Published in final edited form as:

J Chem Theory Comput. 2018 January 09; 14(1): 426–443. doi:10.1021/acs.jctc.7b00775.

Systematic testing of belief-propagation estimates for absolute free energies in atomistic peptides and proteins

Rory M. Donovan-Maiye[†], Christopher J. Langmead[‡], and Daniel M. Zuckerman[¶]

[†]Allen Institute for Cell Science, Seattle, WA

[‡]Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA

[¶]Department of Biomedical Engineering, Oregon Health and Science University, Portland, OR

Abstract

Motivated by the extremely high computing costs associated with estimates of free energies for biological systems using molecular simulations, we further the exploration of existing “belief propagation” (BP) algorithms for fixed-backbone peptide and protein systems. The precalculation of pairwise interactions among discretized libraries of side chain conformations, along with representation of protein side chains as nodes in a graphical model, enables direct application of the BP approach, which requires only ~1 s of single-processor run time after the pre-calculation stage. We use a “loopy BP” algorithm, which can be seen as an approximate generalization of the transfer-matrix approach to highly connected (i.e., loopy) graphs, and has previously been applied to protein calculations. We examine the application of loopy BP to a number of peptides as well as the binding site of the T4 lysozyme L99A mutant. The present study reports on (i) comparison of the approximate BP results with estimates from unbiased estimators based on the Amber99SB forcefield; (ii) investigation of the effects of varying library size on BP predictions; and (iii) a theoretical discussion of discretization effects which can arise in BP calculations. The data suggest that, despite their approximate nature, BP free energy estimates are highly accurate – indeed they never fall outside confidence intervals from unbiased estimators for the systems where independent results could be obtained. Further, we find that libraries of sufficiently fine discretization (which diminish library-size sensitivity) can be obtained with standard computing resources in most cases. Altogether, the extremely low computing times and accurate results suggest the BP approach warrants further study.

1 Introduction

There is immense interest in the computational estimation of biomolecular free energies, for purposes ranging from drug-design to the understanding of fundamental biology.^{1–7}

Unfortunately, calculating these free energy estimates is very computationally intensive using traditional means based on molecular molecular dynamics (MD) simulations.^{8–12}

A popular alternative to time consuming MD-based approaches is computation of a free energy of binding using an empirical “scoring function”.^{13,14} Scoring-based methods focus on protein-ligand interactions, and are fast enough to be suitable for high-throughput virtual screening studies,^{15,16} but this speed is obtained at the price of total or significant receptor

rigidity, essentially eliminating the capacity to estimate protein configurational entropy contributions to the free energy.^{17–22}

Another class of methods based on polymer-growth ideas can estimate free energies accounting for full configurational flexibility without MD simulation or with minimal simulation.^{23–29} These approaches employ (stochastic) sampling of a *discretization* of configuration space. Using pre-calculated Boltzmann-distributed libraries of amino-acid configurations, for example, polymer growth algorithms have been used for absolute free energy calculations of peptides.^{30,31} The polymer growth approach has trouble scaling beyond modest peptide length (~15 residues), but it provides unbiased free energies when enough samples are used in the computation. As will be seen below, an analogous limitation applies in the case of fixed-backbone computations.

An additional algorithm of note is Donaldson's K^* algorithm,³² which uses a branch-and-bound strategy to look for the global minimum energy conformation among a set of discrete rotamers. The key difference between K^* and belief propagation is that K^* explicitly enumerates low-energy configurations whereas message-passing is always operating on the entire (discrete) configuration space. Thus, the K^* approximation excludes nearly all configurations, while message-passing should provide a better estimate of the entropic contributions to the free energy for systems where the probability mass on the configurations ignored by K^* is non-negligible.

Graphical-model algorithms constitute another approach to protein free energy calculations which also exploit discretized configuration spaces. A graphical model consists of nodes representing random variables (e.g., side-chain rotameric configurations) and edges representing statistical couplings (e.g., force field interactions). In this paper, we use a class of undirected graphical models known as Markov Random Fields (MRF), which can be seen as a generalization of the Ising and Potts models.³³

An MRF encodes a partition function Z and corresponding absolute free energy $F = -k_B T \ln Z$, which generally are not amenable to exact calculation due to the usual combinatorial explosion of terms. However, recent work has established the use of MRFs as an attractive alternative to both simulation-based methods or scoring functions, with particularly straightforward application in the context of fixed-backbone protein calculations.^{34–40}

An approximate, deterministic algorithm known as “loopy belief propagation” (loopy BP)⁴¹ operating on pairwise protein MRFs achieves impressively accurate estimates of protein conformational entropies and binding free energies at orders of magnitude lower computational cost than simulation-based methods.^{35,37} Ordinary BP exactly treats MRFs which lack loops (cycles) and have tree-like connectivity (e.g., the simple linear example of Fig. 1). The basic procedure of BP is to iteratively pass sums of interaction terms (called “messages”) back-and-forth between nodes until self-consistency is achieved between the intra-node distribution of states and inter-node interactions; details are given below. “Loopy BP” refers to application of ordinary BP to graphs with loops, which is intuitively reasonable because of the self-consistency criterion, but mathematically is no longer an exact procedure.

In this report, we continue exploring practical and technical aspects of applying loopy BP to peptide and protein systems. Most importantly, we test whether BP provides accurate free-energy estimates for fixed-backbone calculations using a standard force field (Amber99SB) by comparison to established, unbiased polymer growth algorithms.^{30,31} In other research domains, loopy BP has been shown to provide accurate partition-function estimates despite its approximate nature^{41–45} Our study focuses on computing absolute conformational free energies in non-trivial but limited systems appropriate to our goal of quantifying error in this approximate method. Such computations could be extended in the future to estimating binding free energies as part of an appropriate thermodynamic construction.⁴⁶

We also address several issues related to the discretization of side-chain rotameric space. Primarily, we examine how discretization error decays with finer subdivisions of rotamer space by direct calculation on different libraries. Also, instead of using the Dunbrack backbone-dependent rotamer libraries,⁴⁷ which were employed in a prior BP study,³⁷ here we sample rotameric χ angles uniformly to ensure direct correspondence to the definition of the configurational partition function.⁴⁸ Uniform samples may also provide numerical advantages for the evaluation of periodic functions.⁴⁹ The Dunbrack libraries may be seen as an extreme case of weighted rotamer libraries which, as we show in Appendix C, can introduce a subtle error into BP calculations. Finally, some additional clarification is provided regarding the discretization-dependent logarithmic correction term bridging the physical and Shannon entropies.^{37,50}

2 Model Definition: Graph Generation

As shown schematically in Fig. 2, our approach combines standard molecular modeling tools – namely, the force field and a PDB structure for the backbone – with a discrete set of side-chain configurations to define the MRF graph. The MRF is our basic model which defines a partition function and free energy, which may be estimated in different ways. Here, we examine brute-force summation of the full partition function where possible, as well as polymer growth and BP estimates. The present study is limited to rigid-backbone models in order to carefully address methodological issues, but we note that BP has been used with multiple backbone conformers in prior work.³⁷

Every peptide or protein is modeled on the basis of k_i discrete configurations $\mathbf{r}_i$ for each side chain i which *uniformly* sample dihedral-angle space

$$\mathbf{r}_i \quad i = 1, 2, \dots, k_i \quad (1)$$

with the protein backbone taken as fixed. If there are L flexible side chains in the model, the configurational partition function for fixed bond lengths and angles is given by

$$Z^* = \sum_{\text{configs}} \exp \left[-U(\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L)/k_B T \right], \quad (2)$$

where the sum is over all combinations of side chain configurations, and the “*” denotes the discretization-dependence of the partition function, which is discussed below. U is the potential energy function of the force field, here taken to be Amber99SB with a uniform dielectric constant of 60, as has been used previously in the context of methods development.³⁰ Because the pre-calculation stage involves selective omission of large components of a structure, non-pairwise solvent models cannot be readily used.

Although Z^* is discretization-dependent (varies with the number of χ values used for each dihedral), the true free energy is not, and the necessary correction can be calculated³⁷ when the discretization is sufficiently fine-grained. As shown in Appendix A, the free energy corrected for (asymptotically fine) discretization is given by

$$F = -k_B T \log Z^* + k_B T \log K + \text{const.}, \quad (3)$$

where $K = \sum_{i=1}^L k_i$ is the total number of side-chain configurations and the discretization-independent constant primarily accounts for momentum integrations. It is important to note that free energy estimates based on (3) must still be checked regarding whether the discretization is fine enough – and indeed this task is a significant part of the present study. Without the $\log K$ correction term, however, the free energy would continue to change even as discretization became extremely fine.

2.1 Rotameric States: Graph Nodes

Since the backbone, bond angles, and bond lengths are fixed for each graph, the state-space of each node and thus the graph as a whole is determined by the accessible dihedral angles – see Eqs. (1) and (2). The convention that we follow is that each dihedral angle in a residue is allowed to take a set number of states k , evenly sampled around a full rotation of that angle. This number k is the same for each dihedral angle in each residue, though in theory this does not necessarily need to be the case. This uniform sampling does have the advantage of simplifying the free energy formula; non-uniform sampling would entail some form of unweighting the samples back to a uniform distribution in order to compute the correct integrals. This issue is discussed in more detail in Appendix A.

Dihedral angles involving only heavy atoms are treated slightly differently than dihedral angles involving terminal methyl groups. Since the methyl groups possess a three-fold symmetry, they only need to be rotated through one third of a full rotation in order to sample all conformations uniformly. This is consistent with the convention that the contribution to the partition function of symmetrically indistinguishable states is reduced by a factor given by the symmetry.⁴⁸

Although uniform sampling in dihedral space guarantees correctness of the partition function/free energy, it does entail a high cost for residues with a large number of dihedral angles. For instance, for a residue with four dihedral angles, ten states per angle means computing 10^4 states for that residue. Further, the real cost is in computing the pairwise energies for the edges in the graph between these large nodes. If two nodes have 10^4 states

each, the edge between those nodes has 10^8 pairwise combinations of states, which is not feasible in our implementation.

Since this work is largely exploratory, the solution we took to the state-space problem was to avoid using pdb structures with residues containing too many dihedral angles. In practice, the residue with the largest state space that we were able to use was leucine, with two heavy atom dihedrals and two methyl group dihedrals. Sampling at a density of about 10 states per heavy atom dihedral and 2–3 states per methyl group dihedral yielded about 500 states for the nodes and 25,000 states per edge between two leucines. Computing such edges required hours to days in our non-optimized implementation.

2.2 Side chain interactions: Graph Edges

The connectivity of the graphs in the MRFs specifies which residues in the peptide/protein (which are nodes in the graph) directly influence the state of other residues. In a maximally dense graph, all nodes can influence all other nodes, though in practice, constructing such graphs is undesirable and unnecessary. Putting an edge between nodes in the graph that represent residues in the structure which are very far apart alters the properties of the graph only minutely, since the interaction energies of residues that are far apart are so weak. As discussed below and in,^{35,37} including edges between nodes only when the connecting node is within a specified distance is a reasonable approximation. We use the distance between α -carbons of the residues as the input to the cutoff criteria. Other, more sophisticated cutoffs are possible, for instance longer cutoffs for the residues that are physically larger, but we have found that an α -carbon cutoff distance of about 0.8–1.0 nanometers is appropriate for the systems we consider here. Specific values used are noted below when the systems are described. Another reason for restricting the number of edges in a graph is that since constructing the interaction energy tables for the edges of the graph is a bottleneck for the process, keeping the time spent on unimportant edges low is desirable from a practical point of view.

In our work, the graph is constructed from a PDB structure file, based on a C_α -distance cutoff. Once the connectivity of the graph is determined, the potentials in the MRF are constructed according to how many states each node (i.e. residue) can take.

2.3 Force field evaluation: Graph potentials

Once a graph structure for the MRF is generated, a state-space for each node is chosen, and the pairwise and single node “potentials” are computed for each state or pair of states. MRF potentials are not energies but rather are given by the Boltzmann factor for the configuration of the (subset of) nodes involved in that interaction. The single-node potentials are effectively the non-interacting probability distributions for the nodes based on energy terms internal to the residue, and the edge potentials couple the nodes to their neighbors based on inter-residue energy terms.

To compute the node and edge potentials of the Markov random field, it is necessary to be able to manipulate protein structures, and evaluate the potential energies of the structures in different conformations. The other necessary ingredient is the ability to “turn off” or “ignore” large portions of the structure, and only evaluate the energetics of one or two

residues (for nodes and edges respectively. Once the energy for a conformation is computed, the Boltzmann factor of the energy gives the element of the potential function. All Boltzmann factors in this work are computed at 298 K.

To calculate the potential for a single node, the residue corresponding to that node is manipulated to take on all desired conformations (which is a set number per dihedral degree of freedom), and at each of these conformations, the potential energy contributions from atoms only in that residue are tabulated. The Boltzmann factor of these potential energies is then the node potential.

For example, the node potential ϕ of sidechain X that can take on configurations $\mathbf{x} = x_1, \dots, x_k$ would be

$$\forall x_i \in \mathbf{x}, \quad \phi(x_i) = e^{-\beta U^{\text{node}}(x_i)} \quad (4)$$

The computation of the potentials for the edges is only slightly more complicated. All pairwise combinations of conformations for the two residues must be considered, recording the potential energy contributions from all atoms in both residues. Importantly, these potential energies must be modified by subtracting off the contributions internal to each node (both non-bonded and bonded), leaving only the energetics arising from inter-residue interactions. After determining the inter-residue potential energy for each pairwise combination of residue conformations, the Boltzmann factor of that energy is taken, and this matrix is the edge potential. Hence, as we note again, MRF potentials are not to be confused with potential energies.

For example, the edge potential ϕ of sidechains X and Y that can take on configurations $\mathbf{x} = x_1, \dots, x_k$ and $\mathbf{y} = y_1, \dots, y_l$ would be

$$\forall x_i \in \mathbf{x}, y_j \in \mathbf{y}, \quad \phi(x_i, y_j) = e^{-\beta U^{\text{edge}}(x_i, y_j)} \quad (5)$$

where again, this energy function for the edge potential only includes interaction terms across the nodes. Explicitly, if U^{total} is the energy of all interactions in total for the two node system, then

$$\forall x_i \in \mathbf{x}, y_j \in \mathbf{y}, \quad U^{\text{edge}}(x_i, y_j) = U^{\text{total}}(x_i, y_j) - (U^{\text{node}}(x_i) + U^{\text{node}}(y_j)) \quad (6)$$

The precise number of energy calculations needed to construct the nodes and edges of the MRF will depend on the specifics of the system. Different residues have different degrees of freedom, and different backbone conformations will induce different edge connectivity.

Nonetheless, a crude upper bound can be constructed for the number of energy computations needed to construct the MRF. If we take d as the maximal number of degrees of freedom per node in the graph, and use k samples per degree of freedom, then for a graph with $|E|$ edges and $|N|$ nodes, we have a number of energy calls equal to $|E| (k^d)^2 + |N| k^d$. In the worst case of a fully connected graph, this estimate simplifies to $|N| (|N| - 1)/2 (k^d)^2 + |N| k^d$. In practice, when the nodes have heterogeneous degrees of freedom, the computation for the MRF will be dominated by the edge between the two largest nodes.

It is worth noting that after the MRF has been constructed once, point-wise mutations of the structure can be computed relatively quickly, as only the mutated node and its incident edges need be recomputed.

3 Overview of Free Energy Calculations of Graphs

3.1 Brute-Force

The natural point of comparison in evaluating the accuracy of the belief propagation approximation to the partition function is to compare to the exact value, though this is only feasible in certain limited cases. When this gold standard of comparison is practical, we employ a brute-force calculation of the free energy via an exponentially large sum over the partition function, a simplified version of which is shown in listing 1. This approach is simply a naïve sum over an exponentially large number of terms, and is only practical when the graph has very few nodes, and few states per node (in practice, ~ 5 nodes, and ~ 10 – 100 of states per node). The log of this partition function then gives the free energy as in equation 29.

3.2 Polymer Growth

In more challenging cases, we employ a “bronze standard” of a polymer-growth estimate of the free energy. The polymer growth algorithm proceeds by sequentially adding nodes and edges to a graph, until it is fully constructed. After each node is added, the change in the free energy of the graph due to the addition of that node (and all its incident edges) is computed. After all the nodes and edges have been added, the sum of these changes in free energy is found to be the total free energy of the MRF.

Polymer growth has been previously applied to small systems, such as peptides of length ~ 10 residues^{30,31} in the considerably more challenging setting of a totally flexible backbone. Details of the method can be found in,^{30,31} though we will briefly sketch out the procedure as implemented in our work, a simplified version of which is detailed in listing 2. The procedure is illustrated schematically in Fig. 4. The polymer growth free energy estimates are a necessary point of comparison in quantifying the accuracy of belief propagation, because brute-force calculations are not feasible in systems of even modest size.

Whereas both the brute-force and the belief propagation calculations are deterministic, the polymer growth procedure, as we have employed it, is a stochastic algorithm. This is because after each node is added, and the free energies are computed, the number of states of the graph retained for estimation purposes is down-sampled randomly according to the Boltzmann factor of the energy of the state of the graph. Another point of difference is that,

as opposed to end-point methods like brute-force and belief propagation, polymer growth is a perturbative algorithm, which estimates of the free energy of the graph by iteratively adding nodes to the graph and computing changes in free energy.

As an example of our polymer growth implementation, we might start with an empty graph, and add a node to it containing 1000 states. After computing the energy of each state, and finding a free energy difference from the empty graph (taken to have a free energy of 0) via a sum of Boltzmann factors, the state-space is then down-sampled to a smaller number, say 50. In the next round of growth, when we add another node, with say 200 states, we would then calculate the energies of each pairwise combination of old global configurations of the graph (the ones we down-sampled to last time) with all the new states, so in this case 50×200 . In this manner, the combinatoric explosion of state combinations is avoided, and one only pays a cost proportional to the size of the state-space of the individual nodes, multiplied by the number of nodes to which one down-samples.

However, a further cost is also due: since the algorithm is stochastic, one must perform multiple runs in order to obtain converged estimates of the free energy. A last caveat of the polymer growth estimate is that as noted in,⁵¹ at finite sample-size, a nonlinear estimate such as for the free energy is confounded not only by random noise but by systematic bias. This systematic bias is difficult to predict; in practice we increased the number of states retained until the estimate of the free energy stabilized.

3.3 (Loopy) Belief Propagation

Belief propagation⁵² is an algorithm for efficiently computing the exact free energy on graphs without cycles. Unfortunately, cycles (or “loops”) are essential to modeling the interactions between multiple neighboring amino acids in a folded structure. The variant of Pearl’s algorithm⁵³ that we employ, loopy belief propagation,⁴¹ essentially iteratively applies the belief propagation algorithm even though it has no guarantee of converging. However, it has been shown,⁵⁴ that this naïve procedure, when it converges to a stable estimate, is in fact identical to the Bethe approximation for Ising models^{55,56} of the free energy, a well-known method in statistical physics. The Bethe approximation of the free energy comes with no strict guarantees of accuracy in loopy graphs, and the convergence of the algorithm in the general case is still an open problem. However, there is evidence that graphs representing protein structure may be significantly more tractable for the BP algorithm than a hypothetical worst case.³⁷ We refer to loopy belief propagation as BP throughout this work.

3.3.1 Message Passing—The pseudocode for the (loopy) belief propagation algorithm that we implemented is given in listing 3. At a high level, the algorithm passes messages around a graph until they stop changing. The messages are sent from nodes to their neighbors, and convey each node’s belief about what state it thinks its neighbors should be in. In physical terms, a message encodes the current estimate of the distribution of intra-node (internal) states based on prior messages from neighbors. The iteration processes attempts to find a set of intra-node distributions consistent with one another based on the interaction terms.

The way in which these beliefs are calculated is fairly straightforward. The messages (or node beliefs) all are initialized to the uniform distribution, i.e. if a node has k states, the node belief is set to $1/k$ for each state. Other initializations are possible, but in practice using random initializations had no discernible effect. In each round of belief propagation, messages are sent from each node to all of its neighbors (thus $2E$ messages are sent each round, where E is the number of edges in the graph).

Each message from a sending node to a receiving node is computed as follows. The node sending the message, n_s , initializes its message as its node potential, i.e. a vector of the Boltzmann factors of the energy of the node in its various configurations, which is a vector of length k_{n_s} . A list of nodes neighboring n_s is then generated, taking care to exclude the node receiving the message n_r . From each of these neighbor nodes, the sending node gathers all messages incoming to it. These incoming messages are all of the same length k_{n_s} , where k_{n_s} is the number of states of the sending node. The sending node then takes those incoming messages, and multiplies them in an element-wise fashion. Once the sending node has collated the incoming messages by multiplying them together, it then sends a message of its own to the receiving node, by taking this collated message and multiplying it by the edge potential between itself and the receiving node. This edge potential has dimension $k_{n_r} \times k_{n_s}$, where k_{n_r} and k_{n_s} are the number of states of the receiving node and sending node, respectively. Since the outgoing message from the sending node is of length k_{n_s} , and can be thought of as a column vector, the matrix multiplication of the edge potential and the column vector results in a new vector of length k_{n_r} . This product is then normalized, and can be thought of as a probability distribution over k_{n_r} states. This is the final message that is sent to the receiving node; note that it has the same length as the number of states in the receiving node. This process of gathering incoming messages, processing them, and sending outgoing messages is iterated for each node in the graph, and then the process starts over again if the messages haven't stopped changing.

The message passing process is terminated if the messages are within a threshold distance of what they were in the previous round of belief propagation. In particular, for each message we took the sum of the absolute differences of each message, and then took the sum of those sums to define a global change in all of the messages. If that global change was less than a threshold value, the belief propagation was considered to have converged. The default value of the threshold we used was 10^{-12} , which we considered to be fairly stringent. The stringency of the stopping condition can be put into context by noting that each message is a probability distribution over k states, and thus always has an entry of size greater than or equal to $1/k$, and k was never more than about 10^3 .

3.3.2 Computing Final Beliefs—Once the set of messages to and from all the nodes has converged, the beliefs of each node and edge can be computed, and used to approximate the free energy of the Markov random field. Formally, these beliefs are the marginal probabilities of the nodes or edges. For a node, then, the beliefs are equivalent to the (Boltzmann factor) of a discretized potential of mean force – i.e., the probability of each side chain configuration based on integrating over all other residues in the system. Similarly, the beliefs for an edge are the approximated marginal probabilities of all pairwise configurations in each adjacent node.

The node beliefs are computed in a manner almost identical to that of the message passing algorithm, with one slight difference. Since there is no receiving node in this computation of a single node's marginal probability, the incoming messages from all the node's neighbors are used, and none are omitted. Otherwise, the process is identical: the node belief is initialized as the node potential, and then messages from all neighbors are multiplied together element-wise, with the node potential, and the resulting vector is normalized.

Computing the edge beliefs is similar, but slightly more complicated, since there are two nodes contributing beliefs to each edge belief. Say the edge in question, e_{ab} , connects nodes n_a and node n_b . Then the edge belief is initialized as the edge potential ϕ_{ab} , which is a matrix of dimension $k_{n_a} \times k_{n_b}$, where k_{n_a} and k_{n_b} are the number of states in nodes n_a and n_b , respectively. Once the edge belief is initialized, the messages from all nodes incident to nodes n_a and n_b are collated, as in the message passing algorithm. That is, for node n_a , messages from all its neighbors other than n_b are multiplied together and then normalized, and similarly for node n_b omitting the message from node n_a . These modified node beliefs for n_a and n_b are then combined in an outer product, forming another matrix of size $k_{n_a} \times k_{n_b}$. This matrix is then multiplied element-wise by the edge potential. If $\mathbf{x}_a$ and $\mathbf{x}_b$ are the beliefs in the states that nodes n_a and n_b take, this element-wise matrix multiplication can be thought of as evaluating a normalized version of the node potential for these states $\phi_{ab}(\mathbf{x}_a, \mathbf{x}_b)$.

3.3.3 Computing Free Energies from Beliefs—The free energy of the Markov random field is computed by separately computing the entropy and average energy of the graph. In turn the nodes and edges contribute independently and in a pairwise fashion to the entropy and average energy. Let ϕ_n be the node potential of node n , and b_n the node belief, and similarly for the edge potentials ϕ_e and b_e . As a reminder, the beliefs b_n and b_e are the final estimated marginal probability distributions for the nodes and edges, respectively. Then the formula for the contribution of nodes to the average energy is

$$\langle U_{\text{nodes}} \rangle = -k_B T \sum_{n \in \text{nodes}} b_n \cdot \log \phi_n \quad (7)$$

where the log is taken element-wise, and “ $\cdot$ ” indicates a dot product. Similarly, the contribution of edges to the average energy is

$$\langle U_{\text{edges}} \rangle = -k_B T \sum_{e \in \text{edges}} b_e \cdot \log \phi_e \quad (8)$$

The total average energy is then simply

$$\langle U_{\text{total}} \rangle = \langle U_{\text{nodes}} \rangle + \langle U_{\text{edges}} \rangle \quad (9)$$

Appropriately enough, these formulae can be interpreted as taking the expectation of the energy, since

$$\phi_i = e^{-E_i/k_B T} \Rightarrow E_i = -k_B T \log \phi_i \quad (10)$$

and summing over the dot product of this quantity with the beliefs is precisely taking its expectation with respect to the probability density given by the beliefs.

Computing the entropies is similar to computing the average energy, but instead of taking the expectation of the energies of the nodes and edges, we take the expectation of the log of the probability densities, or beliefs. Additionally, there is a somewhat subtle issue in estimating the entropy of a continuous system using a discrete number of samples that must be addressed. The formulae for the node and edge entropies are then

$$S_{\text{nodes}} = -k_B \sum_{n \in \text{nodes}} b_n \cdot \log b_n \quad (11)$$

where the log is taken element-wise, and “ $\cdot$ ” indicates a dot product. Similarly, the contribution of edges to the average energy is

$$S_{\text{edges}} = -k_B \sum_{e \in \text{edges}} b_e \cdot \log b_e \quad (12)$$

However, as pointed out in,³⁷ the naïve sum of these entropy terms is not a valid estimate of the entropy of the system

$$S_{\text{naïve}} = S_{\text{nodes}} + S_{\text{edges}} \neq S_{\text{total}} \quad (13)$$

The correction factor due to discretizing the state-space turns out to be fairly simple: if each of d dihedral degrees of freedom in the graph is sampled using k samples,

$$S_{\text{Total}} = S_{\text{nodes}} + S_{\text{edges}} - k_B \cdot d \log \frac{k}{2\pi} \quad (14)$$

This result is re-derived in appendix A, where the issue is addressed in more depth.

4 Interactive Graph Visualization

Visualizing the graphs in the Markov random fields is extremely helpful in evaluating the interactions in the model. Since the graphs for the Markov random fields are constructed from three-dimensional structures, standard two dimensional visualizations are not particularly informative. The approach we took was to instead just visualize the graph in three dimensions. Additionally, we found it useful to not only inspect the graph topology, but also the physical layout, by superimposing the graph on the physical structure it represents.

To accomplish a simultaneous visualization of the graph and biomolecule structure in three dimensions, we built on the open-source 3dmol.js framework, implementing a simple graph layout superimposed on a pdb structure. The visualization tool is web-based, and available at https://donovanr.github.io/MRF_visualization_3dmol.html. A screenshot of the web interface is shown in Fig. 5.

5 Implementation Details

The time spent generating the edges of the MRF between nodes with a large number of states is the primary bottleneck in the process of extracting a free energy from the graph; the belief propagation algorithm itself runs in a small fraction of that time. We were able to take some rudimentary steps towards mitigating that bottleneck, though since the emphasis of this work not on the efficient construction of the graphs themselves, it was not a strong priority. Because calculation of each edge and each node potential can be performed entirely independently of all other nodes and edges, the process parallelizes trivially, and our Python implementation did take advantage of this fact to run the graph construction in parallel on up to 48 cores. We also note that the energy calls and geometric manipulation routines employed for tabulating the potentials were through a Python interface (OpenMM), and thus very slow compared to, say, the optimized code used in most MD engines. While this led to slow graph generation times in our case, there is no reason why a more optimized implantation could not mitigate this issue, especially considering the parallelizable nature of the task.

All three inference algorithms (brute-force summation, polymer growth, and loop belief propagation) were implemented by us in Python. The graph construction routines were also implemented by us in Python, building on the NetworkX graph library and the Python interface to OpenMM. The MRFs are implemented as annotated undirected graphs using the NetworkX library in Python. Side-chain configurations are generated programmatically on an even grid in dihedral space, and the elements of the node and edge potentials are computed for each side-chain configuration using energy calls to OpenMM. The node and edge potentials are stored as NumPy arrays and accessed as attributes of their corresponding node or edge in the NetworkX graph.

The graph visualization code was implemented by us in JavaScript, building on 3Dmol.js.

All code is available as source at <https://github.com/donovanr/protGM>.

6 Systems and Results

6.1 Validation of Sampling Methodology and Implementations in a Small Test System

When the systems are small enough to compute the free energy of the Markov random field by brute-force, it is straightforward to compare the performance of belief propagation to the gold standard of the brute-force value. However, most systems are too large to compute via brute-force, and in these situations, the belief propagation result needs to be validated against other methods. Here, we use the polymer growth approach, as described in section 3.2. Since the polymer growth algorithm is a stochastic algorithm, in this section we provide evidence that in a simple but realistic setting, BP yields results that are effectively identical with brute-force.

The test system we use to compare brute-force, polymer growth and BP is a threonine-4 peptide, which is the largest system for which we were able to calculate the free energy via brute-force. This peptide contains four threonines joined by peptide bonds, and capped at the C-terminus with N-methyl amide, and at the N-terminus with an acetyl group. We allow the peptide to relax and fold on itself, in order to mimic the potential for clashes in a real structure. After relaxing the structure for 1.0 nanoseconds by conventional molecular dynamics simulation, the final structure was saved as a single pdb file as a reference for graph construction.

With the reference structure determined, the graph is constructed using a fixed backbone, leaving only the side-chain degrees of freedom to vary. Each threonine has one heavy atom χ -angle dihedral degree of freedom, and one terminal methyl group dihedral degree of freedom. The capping groups each have one methyl group dihedral degree of freedom.

For this test system, we set the inter-residue cutoff to be 1.0 nanometers, which yields a fully connected graph between six nodes. Each node was sampled at 11 states per heavy atom χ -angle dihedral degree of freedom, and 3 states per methyl group dihedral degree of freedom. This results in four nodes with 33 states, and two nodes with 3 states, as well as one edge with 9 states, eight edges with 99 states, and six edges with 1089 states. For reference, the naïve size of the state space (as in brute-force) is thus $33^4 \cdot 3^2$, or about ten million states.

This graph is the largest for which we were able to obtain a brute-force estimate of the free energy: the value we obtained was 176.49554277, in units of $k_B T$. We report these values to an excessive number of decimal places not because they are physically relevant to this level of detail, but because *in this model* (as we will see), the agreement between brute-force and belief propagation is so acute.

This system was also useful for validating the accuracy of the polymer growth estimate of the free energy. In using the polymer growth algorithm for this graph, we set the number of states kept in each round of growth to be 100, and also performed 100 independent repetitions of the growth algorithm, adding the nodes in a different random order each time. The results for these polymer growth runs are displayed in Fig. 6. The median of the 100 runs was 176.51809563, and the 95% bootstrapped confidence interval for the median was (176.47527585, 176.55376641), all in units of $k_B T$.

The brute-force value is within in the confidence interval for the polymer growth median, yielding some confidence in the polymer growth approach. Moreover, the polymer growth estimate produces fairly tight bounds for the confidence interval, with the 95% confidence interval spanning only about $0.1 k_B T$.

In addition to validating the polymer growth calculation, we can evaluate the performance of belief propagation against both brute-force and polymer growth in the small system. The belief propagation algorithm is deterministic, and yields a value of $176.495544382 k_B T$. Remarkably, this estimate agrees to the brute-force value to six decimal places; as such, it is also within the bounds of the polymer growth estimate.

6.2 Determining Adequate Sampling Density in a Small Test System

The Threonine-4 peptide is also a useful system for investigating the dependence of the Markov random field free energy estimate upon the sample size used for the nodes and edges. The number of states k per node is a free parameter, and we expect that the free energy estimate will converge to a steady value as $k \rightarrow \infty$, just as the value of a Riemann sum converges to the area under a curve as the number of rectangles becomes large. The art of choosing k , then, is finding a large enough value that the free energy estimate has stabilized, but not so large that either constructing the graph or running belief propagation on the graph becomes prohibitively time or memory intensive.

Although the choice of k will always be system dependent, it is useful to get a sense for reasonable values of this parameter by using a small system like Threonine-4 that can be sampled systematically. The results of our exploration of state-space are shown in Fig. 7.

One fairly striking result from Fig. 7 that is the free energy estimate in this system is largely insensitive to the number of states per methyl-group degree of freedom, as long as that number is greater than one. Another nice result illustrated in the figure is that the free energy estimate seems to converge to a fairly steady value after somewhere around 7–11 states per heavy atom χ -angle degree of freedom.

These numbers are encouraging: a fairly small sample of the each dihedral degree of freedom seems to yield a reasonable estimate for the free energy; prohibitively exhaustive sampling does not appear to be necessary to obtain a reasonably converged result. However, this result should be taken with some healthy skepticism, for a few reasons. The foremost concern is that while each node represented one heavy atom and one methyl-group dihedral degree of freedom, there is no reason to believe that convergence at small sample density in this small system implies convergence in more difficult systems. Another concern is that the values for these free energy estimates were produced using belief propagation. While the previous section demonstrated that belief propagation was impressively accurate in the Threonine-4 system when using 11 states per heavy atom χ -angle dihedral degrees of freedom and 3 states per methyl group dihedral degree of freedom, a skeptic might be concerned that this level of agreement doesn't generalize to other sampling densities. Unfortunately, exploring the parameter space of even this small model with anything other than belief propagation is prohibitively time-intensive. Since these exploratory results are not meant to provide authoritative free energy estimates, but rather give a sense for

reasonable starting points in further investigations of more complicated systems, they find their use as a guide the exploration of more complex models in the following section.

6.3 Exploring Sampling Density in Larger Test Systems

Exploring a full pairwise grid of sample sizes for each combination of heavy atom and methyl group dihedral angle degrees of freedom is prohibitive in larger systems. Informed by the relative insensitivity of the free energy estimates in the previous section to more than two samples per methyl group degree of freedom, we explored primarily along the heavy atom dihedral degrees of freedom.

We studied ten additional peptides in order to assess the effect of both peptide length and side-chain size on the convergence of the energy and entropy estimates. Specifically, we examined capped peptides consisting of four or eight identical amino acids (either Alanine, Threonine, Valine, Leucine, or Tyrosine), after allowing them to relax from an extended initial configuration for 1.0 nanoseconds of molecular dynamics. For each peptide, we constructed a series of Markov random fields at different sampling densities.

The results of these studies are presented in figure 8, with the total free energy of the Markov random field broken down into the energetic and entropic contributions, i.e. $F = \langle U \rangle - TS$. The notation we employ of (m, n) in the figure indicates the number of samples per heavy atom and methyl group dihedral degree of freedom, respectively.

It is important to note that the convergence of the free energy estimates based on the number of samples per node in the graph will always be system dependent, so the figures depicting the convergence (or lack thereof) of these estimates for the test systems must be taken as largely exploratory, and not an authoritative assessment of adequate sample size.

Some broad trends are visible in the data. The first trend is that the energetic contribution to the free energy dominates the entropic contribution (at 298 K). This is sensible, as the backbone is kept fixed in these studies, freezing out otherwise important contributions to the entropy (a justification for doing so might be that in larger protein structures, the backbone is relatively stable, though of course further quantification of the effects of backbone flexibility on these estimates is desirable). Another trend visible in the data is that the estimate of the average energy seems to stabilize more quickly than the estimate of the entropy, at least in relative terms. Finally, the convergence of the entropy estimate seems to be worse for the peptides the larger they are, and the larger their side-chains are, though the convergence is only indisputably absent for the Leucine-8 system.

The overall satisfactory behavior of BP for the peptides suggests further applications to more complex systems is warranted, particularly if compared to independent unbiased calculations.

6.4 Comparing belief propagation free energies to statistically exact estimates in the binding pocket of a Protein: T4 Lysozyme Mutant

The binding pocket of a cavity-containing mutant of T4 Lysozyme (L99A), with the RSCB id of 1L83 (hereafter “lysozyme” for short),⁵⁷ provides an ideal system in which to investigate the speed and accuracy of belief propagation in a larger system.

The lysozyme system is composed of 164 amino acids, a significantly larger structure than the small peptides used previously. However, the larger structure, which is conformationally stable, provides a setting in which freezing the backbone when constructing the Markov random field is more reasonable. Eventually, one would want to sample multiple backbone conformations, as in,³⁷ to capture and assess the effects of the backbone degrees of freedom.

The binding pocket of the lysozyme mutant, where benzene can dock, has 11 amino acids whose α -carbons are within 0.5 nanometers of the benzene position: Ile78, Leu84, Val87, Tyr88, Leu91, Ala99, Val103, Val111, Leu118, Leu121, Phe153. Working in the interior of a protein is a fairly challenging environment in which to find favorable side-chain conformations due to tight packing,⁵⁸ a situation largely absent in the peptide systems investigated previously.

6.4.1 Artificial “Binding Pocket” Peptide—In order to tease apart the effects of the side-chain packing on the combinatorics of the state-space of the different side-chains, we initially examined an artificial test system: a peptide composed of the amino acids in the binding pocket. This allowed us to work in a slightly “easier” setting while still exploring parameters relevant to the lysozyme system. The peptide is shown in Fig. 9, and has sequence Ace0, Ile1, Val2, Tyr3, Leu4, Ala5, Val6, Val7, Leu8, Leu9, Phe10, Nme11. Backbone coordinates were generated by allowing the structure to relax for 1.0 nanoseconds by conventional molecular dynamics simulation from a fully extended configuration.

Generating the edge potentials for the Markov random field was the bottleneck in investigating this system: sampled at 11 states per heavy atom χ -angle degree of freedom, and 2 states per methyl group degree of freedom, it took approximately five days to generate the full Markov random field, running in parallel on eight cores. As noted already, our implementation was not optimized.

Since this model is too large for a brute-force computation, the polymer growth sampling algorithm was employed as a bronze standard to which belief propagation results can be compared. The polymer growth algorithm took approximately 12 hours to run 100 replicates in serial, retaining 1000 states per round of growth; for larger or smaller numbers of states kept per round, the run-time scaling is linear. The MRF generation time was given above and not included in the values just noted.

The belief propagation free estimates for the entropy and average energy of this system take approximately one second to compute, and they agree with the converged polymer growth estimate to at least $\pm 0.1 k_B T$, as seen in figure 10. This is a striking indication that similar to previous results using smaller state-spaces,³⁷ peptide or protein graphs large state-spaces pose no difficulties for the belief propagation algorithm.

The negative entropy values appearing in Fig. 10 are not physically meaningful but merely reflect the arbitrary choice of an offset constant based on our discretization scheme. See Appendix B.

One difficulty in using the polymer growth approach in a reasonably complex system such as this one is that, as noted in section 3.2, polymer growth is systematically biased at finite sample size. Here we keep increasing the number of samples kept per round of growth until the estimates converge, but even at sample sizes that entail a week running polymer growth, the estimate is still changing slightly as the sample size increases. Because atomic force fields are not believed to be accurate to much more than ~ 1 kcal/mole ($1k_B T \approx 0.6$ kcal/mol),^{59,60} and as long as the polymer growth estimate converges to significantly within that tolerance, it can be considered adequate for our purposes.

A last difficulty with the polymer growth algorithm in this system is that at low numbers of samples per round of growth, it often fails to produce any estimate at all. Specifically, the algorithm reaches a dead-end where it can't add any states from a new node that aren't clashes with any of the global configurations from the last round of growth. This is a general difficulty common sequential importance sampling methods such as polymer growth.²⁹ For instance, when ten states kept per round of growth, only 65 of the 100 polymer growth replicates completed growing the entire structure, though by the time 500 states are kept per round, 99% or greater of the replicates complete the growth process. In calculating the statistics in the figure below, these incomplete runs are omitted, though they do imply that the statistics for the growth runs with smaller numbers of states should be viewed with some skepticism.

6.4.2 Lysozyme Pocket—While the results for the binding pocket surrogate peptide are encouraging, a more honest evaluation of the accuracy of the belief propagation algorithm in the context of atomically dense protein structures should involve some explicit difficulty in side-chain positioning, as protein side-chain torsional entropies are known to affect protein-ligand interactions.⁵⁸ The binding pocket of a mutant lysozyme protein (PDB id 1L83) provides a convenient test-bed in which to evaluate the agreement of belief propagation and exact methods in such a situation. The structure of the lysozyme protein is shown in Fig. 11. The Markov random field for this structure was constructed with a 1.0 nanometer cutoff between the α -carbons of the residues, and used 11 states per heavy atom χ -angle dihedral degree of freedom, and 2 states per methyl group dihedral degree of freedom.

In this initial of study, we restricted the system to a graph containing just the residues in the binding pocket: Ile78, Leu84, Val87, Tyr88, Leu91, Ala99, Val103, Val111, Leu118, Leu121, Phe153. The binding pocket is defined as anything within 0.5 nm of the benzene molecule that is docked in the cavity of the lysozyme crystal structure.⁵⁷ Interactions with residues outside the binding pocket were not treated in this calculation (although they are accounted for in a subsequent computation, below). The free energy of this system was estimated using both polymer growth and belief propagation, in a process identical to the one described above for the binding pocket surrogate peptide.

The agreement between polymer growth and belief propagation is more difficult to achieve in this system than in the less tightly constrained peptide surrogate system. The polymer growth algorithm required extensive sampling, necessitating 10,000 states per round of growth in its computation, demanding a week-long run-time for this estimate. At this level of sampling, though, we do see agreement between the polymer growth values and the belief propagation values, as in figure 12.

In this more realistic environment, the polymer growth algorithm had more difficulty completing the growth process without dead-ending in a set of clash states. With 10 states kept per round of growth, only 65% of the 100 replicates completed, increasing to 90% at 500 states kept per round, and 95% at 5000 states per round. Nevertheless, the polymer growth did converge to a reasonably steady result when 500 or more states per round of growth were kept, as shown in Fig. 12. As in the binding pocket peptide calculations, the polymer growth algorithm took approximately 12 hours running in serial to perform 100 replicates of growth, when set to retain 1000 states per round, and the belief propagation algorithm took approximately one second to compute its free energy estimate. Both calculations use the same pre-constructed Markov random field as input, and so the cost of constructing the energy tables in the MRF, which can be considerable one-time cost (up to days in some of our cases), is not included in this run time.

If we take the converged polymer growth estimates to probabilistically bound the exact answer, the margin of error by which belief propagation misses this exact answer is impressively small. Though the tightness of such bounds is by nature system-dependent, these excellent results for the lysozyme binding pocket are very encouraging, and confirm both the received wisdom that loopy BP is uncannily accurate even when it is not guaranteed to be so,⁴¹ and that the Bethe approximation to the free energy for MRFs of peptides and proteins can be remarkably close to the true free energy.³⁷

6.4.3 F for a Lysozyme Pocket Mutant—As a last investigation of the effect that sampling density has on free energy estimates, we calculated the change in free energy due to a mutation in the binding pocket of the lysozyme structure.

Unlike in the previous section, where we worked with a small subset of the total structure, in this case we incorporate more of the residues to give context to the binding pocket interactions. To do so, we take advantage of the flexibility available in constructing the Markov random field, and include as nodes in the graph all residues which are neighbors of the residues in the binding pocket, but only allow these nodes to take on one state. That is, these border residues are frozen in their PDB rotamers but they interact with the flexible side chains in the binding pocket. The nodes in the binding pocket, as before, are sampled with 11 states per heavy atom χ -angle dihedral degree of freedom, and 3 states per methyl group dihedral degree of freedom. This larger Markov random field has the same state-space complexity as the one focused exclusively on the binding pocket, but the extra singleton nodes influence the states that the fully sampled nodes prefer to occupy, removing some of the boundary artifacts implicit in the previous calculation.

Unfortunately, while the larger MRF posed little difficulty for the belief propagation algorithm, the polymer growth algorithm was not able to sample it due to clashes overwhelming the sampling as the polymer grew, leaving us without a standard against which the performance of belief propagation can be compared for this system. Further work modifying the polymer growth algorithm could perhaps alleviate this issue, but as it stands, these final belief propagation results are presented without any point of comparison, and are of use primarily to illustrate the effect that sampling density has on the BP free energy estimates in a complex calculation.

Fig. 13 shows the belief propagation estimates of the change in free energy of the lysozyme system due to mutating Leucine-111 to a glycine residue, broken up into the constituent changes in entropy and average energy. As the number of samples per dihedral degree of freedom increases, the estimates also continue to change, indicating that not enough samples per degree of freedom have been taken to generate confident estimates of the change in free energy. However, the estimated change in entropy does show some signs of converging, though further sampling would be necessary to confirm this.

7 Discussion

Overall, our data indicate that reasonably “converged” free energy estimates can be obtained for Markov random fields of fixed-backbone peptides and a sample protein binding pocket. Here, convergence refers to the behavior of the estimates with increasingly fine discretization of the dihedral angles, which was not studied to our knowledge in prior applications of BP and MRFs to proteins. The BP computations themselves are extremely rapid, suggesting further work on the methodology could be very fruitful. In the longer term, such MRF/BP computations may be useful as part of a docking pipeline accounting systematically for conformational entropy - either indirectly to generate protein conformational ensembles or directly by developing a MRF including ligand interactions. The latter case might be feasible if interaction table construction can be accelerated based on three-dimensional grid representations for rigid fragments. We note that the MRF/BP approach has already been applied to protein-protein docking [CITE LANGMEAD] and further progress may be possible in that arena building on the present study.

There are many facets of this work that could be improved or expanded upon. Most practically, the current bottleneck in estimating free energies using Markov random fields is the generation of the Markov random fields themselves. Our pipeline uses OpenMM to calculate the interaction energies for the node and edge potentials. This library is remarkably flexible, but the Python interface by which we access the energetics information is quite slow compared to state of the art lower-level implementations such as those used in molecular dynamics packages or as implemented in.³⁷ At the level of sampling used here, this bottleneck is a practical hurdle, not a theoretical one, and there is no reason that implementing a more efficient graph generation pipeline could not result in large speedups in the time expended generating the Markov random fields. For instance, a system of 20 amino acids with 100 states per node requires computing the configurational energy for $20 \cdot (10^2)$ nodes states calls and $(20 \cdot 19/2) \cdot (10^2)^2$ edge states in a fully connected MRF, for a total of about two million energy calls. At 2 fs per MD time step, this is equivalent to the

number of energy computations in a 4 ns MD trajectory for a system of two residues with all forces other than electrostatics and dihedral forces frozen out, which could be computed in an embarrassingly parallel manner. Based on current benchmarks for small systems,⁶¹ a run-time on the order of a minute would be achievable for pre-computing the MRFs for our systems. This is orders of magnitude faster than our current proof of principle implementation, and fast enough for quick turn-around screening applications.

The quality of the model embodied in the current MRF graphs is a limitation, most notably the solvent description. The solvent model for the graphs is currently a simple uniform relative dielectric constant of 60.0. The most obvious improvement would be to use a distance-dependent dielectric constant⁶² which has been employed previously for pre-calculation of fragment interactions.⁶³ Non-pairwise solvent effects would be difficult to incorporate and hence appear to constitute a fundamental limitation of current MRFs. Potential users of the MRF/BP approach have to make a decision regarding the importance of speed and conformational sampling vs. model accuracy.

Even if generating the Markov random field is not slow enough to be the main bottleneck in the process, at some point the number of states sampled will be too large to hold in memory and perform computations with. This motivates the thought that instead of uniformly sampling states in dihedral space, one could be more selective in both generating and retaining samples. States that are explicit clashes and have effectively zero probability of occurring could be pruned away, saving a significant fraction of space. Additionally, one could consider sampling the dihedral degrees of freedom at different densities, depending on their relative importance in generating large conformational changes in the side-chain. That is, the χ_1 angles might be sampled at a higher density than the χ_2 angles, with coarser sampling as one moves down the side-chain. Further reduction in MRF complexity could be achieved by making more extensive use of nodes with singleton, or otherwise severely reduced state-spaces, employed for portions of the structure one has less interest in studying.

More radically, one could consider abandoning uniform sampling, and attempt to use, say, a Boltzmann sampled ensemble of states for the nodes as in prior polymer-growth work.^{30,31} This approach would require very careful attention to weighting the samples, as it is quite easy to arrive at an incorrect estimate of the partition function when the states account for different proportions of state space. For instance, the logarithmic correction of equation 25 is no longer entirely accurate, as the states no longer evenly divide up the volume of state space. The advantage to a “hands-off” approach like Boltzmann sampling is appealing though, as it would allow one to include arbitrary degrees of freedom in the samples, and it would be worth investigating.

Throughout this exploratory work, we have used a fixed backbone in all of the systems. To accurately capture the physics of the system these degrees of freedom need to be incorporated. Drawing samples from a backbone ensemble, and generating graphs for each backbone, as in³⁷ is an option. Another approach that would account for small flexing motions would be to include in the nodes some amount of flexibility in the backbone degrees of freedom, but not enough to alter the topology of the graph. This would expand

the size of the node and edge potentials by a multiplicative factor, so the benefit of doing so would have to be weighed against the cost of generating a larger Markov random field.

It is interesting to note that Minh has recently developed an exact free energy formalism for binding based on a sum over rigid receptor conformations.^{64,65} Insofar as the BP approach described here already includes further degrees of freedom necessary to the partition function, it could be of interest to combine the two approaches.

8 Conclusions

The goal of this investigation was to explore the performance of belief propagation (BP) approximations of the free energy of Markov random fields (MRFs) constructed using the energetic interactions of peptides and proteins based on a standard all-atom force field (Amber99SB). We compared BP free energy predictions to exact methods on fixed MRFs and checked convergence/self-consistency as the sampling density of the MRF is increased. In the regime where belief propagation can be compared to an exact brute-force result (small peptides), the belief propagation result agreed with brute-force to a startlingly high level of accuracy and precision. In larger systems where brute-force approaches fail and a polymer-growth estimate must be used as a point of comparison (lysozyme binding pocket and surrogate peptide), the belief propagation estimate is within the fairly tight bounds of the converged polymer growth estimates. In both large and small systems, the belief propagation algorithm takes about a second to compute its estimates, while the brute force and polymer growth estimates take anywhere from hours to days to run on equivalent hardware.

Building on previous work,^{35–40,66,67} the present study reinforces the potential of BP calculations for equilibrium molecular mechanics calculations. In particular, we have begun to clarify the requirements for constructing peptide and protein MRFs of sufficient accuracy, and our results suggest that BP calculations using the latest hardware and BP-specialized software could be extremely powerful. The present study fully includes side-chain flexibility but more follow-up work clarifying model requirements would be a useful complement to prior BP calculations with backbone flexibility.³⁷

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We very much appreciate helpful discussions with Ramu Anandakrishnan, David Koes, Justin Spiriti, and Ernesto Suarez, as well as support from the National Science Foundation under grant MCB-1119091.

References

1. Gilson MK, Zhou HX. Annual review of biophysics and biomolecular structure. 2007; 36:21.
2. Park S, Khalili-Araghi F, Tajkhorshid E, Schulten K. The Journal of chemical physics. 2003; 119:3559–3566.
3. Woo HJ, Roux B. Proceedings of the National Academy of Sciences of the United States of America. 2005; 102:6825–6830. [PubMed: 15867154]
4. Singh N, Warshel A. Proteins: Structure, Function, and Bioinformatics. 2010; 78:1705–1723.

5. Dror RO, Dirks RM, Grossman J, Xu H, Shaw DE. Annual review of biophysics. 2012; 41:429–452.
6. Gumbart JC, Roux B, Chipot C. Journal of chemical theory and computation. 2012; 9:794–802.
7. Hansen N, Van Gunsteren WF. Journal of chemical theory and computation. 2014; 10:2632–2647. [PubMed: 26586503]
8. Rao SN, Singh UC, Bash PA, Kollman PA. Nature. 1987; 328:551–554. [PubMed: 3302725]
9. Meirovitch H. Current opinion in structural biology. 2007; 17:181–186. [PubMed: 17395451]
10. Shivakumar D, Williams J, Wu Y, Damm W, Shelley J, Sherman W. Journal of chemical theory and computation. 2010; 6:1509–1519. [PubMed: 26615687]
11. Borhani DW, Shaw DE. Journal of Computer-Aided Molecular Design. 2012; 26:15–26. [PubMed: 22183577]
12. Chipot C. Wiley Interdisciplinary Reviews: Computational Molecular Science. 2014; 4:71–89.
13. Verdonk ML, Cole JC, Hartshorn MJ, Murray CW, Taylor RD. Proteins: Structure, Function, and Bioinformatics. 2003; 52:609–623.
14. Trott O, Olson AJ. Journal of computational chemistry. 2010; 31:455–461. [PubMed: 19499576]
15. Ma DL, Chan DSH, Leung CH. Chemical Society Reviews. 2013; 42:2130–2141. [PubMed: 23288298]
16. Sastry GM, Adzhigirey M, Day T, Annabhimoju R, Sherman W. Journal of computeraided molecular design. 2013; 27:221–234.
17. Leach AR, Shoichet BK, Peishoff CE. Journal of medicinal chemistry. 2006; 49:5851–5855. [PubMed: 17004700]
18. Moitessier N, Englebienne P, Lee D, Lawandi J, Corbeil C. British journal of pharmacology. 2008; 153:S7–S26. [PubMed: 18037925]
19. Huang SY, Zou X. International journal of molecular sciences. 2010; 11:3016–3034. [PubMed: 21152288]
20. Sousa S, Ribeiro A, Coimbra J, Neves R, Martins S, Moorthy N, Fernandes P, Ramos M. Current medicinal chemistry. 2013; 20:2296–2314. [PubMed: 23531220]
21. Rentzsch R, Renard BY. Briefings in bioinformatics. 2015:bbv008.
22. Chen YC. Trends in pharmacological sciences. 2015; 36:78–95. [PubMed: 25543280]
23. Meirovitch H. Journal of Physics A: Mathematical and General. 1982; 15:L735.
24. Meirovitch H. Physical Review A. 1985; 32:3699.
25. Garel T, Orland H. Journal of Physics A: Mathematical and General. 1990; 23:L621.
26. Grassberger P. Physical Review E. 1997; 56:3682.
27. Grassberger P. Computer Physics Communications. 2002; 147:64–70.
28. Liu JS, Chen R. Journal of the American statistical association. 1998; 93:1032–1044.
29. Liu, JS. Monte Carlo strategies in scientific computing. Springer Science & Business Media; 2008.
30. Zhang X, Mamonov AB, Zuckerman DM. Journal of computational chemistry. 2009; 30:1680–1691. [PubMed: 19504588]
31. Lettieri S, Mamonov AB, Zuckerman DM. Journal of Computational Chemistry. 2010; 32:1135–1143. [PubMed: 21387340]
32. Donald, B. Computational Molecular Biology. MIT Press; 2011. Algorithms in Structural Molecular Biology.
33. Kindermann, R., Snell, L. Markov random fields and their applications. American Mathematical Society; 1980.
34. Yanover C, Weiss Y. Approximate inference and protein-folding. Advances in neural information processing systems. 2003:1481–1488.
35. Kamisetty H, Xing E, Langmead C. J Comp Bio. 2008; 15:755–766.
36. Kamisetty, CH. Langmead Conformational Free Energy of Protein Structures: Computing Upper and Lower bounds. Proc. Structural Bioinformatics and Computational Biophysics (3DSIG); 2008; p. 23-24.
37. Kamisetty H, Ramanathan A, Bailey-Kellogg C, Langmead CJ. Proteins. 2011; 79:444–62. [PubMed: 21120864]

38. Razavian NS, Kamisetty H, Langmead CJ. BMC genomics. 2012; 13(Suppl 1):S5.
39. Langmead, CJ. Protein Conformational Dynamics. Springer International Publishing; Cham: 2014. p. 87-105.
40. Kamisetty H, Ghosh B, Langmead CJ, Bailey-Kellogg C. Journal of Computational Biology. 2015; 22:474–486. [PubMed: 25973864]
41. Murphy, KP., Weiss, Y., Jordan, MI. Loopy belief propagation for approximate inference: An empirical study. Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence; 1999. p. 467-475.
42. Heskes T. Stable fixed points of loopy belief propagation are local minima of the bethe free energy. Advances in neural information processing systems. 2002:343–350.
43. Tatikonda, SC. Convergence of the sum-product algorithm. Information Theory Workshop, 2003. Proceedings. 2003 IEEE; 2003. p. 222-225.
44. Ihler AT, Fisher JW, Willsky AS. Message errors in belief propagation. Advances in Neural Information Processing Systems. 2004:609–616.
45. Mooij JM, Kappen HJ. IEEE Transactions on Information Theory. 2007; 53:4422–4437.
46. Gilson MK, Given JA, Bush BL, McCammon JA. Biophysical journal. 1997; 72:1047. [PubMed: 9138555]
47. Canutescu AA, Shelenkov AA, Dunbrack RL. Protein science : a publication of the Protein Society. 2003; 12:2001–14. [PubMed: 12930999]
48. Zuckerman, D. Statistical Physics of Biomolecules: An Introduction. Taylor & Francis; 2010.
49. Weideman JAC. The American mathematical monthly. 2002; 109:21–36.
50. Jaynes ET. Physical review. 1957; 106:620.
51. Zuckerman DM, Woolf TB. Physical Review Letters. 2002; 89:180602. [PubMed: 12398588]
52. Yedidia JS, Freeman WT, Weiss Y. Exploring artificial intelligence in the new millennium. 2003; 8:236–239.
53. Pearl J. Artificial intelligence. 1986; 29:241–288.
54. Yedidia JS, Freeman WT, Weiss Y. Advances in neural information processing systems. 2001; 13
55. Bethe HA. Proceedings of the Royal Society of London Series A, Mathematical and Physical Sciences. 1935; 150:552–575.
56. Kikuchi R. Physical review. 1951; 81:988.
57. Eriksson A, Baase W, Wozniak J, Matthews B. Nature. 1992; 355:371–373. [PubMed: 1731252]
58. DuBay KH, Geissler PL. Journal of molecular biology. 2009; 391:484–497. [PubMed: 19481551]
59. Kaminski GA, Friesner RA, Tirado-Rives J, Jorgensen WL. The Journal of Physical Chemistry B. 2001; 105:6474–6487.
60. Shirts MR, Pitner JW, Swope WC, Pande VS. The Journal of chemical physics. 2003; 119:5740–5761.
61. [Accessed: 2017-10-19] Amber 16 Benchmarks. ImplicitSolvent. <http://ambermd.org/gpus/benchmarks.htm#>
62. Warshel A, Levitt M. Journal of molecular biology. 1976; 103:227–249. [PubMed: 985660]
63. Spirti J, Zuckerman DM. Journal of chemical theory and computation. 2014; 10:5161. [PubMed: 25400525]
64. Minh DD. The Journal of chemical physics. 2012; 137:104106. [PubMed: 22979849]
65. Xie B, Nguyen TH, Minh DD. Journal of Chemical Theory and Computation. 2017
66. Balakrishnan S, Kamisetty H, Carbonell J, Lee S, CJL. Proteins: Structure, Function, and Bioinformatics. 2011; 79:1061–1078.
67. Razavian, NS. PhD Dissertation. Carnegie Mellon University; 2013.

A Correcting for Discretization in Partition Function Estimates

Following Zuckerman,⁴⁸ the entropy of a continuous physical system is given by

$$S/k_B = - \int_V \rho(x) \log (\rho(x)V_0) dx, \quad (15)$$

Where V_0 is an arbitrary fiducial volume introduced for the sake of dimensional consistency, and which cancels upon computing differences in entropy. This equation for the entropy resembles the $\sum p \log p$ formula, but some further manipulations are needed before it is of use in a discretely sampled space.

To discretize the entropy formula so that it is applicable when dealing with a finite number of samples, we approximate it by a Riemann sum in an arbitrary number of dimensions:

$$S/k_B = - \int_V \rho(x_i) \log (\rho(x)V_0) dx \quad (16)$$

$$\approx - \sum_i^N \rho(x_i) \log (\rho(x)V_0) \Delta x_i \quad (17)$$

where the equality holds as $N \rightarrow \infty$ and all $\Delta x_i \rightarrow 0$. Here, Δx_i here is not just one dimensional, but an arbitrarily dimensioned sub-volume of configuration space. Note that N here is the number of rectangles in the integral, and not the usual statistical mechanics usage of the number of particles in a system.

When using a finite number of samples to represent a distribution, the probability of each sample p_i (that you might get from e.g. a marginal probability in belief propagation) is related to the probability *density* ρ by $p_i = \rho(\Delta x_i) \Delta x_i$. That is, the area of a rectangle (in the Riemann sum) is equal to its width times its height. Substituting in, we get

$$S/k_B = - \sum_i^N \left[\left(\frac{p_i}{\Delta x_i} \right) \log \left(\frac{p_i}{\Delta x_i} V_0 \right) \right] \Delta x_i \quad (18)$$

$$= - \sum_i^N p_i \log \left(\frac{p_i}{\Delta x_i} V_0 \right) \quad (19)$$

$$= - \sum_i^N p_i \log p_i - \sum_i^N p_i \log \frac{V_0}{\Delta x_i} \quad (20)$$

For a uniform discretization, $\Delta x = V/N$, so

$$S/k_B = - \sum_i^N p_i \log p_i - \sum_i^N p_i \log \frac{V_0}{V/N} \quad (21)$$

$$= - \sum_i^N p_i \log p_i - \log \frac{V_0}{V/N} \sum_i^N p_i \quad (22)$$

$$= - \sum_i^N p_i \log p_i - \log \frac{V_0}{V/N} \quad (23)$$

$$= - \sum_i^N p_i \log p_i - \log \frac{N}{V/V_0} \quad (24)$$

$$= - \sum_i^N p_i \log p_i - \log N + \log V/V_0 \quad (25)$$

So we see the logarithmic correction term ($-\log N$), as used in e.g.³⁷ for discretizing a continuous system, emerge naturally from a statistical physics treatment of the problem – it's just accounting for bin widths when translating from an integral to a Riemann sum. Lastly, we note that the constant term, $\log V/V_0$, will cancel when computing entropy differences, for example the change in entropy upon binding, even when using entropies computed using a different number of states.

When we apply this formula in Eq. (3), we choose 2π for V_0 and use k rather than N for the number of states.

B Corrections for $\hat{Z}$ and F, but not E

Similarly, we must correct computations of the configurational partition function $\hat{Z}$ when approximating it using a finite number of samples:

$$\hat{Z} = \int_V e^{-\beta U(x)} dx \quad (26)$$

$$\approx \sum_i^N e^{-\beta U(x_i)} \Delta x_i \quad (27)$$

where the sum becomes exact in the limit $N \rightarrow \infty$ and all $x_i \rightarrow 0$. If we use a uniform spatial grid, $\Delta x_i = \Delta x = \frac{V}{N}$ is a constant, and we can factor it out of the sum:

$$\hat{Z} = \frac{V}{N} \sum_i^N e^{-\beta U(x_i)} \quad (28)$$

Where again, it is useful to think of $e^{-\beta U(x_i)} \Delta x$ as the area of a box in a Riemann sum.

We can transform into the free-energy picture, using $Z = \hat{Z}/V_0 = e^{-\beta F}$:

$$F = -\frac{1}{\beta} \log(\hat{Z}/V_0) \quad (29)$$

$$= -\frac{1}{\beta} \log\left(\frac{V/V_0}{N} \sum_i^N e^{-\beta U(x_i)}\right) \quad (30)$$

$$= -\frac{1}{\beta} \log\left(\sum_i^N e^{-\beta U(x_i)}\right) + \frac{1}{\beta} \log \frac{N}{V/V_0} \quad (31)$$

$$= \left[-\frac{1}{\beta} \log\left(\sum_i^N e^{-\beta U(x_i)}\right) + \frac{1}{\beta} \log N - \frac{1}{\beta} \log(V/V_0) \right] \quad (32)$$

and we see the same correction appear here as in the entropy calculation.

It turns out that the average energy formula doesn't need a correction, since the correction terms cancel in a linear average:

$$\langle E \rangle = \frac{1}{\hat{Z}} \int_V U(x) e^{-\beta U(x)} dx \quad (33)$$

$$\approx \frac{1}{Z} \sum_i^N U(x_i) e^{-\beta U(x_i)} \Delta x \quad (34)$$

$$\frac{1}{Z} \frac{V}{N} \sum_i^N U(x_i) e^{-\beta U(x_i)} \quad (35)$$

$$= \frac{\sum_i^N U(x_i) e^{-\beta U(x_i)}}{\sum_i^N e^{-\beta U(x_i)}} \quad (36)$$

So the only real correction is for the entropy, but when computing the free energy or partition function directly, that same correction must also be included.

The constant term in equations 25 and 32 is worth discussing briefly. While the $\log N$ term allows entropy and free energy estimates made using different numbers of samples to be compared to each other, the constant term $\log V/V_0$ renders the overall estimate arbitrary, up to a constant. The value of V in our computations is the volume of configurational space explored in sampling states for the Markov random fields. Since we only explore side-chain dihedral degrees of freedom, this volume is directly proportional to the number of side-chains represented in the MRF. The volume explored for each side-chain dihedral is 2π , so if there are d dihedral degrees of freedom in the graph, $V = (2\pi)^d$. Similarly, if each dihedral in the graph is sampled using k states, then the total number of states is $N = k^d$. Lastly, since V_0 is treated as an arbitrary constant (in some ways similar to the standard 1M concentration needed in calculating binding affinities), we are free to set it to 1 radian, with the foreknowledge that the specific values we assign it will be immaterial in physical calculations. Plugging in these values for the number of states and the volumes, we get

$$-\log N + \log V/V_0 = -\log k^d + \log (2\pi)^d/1 \quad (37)$$

$$= -\log \frac{k^d}{(2\pi)^d} \quad (38)$$

$$= -d \log \frac{k}{2\pi} \quad (39)$$

which is the form of the correction we use, as is equation 14.

As a last note, it is worth drawing attention to the negative entropy values that appear at times in the figures in this work. These negative values are solely the result of our choice of 1 radian for V_0 , and do not reflect any sort of spooky physics going on. For instance, had we chosen 2π radians instead of one, the entropies would all be positive, but since at the end of the day it is entropy *differences* that matter, the choice of V_0 only affects the computation by way of numerical stability. For this purpose, any number on the order of 2π is adequate.

C Uniform Sampling Computes the Correct Partition Function

In constructing a Markov random field to approximate the free energies of biomolecules, there is considerable freedom in choosing the state space of the model. An immediate simplification is to only consider dihedral degrees of freedom in the side-chains of the amino acids, which is what we have done in this work. In analyzing the correctness of partition function (or equivalently, free energy) computations, it is simplest to ignore belief propagation or polymer growth or other particular algorithms, and focus instead on brute-force computation, since all three converge on the same answer, but brute-force is the simplest to reason about.

In fact, the simplest non-trivial model system to analyze is a graph with just one node, and hence no edges. This one node graph represents a fixed-backbone peptide with one amino acid, and let us further assume that the single side-chain has only one dihedral angle degree of freedom. We start with the definition of the partition function for the system:

$$Z = \int_0^{2\pi} e^{-\beta U(\theta)} d\theta \quad (40)$$

Since there is no Jacobian factor for the dihedral angle, uniform sampling over the state space volume ($V = [0, 2\pi)$) is the proper way to discretize the integral. A uniform discretization of this integral yields

$$Z \approx \frac{2\pi}{N} \sum_i^N e^{-\beta U(\theta_i)} \quad (41)$$

Now consider the situation from the belief propagation point of view: if the states for the node are sampled uniformly, and plugged in our Markov random field as usual, this is precisely the (correct) answer we will get.

On the other hand, one might be tempted to sample according to, say, a Boltzmann distribution. In this case, in plugging such states into a Markov random field and treating them as usual, we would get the wrong answer. This is most easily illustrated in the integral formulation, where the effect of the Boltzmann weighting of the states, $\rho(\theta)$, can be seen to induce a sort of double-counting of the weights:

$$\begin{aligned}
Z_{\text{wrong}} &= \int_0^{2\pi} e^{-\beta U(\theta)} \rho(\theta) d\theta \\
&= \int_0^{2\pi} e^{-\beta U(\theta)} \frac{e^{-\beta U(\theta)}}{Z(\beta)} d\theta \\
&= \frac{1}{Z(\beta)} \int_0^{2\pi} e^{-2\beta U(\theta)} d\theta \\
&= \frac{Z(2\beta)}{Z(\beta)}
\end{aligned}$$

It might be possible to correct for this effect, but the simple approach taken here is to eschew sophisticated sampling techniques and use a uniform sampling of states.

Algorithms

Below we proved pseudocode for the three main algorithms used in our work. Actual code is available at <https://github.com/donovanr/protGM> or in the supplementary material.

Algorithm 1

Brute-force computation of the free energy

```

procedure BruteForce( $G$ )
   $S \leftarrow \text{states}(n_1) \otimes \text{states}(n_2) \otimes \dots \otimes \text{states}(n_N)$   ▷ All possible configurations of all nodes
   $Z \leftarrow 0$ 
  for  $s \in S$   ▷ For each configuration of all nodes
     $z_{\text{nodes}} \leftarrow 1$ 
    for  $n \in \text{nodes}(G)$   ▷ Multiply all the node potentials
       $z_{\text{nodes}} \leftarrow z_{\text{nodes}} \cdot \phi_n(s_n)$ 
     $z_{\text{edges}} \leftarrow 1$ 
    for  $(n_a, n_b) \in \text{edges}(G)$   ▷ Multiply all the edge potentials
       $z_{\text{edges}} \leftarrow z_{\text{edges}} \cdot \phi_{n_a n_b}(s_{n_a}, s_{n_b})$ 
     $Z \leftarrow Z + (z_{\text{nodes}} \cdot z_{\text{edges}})$   ▷ Sum
   $\beta F \leftarrow -\log(Z)$ 
  return  $\beta F$ 

```

Algorithm 2

Polymer growth algorithm for computing the free energy

```

procedure PolymerGrowth( $G$ , sample size =  $K$ )
   $N \leftarrow |\text{nodes}(G)|$ 
   $\beta F \leftarrow 0_N$ 
   $\beta U \leftarrow 0_K$ 
   $S_{\text{saved}} \leftarrow \{\}$ 
  for  $n \in 1, \dots, N$ 
     $\beta U \leftarrow 0_{K \times \text{states}(n)}$ 

```

```

 $S^{\text{new}} \leftarrow \mathbf{0}_{K \times n \times \text{states}(n)}$ 
for  $s_i \in \text{states}(n)$ 

  for  $s_j^{\text{saved}} \in S^{\text{saved}}$ 

     $s_{ij}^{\text{new}} \leftarrow [(s_j^{\text{saved}})_1, \dots, (s_j^{\text{saved}})_{n-1}, s_i]$ 

     $\Delta\beta U_{ij} \leftarrow \beta U(s_{ij}^{\text{new}}) - \beta U(s_j^{\text{old}})$ 

     $\Delta\beta F_n \leftarrow -\log\left(\frac{1}{K} \sum_{ij} e^{\Delta\beta U_{ij}}\right)$ 

   $S^{\text{saved}} \leftarrow \text{downsample}(S^{\text{new}}, \text{size} = K, \text{weight} = \text{Boltzmann})$ 

 $\beta F \leftarrow \sum_{n \in 1, \dots, N} \beta F_n$ 

 $\beta U \leftarrow \frac{1}{K} \sum_{s \in S^{\text{saved}}} \beta U(s)$ 

 $\beta TS \leftarrow \beta U - \beta F$ 

return  $\beta F$ 

```

Algorithm 3

Belief propagation computation of the free energy

```

procedure BeliefPropagation( $G$ )
  while not converged
    for  $n_i \in \text{nodes}(G)$ 
      for  $n_j \in \mathcal{N}(n_i)$ 
         $\mu_{j \rightarrow i} \leftarrow \sum_{\text{states}(n_j)} [\phi_i(x_i, x_j) \phi_j(x_j) \prod_{n_k \in \mathcal{N}(n_j) \setminus n_i} \mu_{k \rightarrow j}]$ 
         $\mu_{j \rightarrow i} \leftarrow \mu_{j \rightarrow i} / \sum_{\text{states}(n_j)} \mu_{j \rightarrow i}$ 
      for  $n_i \in \text{nodes}(G)$ 
         $b_i \leftarrow \phi_i(x_i) \prod_{n_j \in \mathcal{N}(n_i)} \mu_{j \rightarrow i}$ 
         $b_i \leftarrow b_i / \sum_{\text{states}(n_i)} b_i$ 
         $\beta U_i^{\text{node}} \leftarrow -\sum_{\text{states}(n_i)} b_i \log(\phi_i(x_i))$ 
         $\beta TS_i^{\text{node}} \leftarrow -(1 - |\mathcal{N}(n_i)|) \sum_{\text{states}(n_i)} b_i \log(b_i)$ 
      for  $(n_i, n_j) \in \text{edges}(G)$ 
         $b_{ij} \leftarrow \phi_i(x_i, x_j) (\phi_j(x_j) \prod_{n_k \in \mathcal{N}(n_j) \setminus n_i} \mu_{k \rightarrow j}) (\phi_i(x_i) \prod_{n_k \in \mathcal{N}(n_i) \setminus n_j} \mu_{k \rightarrow i})$ 
         $b_{ij} \leftarrow b_{ij} / \sum_{\text{states}(n_i, n_j)} b_{ij}$ 
         $\beta U_{ij}^{\text{edge}} \leftarrow -\sum_{\text{states}(n_i, n_j)} b_{ij} \log(\phi_{ij}(x_i, x_j))$ 
         $\beta TS_{ij}^{\text{edge}} \leftarrow -\sum_{\text{states}(n_i, n_j)} b_{ij} \log(b_{ij})$ 

```

$\beta U \leftarrow \sum_{n_i \in \text{nodes}(\mathbf{G})} \beta U_i^{\text{node}} + \sum_{(n_i, n_j) \in \text{edges}(\mathbf{G})} \beta U_{ij}^{\text{edge}}$

$\beta TS \leftarrow \sum_{n_i \in \text{nodes}(\mathbf{G})} \beta TS_i^{\text{node}} + \sum_{(n_i, n_j) \in \text{edges}(\mathbf{G})} \beta TS_{ij}^{\text{edge}}$

$\beta F \leftarrow \beta U - \beta TS$

return βF

Figure 1.

Mapping a molecular mechanics model to a Markov random field (MRF). MRFs generalize the Ising model to arbitrary graphs, arbitrary state-spaces per node, and arbitrary interactions between nodes. Here we use MRFs for protein side chains X_i on a rigid backbone that experience ϕ “potentials” – which are Boltzmann factors of interaction energies in a MRF. Random variables (side-chains) are represented as circles, and interaction potentials (both self and pairwise) are represented as black squares.

Figure 2.

Graphs are constructed from a PDB structure, with node states specified by a set of poses per side-chain, and node and edge energies given by the Amber99SB forcefield. Once the graph is constructed, the free energy and other statistics can be computed by a variety of means. Here we apply belief propagation, and compare its performance to brute-force and polymer growth.

Figure 3.

Example of a MRF for a peptide. Each residue is a node, with a fixed number of states per dihedral angle. Nodes interact with their neighbors if they are within a tunable cutoff distance (here 0.8 nm).

Figure 4.

Polymer growth procedure for a graph with three nodes. Nodes are added successively, and at each step the free energy is computed from an ensemble of states sampled in the previous step. The total free energy is then calculated using the sum of the free energy differences tabulated during the growth process. Note that the free energies F_{AB} and F_{ABC} include *all* node and edge terms for those graphs.

Markov Random Field Topology Visualization with 3Dmol.js

Figure 5.

Web-based 3D visualization of the graph structure of the Markov random fields used in our computations. The interface allows the visualization of graphs for arbitrary pdb structures, with a user-determined cutoff distance between residues. The user is able to zoom, pan, and rotate the structure, as well as specify different visualization options. Shown here is the T4 lysozyme L99A mutant based on PDB code 1L83.

Figure 6.

Free energy (F) results for a test system, Thr-4. The median of 100 polymer growth estimates is shown as a vertical blue line, the brute-force value as a red line, and the belief propagation value as a green line. The brute-force and belief propagation values are close enough that the brute-force value is masked by the belief propagation value. The 95% confidence interval of the polymer growth median is shown as a blue band around the median.

Figure 7.

Free energy estimates in Threonine-4, for different numbers of states per degree of freedom. The horizontal axis indexes the number of states per heavy atom chi angle degree of freedom, while the colors correspond to different choices of the number of states per methyl-group dihedral angle degree of freedom (“hchis”, for short).

Figure 8.

Belief propagation results for a series of peptides. Energy (blue) and entropy (green) estimates for ten homopolymer peptides (Alanine-4, Alanine-8, Threonine-4, Threonine-8, Valine-4, Valine-8, Leucine-4, Leucine-8, Phenylalanine-4, and Phenylalanine-8) are shown at different discretizations. Additionally, the total free energy is plotted in black, along with the energy. The pairs of numbers on the x-axis indicate how many states per heavy atom and methyl group degree of freedom are used in the calculation. As the number of states per dihedral degree of freedom increases from one to five, both the energy and entropy estimates converge to a stable value ($\pm \sim 1$ kcal/mol).

Figure 9. Structure of the binding pocket surrogate peptide. The graph is highly connected, containing 54 out of 66 possible edges.

Figure 10.

Free energy estimates of the binding pocket surrogate peptide, broken up into the constituent entropy and average energy estimates, as well as the total Free energy. The value of the polymer growth estimate converges to within $\pm 0.1 k_B T$ of the belief propagation value by the time 500 states per round of growth are kept, with better convergence as the number of states kept per round increases.

Figure 11. Structure of the lysozyme mutant and MRF used in this study. Nodes sampled with multiple configuration are shown as gray spheres. Edges are shown between these nodes and also with nodes sampled only with a single configuration.

Figure 12.

Results for the binding pocket of the lysozyme protein, broken up into the constituent entropy and average energy estimates. The value of the polymer growth estimate of the average energy converges to within $\pm 1.0 k_B T$ of the belief propagation value using 500 states per round of growth, though the belief propagation value is slightly outside the 95% confidence interval for the estimates until 10,000 states are used per round of growth.

Figure 13.

The belief propagation estimate for the change in free energy of the lysozyme structure due to mutating Leucine-111 to a Glycine. The change in free energy is broken down into the change in average energy (green), and the change in entropy at 298 K (blue). Additionally, the total free energy is plotted in black, along with the energy. The computation is performed at four different sampling densities, always with two samples per methyl group dihedral degree of freedom, and between 3 and 11 samples per heavy atom χ -angle dihedral degree of freedom. The fluctuation in the change in average energy computation indicates that more samples are needed in order to reach a realistic estimate. The estimate of the change in entropy displays somewhat better behavior.