [0, \infty)$ such that $C(a_i, \mathbf{q}_k)$ counts the actual number of neighbors $a_j \in \mathbb{A}_i'$ that overlap the sample point $\mathbf{q}_{i,k} \in T_i(Q)$. Three different states for a sample point are observed in terms of the changes in $C_{i,k} := C(a_i, \mathbf{q}_k)$:

1. 'Not overlapped' or 'exposed' $(C_{i,k} = 0)$. In this case, displacing any neighbor either keeps the state at $C_{i,k} = 0$ or changes it to $C_{i,k} = 1$, where the latter case affects the contribution to SASA. Hence the inner loop needs to be repeated for all neighbors (i.e., for $3|\mathbb{A}_i'|$ times).
2. 'Critically overlapped' $(C_{i,k} = 1)$. In this case, the only neighbor whose displacement may change the sample point's state to $C_{i,k} = 0$ is the one that originally overlapped it, and displacing any other neighbor either keeps the state at $C_{i,k} = 1$ or changes it to $C_{i,k} = 2$, both of which correspond to overlapped states that does *not* affect the contribution to SASA. Hence the inner loop is

Algorithm 1: SASA enumeration algorithm for solvation free energy and force computation.

Input: $\mathbf{r}_i, R_i^{\text{off}}, \gamma_i, Q_i,$ and $\mathbb{A}'_i$ for all $a_i \in \mathbb{A}$ $(1 \le i \le n)$;
Output: G_i^{cav} and $\mathbf{F}_i^{\text{cav}}$ for all $a_i \in \mathbb{A}$ $(1 \le i \le n)$;

for $1 \le i \le n$ (seq. or in $\|$) **do**
  Step 1: Energy Computation:
  initialize $f_i^{\text{exp}} \leftarrow 1$;
  initialize $G_i^{\text{cav}} \leftarrow G_{i,0}^{\text{cav}} \leftarrow 4\pi\gamma_i(R_i^{\text{off}})^2$;
  for $1 \le k \le |Q_i|$ (seq. or in $\|$) **do**
    initialize $C_{i,k} \leftarrow 0$; $j_{i,k}^{\text{over}} \leftarrow -1$;
    for $j =$ *indices of atoms in* $\mathbb{A}'_i$ (seq.) **do**
      if $\|\mathbf{q}_{i,k} - \mathbf{r}_j\|_2 \le R_j^{\text{off}}$ **then**
        increment $C_{i,k} \leftarrow C_{i,k} + 1$;
        if $C_{i,k} = 1$ **then**
          //Save critical neighbor index:
          write $j_{i,k}^{\text{over}} \leftarrow j$;
          atomic read+modify+write
            $f_i^{\text{exp}} \leftarrow f_i^{\text{exp}} - 1/|Q_i|$;
            $G_i^{\text{cav}} \leftarrow G_i^{\text{cav}} - G_{i,0}^{\text{cav}}/|Q_i|$;
        else
          if $C_{i,k} \ge 2$ **then**
            write $j_{i,k}^{\text{over}} \leftarrow -1$;
            break;
  Step 2: Force Computation:
  initialize $f_{i,1}^{\text{exp}} \leftarrow f_{i,2}^{\text{exp}} \leftarrow f_{i,3}^{\text{exp}} \leftarrow f_i^{\text{exp}}$;
  initialize $F_{i,1}^{\text{cav}} \leftarrow F_{i,2}^{\text{cav}} \leftarrow F_{i,3}^{\text{cav}} \leftarrow 0$;
  Synchronize for all $1 \le i \le n$;
  for $1 \le k \le |Q_i|$ (seq. or in $\|$) **do**
    for $1 \le s \le 3$ (seq. or in $\|$) **do**
      if $C_{i,k} = 0$ **then**
        for $j =$ *indices of atoms in* $\mathbb{A}'_i$ (seq.) **do**
          if $\|\mathbf{q}_{i,k} - (\mathbf{r}_j + \delta r\mathbf{e}_s)\|_2 \le R_j^{\text{off}}$ **then**
            atomic read+modify+write
              $f_{i,j,s}^{\text{exp}} \leftarrow f_{i,j,s}^{\text{exp}} - 1/|Q_i|$;
              $F_{i,s}^{\text{cav}} \leftarrow F_{i,s}^{\text{cav}} - G_{i,0}^{\text{cav}}/(|Q_i|\delta r)$;
              $F_{j,s}^{\text{cav}} \leftarrow F_{i,s}^{\text{cav}} + G_{i,0}^{\text{cav}}/(|Q_i|\delta r)$;†
            break;
      else
        if $C_{i,k} = 1$ **then**
          write $j \leftarrow j_{i,k}^{\text{over}}$; //note: $j_{i,k}^{\text{over}} \ne -1$
          if $\|\mathbf{q}_{i,k} - (\mathbf{r}_j + \delta r\mathbf{e}_s)\|_2 > R_j^{\text{off}}$ **then**
            atomic read+modify+write
              $f_{i,j,s}^{\text{exp}} \leftarrow f_{i,j,s}^{\text{exp}} + 1/|Q_i|$;
              $F_{i,s}^{\text{cav}} \leftarrow F_{i,s}^{\text{cav}} + G_{i,0}^{\text{cav}}/(|Q_i|\delta r)$;
              $F_{j,s}^{\text{cav}} \leftarrow F_{i,s}^{\text{cav}} - G_{i,0}^{\text{cav}}/(|Q_i|\delta r)$;†
            break;
  write $\mathbf{F}_i^{\text{cav}} \leftarrow (F_{i,1}^{\text{cav}}, F_{i,2}^{\text{cav}}, F_{i,3}^{\text{cav}})$;

//† The instructions that require architecture-specific mutex.

repeated only 3 times after displacing that critical neighbor along the 3 Cartesian axes.

3. 'Multiply overlapped' ($C_{i,k} \ge 2$). In this case, displacing any neighbor either keeps the state at $C_{i,k} \ge 2$ or changes it to $C_{i,k} = 1$, both of which correspond to overlapped states that does *not* affect the contribution to SASA. Hence the inner loop need not be repeated at all.

Therefore, the only changes that contribute a nonzero value to $f_{i,j,s}^{\text{exp}} - f_i^{\text{exp}}$ $(s = 1, 2, 3)$ are those from exposed ($C_{i,k} = 0$) to critically overlapped ($C_{i,k} = 1$) and vice versa, thus a significant amount of computation time can be saved by early detection of the rest. An atom $a_j \in \mathbb{A}'_i$ is called a 'critical neighbor' of the atom $a_i \in \mathbb{A}$ with respect to a sample point $\mathbf{q}_{i,k} \in Q_i$ along a particular direction $\mathbf{e}_s$ $(s = 1, 2, 3)$, if a finite displacement $\delta r\mathbf{e}_s$ results in such a change. As a direct consequence of geometry, if a_j is a critical neighbor of a_i along $+\mathbf{e}_s$, then a_i is also a critical neighbor of a_j along $-\mathbf{e}_s$, both with respect to the same sample point. Therefore, a pair of neighbor atoms $a_i, a_j \in \mathbb{A}$ exchange a solvation force $\pm\delta\mathbf{F}^{\text{cav}} = \pm 4\pi\gamma_i(R_i^{\text{off}})^2/(N\delta r)\mathbf{e}_s$ due to their overlap at the sample point $\mathbf{q}_{i,k}$ if and only if they are critical neighbors with respect to $\mathbf{e}_s$. The improved algorithm (based on the integer-valued $C_{i,k}$) is different from the original (based on the binary-valued $B_{i,k}$) in that the first iteration of the sequential inner loop for computing $C_{i,k}$ terminates after the *second* (rather than the *first*) overlap is encountered, because all $C_{i,k} \ge 2$ have equivalent implications according to the above rules.[9] During this step, the value of f_i^{exp} is initialized to 1 for each atom, and every time a sample point with $C_{i,k} = 1$ or 2 is discovered, f_i^{exp} is decremented by $1/N$. The next 3 repetitions of the inner loop per neighbor atom displacement depend on the aforementioned rules based on the value of $C_{i,k}$. The 3 variants of the exposure ratio $f_{i,j,s}^{\text{exp}}$ $(s = 1, 2, 3)$ are initialized to f_i^{exp} for each atom with respect to displacements in all of its neighbors. Every time a sample point with $C_{i,k} = 0$ or 1 is encountered, the inner loop is repeated with displaced neighbor coordinates to discover the critical neighbors, each adding $\pm 1/N$ to $f_{i,j,s}^{\text{exp}}$.

Significant speed-ups are achieved in terms of the average time, a rigorous analysis of which is not possible without assumptions on the spatial distribution of atoms. However, the worst case time complexity is still $O(nN)$ for the sequential algorithm. One could easily parallelize the algorithm at the outer loops over the atoms and sample points, while the inner loops over the neighbor atoms is best implemented sequentially. On a simple CRCW PRAM machine with common conflict resolution (briefly introduced in Appendix C.1), the parallel running time of $O(nN/P)$ can be achieved in theory using P processors (i.e., linear speed-up), which leads to $O(n)$ if we have $P = O(N)$ processors at our disposal—not far from reality when using GPUs. However, there are more complications to the machine architecture in practice, as will be addressed in Section 4.

[9] Hence one could redefine to $C : \mathbb{A} \times Q \to \{0, 1, \text{“2 or more”}\}$ to implement the same trick with only 3 distinct flags, as in Algorithm 1.

The complete process is described in pseudo-code in Algorithm 1. The instructions marked by a dagger (†) modify variables that belong to different atoms iterated in parallel by the outer-most loop, hence require a mutex with nuances that depend on the architecture as described in Section 4.

3.5 Prefix Sum Calls

There are multiple references in Protofold II to the generic prefix sum routine—explained in Appendix B, which can be performed using optimal sequential and parallel algorithms in linear number of steps—that emerge naturally as a consequence of the linear topology of the polypeptide backbone:

1. Computing link transformations from successive matrix multiplications in (7) and (8), in which the domain is $\Sigma := \mathrm{SO}(3)$ (represented by 3×3 rotation matrices) and the operator $\oplus$ is the matrix multiplication.

2. Computing atom center coordinates from successive vector summations in (9) and (10), in which the domain is $\Sigma := \mathbb{R}^3$ (represented by 3×1 column matrices) and the operator $\oplus$ is the vector summation.

3. Computing joint torques from successive superposition of the contributions of each link on the preceding joints in the chain using (27) and (29), in which the domain is $\Sigma := \mathbb{R}^6$ (represented by 6×1 column matrices) and the operator $\oplus$ is the inner product.

The first item clearly takes $O(n)$ steps, while the latter two take $O(l) = O(m)$ steps (which is also $O(n)$). To explain the last item further, let $[J_k]$ be the k^{th} column of the Jacobian matrix $[J]$ and $[\mathbf{P}_h] := [\mathbf{T}_h^{\text{link}} \ \mathbf{F}_h^{\text{link}}]^{\mathrm{T}}$ be the so-called generalized force on the right-hand side of (27) on the h^{th} link along the chain, both of which are 6×1 column matrices. The contribution of $\mathbf{P}_h$ on the k^{th} joint is obtained as the inner product of the two matrices $\eta_{k,h} = [\mathbf{J}_k]^{\mathrm{T}}[\mathbf{P}_h]$ arranged into the following matrix:

$$[\eta] = \begin{bmatrix} \mathbf{J}_1^{\mathrm{T}}\mathbf{P}_1 & \mathbf{J}_1^{\mathrm{T}}\mathbf{P}_2 & \cdots & \mathbf{J}_1^{\mathrm{T}}\mathbf{P}_l \\ 0 & \mathbf{J}_2^{\mathrm{T}}\mathbf{P}_2 & \cdots & \mathbf{J}_2^{\mathrm{T}}\mathbf{P}_l \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{J}_l^{\mathrm{T}}\mathbf{P}_l \end{bmatrix}, \qquad (37)$$

where $[\eta]$ is an $l \times l$ upper-triangular matrix made of the torque contributions $\eta_{k,h}$, whose h^{th} column's upper nonzero elements form the $h \times 1$ column matrix $[\boldsymbol{\eta}_h]$ introduced in (27). Note that each link only affects the preceding joints in the chain, hence $\eta_{k,h} = 0$ for all $h \leq k-1$. The total torque joints $\tau_k (1 \leq k \leq l)$ can be obtained as a summation over the rows of the above matrix via (29). In Protofold I [46–48] this was accomplished by scanning through the terms along the columns in (37), which took $l(l+1)/2 = O(l^2)$ operations. In Protofold II we perform row scanning of the matrix, starting from the bottom row and moving upwards. More specifically, by factoring out the Jacobian terms $[\mathbf{J}_k^{\mathrm{T}}]$ in each row of (37) and aggregating the generalized forces into $[\mathbf{P}_k^{\text{agg}}] = \sum_{h=k}^{l} [\mathbf{P}_h]$, (29) yields $\tau_k = [\mathbf{J}_k]^{\mathrm{T}}[\mathbf{P}_k^{\text{agg}}]$ as the sum of each row. Then $[\mathbf{P}_k^{\text{agg}}]$ can be obtained in $O(1)$ time from $[\mathbf{P}_{k+1}^{\text{agg}}]$ as $[\mathbf{P}_k^{\text{agg}}] = [\mathbf{P}_k] + [\mathbf{P}_{k+1}^{\text{agg}}]$, which leads to a total of $O(l)$ prefix computation steps.

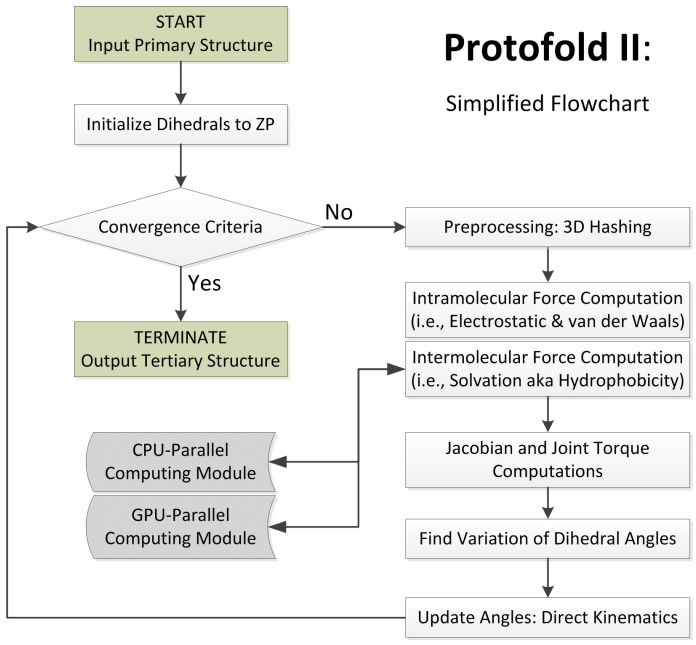

Figure 5: The main process flowchart of Protofold II

4 Implementation

Figure 5 is a schematic of the Protofold II architecture elaborated in Section 4.1. The remainder of this section is dedicated to the parallel the implementation of Algorithm 1 on the CPU in Section 4.2 and on the GPU in Section 4.3, which are identified by the alternative shaded modules in Fig. 5.

4.1 Protofold II Architecture

Unlike Protofold I [46–48] that was programmed in Matlab®, Protofold II is reprogrammed with a new architecture in C++. The CPU- and GPU-parallel algorithms are implemented as external modules and linked to the main application thread as dynamic link libraries (DLL), which can be integrated into other folding packages.

As depicted in Fig. 5, a typical KCM simulation in Protofold II can be summarized into the following steps:

1. **Input:** The user specifies a primary structure (i.e., AA sequence information) to the interface.

2. **Preprocessing:** The program constructs the AA chain using the structural assumptions given in Section 2.1 to arrange the atoms into the consecutive peptide planes. The double-bond angles are all set to the fixed values of $\omega_i = 0^\circ$ (cis) or -180° (trans) and the body vectors are assigned with the values given in Table 1.

3. **Initialization:** The main chain dihedral angles are initialized as $\phi_i^0 = \psi_i^0 = -180^\circ$ and the side chain dihedral angles are initialized to rotameric default values $\chi_{i,k}^0$ [45] for $1 \leq i \leq n, 1 \leq k \leq l_i \leq 4$—i.e., set all $\theta_{j,k}^0 = 0^\circ$ in (1) through (3) referred to as ZP initial conditions.

4. **Forward Kinematics:** The conformation variables summarized in Table 2 are converted to the Cartesian coordinates of the individual atoms by using the sequence of rigid body transformations described in Section 2.1.

5. **Coordinate Hashing:** Using the 3D grid data structure presented in Section 3.2, the atom coordinates are arranged into buckets for fast neighborhood queries based on the cut-off distances.

6. **Force Computations:** The free energy- and force-fields are computed from the atom coordinates using the equations given in Section 2.2. This is where the CPU- or GPU-parallel modules are called for computing the solvation effects.

7. **Torque Computations:** The forces on the atoms are converted to joint torques using the Jacobian transformation described in Section 2.2.

8. **KCM Stepping:** The kinetostatic effect of the joint torques are computed using the simple steepest-descent stepping explained in Section 2.3.

9. **Termination:** If the convergence criteria is met, the program terminates; otherwise it repeats the steps 4 through 8 above.

10. **Output:** The intermediate (every several frames) and final conformations in PDB format, the variations of the dihedral angles and free energy terms, and the performance measures (e.g., running times of different steps) are exported by the program.

These steps characterize the process of arriving from sequence configuration (i.e., primary structure) to stable 3D conformation (i.e., tertiary structure) without any additional assumptions. Although this is the ultimate goal of protein folding, it is rather ambitious to obtain results that are consistent with experimental observations except in the case of relatively short chains; e.g., folding simulation of $\alpha-$helix coiling described in section 5.1. This is due to a variety of reasons ranging from the sensitivity of the folding pathway to the physical parameters (e.g., adjusted coefficients in the empirical force-field equations) to the sensitivity of the spatial structure of long chains to simplifying geometric assumptions (e.g., the exact planarity of the peptide planes).

Additional Functionalities. In order to enable addressing certain computer-aided structural studies on real proteins effectively in spite of the aforementioned difficulties, we found it imperative to include the following additional functionalities in Protofold II:

- The user has the option to 1) specify only sequence data, from which the 'canonical' peptide plane geometry (i.e., assuming exact planarity $\omega_i = 0^\circ$ (cis) or -180° (trans) and average lengths in Table 2); or 2) import the protein structure as a PDB file and retain the peptide group geometry as-read when constructing the rigid links.

- The user has the option to limit the mobility of the linkage by fixing as many dihedral angles as desired. This enables folding studies at multiple levels and different scales. For example, it is possible to group collections of AAs (e.g., secondary elements, motifs, domains, etc.) into presumed rigid bodies and limit the DOF to deformations at the loops connecting them.

- In addition to the ZP initial conditions, the user may choose to use other initial conditions, including but not limited to completely random initial conditions or the native conformation perturbed by arbitrary (deterministic or randomized) changes to certain dihedral angles.

- When importing PDB files, the program eliminates water molecules—since their effect is implicitly incorporated by the solvation energies—but retains other heteroatoms (e.g., metal ions, co-factors, substrates, etc.) and includes them among chain atoms when computing the force-field. This is crucial since the proper folding of many proteins is dependent on these agents.

In addition to the above features, the following need to be included in future versions:

- The program currently supports monomeric protein folding in its simplest topology. It is desirable to enable multimeric protein folding by maintaining multiple chains bound together (i.e., quaternary structure) and more complex topologies induced by other effects (e.g., disulfide bonds, hydrogen bonds, lipidation, etc.)

- the simplistic steepest-descent search process presented in Section 2.3 has not evolved much since Protofold I [46–48]. Our numerical experiments suggest that better optimization algorithms such as a hybrid Monte Carlo sampling combined with steepest-descent or conjugate-gradients KCM[10] could be more effective in avoiding local minima and enable faster convergence to the global minimum.

Parallel Implementations. As demonstrated in Section 3.4, the solvation energy and force computations using (24) and Algorithm 1 are the most time-consuming steps of each KCM iteration, mainly due to the large number of sample points $|Q| = N \gg 1$ required to enumerate the offset sphere of each atom for an adequate approximation of SASA and its gradient. To benefit from the single-instruction multiple-data

[10]This module is already implemented into Protofold II but not tested yet, as the focus of this article is on the improved model and implementation of the force-field.

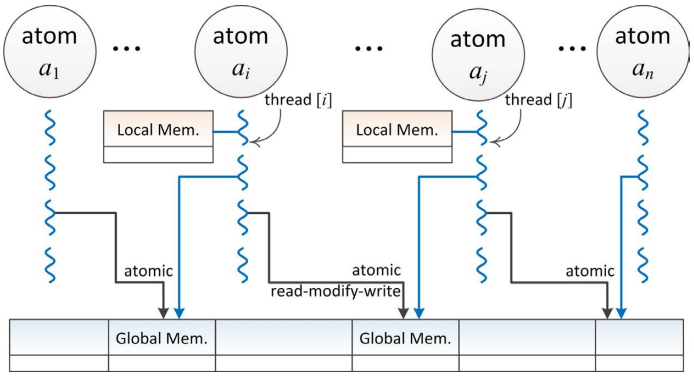

Figure 6: Thread execution model on the CPU.

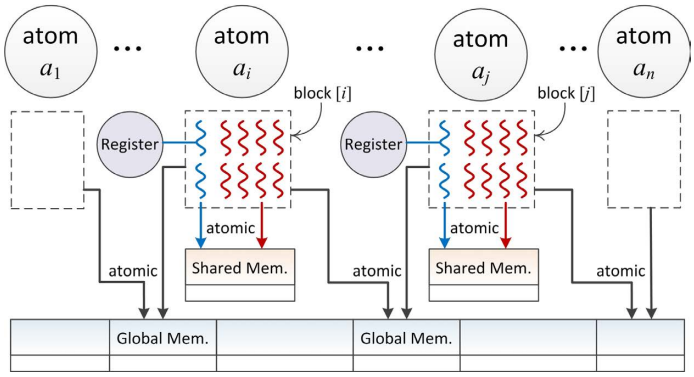

Figure 7: Thread execution model on the GPU.

(SIMD) characteristic of Algorithm 1, the variables pertaining to different atoms are assigned to different processors. The two terms on the right-hand side of (24) are computed concurrently by different processors assigned to $a_i, a_j \in \mathbb{A}$ and broadcasted to each other to minimize the computational work. An immediate consequence is an additional communication overhead and possible network contention due to concurrent write attempts. Such a trade-off between computation and communication intensities is a common characteristic of parallel algorithms [85], and will be considered here for code optimization.

Here we focus on the implementation of the SIMD Algorithm 1 using two parallel computing models; namely,

- one that is designed for coarse-grained multiprocesser machines such as multi-core CPUs (Section 4.2); and
- another that is designed for fine-grained multiprocesser machines such as many-core GPUs (Section 4.3).

4.2 CPU-Parallel Implementation

The first parallel implementation targets a coarse-grained shared-memory multiprocessor machine, i.e., one with a multi-core multi-thread CPU. Given n atoms n threads are generated, assigning one thread per atom a_i $(1 \leq i \leq n)$. The neighborhood information (i.e., the list of indices of all $a_j \in \mathbb{A}_i$ defined in (31) for each atom $a_i \in \mathbb{A}$) is constructed and saved in the *global* memory shared among all processors, hence can be accessed concurrently from different threads. Each thread stores the sample point coordinates, the SASA and its gradient, and the resulting solvation energy and force components in the *local* memory of the processor. The thread iterates sequentially over the sample points on the offset sphere, and a counter variable that keeps track of the number of overlapped sample points is initialized within the scope of the thread. For each sample point, the coordinates are computed and tested sequentially against all neighbors to obtain $C_{i,k}$ $(1 \leq i \leq n, 1 \leq k \leq |Q_i|)$.

Once the exposure states are obtained for the original configuration of the neighbors, the thread loops over all neighbors one more time to examine the effects of their displacement along the 3 coordinate axes one at a time. If certain criteria given in the previous section are met, the pair of force components $\pm\delta F^{\text{cav}} = \pm 4\pi\gamma_i (R_i^{\text{off}})^2/(N\delta r)$ need to be added to the total solvation forces of two neighbor atoms a_i and a_j along the proper coordinate axis, and in opposite directions. This results in two write operations per incidence, the first of which modifies $\mathbf{F}_i^{\text{cav}}$ of a_i, which is safely assigned to the current thread and occurs in the local memory without any concern related to communication between the threads. The second write operation, on the other hand, modifies $\mathbf{F}_j^{\text{cav}}$ of a_j, a variable assigned to a different thread. This requires communication between the two threads, and has to be implemented using atomic write operations into the global memory to guarantee mutual exclusion. Figure 6 shows the multi-threading scheme for the CPU-parallel algorithm. The algorithm is implemented using the OpenMP library. Although linear speed-up is expected in theory on an abstract CRCW PRAM, the actual speed-up is sublinear (as depicted in Section 5.2) in practice due to bus traffic, network contention, cache invalidations, and serialized operations.

CPU Optimization. The number of CPU cores is generally much smaller that the number of atoms ($p \ll n$). Nevertheless, it is good practice to generate more threads than the number of cores to maximize the performance by keeping the processors saturated at all times with computational work. Accessing global memory incurs latency at the incidence of a cache miss and multithreading is a standard technique for hiding such latencies. The computation instructions are interleaved with memory access instructions, hence every time one thread is accessing the global memory the processor can switch the context to a different thread. Other optimization attempts include using local memories instead of global memories whenever possible, and avoiding multiple computations of constant parameters or variables that are used repeatedly.

4.3 GPU-Parallel Implementation

The second parallel implementation targets a fine-grained machine with a hierarchical memory architecture, i.e., one with a many-core many-thread GPU. Given n atoms, a lin-

ear grid of n blocks is generated, assigning one block per atom a_i $(1 \leq i \leq n)$. Each block is further divided into $N = |Q|$ threads, assigning one thread per sample point $\mathbf{q}_k \in Q$ $(1 \leq q \leq n)$ on the unit offset sphere. Prior to GPU kernel execution, the neighborhood information (i.e., the list of indices of all $a_j \in \mathbb{A}_i$ defined in (31) for each atom $a_i \in \mathbb{A}$) is transferred from the CPU (i.e., *host*) memory to the GPU (i.e., *device*) memory. For each thread, the iteration over different neighbors is performed sequentially, similar to the CPU-parallel code presented in 4.2. The solvation energy and force components, a counter that keeps track of the number of overlapped sample points, and exposure states are initialized in the *shared* memory of the blocks, which require atomic operations for access safety by multiple threads, while sample point coordinates are stored in the *registers* that are local to each thread. Each sample point is tested sequentially against all neighbors to obtain $C_{i,k}$ $(1 \leq i \leq n, 1 \leq k \leq |Q_i|)$.

Once the exposure states are obtained for the original configuration of the neighbors, the thread loops over all neighbors one more time to examine the effect of their displacement along the 3 coordinate axes one at a time. Similar to the CPU-based implementation, whenever the pair of force components $\pm\delta F^{\text{cav}} = \pm 4\pi\gamma_i (R_i^{\text{off}})^2/(N\delta r)$ need to be added to the total solvation forces of two neighbor atoms a_i and a_j along the proper coordinate axis, the write operation that modifies $\mathbf{F}_i^{\text{cav}}$ of a_i happens atomically in the shared memory. This ensures mutual exclusion between threads of the same block. On the other hand, the write operation that modifies $\mathbf{F}_j^{\text{cav}}$ of a_j happens atomically in the global memory to ensure mutual exclusion between blocks of the same grid. Figure 7 shows the multi-threading scheme for the GPU-parallel algorithm. The algorithm is implemented using NVIDIA's compute-unified device architecture (CUDA). Kernel invocation is carried out synchronously within the default CUDA stream, hence synchronization between blocks is automatically guaranteed, while barrier synchronization is needed between threads of the same block.

GPU Optimization. The optimization attempts can be categorized as memory, execution, instruction, and flow-control optimization.

- Memory optimization is the most effective of all, as demonstrated by the results in the Section 5.2. In contrast to the CPU-parallel algorithm that makes most references through the cached global memory, the GPU-parallel algorithm transfers the coordinates, radii, solvation parameters, and neighbor index lists for each atom into the shared memory to minimize the number of global memory references. The variables that are exclusive to the threads, on the other hand, such as the exposure states or sample coordinates are allocated in the registers. However, the limited amount of shared memory and register resources are limited on the streaming multiprocessor (SM) imposes a restriction on the number of resident blocks on the SM and can adversely affect thread occupancy at any time during the simulation. Therefore, one needs to avoid excessive variable definitions within the scope of the GPU kernels.

- For execution level optimization, the kernels should be executed with proper granularity to maximize SM thread occupancy. Specifying a larger number of threads per block generally contributes to latency hiding, but is limited by the architecture as well as the on-chip memory resources. The number of threads is the same as the sample size $N = |Q|$ a proper choice of which is a trade-off between accuracy and performance.

- For instruction level optimization, the transcendental math functions are converted to their intrinsic alternatives that are executed on the special function units (SFU) of the CUDA cores.

- Flow-control optimization is realized by avoiding multiple execution paths within the same block, which might lead to thread divergence and serialization within the same warp. In particular, when checking for overlaps between neighbor atoms and sample points, the conditional (e.g., if/else/then) statements are set in such a way that one of the two execution paths is always null.

The near-optimal conditions are reached by successive experimentation and modification of the code. For more information regarding the GPU architecture and terminology, see Appendix C.2.

5 Results & Discussion

This section presents a preliminary assessment of the model and implementation enhancements from Protofold I [46–48] to Protofold II. The folding process is simulated and assessed at multiple levels, ranging from the formation of secondary structural elements (e.g., $\alpha-$helix coiling or $\beta-$strands formation) from an open chain to tertiary interactions between secondary elements or across larger domains that can be assumed to be rigid in real protein examples.

In Section 5.1 we discuss the impact of introducing solvation effects on the folding process of secondary structural elements starting from different initial conditions. We present some performance measures in Section 5.2 to validate the practical benefits of algorithmic improvements (e.g., coordinate hashing) as well as implementation improvements (i.e., CPU- and GPU-parallel computing). Finally, we look at a few real protein molecules in Section 5.3 and examine the energy variations in the neighborhood of the native structures.

5.1 The Folding Process

The simplest structural elements that are ubiquitous across many protein domains are $\alpha-$helices and $\beta-$strands. Here we start by considering simple test runs on relatively short (e.g., 10-20 residues long) peptide chains made of Ala residues

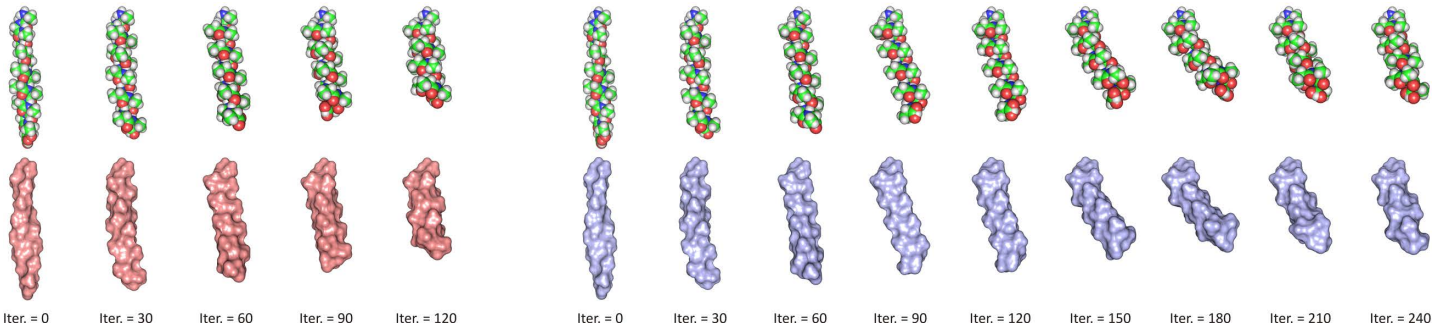

Figure 8: Left-handed $\alpha-$helix formation for a 15-residue polyalanine chain in vacuum (left) and in water (right) starting from $\phi_i^0 = \psi_i^0 = +10^\circ$ using Protofold II. Initial conditions and solvation effects dramatically affect the folding pathway.

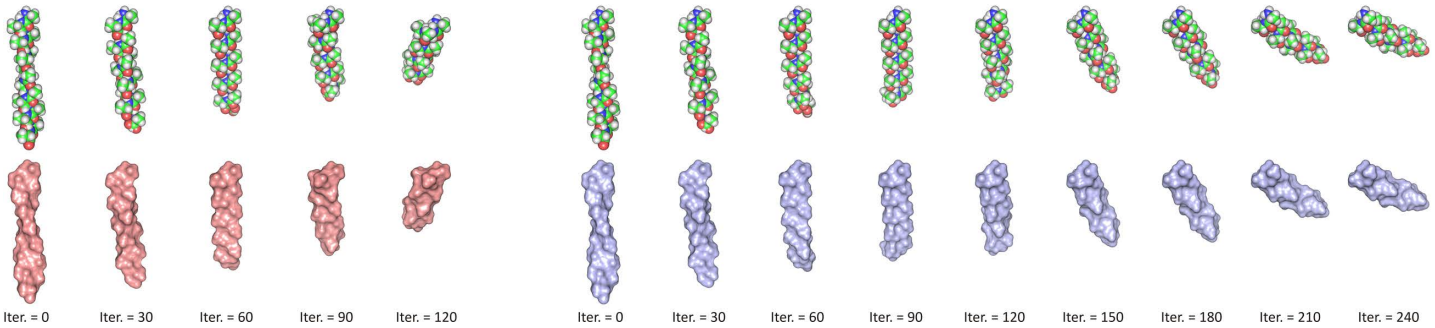

Figure 9: Right-handed $\alpha-$helix formation for a 15-residue polyalanine chain in vacuum (left) and in water (right) starting from $\phi_i^0 = \psi_i^0 = -10^\circ$ using Protofold II. Initial conditions and solvation effects dramatically affect the folding pathway.

(typically used as the benchmark AA type in its most common L-stereoisomeric form) to visualize their compliance into secondary structural elements.

Alpha Helix Formation. First, we run four tests on a 15-residue chain starting from two different initial conditions, for both of which we simulate the folding process without and with solvation effects taken into account:

1. Starting from the (slightly pre-coiled) initial conditions $\phi_i^0 = \psi_i^0 = +10^\circ$ for all $1 \leq i \leq 15$ the chain folds into a left-handed $\alpha-$helix as depicted in Fig. 8. The energy variation during KCM iterations is given in Fig. 10.

2. Starting from the (slightly pre-coiled) initial conditions $\phi_i^0 = \psi_i^0 = -10^\circ$ for all $1 \leq i \leq 15$ the chain folds into a right-handed $\alpha-$helix as depicted in Fig. 9. The energy variation during KCM iterations is given in Fig. 11.

The folding process in vacuum, i.e., without considering the solvation effects, emulates the behavior in the absence of the polar solvent, e.g., in membrane proteins extended along the nonpolar lipid bilayer or in secondary structural elements wrapped inside the hydrophobic core of globular proteins [3]. On the other hand, the folding process in water, i.e., with the presence of the solvation effects in addition to the intramolecular interactions, emulates the formation of elements that reside at the hydrophilic surface of globular proteins [3].

In the case of $\alpha-$helix formations in Figs. 8 and 9, the hand of the initial coil determines the hand of the final helix.[11] This is due to the gradient descent nature of the KCM search algorithm that tends to converge to different local minima depending on the initial state. The effects of the solvation are hard to observe in these examples with the energetically unchallenged helical structures due to proper stacking of the atoms favored by all considered effects. In both cases the van der Waals and solvation effects work in the same direction until the steric clash prevents the helix to coil further.

Figures 10 and 11 are plots of the free energy variations versus KCM iteration number for the four runs described above. Note that in all four cases (top and bottom plots) the solvation energy is evaluated and plotted, but only in two of them (bottom plots) its effects are applied to deform the chain. For both left and right-handed helix formation, the inclusion of solvation effects clearly changes the folding pathway and increases the number of iterations before convergence from around 150 to 300. However, in either case the solvation free energy changes are not as significant as those of intramolecular (particularly van der Waals) effects. Another important observation is that the right-handed $\alpha-$helix exhibits a notably more stable conformation than the left-handed $\alpha-$helix with about $\sim 40-50$ kcal per mol lower total free energy state—to be accurate, 43.8 and 45.9 kcal per mol for the

[11] The surface visualizations can be deceiving where the right-handed helix appears to have a left-handed twist and vice versa. This is due to the transversal ridges and grooves formed in between the side chains [3].

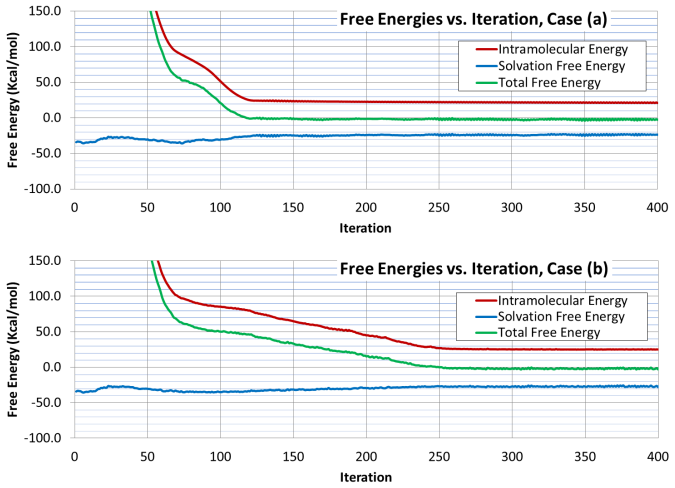

Figure 10: Free energy variations during folding of a 15-residue polyalanine chain into a left-handed $\alpha-$helix in vacuum (top) and in water (bottom) using Protofold II.

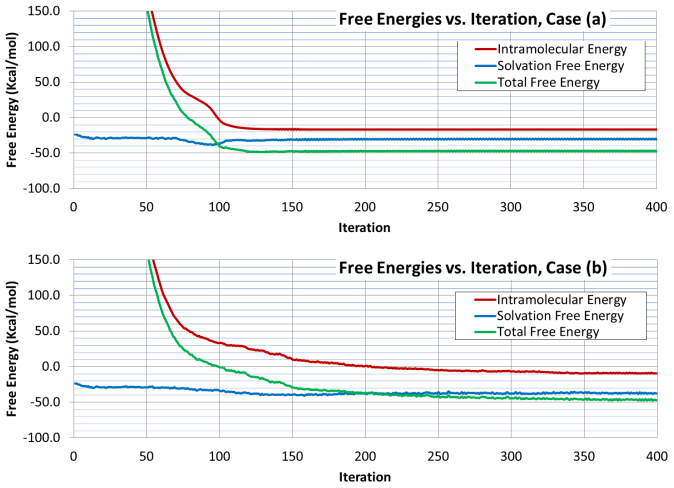

Figure 11: Free energy variations during folding of a 15-residue polyalanine chain into a right-handed $\alpha-$helix in vacuum (top) and in water (bottom) using Protofold II.

Table 4: Final conformations of a 15-residue polyalanine chain folded in vacuum using Protofold II.

	Left-handed coil $\phi_i^0 = \psi_i^0 = +10°$				Right-handed coil $\phi_i^0 = \psi_i^0 = -10°$			
	In vacuum		In water		In vacuum		In water	
i	$\phi_i(°)$	$\psi_i(°)$	$\phi_i(°)$	$\psi_i(°)$	$\phi_i(°)$	$\psi_i(°)$	$\phi_i(°)$	$\psi_i(°)$
1	+10.0	+118	−7.20	+21.5	−10.0	−59.1	+15.1	+54.4
2	+60.2	+53.9	+59.9	+47.6	−82.4	−41.1	−107	−47.1
3	+58.9	+36.8	+58.6	+40.0	−76.9	−24.8	−91.9	+11.3
4	+56.8	+45.4	+57.9	+45.1	−75.7	−34.4	−81.1	−34.4
5	+57.9	+42.2	+58.8	+40.3	−75.8	−28.4	−83.1	−23.8
6	+56.7	+43.7	+57.8	+43.6	−75.3	−31.3	−76.7	−32.1
7	+57.3	+42.7	+57.9	+43.6	−76.1	−29.3	−79.4	−28.9
8	+56.8	+44.0	+58.1	+41.6	−75.6	−30.5	−77.1	−28.9
9	+56.8	+42.3	+58.3	+42.1	−76.3	−29.2	−79.8	−29.2
10	+56.5	+45.5	+58.4	+43.3	−76.5	−29.7	−78.4	−28.3
11	+56.2	+42.4	+57.9	+41.7	−77.5	−29.5	−79.4	−28.6
12	+55.1	+48.4	+56.3	+46.0	−75.6	−33.5	−78.9	−28.7
13	+53.5	+45.5	+55.6	+45.0	−72.1	−39.0	−75.0	−33.4
14	+49.2	+59.4	+52.3	+52.6	−63.1	−44.4	−72.6	−40.0
15	+53.2	+69.8	+53.2	+72.7	−64.3	−53.5	−65.4	−48.0
Ave.	+53.0	+52.0	+52.9	+44.4	−70.2	−35.9	−74.1	−24.4

entire chain, i.e., 2.9 and 3.1 kcal per mol per AA residue, without and with solvation effects, respectively. Although this is qualitatively consistent with the expectation of right-handed coiling being favored by L-alanine chains, the energy differences are higher than the ones reported in earlier studies (e.g., MD results in [86]). However, a meaningful comparison would require using identical simulation parameters, which is beyond the scope of this paper.

The final dihedral angles for all 15 Ala residues corresponding to the folded (i.e., stable) conformations, obtained after a large enough number of iterations, are given in Table 4.

Ramachandran Plots. To examine the local effects of energetics, Ramachandran plots (for a pair of Ala residues in tandem) are generated by Protofold II using the energy-field presented in Section 2.2. The plots in Fig. 12 show the energy variations across different pairs (ϕ, ψ) of dihedral angles, without and with considering the solvation effects. The vertical high-energy regions in the middle corresponds to prohibitive steric clashes between the atoms. The low-energy regions around it, on the other hand, correspond to the geometric relations between consecutive peptide planes that, when repeated for a segment of multiple residues along the chain, create secondary structural elements such as coiled $\alpha-$helices and flat $\beta-$strands. Although the solvation effects do not significantly alter the shape of the energy profile, there is a certain amount of noise added due to the discrete nature of the enumeration algorithm presented in Section 3.4.

To observe the effects of solvation, one needs to carry out more extensive simulations on larger data sets with different chain lengths and various initial conditions. We carried out KCM runs on 2,000 independent polyalanine chains of random lengths in the range $10 \leq m \leq 20$ starting from random initial angles in the range $-90° \leq \phi_i^0, \psi_i^0 \leq +90°$. The tests were run separately without and with considering the solvation effects using the same random seed. The resulting dihedral angles for all 27,717 Ala residues of all chains[12] are plotted in Fig. 13. One can observe multiple concentration areas that clearly correspond to left- and right-handed $\alpha-$helices and flat $\beta-$strands, the former two helical folds being more populated on the plots. Zooming further on the two $\alpha-$regions reveals multiple local minima where the points are concentrated more, which correspond to different subtypes of $\alpha-$helices. Comparing the two plots in Fig. 13 reveals that the solvation effects do not significantly change the locations of the local minima. However, the energy profile is relaxed around the local minima where it has sharp cracks traced by the concentrated points along the valleys of the intramolecular energy landscape.

[12]The angles for the first AA residue of all chains are eliminated as outliers since they differ dramatically from those of the subsequent residues due to the anchoring at the N-terminus (see Table 4).

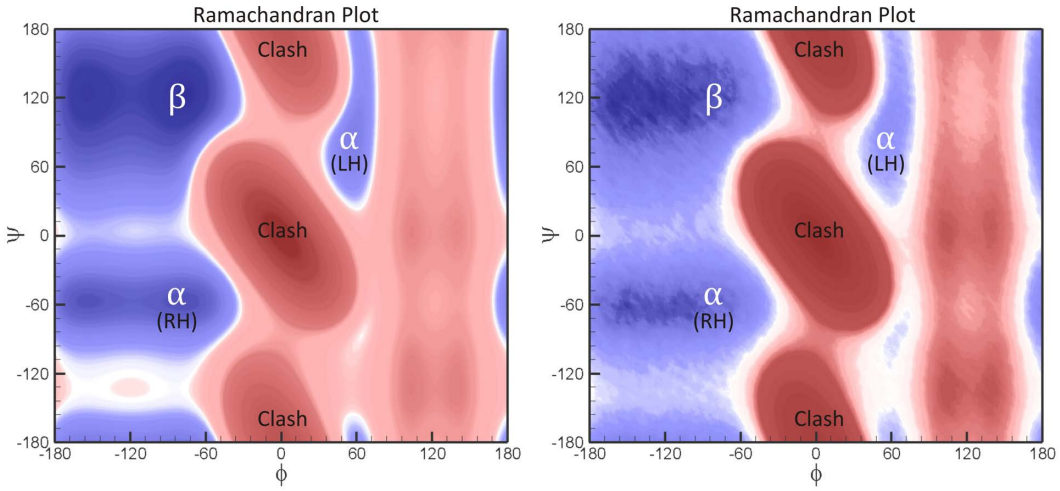

Figure 12: The Ramachandran plots for the energy variations of a pair of Ala residues in vacuum (i.e., without solvation effects) (left) and in water (i.e., with solvation effects) (right) using Protofold II.

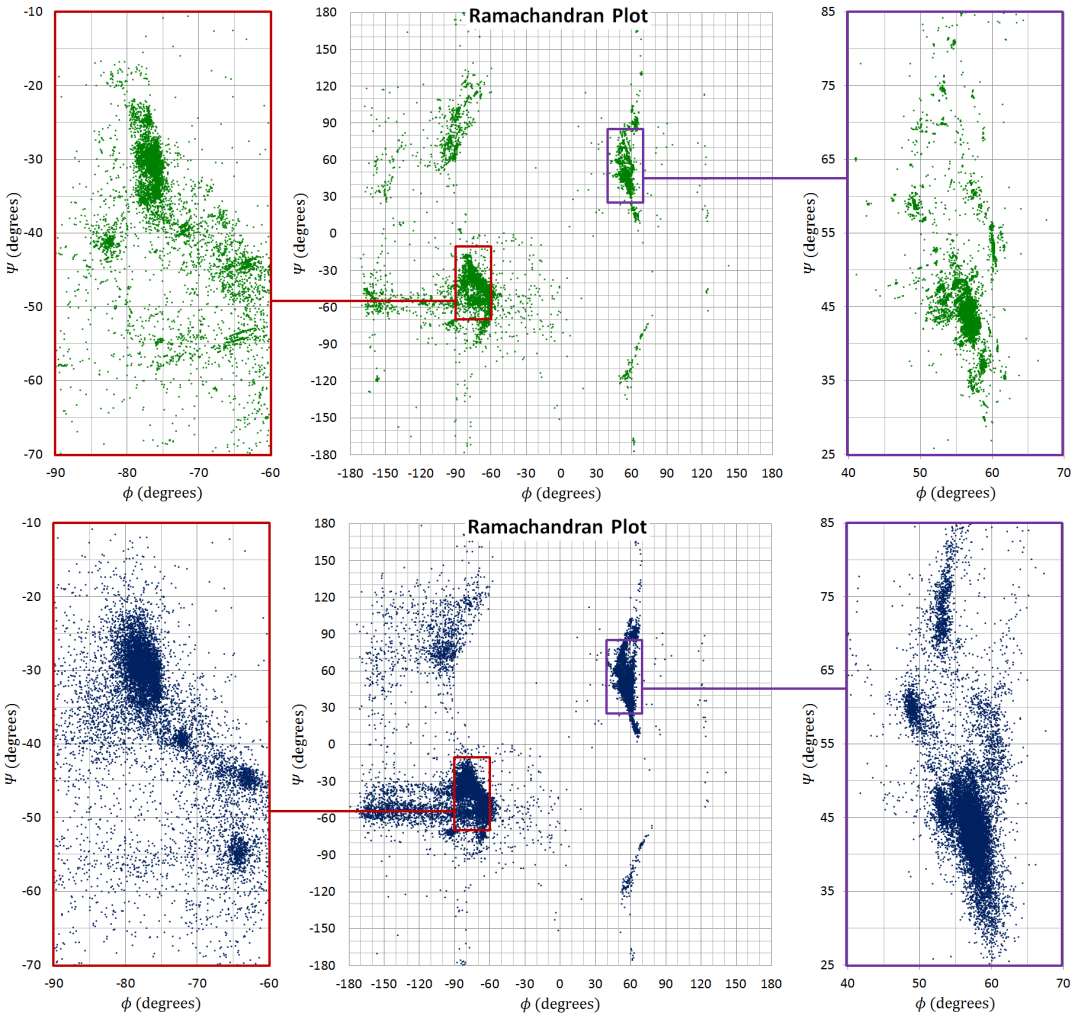

Figure 13: The Ramachandran plots for the folded backbone conformation of 2,000 10- to 20-residue polyalanine chains in vacuum (i.e., without solvation effects) (top) and in water (i.e., with solvation effects) (bottom) starting from random initial conditions $-90^\circ \leq \phi_i^0, \psi_i^0 \leq +90^\circ$ using Protofold II.

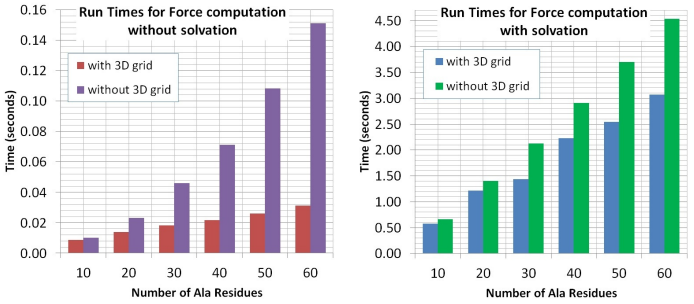

Figure 14: The effect of 3D hashing on the force computation times of a 60-residue polyalanine chain with and without solvation effects (on **C-1**).

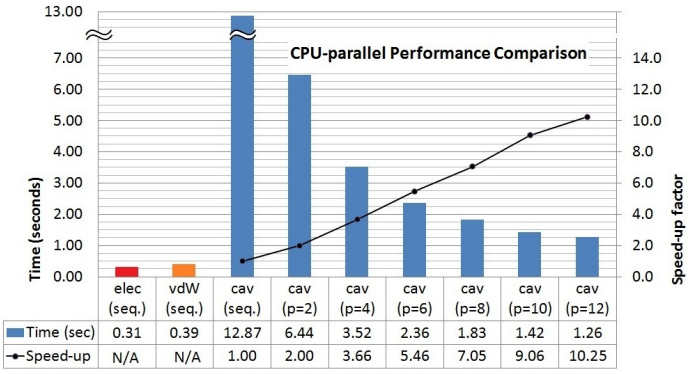

Figure 15: Sequential running times for electrostatic, van der Waals, and solvation forces, and CPU-parallel running times of the latter for a 1,200-residue polypeptide chain (on **C-1**).

5.2 Computation Times

Next, we demonstrate the substantial performance improvements in Protofold II as a result of introducing the algorithms and data structures presented in Section 3. The running times are reported and compared on two computer systems, namely:

- **C-1**: Dell Precision T7500 workstation with an Intel® Xeon® E5645 CPU (12 cores, clock rate 2.40 GHz, and host memory 24 GB). The system is equipped with one NVIDIA Quadro® 4000 GPU (256 CUDA cores with compute capability (CC) 2.0 and device memory 2 GB).
- **C-2**: Dell Precision T7600 workstation with an Intel® Xeon® E5-2687W CPU (32 cores, clock rate 3.10 GHz, and host memory 64 GB). The system is equipped with two graphics cards: a NVIDIA Quadro® K5000 GPU (1,536 CUDA cores with CC 3.0 and device memory 4 GB) and a NVIDIA Tesla® K20C GPU (2,496 CUDA cores with CC 3.5 and device memory 5 GB).

Effect of Hashing. Figure 14 shows the running times of the force computation step (on **C-1**) in a single KCM iteration for folding polyalanine chains of different lengths with and without 3D hashing presented in Seciton 3.2 both in vacuum (i.e., without considering solvation effects) and in water (i.e., with considering solvation effects). In both cases, the results show a significant reduction in the running times with hashing (e.g., up to 4.6× in vacuum and 1.5× in water for $m = 60$ Ala residues), and the difference scales with the size of the molecule. Nevertheless, the solvation force computations remain the bottleneck of the simulation (using sequential CPU implementation) and adversely affect the speed-up gained from hashing.

Parallel Computing. The first 3 columns on the left in Fig. 15 show the sequential CPU running times (on **C-1**) for the electrostatic, van der Waals, and nonpolar solvation forces in a single KCM iteration. The results were obtained for a polypeptide chain composed of $1,200$ residues that are randomly selected from Ala, Cys, and Ser AAs—the latter two being equivalent to replacing the H in the nonpolar Ala side chain with SH and OH, respectively, resulting in polar side chains. It is clearly observed that with a single CPU core, the first two terms take a very small fraction of the total time (around 5–6% of the total force computation time per iteration) and the solvent effects are clearly the bottleneck.

The same column chart in Fig. 15 also shows the running times and corresponding speed-ups (on **C-1**) for the CPU-parallel computation of the solvation force using up to 12 CPU cores. An almost linear speed-up is achieved by increasing the number of processors (e.g., $\sim 10\times$ with 12 cores). However, the solvation force calculation is still about an order of magnitude slower than that of the other two force types.

Figure 16 compares the running times and corresponding speed-ups (on **C-1**) of the CPU- and GPU-parallel implementations for polypeptides of different lengths, ranging from $m = 200$ (i.e., $\sim$ 2K atoms) to $1,200$ AAs (i.e., $\sim$ 13K atoms). To depict the importance of memory optimization presented in Section 4.3, the running times are shown for two different cases; namely, one that uses global memory for communications between all threads of all blocks, and the optimized code making extensive use of shared memories for communications between threads within the same block. It is interesting to note that when the GPU shared memory is not utilized, the results do not show an improvement over the 12-core CPU implementation due to large global memory access latencies. However, proper shared memory usage results in a huge performance improvement that scales by protein size, e.g., from $20\times$ for $m = 200$ AAs to $70\times$ for $m = 1,200$ AAs with respect to the sequential run on a single CPU.

The computations are repeated for even larger molecules in Fig. 17, ranging from $m = 1,000$ AAs (i.e., $\sim$ 10K atoms) to $6,000$ AAs (i.e., $\sim$ 64K atoms), enabled by leveraging a more powerful machine (i.e., **C-2**). For the case of $m = 1,000$ AAs, **C-2** yields a two-fold speed-up on the CPU and a three-fold speed-up on the GPU compared to **C-1**. As depicted in Fig. 17, significantly higher and more consistent CPU speed-ups of 16-18× and GPU speed-ups of 90-100× (for Quadro® K5000) and 270-290× (for Tesla® K20C) are

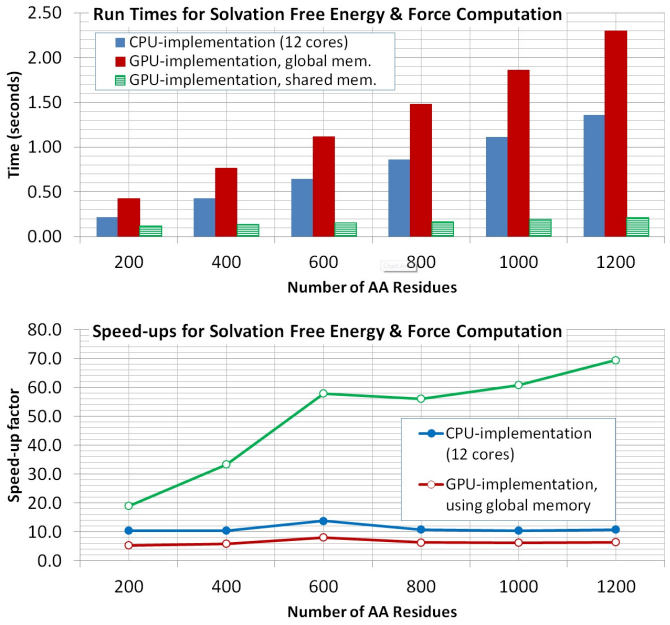

Figure 16: CPU- and GPU-parallel running times (top) and speed-ups (bottom) with and without memory optimization for polypeptide chains of various lengths (on **C-1**).

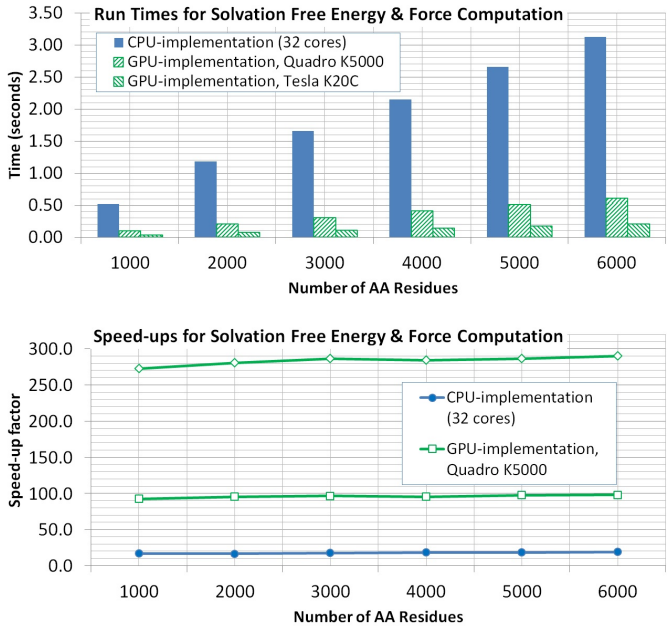

Figure 17: CPU- and GPU-parallel running times (top) and speed-ups (bottom) with memory optimization on two GPUs for polypeptide chains of various lengths (on **C-2**).

observed. These observations imply proper scalability of the data-parallel implementation with molecular size.

Even for molecules with tens of thousands of atoms, each force-field computation takes only a fraction of a second per iteration. This enables fast KCM simulation of folding for large proteins over extended periods of time via Protofold II, which wouldn't be tractable via Protofold I [46–48].

5.3 Real Examples

Having considered the folding of secondary structural elements with complete flexibility (i.e., each peptide plane being treated as a separate rigid body), we proceed to study the tertiary interaction between larger rigid units in real proteins. We report on the following case studies:

- The interactions between multiple rigid $\alpha-$helices that rotate around flexible loops within the containing motifs/domains are considered. Two examples from the PDB are used for this purpose: Myoglobin (PDB: 1TES) and Troponin-C (PDB: 2JNF).
- The interactions between multiple rigid domains that are connected via flexible loops within the containing monomeric unit are examined. The example of Gamma-B Crystallin (PDB: 1GCS) is used for this purpose.

These PDB structures that are obtained from X-ray crystallography (e.g., 1TES and 1GCS) do not contain the H atoms, hence are first preprocessed using Duke University's MolProbity server [87,88] which predicts and adds the H atoms to the coordinates information before importing the structure into Protofold II. The PDB structures that are obtained from nuclear magnetic resonance (NMR) spectrometry (e.g., 2JNF), on the other hand, already contain the H atoms positions. The size of the molecules, i.e., the number of AA residues m and the number of atoms n' and n with and without the H atoms included, respectively, are given in Table 5.

Table 5: The size of the analyzed protein examples.

Protein Name	PDB Code	m	n'	n
Myoglobin	1TES	154	1,231	2,478
Troponin-C	2JNF	158	1,232	2,401
Gamma-B Crystallin	1GCS	174	1,474	2,844

Secondary Structural Interactions. Let us first consider the energy variations when an $\alpha-$helix of an $\alpha-$domain is reoriented from its stable conformation with respect to the rest of the bundle, as illustrated in Figs. 18 and 20.

Myoglobin (PDB: 1TES) is an oxygen binding muscle protein that is composed of a single 'globin fold' domain, which is an $\alpha-$domain motif consisting of a bag of 8 $\alpha-$helices per domain (denoted A through H) arranged at $\sim +90°$ and $+50°$ angles with respect to each other, as shown in Fig. 18. This arrangement creates a hydrophobic pocket in the interior that wraps the stabilizer co-factor known as 'heme group' [89]. Assuming that the helices are rigid, we examine the energy variations due to dihedral rotations at the loops that connect the two end $\alpha-$helices; namely, local changes in (ϕ_i, ψ_i) for $i = 21$ and $i = 125$ where the A and H helices are hinged, respectively.

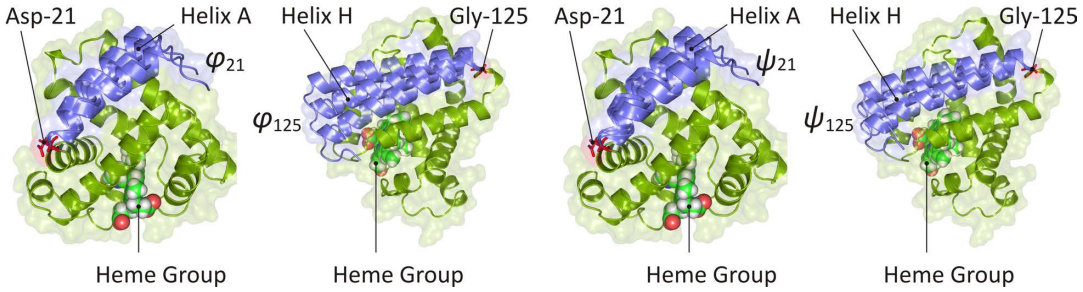

Figure 18: Myoglobin (PDB: 1TES) hinged at Asp-21 and Gly-125 for variations in ϕ_{21}, ϕ_{125} (left) and ψ_{21}, ψ_{125} (right).

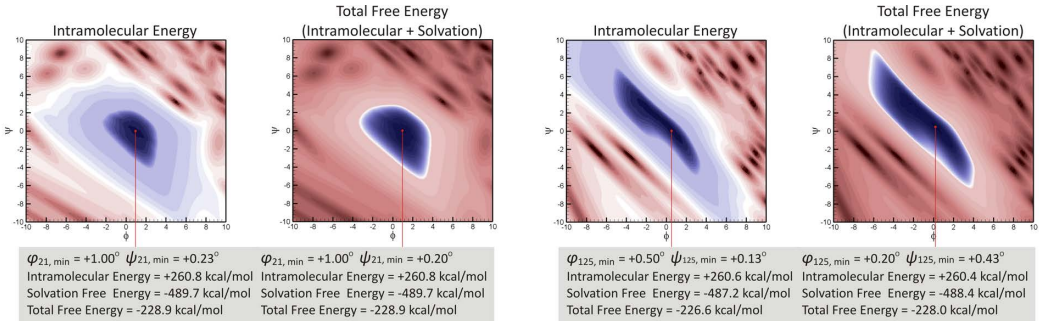

Figure 19: Intramolecular versus total (i.e., solvation included) free energy landscape for Myoglobin (PDB: 1TES) in the vicinity of the native conformation versus variations in (ϕ_{21}, ψ_{21}) (left) and (ϕ_{125}, ψ_{125}) (right).

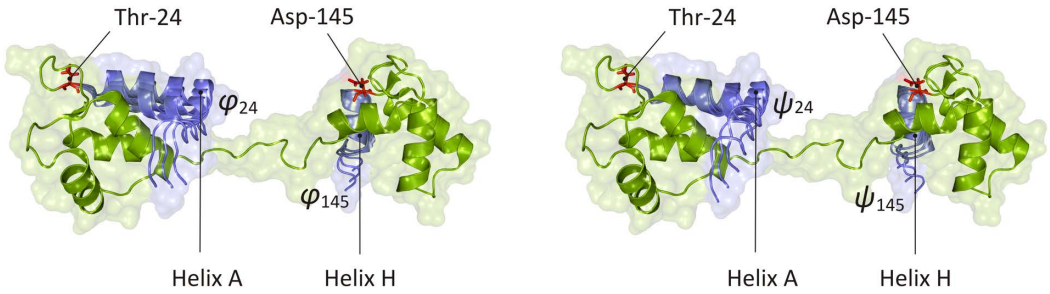

Figure 20: Troponin-C (PDB: 2JNF) hinged at Thr-24 and Asp-145 for variations in ϕ_{24}, ϕ_{145} (left) and ψ_{24}, ψ_{145} (right).

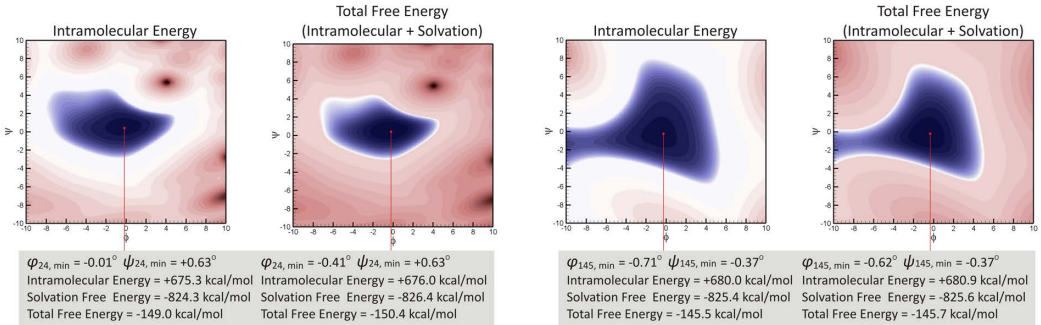

Figure 21: Intramolecular versus total (i.e., solvation included) free energy landscape for Troponin-C (PDB: 2JNF) in the vicinity of the native conformation versus variations in (ϕ_{24}, ψ_{24}) (left) and (ϕ_{145}, ψ_{145}) (right).

Troponin-C (PDB: 2JNF) is a calcium binding muscle protein that is composed of two 'EF-hand' domains, which are $\alpha-$domain motifs consisting of a bundle of 4 $\alpha-$helices per domain (denoted A through H) arranged in an up-and-down anti-parallel conformation [89], as shown in Fig. 20. There is a long pseudo-helical segment that connects the two globular domains. Assuming that the helices are rigid, we examine the energy variations due to dihedral rotations at the loops that connect the two end $\alpha-$helices; namely, local changes in (ϕ_i, ψ_i) for $i = 24$ and $i = 145$ where the A and H helices are hinged, respectively.

Figures 19 and 21 show the free energy variations in vacuum (i.e., without considering solvation effects) and in water (i.e., with considering solvation effects) for the two protein domains as the end $\alpha-$helices A and H of each are rotated within a range of $\pm 10^\circ$ of the native (ϕ_i, ψ_i) at the hinged loops.[13] In all four cases the energy model of Protofold II exhibits a local minima within $\pm 1^\circ$ of the native conformation. We also observe that the solvation effects contribute a significant amount to the total free energy; however, the SASA variations are so small in the considered neighborhood that the location of the local minima is almost unchanged. The van der Waals effects appear to be dominant in this neighborhood, manifested as the shape complementarity between the ridges and grooves of the mobile $\alpha-$helix and those of the other helices in the bundle [90]. However, when variations across larger angular ranges are considered, the solvation effects are expected to play a more determining role.

Tertiary Structural Interactions. Lastly, we consider the energy variations when a rigid domains of a protein is reoriented with respect to another domain against which it is packed into a stable structure.

Gamma-B Crystallin (PDB: 1GCS) is an eye-lens protein that is made of two similar domains that are 40% identical in sequence [89]. Each domain is composed of two anti-parallel $\beta-$sheets each made of 4 $\beta-$strands with the same arrangement topology [89], as shown in Fig. 22. Assuming that these domains are rigid, we analyze the energy variations due to rotating one domain with respect to the other at one of the residues that belong to the connecting loop (e.g., $i = 81$). To observe the global effects of solvation, we allow both dihedral angles (ϕ_{81}, ψ_{81}) to vary over the entire range of $\pm 180^\circ$ from the native values. To facilitate visualization, this time we consider only one angle's variation at a time.

Figures 23 and 24 show the variations of the different energy terms.[14] It appears that the van der Waals effects are dominant in determining the profile of the energy well near the native conformation, which can be attributed to the extensive contact interface between the two domains in Fig. 22. The electrostatic and solvation effects substantially change the energy landscape thus can dramatically affect the folding

[13]The colormaps are generated using a nonlinear but consistent contouring scheme.

[14]To enabel logarithmic plots, each energy ordinate is offset by a constant value to shift its minimum to (the arbitrary positive value of) 100 kcal per mol.

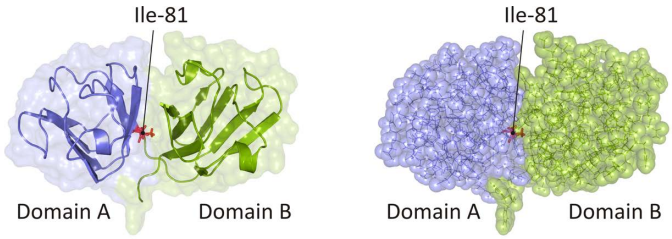

Figure 22: Gamma-B Crystallin (PDB: 1GCS) hinged at Ile-81 for variations in ϕ_{81} and ψ_{81}, one at a time.

pathway. However, they do not cause a significant change in the location of the energy minimum. Once again, the solvation energy is noisier due to the discrete nature of the enumeration algorithm in Section 3.4. Another interesting observation is that the electrostatic energy has discontinuities due to the cut-off approximation when pairs of atoms are farther than 9.0 $\mathring{A}$. Although it does not seem to affect the minimum location in this example, larger cut-off distances might be necessary for analyzing large proteins, since the accumulation of the pairwise errors grows quadratically with the number of atoms.

6 Conclusion

The KCM approach to protein folding [44, 45] originally implemented into Protofold I [46–48] provides a promising fast alternative to the popular MD simulation and MC sampling methods by

1. modeling the protein chain as a kinematic linkage with restricted DOF to which the well-studied principles from mechanism synthesis and robotics can be readily applied; and

2. replacing the 2nd-order dynamic response with 1st-order kinetostatic integration of the equations of motion to facilitate convergence to the free energy minima.

In the present work, we introduced major model and implementation improvements in Protofold II by

1. incorporating the solvation effects that characterize the hydrophobic effect, i.e., the entropic changes due to cavity formation in the aqueous solvent;

2. taking advantage from efficient auxiliary algorithms and data structures to improve the computational complexity from $O(n^2)$ to expected $O(n)$; and

3. implementing fast and relatively accurate evaluation of the SASA and its gradient for solvation energy- and force-field computation, respectively, in parallel on both CPU and GPU.

The presented enumeration algorithm for the latter provides a fast approximate method, in which the degree of accuracy is traded off with the performance by a proper choice

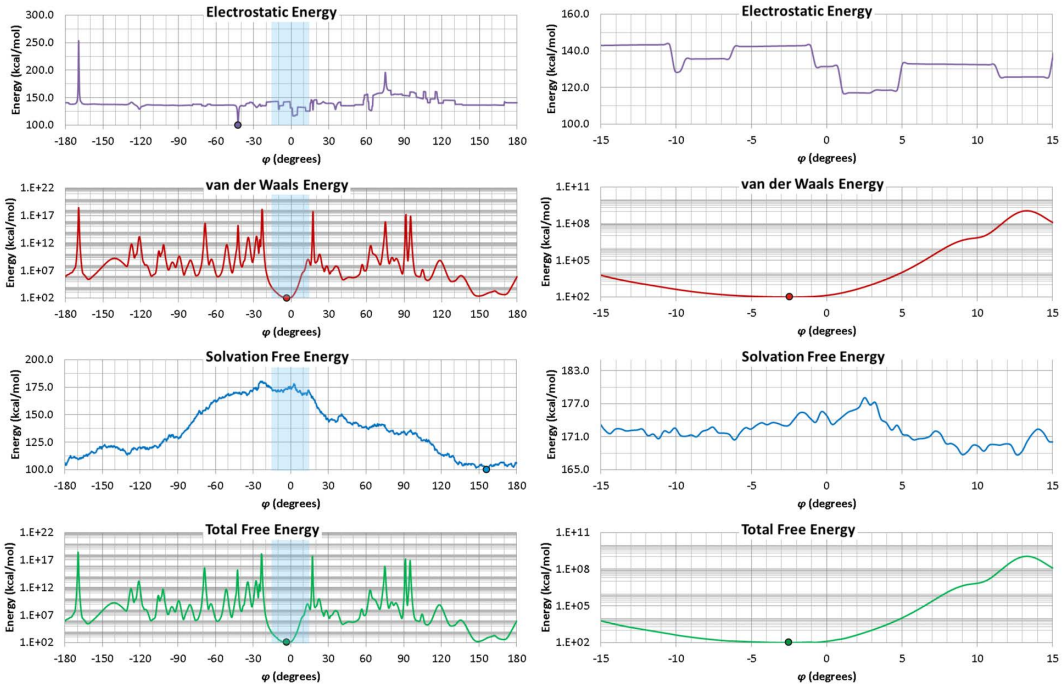

Figure 23: Energy variations for Gamma-B Crystallin (PDB: 1GCS) versus changes in ϕ_{81} for fixed native ψ_{81}.

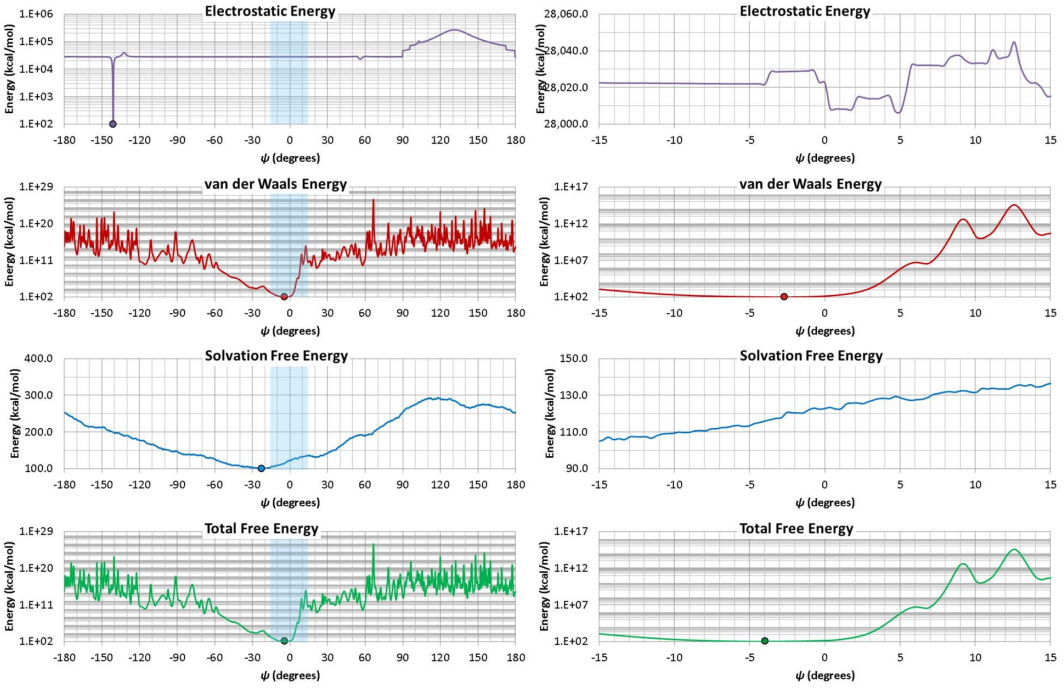

Figure 24: Energy variations for Gamma-B Crystallin (PDB: 1GCS) versus changes in ψ_{81} for fixed native ϕ_{81}.

of the sample size. We argued that the inclusion of the solvation free energy into the mix can significantly affect the folding pathway for water-soluble proteins whose folding *in vivo* is dominated by such effects. We also demonstrated that the performance gain of the GPU-accelerated implementation scales properly with the number of atoms, achieving up to two orders of magnitude in speed-ups after memory optimization.

Protofold II has been completely rearchitectured and is evolving into a versatile analysis toolbox for studying the kinematic and structural behavior of molecular chains in protein engineering applications such as the design of nanomanipulators [53, 54] among the ongoing projects. A hybrid force-field model was used, composed of

1. the AMBER model [67] (for noncovalent interactions); and
2. supplemental terms similar to the CHARMM model [78] and the GROMOS model [64] (for solvation effects) except for the probabilistic SASA estimation [63] replaced with our own surface sampling algorithm.

This model is by no means versatile enough to enable addressing the ultimate goal of arriving from sequence to 3D structure at a click of a button. Even though predicting folding of real 3D structures requires further developments, this study represents a major step toward this goal.

7 Acknowledgements

The authors are thankful to Sanguthevar Rajasekaran from the Department of Computer Science and Engineering and to Andrei Alexandrescu and Victoria Robinson from the Department of Molecular and Cellular Biology at UConn for providing instructive insight in this interdisciplinary project.

This work was supported in part by the National Science Foundation grants CMMI-1200089, CNS-0927105, and CMMI-1462759. The responsibility for any errors and omissions lies solely with the authors.

A Peptide Chains

This appendix overviews the structural biochemistry of peptide chains. Amino acids (AAs) are composed of a central carbon atom (denoted C_α) attached to 4 chemical components; namely a carboxylate group ($-COO^-$), an amino group ($-NH_3^+$), and a hydrogen atom ($-H$), common among all types, and a variable side chain (denoted $-R$) [3]. The amino group of one AA reacts with the carboxyl group of another to form a 'peptide bond,' eliminating a water molecule. This so-called 'condensation reaction' repeats over and over again to form a 'peptide chain' [3].

As depicted in Fig. 1, the 3D structure of the peptide chain can be uniquely represented by a set of bond lengths, and two sets of angles, namely the angles between adjacent bonds that share one atom (referred to as 'bond angles'), and those describing rotation around the bonds (referred to as 'torsion angles' or 'dihedral angles'). It is reasonable to assume that the bond lengths and bond angles are constant [45], thus the dihedral angles exclusively specify the protein conformation. For a protein with m AA residues denoted by AA_i ($1 \leq i \leq m$) numbered in order from N-terminus to C-terminus, the 3 set of dihedral angles in the main chain are defined for $1 \leq i \leq m$ as

- the rotation angle ω_i around the backbone $C-N$ bond that connects the residues AA_i and AA_{i+1};
- the rotation angle ϕ_i around the backbone $N-C_\alpha$ bond in the residue AA_i; and
- the rotation angle ψ_i around the backbone $C_\alpha-C$ bond in the residue AA_i.

Based on high resolution X-ray crystallographic studies, the angle ω_i is very close to either 0° (the 'cis' conformation) or 180° (the 'trans' conformation) [3], and the 6 atoms in the peptide group $C_\alpha-CO-NH-C_\alpha$ are approximately coplanar, forming the so-called 'peptide plane' [45]. Due to the partial double-bond characteristic of the peptide bond $C-N$, the peptide groups are almost rigid, hence modeled as rigid links hinged to the preceding and following peptide groups along the main chain [45]. These planes rotate about the $N-C_\alpha$ and $C_\alpha-C$ bonds, which can be thought of as *revolute joints*. Hence the 'main chain dihedral angles' ϕ_i and ψ_i completely specify the conformation of the backbone. In addition, each side chain can be treated as a shorter linkage which can add up to 4 extra links with their associated joint angles, called 'side chain dihedral angles' ($\chi_{i,1}$ to $\chi_{i,4}$). Therefore, the whole protein chain can be modeled as an open kinematic linkage, conformation of which is fully specified by a set of main chain and side chain dihedral angles. The resulted model has a reduced number of DOF of $2m+\sum_{i=1}^{m} \mathrm{DOF}(R_i)$, where the DOF of the side chain R_i is determined by the number of its side chain dihedral angles.

B Prefix Computation

The prefix sum problem is fundamental to numerous important algorithms, and is defined as follows. Given a finite ordered sequence of elements $X = (x_1, x_2, \cdots, x_n) \in \Sigma^n$ and an arbitrary binary operator $\oplus : \Sigma \times \Sigma \to \Sigma$ that is $O(1)-$time computable and associative (i.e., $(x \oplus y) \oplus z = x \oplus (y \oplus z)$), compute another sequence $Y = (y_1, y_2, \cdots, y_n) \in \Sigma^n$ where $y_1 = x_1$, $y_2 = x_1 \oplus x_2$, $\cdots$, $y_n = x_1 \oplus x_2 \oplus \cdots \oplus x_n$; in other words $y_i = y_{i-1} \oplus x_n$ for $1 \leq i \leq n$ where y_0 is the left-identity element (i.e., $y_0 \oplus x = x$ for all $x \in \Sigma$).

It is trivial to show that the prefix sums can be computed sequentially in $O(n)$, which is optimal. In addition, there are work-optimal parallel algorithms with a total computational work of $TP = O(n)$ that carry out the prefix computation in $T = O(\log n)$ time using $P = O(n/\log n)$ CREW PRAM or in $T = O(\log n/\log\log n)$ time using

$P = O(n \log\log n / \log n)$ CRCW PRAM processors with common conflict resolution [70]—see Appendix C.1 for details regarding PRAM.

C Parallel Computing

C.1 Abstract Machines

The parallel random-access machine (PRAM) is a shared-memory abstract parallel computation model, typically assigned with the exclusive/concurrent-read (ER/CR) and exclusive/concurrent-write (EW/CW) attributes [85]. The most common attributes are CREW and CRCW, noting that multiple processors can concurrently read a memory cell but only one can write at a time to prevent race conditions. To enable concurrent writing, one needs to resolve possible conflicts with typical mechanisms such as 1) 'common' meaning that all processors attempt to write the same value); 2) 'arbitrary' meaning that one processor's attempt succeeds at random; and 3) 'priority' meaning that the processors are prioritized by a prespecified order. Note that a CREW algorithm can always run in the same (if not fewer) number of steps on a CRCW machine.

C.2 GPU SIMT Model

A typical CUDA GPU program proceeds as follows. The data is first transferred from the CPU (i.e., *host*) memory to the GPU (i.e., *device*) memory. The host application invokes the so-called kernels on the GPU with specified granularity, i.e., issuing a 1D, 2D, or 3D grid of blocks, each block being a 1D, 2D, or 3D array of threads that are sent in groups of 16 or 32 (called 'warps') to one of the streaming multiprocessors (SM). The threads within the same block can access the fast shared memory banks on the SM, and communication across blocks is done using the global memory. The computed results are transferred from the device memory back to the host memory.

There are different types of GPU memory locations, classified into two groups: 1) device (i.e., off-chip) memory including global and local memories; and 2) on-chip memory including shared memory, cache, and registers. The access latencies to the on-chip are much less (around 100× faster) than those of the off-chip memory.

References

[1] Tavousi, P., Behandish, M., Kazerounian, K., and Ilies, H. T., 2013. "An improved free energy formulation and implementation for kinetostatic protein folding simulation". In Proccedings of the 2013 ASME International Design and Engineering Technical Conferences and Computer and Information in Engineering Conference (IDETC/CIE'2013), no. DETC2013-12671, American Society of Mechanical Engineers (ASME), pp. V06AT07A006:1–13.

[2] Behandish, M., Tavousi, P., Ilies, H. T., and Kazerounian, K., 2013. "GPU-accelerated parallel computation of free energy for kinetostatic protein folding simulation". In Proccedings of the 2013 ASME International Design and Engineering Technical Conferences and Computer and Information in Engineering Conference (IDETC/CIE'2013), no. DETC2013-12675, American Society of Mechanical Engineers (ASME), pp. V02AT02A009:1–12.

[3] Kuriyan, J., Konforti, B., and Wemmer, D., 2012. *The Molecules of Life: Physical and Chemical Principles*. Garland Science, Taylor & Francis Group.

[4] Anfinsen, C. B., 1973. "Studies on the principles that govern the folding of protein chains". *Science,* **181**(4096).

[5] van Gunsteren, W. F., 1988. "The role of computer simulation techniques in protein engineering". *Protein Engineering,* **2**(1), pp. 5–13.

[6] Chirikjian, G. S., Kazerounian, K., and Mavroidis, C., 2005. "Analysis and design of protein based nanodevices: Challenges and opportunities in mechanical design". *Journal of Mechanical Design,* **127**(4), pp. 695–698.

[7] Echenique, P., 2007. "Introduction to protein folding for physicists". *Contemporary Physics,* **48**(2), pp. 81–108.

[8] Chothia, C., and Lesk, A. M., 1986. "The relation between the divergence of sequence and structure in proteins". *The EMBO journal,* **5**(4), p. 823.

[9] Martí-Renom, M. A., Stuart, A. C., Fiser, A., Sánchez, R., Melo, F., and Šali, A., 2000. "Comparative protein structure modeling of genes and genomes". *Annual Review of Biophysics and Biomolecular Structure,* **29**(1), pp. 291–325.

[10] Krieger, E., Nabuurs, S., and Vriend, G., 2003. "Homology modeling". *Methods of Biochemical Analysis,* **44**, pp. 509–524.

[11] Bowie, J., Luthy, R., and Eisenberg, D., 1991. "A method to identify protein sequences that fold into a known three-dimensional structure". *Science,* **253**(5016), pp. 164–170.

[12] Ginalski, K., Grishin, N. V., Godzik, A., and Rychlewski, L., 2005. "Practical lessons from protein structure prediction". *Nucleic Acids Research,* **33**(6), pp. 1874–1891.

[13] Moult, J., Fidelis, K., Rost, B., Hubbard, T., and Tramontano, A., 2005. "Critical Assessment of methods of protein Structure Prediction (CASP)–round 6". *Proteins: Structure, Function, and Bioinformatics,* **61**(S7), pp. 3–7.

[14] Bradley, P., Misura, K. M. S., and Baker, D., 2005. "Toward high-resolution *de novo* structure prediction for small proteins". *Science,* **309**(5742), pp. 1868–1871.

[15] Schueler-Furman, O., Wang, C., Bradley, P., Misura, K., and Baker, D., 2005. "Progress in modeling of protein structures and interactions". *Science,* **310**(5748), pp. 638–642.

[16] Osguthorpe, D. J., 2000. "*Ab Initio* protein folding". *Current Opinion in Structural Biology,* **10**(2), pp. 146–152.

[17] Bonneau, R., and Baker, D., 2001. "*Ab Initio* protein structure prediction: Progress and prospects". *Annual Review of Biophysics and Biomolecular Structure,* **30**(1), pp. 173–189.

[18] Hansmann, U. H. E., and Okamoto, Y., 1994. "Comparative study of multicanonical and simulated annealing algorithms in the protein folding problem". *Physica A: Statistical Mechanics and its Applications,* **212**(3), pp. 415–437.

[19] Simons, K., Kooperberg, C., Huang, E., and Baker, D., 1997. "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions". *Journal of Molecular Biology,* **268**(1), pp. 209–225.

[20] Li, Z., and Scheraga, H., 1987. "Monte Carlo-minimization approach to the multiple-minima problem in protein folding". *Proceedings of the National Academy of Sciences,* **84**(19), pp. 6611–6615.

[21] David, J., and Doye, J. P. K., 1997. "Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms". *The Journal of Physical Chemistry A,* **101**(28), pp. 5111–5116.

[22] Nayeem, A., Vila, J., and Scheraga, H. A., 2004. "A comparative study of the simulated-annealing and Monte Carlo-with-minimization approaches to the minimum-energy structures of polypeptides:[met]-enkephalin". *Journal of Computational Chemistry,* **12**(5), pp. 594–605.

[23] Prentiss, M., Wales, D. J., and Wolynes, P. G., 2008. "Protein structure prediction using basin-hopping". *Journal of Chemical Physics,* **128**(22), p. 225106.

[24] Schug, A., and Wenzel, W., 2004. "Predictive in silico all-atom folding of a four-helix protein with a free-energy model". *Journal of the American Chemical Society,* **126**(51), pp. 16736–16737.

[25] Schug, A., and Wenzel, W., 2006. "An evolutionary strategy for all-atom folding of the 60-amino-acid bacterial ribosomal protein l20". *Biophysical Journal,* **90**(12), pp. 4273–4280.

[26] Verma, A., Gopal, S. M., Oh, J. S., Lee, K. H., and Wenzel, W., 2007. "All-atom *De Novo* protein folding with a scalable evolutionary algorithm". *Journal of Computational Chemistry,* **28**(16), pp. 2552–2558.

[27] Abagyan, R. A., and Totrov, M., 1994. "Biased probability Monte Carlo conformational searches and electrostatic calculations for peptides and proteins". *Journal of Molecular Biology,* **235**(3), pp. 983–1002.

[28] Abagyan, R. A., and Totrov, M., 1999. "*Ab Initio* folding of peptides by the optimal-bias Monte Carlo minimization procedure". *Journal of Computational Physics,* **151**, pp. 402–421.

[29] Carr, J. M., and Wales, D. J., 2005. "Global optimization and folding pathways of selected α-helical proteins". *Journal of Chemical Physics,* **123**(23), p. 234901.

[30] Klenin, K., Strodel, B., Wales, D., and Wenzel, W., 2011. "Modelling proteins: Conformational sampling and reconstruction of folding kinetics". *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics,* **1814**(8), pp. 977–1000.

[31] Gear, C. W., 1971. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall.

[32] Beeman, D., 1976. "Some multistep methods for use in molecular dynamics calculations". *Journal of Computational Physics,* **20**(2), pp. 130–139.

[33] Swope, W. C., Andersen, H. C., Berens, P. H., and Wilson, K. R., 1982. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters". *Journal of Chemical Physics,* **76**(1), pp. 637–649.

[34] Hockney, R. W., and Eastwood, J. W., 1988. *Computer Simulation Using Particles*. Taylor & Francis.

[35] van Gunsteren, W., and Berendsen, H., 1988. "A leap-frog algorithm for stochastic dynamics". *Molecular Simulation,* **1**(3), pp. 173–185.

[36] Ricci, A., and Ciccotti, G., 2003. "Algorithms for Brownian dynamics". *Molecular Physics,* **101**(12), pp. 1927–1931.

[37] Guarnieri, F., and Still, W., 2004. "A rapidly convergent simulation method: Mixed Monte Carlo/stochastic dynamics". *Journal of Computational Chemistry,* **15**(11), pp. 1302–1310.

[38] Ciccotti, G., and Kalibaeva, G., 2004. "Deterministic and stochastic algorithms for mechanical systems under constraints". *Philosophical Transactions-Royal Society Of London Series A Mathematical Physical And Engineering Sciences,* **362**, pp. 1583–1594.

[39] Rojnuckarin, A., Kim, S., and Subramaniam, S., 1998. "Brownian dynamics simulations of protein folding: Access to milliseconds time scale and beyond". *Proceedings of the National Academy of Sciences,* **95**(8), pp. 4288–4292.

[40] Gabdoulline, R. R., and Wade, R. C., 2001. "Protein-protein association: Investigation of factors influencing association rates by Brownian dynamics simulations". *Journal of Molecular Biology,* **306**(5), pp. 1139–1155.

[41] Ando, T., and Yamato, I., 2005. "Free energy landscapes of two model peptides: α-helical and β-hairpin peptides explored with Brownian dynamics simulation". *Molecular Simulation,* **31**(10), pp. 683–693.

[42] Frembgen-Kesner, T., and Elcock, A. H., 2009. "Striking effects of hydrodynamic interactions on the simulated diffusion and folding of proteins". *Journal of Chemical Theory and Computation,* **5**(2), pp. 242–256.

[43] Scheraga, H., Khalili, M., and Liwo, A., 2007. "Protein-folding dynamics: Overview of molecular simulation techniques". *Annual Reveiw of Physical Chemistry,* **58**, pp. 57–83.

[44] Kazerounian, K., 2004. "From mechanisms and robotics to protein conformation and drug design". *Journal of Mechanical Design (JMD),* **126**(1), pp. 40–45.

[45] Kazerounian, K., Latif, K., Rodriguez, K., and Alvarado, C., 2005. "Nano-kinematics for analysis of protein molecules: Analysis and design of protein based nanodevices". *Journal of Mechanical Design (JMD),* **127**(4), pp. 699–711.

[46] Kazerounian, K., Latif, K., Rodriguez, K., and Alvarado, C., 2004. "ProtoFold: Part I–nanokinematics for analysis of protein molecules". In Proceedings of the 2004 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE'2004), no. DETC2004-57243, American Society of Mechanical Engineers (ASME), pp. 645–658.

[47] Kazerounian, K., Latif, K., and Alvarado, C., 2004. "ProtoFold: Part II–a successive kineto-static compliance method for protein conformation prediction". In Proceedings of the 2004 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE'2004), no. DETC2004-57247, American Society of Mechanical Engineers (ASME), pp. 669–677.

[48] Kazerounian, K., Latif, K., and Alvarado, C., 2005. "Protofold: A successive kinetostatic compliance method for protein conformation prediction". *Journal of Mechanical Design (JMD),* **127**, pp. 712–717.

[49] Shahbazi, Z., Ilieş, H. T., and Kazerounian, K., 2010. "Hydrogen bonds and kinematic mobility of protein molecules". *Journal of Mechanisms and Robotics (M&R),* **2**(2), pp. 021009:1–9.

[50] Shahbazi, Z., Pimentel, T. A. P. F., Ilieş, H., Kazerounian, K., and Burkhard, P., 2010. "A kinematic observation and conjecture for creating stable constructs of a peptide nanoparticle". In *Advances in Robot Kinematics: Motion in Man and Machine*, J. Lenarcic and M. M. Stanisic, eds. Springer Netherlands, pp. 203–210.

[51] Shahbazi, Z., and Demirtaş, A., 2015. "Rigidity analysis of protein molecules". *Journal of Computing and Information Science in Engineering (JCISE),* **15**(3), pp. 031009:1–6.

[52] Shahbazi, Z., 2015. "Mechanical model of hydrogen bonds in protein molecules". *American Journal of Mechanical Engineering,* **3**(2), pp. 47–54.

[53] Tavousi, P., and Ilieş, H. T., 2015. "Synthesizing functional mechanisms from a link soup". In Proccedings of the 2015 ASME International Design and Engineering Technical Conferences and Computer and Information in Engineering Conference (IDETC/CIE'2015), no. DETC2015-47311, American Society of Mechanical Engineers (ASME), pp. V05CT08A044:1–14.

[54] Tavousi, P., and Ilieş, H. T., 2016. "Synthesizing functional mechanisms from a link soup". *Journal of Mechanical Design (JMD),* (to appear).

[55] Mashayak, S. Y., and Tanner, D. E., 2011. Comparing solvent models for molecular dynamics of protein. Technical report, University of Illinois at Urbana-Champaign.

[56] Jorgensen, W. L., and Tirado-Rives, J., 2005. "Potential energy functions for atomic-level simulations of water and organic and biomolecular systems". *Proceedings of the National Academy of Sciences of the United States of America,* **102**(19), pp. 6665–6670.

[57] Wang, H., Junghans, C., and Kremer, K., 2009. "Comparative atomistic and coarse-grained study of water: What do we lose by coarse-graining?". *The European Physical Journal E: Soft Matter and Biological Physics,* **28**(2), pp. 221–229.

[58] Izvekov, S., and Voth, G. A., 2005. "Multiscale coarse graining of liquid-state systems". *Journal of Chemical Physics,* **123**, p. 134105.

[59] Roux, B., and Simonson, T., 1999. "Implicit solvent models". *Biophysical Chemistry,* **78**(1), pp. 1–20.

[60] Eisenberg, D., and McLachlan, A., 1986. "Solvation energy in protein folding and binding". *Nature,* **319**, pp. 199–203.

[61] Richmond, T. J., 1984. "Solvent accessible surface area and excluded volume in proteins: Analytical equations for overlapping spheres and implications for the hydrophobic effect". *Journal of Molecular Biology,* **178**(1), pp. 63–89.

[62] Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., Swaminathan, S., and Karplus, M., 2004. "CHARMM: A program for macromolecular energy, minimization, and dynamics calculations". *Journal of Computational Chemistry,* **4**(2), pp. 187–217.

[63] Wodak, S. J., and Janin, J., 1980. "Analytical approximation to the accessible surface area of proteins". *Proceedings of the National Academy of Sciences,* **77**(4), pp. 1736–1740.

[64] Fraternali, F., and van Gunsteren, W. F., 1996. "An efficient mean solvation force model for use in molecular dynamics simulations of proteins in aqueous solution". *Journal of Molecular Biology,* **256**(5), pp. 939–948.

[65] van Gunsteren, W. F., Daura, X., and Mark, A. E., 2002. "GROMOS force field". *Encyclopedia of Computational Chemistry.*

[66] Allison, J. R., Boguslawski, K., Fraternali, F., and van Gunsteren, W. F., 2011. "A refined, efficient mean solvation force model that includes the interior volume contribution". *The Journal of Physical Chemistry B,* **115**(15), pp. 4547–4557.

[67] Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S., and Weiner, P., 1984. "A new force field for molecular mechanical simulation of nucleic acids and proteins". *Journal of the American Chemical Society,* **106**(3), pp. 765–784.

[68] Cornell, W., Cieplak, P., Bayly, C., Gould, I., Merz, K., Ferguson, D., Spellmeyer, D., Fox, T., Caldwell, J., and Kollman, P., 1995. "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules". *Journal of the American Chemical Society,* **117**(19), pp. 5179–5197.

[69] Weiser, J., Shenkin, P. S., and Still, W. C., 1999. "Approximate atomic surfaces from linear combinations of pairwise overlaps (LCPO)". *Journal of Computational Chemistry,* **20**(2), pp. 217–230.

[70] Cole, R., and Vishkin, U., 1989. "Faster optimal parallel prefix sums and list ranking". *Information and Computation,* **81**(3), pp. 334–352.

[71] Gupta, K. C., 1986. "Kinematic analysis of manipulators using the zero reference position description". *The International Journal of Robotics Research,* **5**(2), pp. 5–13.

[72] Subramanian, R., and Kazerounian, K., 2007. "Improved molecular model of a peptide unit for proteins". *Journal of Mechanical Design (JMD),* **129**(11), pp. 1130–1136.

[73] Bajaj, C., and Zhao, W., 2010. "Fast molecular solvation energetics and forces computation". *SIAM Journal on Scientific Computing,* **31**(6), pp. 4524–4552.

[74] Fogolari, F., Brigo, A., and Molinari, H., 2002. "The Poisson–Boltzmann equation for biomolecular electrostatics: A tool for structural biology". *Journal of Molecular Recognition,* **15**(6), pp. 377–392.

[75] Still, W. C., Tempczyk, A., Hawley, R. C., and Hendrickson, T., 1990. "Semianalytical treatment of solvation for molecular mechanics and dynamics". *Journal of the American Chemical Society,* **112**(16), pp. 6127–6129.

[76] Onufriev, A., Case, D., and Bashford, D., 2002. "Effective Born radii in the generalized Born approximation: the importance of being perfect". *Journal of Computational Chemistry,* **23**(14), pp. 1297–1304.

[77] Rappe, A., and Casewit, C., 1997. *Molecular Mechanics Across Chemistry.* University Science Books.

[78] Wesson, L., and Eisenberg, D., 1992. "Atomic solvation parameters applied to molecular dynamics of proteins in solution". *Protein Science,* **1**(2), pp. 227–235.

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

[80] Sharp, K. A., Nicholls, A., Friedman, R., and Honig, B., 1991. "Extracting hydrophobic free energies from experimental data: Relationship to protein folding and theoretical models". *Biochemistry,* **30**(40), pp. 9686–9697.

[81] Lee, B., and Richards, F., 1971. "The interpretation of protein structures: Estimation of static accessibility". *Journal of Molecular Biology,* **55**(3), pp. 379–400, IN3–IN4.

[82] Wolfenden, R., Andersson, L., Cullis, P. M., and Southgate, C. C. B., 1981. "Affinities of amino acid side chains for solvent water". *Biochemistry,* **20**(4), pp. 849–855.

[83] Muller, M. E., 1959. "A note on a method for generating points uniformly on $n-$dimensional spheres". *Communications of the ACM (CACM),* **2**(4), pp. 19–20.

[84] Yershova, A., and LaValle, S. M., 2004. "Deterministic sampling methods for spheres and $so(3)$". In Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA'2014), Vol. 4, Institute of Electrical and Electronics Engineers (IEEE), pp. 3974–3980.

[85] Maggs, B. M., Matheson, L. R., and Tarjan, R. E., 1995. "Models of parallel computation: A survey and synthesis". In Proceedings of the 28th Hawaii International Conference on System Sciences, Vol. 2, Institute of Electrical and Electronics Engineers (IEEE), pp. 61–70.

[86] Lins, R. D., and Ferreira, R., 2006. "The stability of right- and left-handed alpha-helices as a function of monomer chirality". *Química Nova,* **29**(5), pp. 997–998.

[87] Davis, I. W., Leaver-Fay, A., Chen, V. B., Block, J. N., Kapral, G. J., Wang, X., Murray, L. W., Arendall, W. B., Snoeyink, J., Richardson, J. S., and Richardson, D. C., 2007. "MolProbity: All-atom contacts and structure validation for proteins and nucleic acids". *Nucleic Acids Research,* **35**, pp. W375–W383.

[88] Chen, V. B., Arendall, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S., and Richardson, D. C., 2009. "MolProbity: All-atom structure validation for macromolecular crystallography". *Acta Crystallographica Section D: Biological Crystallography,* **66**(1), pp. 12–21.

[89] Petsko, G. A., and Ringe, D., 2004. *Protein Structure and Function.* New Science Press.

[90] Eilers, M., Patel, A. B., Liu, W., and Smith, S. O., 2002. "Comparison of helix interactions in membrane and soluble $\alpha-$bundle proteins". *Biophysical Journal,* **82**(5), pp. 2720–2736.