Protein Design by Sampling an Undirected Graphical Model of Residue Constraints

John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg

Abstract— **This paper develops an approach for designing protein variants by sampling sequences that satisfy residue constraints encoded in an undirected probabilistic graphical model. Due to evolutionary pressures on proteins to maintain structure and function, the sequence record of a protein family contains valuable information regarding position-specific residue conservation and coupling (or covariation) constraints. Representing these constraints with a graphical model provides two key benefits for protein design: a probabilistic semantics enabling evaluation of possible sequences for consistency with the constraints, and an explicit factorization of residue dependence and independence supporting efficient exploration of the constrained sequence space. We leverage these benefits in developing two complementary MCMC algorithms for protein design: constrained shuffling mixes wild-type sequences position-wise and evaluates graphical model likelihood, while component sampling directly generates sequences by sampling clique values and propagating to other cliques. We apply our methods to design WW domains. We demonstrate that likelihood under a model of wild-type WWs is highly predictive of foldedness of new WWs. We then show both theoretical and rapid empirical convergence of our algorithms in generating high-likelihood, diverse new sequences. We further show that these sequences capture the original sequence constraints, yielding a model as predictive of foldedness as the original one.**

Index Terms— **Protein design, residue coupling, graphical models, Markov chain Monte Carlo (MCMC)**

I. Introduction

Protein engineering seeks to produce amino acid sequences with desired characteristics, such as specified structure [2], [6] or catalytic activity [9], [14]. This is a difficult problem due to the complex relationship between the choices of amino acid types and their impact on structure and function. Consequently, a number of different general approaches have been pursued. Homology-based approaches design new proteins to satisfy patterns observed in existing proteins, such as sequence motifs [11] or polar-nonpolar patterning [5]. Structure-based approaches employ biophysical models to predict the energies of possible amino acid substitutions, and develop sophisticated optimization techniques to identify low-energy sequences [2], [6], [9]. It is also possible to incorporate experimental measurements of the effects of amino acid choices [8], [12], or to work with entire sequential fragments rather than single amino acids [14], [18], [22]–[24].

J. Thomas and C. Bailey-Kellogg are with the Department of Computer Science, Dartmouth College, Hanover, NH 03755.
Email: {jthomas, cbk}@cs.dartmouth.edu
N. Ramakrishnan is with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24061.
Email: naren@cs.vt.edu

Interactions between residues have always been at the heart of structure-based design approaches, but most homology-based design work has focused on conservation (common amino acid types at particular residue positions), thus treating different residue positions as independent. However, Ranganathan and colleagues recently demonstrated convincingly the importance of *residue coupling* (common pairs of amino acid types at particular pairs of positions, also known as correlated mutations or co-evolving residues). In designing new, stably folded [19] and functional [17] WW domains, they showed that accounting for residue coupling, in addition to conservation, was to some extent both necessary and sufficient for viability. Their approach ensures that the new sequences have the same conservation statistics as wild-type ones, and optimizes for reproducing the observed coupling statistics, according to their 'statistical coupling analysis' measure [10].

There are some potential pitfalls one faces in generating new sequences consistent with observed conservation and coupling. Simply reproducing the overall conservation and coupling statistics (as in the Ranganathan approach) may not be sufficient to capture the characteristics of the underlying protein family. In the multiple sequence alignment in Fig. 1, observe that if we permute column 6 so that methionine is replaced by threonine and vice versa, the permutation retains the aggregate conservation and coupling statistics but the residue combination of serine at position 3 and threonine at 6 has not been observed in the extant sequences and might be violating some underlying constraint. (This might explain why some of the designed sequences from [19] did not adopt the native fold.) Thus it is desirable to ensure that the designed sequences directly satisfy the underlying sequence constraints, rather than indirectly display the same amount of conservation and coupling.

Another possible issue involves "double counting" the statistics. For example, in Fig. 1, positions 1, 7, and 8 are each pairwise coupled. We can design residues at these positions to satisfy all three pairwise constraints but it might be the case that by satisfying just 1–8 and 7–8, we will also (mostly) satisfy 1–7. Thus it might not be desirable to directly reproduce all the observed pairwise statistics. By recognizing that the 1–7 coupling is mediated through residue 8, we can better explore constrained sequence space, since the indirect 1–7 constraint isn't given undue influence by contributing both directly and indirectly.

Contributions: This paper develops algorithms for protein design using probabilistic graphical models to properly represent and exploit residue constraints. We formulate protein design as a two-step process: learning an undirected graphical

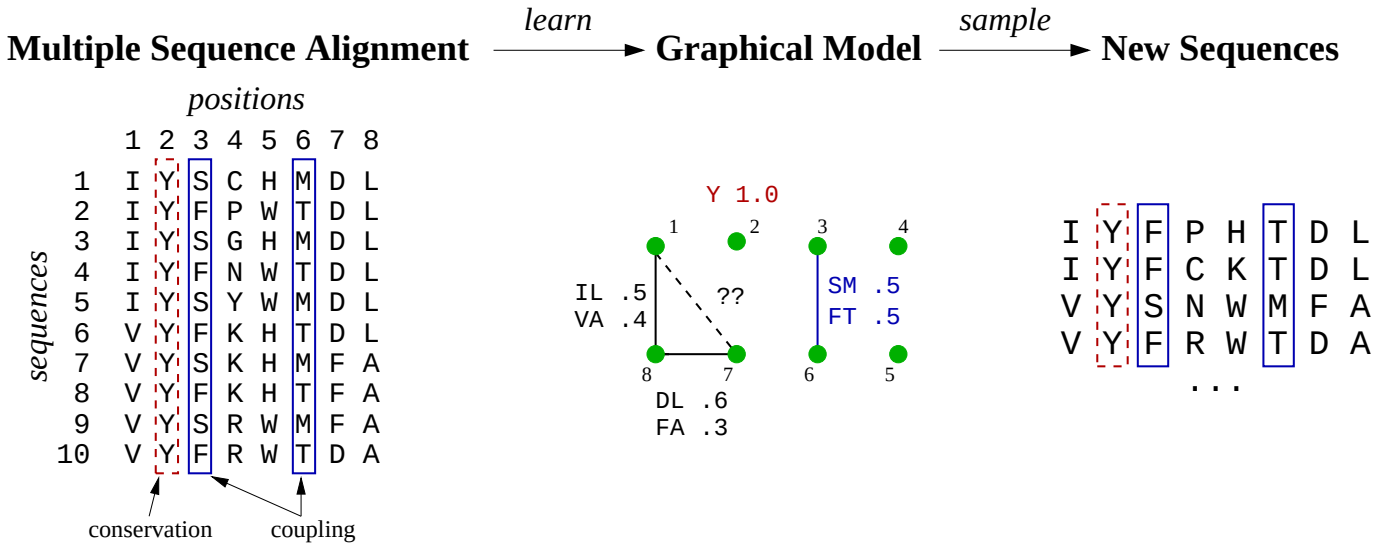

Fig. 1. Given an MSA (left), conservation and coupling constraints are inferred and summarized into a graphical model (middle) which captures conditional independence relationships through its edges. Residues 3 and 6 are coupled—when residue 3 is small and polar (S), residue 6 is large and hydrophobic (M). Likewise, when residue 3 is large and hydrophobic (F), residue 6 is small and polar (T). Through its clique potentials (not shown here), the model captures probability distributions for subsets of residues. Sampling from the model (right) then yields new sequences that obey the underlying constraints.

model that captures the essential constraints underlying a protein family, and using the model *generatively* to produce new sequences. The first step, covered in our prior publication in *TCBB* [20], provides a probabilistic basis for evaluating the quality of new sequences with respect to satisfaction of protein family constraints. While predicting protein folding and activity from sequence alone remains beyond the state of the art, by satisfying the learned constraints, we seek to improve the "hit rate" of new sequences that are folded and functional (as Ranganathan demonstrated). With this in mind, protein design entails sampling high-likelihood sequences from such a graphical model. While such sampling is difficult in general (due to the presence of cycles and their possible non-chordality), we develop two new sampling techniques targeted to protein design, and show that they are good both in theory and in practice. We prove theoretical convergence of the algorithms, and, in application to design of new WW domains, we demonstrate that they converge quickly while generating new sequences that meet the modeled constraints and are reflective of (but different from) the original sequences.

II. Methods

A. Graphical Models of Residue Coupling

Our sampling methods take as input an undirected graphical model, also known as a Markov random field or a log-linear model, encoding the residue constraints that should be satisfied in the designed sequences. We focus here on constraints (conservation and coupling) inferred from the sequence record, and we call the resulting graphical model a *graphical model of residue coupling (GMRC)* [20]. It remains interesting future work to integrate such a model with undirected graphical models that capture energetics [4], as well as experimentally-identified constraints (e.g., from mutagenesis and double-mutant cycles). In fact, the proofs below for theoretical convergence of our sampling methods do not depend on how the graphical model was constructed.

Undirected graphical models have the form $G = (\mathbf{V}, E)$, where vertices in $\mathbf{V}$ are random variables, one for each residue position, and the edges in E encode independence relationships between the random variables—a vertex is conditionally independent of all other vertices, given its immediate neighbors. (Throughout, random variables are represented using uppercase letters and their values using lowercase letters; sets of variables appear in bold.) G defines a probability distribution function (pdf) $p(\mathbf{V} = \mathbf{r})$ on residue types $\mathbf{r}$ for $\mathbf{V}$, computed by combining scores ("potentials") for the cliques in the graph. We use $p(\mathbf{r})$ as shorthand for $p(\mathbf{V} = \mathbf{r})$.

$$p(\mathbf{r}) = \frac{1}{Z} \prod_{\mathbf{c} \in C(G)} \phi_{\mathbf{c}}(\mathbf{r}_{\mathbf{c}}) \tag{1}$$

Here, $C(G)$ is the set of all cliques (typically maximal cliques) in G, Z is a suitable normalizing factor to ensure that p is a pdf, and the ϕ_c are the potential functions. The potential functions are not necessarily probabilities and are best thought of as compatibility functions over the combinatorial space of residue values. For the special case that G is a chordal graph (also known as triangulated, recursively simplicial, or decomposable), the structure of the potential functions satisfies:

$$\prod_{\mathbf{c} \in C(G)} \phi_{\mathbf{c}}(\mathbf{r}_{\mathbf{c}}) = \frac{\prod_{\mathbf{c}} p_{\mathbf{c}}(\mathbf{r}_{\mathbf{c}})}{\prod_{\mathbf{a} \in \text{cliqueadj}(G)} p_{\mathbf{a}}(\mathbf{r}_{\mathbf{a}})} \tag{2}$$

In this case, the potentials are given by the product of marginals defined over the cliques divided by the product of marginals defined over the clique adjacencies $\mathbf{a}$, which could be nodes, edges, or general subgraphs [7]. Thus each potential is either a conditional or a joint marginal distribution.

Given a multiple sequence alignment (MSA) $\mathcal{S}$ for a protein family, we use the following estimator for $p_{\mathbf{c}}(\mathbf{r}_{\mathbf{c}})$, the

probability of a set of amino acid types $\mathbf{r_c}$ for a clique:

$$p_{\mathbf{c}}(\mathbf{r_c}) = \frac{f_{\mathbf{c}}(\mathbf{r_c}) + \frac{\rho|\mathcal{S}|}{21^{|\mathbf{c}|}}}{|\mathcal{S}|(1+\rho)} \quad (3)$$

Here $f_{\mathbf{c}}(\mathbf{r_c})$ is the frequency, in $\mathcal{S}$, of the set of amino acid types, $|\mathcal{S}|$ is the total number of sequences, $|\mathbf{c}|$ is the cardinality of the clique, 21 arises from the 20 amino acid symbols plus a 'gap' symbol, and ρ is a parameter that weights the importance of missing data. Notice that when $\rho = 0$, the estimator is completely determined by the data in the MSA. Larger values of ρ accommodate unseen data, enable factorization according to the Hammersley-Clifford theorem [7], and support greater exploration of constrained sequence space when sampling.

Techniques to learn GMRCs are covered elsewhere [20]. Briefly, these techniques employ conditional mutual information to assess coupling and a greedy algorithm to factor statistically-significant couplings into a graphical model.

B. Sampling a Graphical Model of Residue Coupling

Sampling from a directed graphical model, i.e., a Bayesian network, is straightforward; a topological sort of the vertices suggests the order in which values for the random variables must be generated [1]. Similarly, for undirected graphical models with chordal graphs (graphs where the maximal cliques can be arranged to form a junction-tree), efficient sampling procedures exist [16]. Sampling from an arbitrary undirected graphical model is more difficult, however, due to the presence of cycles and non-chordality (and thus no tree ordering).

Here, we present two MCMC (Markov Chain Monte Carlo) methods for sampling GMRCs. To show that the sampling methods systematically explore the constrained sequence space, we show that the underlying Markov chains have stationary distributions, i.e., they are irreducible, aperiodic, and positive recurrent [3]. In the Markov chain for the first method, the states correspond to MSAs, whereas in the second they correspond to individual sequences. A Markov chain is irreducible if it is possible to get from any *valid* state (a state with nonzero likelihood) to any other valid state. A chain is aperiodic if it does not systematically cycle; to prove aperiodicity, it is sufficient to show that every state can self loop (since any periodic cycle can be broken by adding such a self loop). A chain is positive recurrent if each valid state has a finite expected return time.

More formally, let X_k denote the state of the chain at time k. Irreducibility requires that the probability of being in state j at time n given that we are in state i at time 0 is positive for some n:

$$p(X_n = j | X_0 = i) > 0 \quad (4)$$

The period k of the chain is given by:

$$k = \gcd\{n : p(X_n = i | X_0 = i) > 0\} \quad (5)$$

where gcd is the greatest common divisor. k is 1 when the chain is aperiodic. Finally, the chain is positive recurrent if all states have an expected finite expected return time. Let T_i be a random variable denoting the return time to state i:

$$T_i = \min\{n : X_n = i | X_0 = i\} \quad (6)$$

Algorithm 1 ConstrainedShuffling(G, $\mathcal{S}$)

Input: undirected graphical model G defined over MSA $\mathcal{S}$ (involving m sequences of n residues)
Output: MSA $\mathcal{S}$' of sampled sequences

```
S' ← column-wise permutation of S
v ← average_{s∈S'} p(s)
while not converged do
    c ← random column ∈ [1, n]
    s, t ← random rows ∈ [1, m] s.t. s ≠ t
    swap S'[s, c] and S'[t, c]
    v' ← average_{s∈S'} p(s)
    if v' < v then
        undo the swap with probability 1 − e^{v'−v}
    end if
end while
```

To be positive recurrent we need $E[T_i]$ to be finite.

1) Constrained Shuffling: Our first algorithm generates a small set of high-likelihood new sequences. It is a variant of the procedure used by Ranganathan and colleagues [19] but goes beyond simply recreating pairwise statistics and instead seeks to generate a set of high-likelihood sequences, each satisfying the constraints. Algorithm 1 gives the pseudocode. It begins by independently shuffling the columns of an MSA. A 'move' then entails swapping amino acids in two random sequences at a single random column. Notice that a move preserves the same amino acid composition as the original MSA. A move is accepted if the average log likelihood of the new sequences is improved. Otherwise, it is accepted with probability proportional to the change in score. The procedure continues until convergence; e.g., a user-specified number of iterations is reached or the distribution of new sequences is sufficiently good.

We now analyze the Markov chain induced by constrained shuffling, in order to show that in some sense it "properly" samples the distribution. A state in the chain corresponds to an entire MSA. Intuitively, the Markov chain is irreducible since we can get from any permutation of the MSA to any other with the correct sequence of pairwise swaps, each of which has a nonzero probability of being accepted. This nonzero probability is also sufficient to ensure a finite expected return time for each state, so the Markov chain is positive recurrent. The Markov chain is aperiodic since each state can self loop either by rejecting a move or by swapping two of the same amino acids. Let us now more formally characterize these properties.

Theorem 1: Algorithm 1 (ConstrainedShuffling) induces a stationary distribution.

Proof: Each state in constrained shuffling corresponds to a shuffled MSA. Let m be the number of rows and n be the number of columns in the MSA. Also, let d be the largest difference between the average log likelihood of two states separated by one move. This term is important because a move in constrained shuffling that reduces the average log likelihood scores will still be accepted with probability dependent on the change in average log likelihood score. Using d, the probability of accepting a bad move is lower bounded by

e^{-d}. To show that each state is irreducible, we show that the probability of moving from any state i to state j in time mn is greater than 0 (see Eq. 4). One way to move from state i to state j in time mn is to permute each column successively until it is equal to columns in state j. If we do this, then the probability of being in state j at time mn after starting at state i at time 0 is

$$\begin{aligned} p(X_{mn} = j | X_0 = i) &\geq \prod_{c=1}^{n} \prod_{r=1}^{m} \frac{1}{n} \frac{1}{m^2} \cdot e^{-d} \\ &= \frac{e^{-dmn}}{n^{mn} m^{2mn}} > 0 \end{aligned} \tag{7}$$

so every state is irreducible. To show every state j is aperiodic, we can simply show that every state can self-loop (see Eq. 5). For ease of exposition, we assume that the there is at least one column in the input MSA that has two of the same amino acids—almost certainly true of any protein family. However, even if this is not the case, each state is still aperiodic (the proof involves showing that every state can return to itself in both 2 and 3 steps). If there is a column with two of the same amino acids, then one way for a state j to self-loop is to select that column and then swap two of the same amino acids. The probability of this happening is

$$p(X_1 = j | X_0 = j) \geq \frac{1}{n} \cdot \frac{1}{m^2} > 0 \tag{8}$$

so every state is aperiodic. Finally, we show that every state j is positive recurrent by showing that its expected return time is finite (see Eq. 6). The expected return time, $E[T_j]$ for state j is

$$\begin{aligned} E[T_j] &= \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \cdot p(X_k = j) \\ &\leq \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \end{aligned}$$

Recall that no matter which state constrained shuffling is in at time k, there is a nonzero probability that it will reach state j at time mn (Eq. 7). Therefore,

$$\begin{aligned} E[T_j] &\leq \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \\ &\leq \sum_{k=1}^{mn} k + \sum_{k=mn+1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \\ &\leq \frac{mn(mn+1)}{2} + \sum_{k=mn+1}^{\infty} k \cdot \left[1 - \frac{e^{-dmn}}{n^{mn} m^{2mn}}\right]^{k-mn} \end{aligned}$$

Since both m and n are finite, the first part of the sum is finite. To show that the infinite summation is finite, we employ the ratio test

$$\begin{aligned} &\lim_{k \to \infty} \frac{(k+1) \cdot \left[1 - \frac{e^{-dmn}}{n^{mn} m^{2mn}}\right]^{k+1-mn}}{k \cdot \left[1 - \frac{e^{-dmn}}{n^{mn} m^{2mn}}\right]^{k-mn}} \\ &= \left[1 - \frac{e^{-dmn}}{n^{mn} m^{2mn}}\right] \cdot \lim_{k \to \infty} \frac{k+1}{k} < 1 \end{aligned} \tag{9}$$

so the sum is finite and thus all states are positive recurrent. ■

2) Component Sampling: Constrained shuffling designs a fixed number of sequences from a given, also fixed, pool of amino acids. For protein design, we might desire to generate all the sequences that meet the constraints encoded by the model, using possibly new amino acid combinations not encountered in the given pool. To achieve this goal, we present our second algorithm: component sampling. At an abstract level, component sampling is a Gibbs sampler [3] with moves defined over cliques instead of vertices. At each step in the algorithm, a move is generated from the current state (a sequence) to a new state. The central question is how to generate a move. A move that simply changes the value for a single vertex can get stuck in a local minimum or, worse, generate sequences that don't conform to the model. To avoid this problem, we must make moves at the level of cliques. Unfortunately, changing the value of a single clique can cause the clique values to be invalid for neighboring cliques (cliques that contain one of the changed vertices). The neighboring cliques must thus be given new values, conditioned on the original clique value. These changes can cause invalid values further downstream, so those cliques must also be sampled, conditioned on all the values so far. This process of fixing downstream clique values continues until the entire connected component from the original clique has been sampled. We named this approach *component sampling* to reflect this propagation process.

This procedure is similar to the algorithms used to sample undirected graphical models with chordal graphs [7]. However, with chordal graphs, the order of the components is determined since the clique relationships form a tree. To sample these distributions, simply sample the cliques in the tree order. For arbitrary graphs, however, the cliques can form cycles and thus selections made early in a component may later be found to be incompatible with ones chosen later. It is possible to make our GMRCs chordal by "triangulating", i.e., adding edges to non-chordal cycles in order to make them chordal. The traditional reason for such triangulation is to improve efficiency of inference by helping determine a suitable elimination ordering of nodes. However, this approach introduces artificial constraints into the model, which is bad both for learning the model and for sampling from it. In particular, by seeking to satisfy the extra constraints, the new sequences would be able to sample only a subset of the sequence space that is valid under the original model.

To avoid the problems discussed above, we randomly determine the sampling order for each component. This is equivalent to choosing a random tree for each sample. The resulting sample can still be unsatisfiable, but since the tree is determined randomly, systematic unsatisfiable situations don't arise and therefore the sampled distribution isn't expected to deviate significantly from the true underlying distribution.

Algorithm 2 provides the details of our component sampling algorithm (observe that it takes as input any graphical model of residue constraints, not just our GMRC). A single move consists of sampling clique values in a connected component, propagating to neighboring cliques breadth-first (in random order from each clique). The process continues until convergence, e.g., enough sequences are generated, or their distri-

Algorithm 2 ComponentSampling(G)

Input: undirected graphical model G
Output: set of sequences sampled from G

$S \leftarrow \emptyset$
while not converged **do**
  $c \leftarrow$ random clique from G, with equal probability
  $S_c \leftarrow$ random AA types r_c, with probability $p(r_c)$
  $Q \leftarrow$ queue initialized as random permutation of cliques neighboring c in G
  while Q is not empty **do**
    $D \leftarrow$ dequeue from Q
    $A \leftarrow$ vertices already assigned in D
    if $A \neq D$ **then**
      $S_{D-A} \leftarrow$ random AA types r_{D-A} with probability $p_{D-A}(r_{D-A}|r_A)$
      enqueue onto Q a random permutation of cliques neighboring D in G
    end if
  end while
  output S
end while

bution is sufficiently good. Notice that the values for each component are selected according to their likelihood under the model. Since the components are conditionally independent of each other, the likelihood of generating a whole sequence is the product of the likelihood of each of its components.

In order to focus sampling on only the most representative sequences, we assign zero probability to unobserved clique values (using $\rho = 0$ in Eq. 3). Note that this does limit the amino acid choices we have during design. This limitation restricts the algorithm to generate only sequences with amino acids we have seen in the MSA before. Unlike constrained shuffling, this restriction does not leave us with a fixed pool of amino acids to consider—choosing an amino acid for one sequence does not preclude it being used again in another. However, using zero probabilities for some clique values does mean that the sampling procedure can get stuck, with no value remaining for a clique that is consistent with the values chosen so far. In this case, the move is rejected. To avoid being systematically stuck, the order in which the cliques are visited is randomized at each move. It still is possible to get stuck, but the dead ends are not systematic and therefore are unlikely to cause large deviations from the true distribution even when the graph is not chordal.

A state in the Component Sampling Markov chain is an individual sequence. Intuitively, the Markov chain is irreducible since we can get from any sequence to any valid one by individually replacing each component, and there are nonzero probabilities of choosing the components in the correct order and selecting the correct values for each component (the new sequence is valid so it has a positive likelihood). This nonzero probability also guarantees that every state will have a finite expected return time, so the Markov chain is positive recurrent. The Markov chain is aperiodic since a state can self loop by "replacing" a component with its original values.

Theorem 2: Algorithm 2 (ComponentSampling) induces a stationary distribution.

Proof: Each state in component sampling corresponds to a single sequence with positive likelihood. That is, a state j represents a sequence s_j where $p(s_j) > 0$. Let C be the set of all components in the graphical model G, t be the total number of components, and C_l be the lth component. The number of cliques in a component (or set of components) is denoted $|\cdot|$. Thus, $|C|$ corresponds to the number of cliques in G. To show that each state is irreducible we show that the probability of moving from any state i to state j in some time n is greater than 0 (see Eq. 4). One way to move from state i to state j is by successively selecting each component C_l for replacement and then choosing the values of $s_j(C_l)$ (amino acid values corresponding to those in sequence s_j) to replace the current component. If we do this for each component, then the probability of being in state j at time t after starting at state i at time 0 is

$$\begin{aligned} p(X_t = j|X_0 = i) &= \prod_{l=1}^{t} \frac{|C_l|}{|C|} \cdot p_{C_l}(s_j(C_l)) \\ &= \prod_{l=1}^{t} \frac{|C_l|}{|C|} \cdot \prod_{l=1}^{t} p_{C_l}(s_j(C_l)) \\ &= p(s_j) \prod_{l=1}^{t} \frac{|C_l|}{|C|} > 0 \end{aligned} \tag{10}$$

so every state is irreducible. To show that every state j is aperiodic we can simply show that every state can self-loop (see Eq. 5). One way for a state j to self-loop is to select the first component for replacement and choose $s_j(C_1)$ as the value to be replaced. If we do this, then the probability of being is state j at time 1 given that we were in state j at time 0 is

$$p(X_1 = j|X_0 = j) \geq \frac{|C_1|}{|C|} \cdot p_{C_1}(s_j(C_1)) > 0 \tag{11}$$

so every state is aperiodic. Finally, we show that every state j is positive recurrent by showing that its expected return time is finite (see Eq. 6). The expected return time, $E[T_j]$ for state j is

$$\begin{aligned} E[T_j] &= \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \cdot p(X_k = j) \\ &\leq \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \end{aligned}$$

Recall that no matter which state component sampling is in at time k, there is a nonzero probability that it will reach state j at time $k + t$ (Eq. 10). Therefore,

$$\begin{aligned} E[T_j] &\leq \sum_{k=1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \\ &\leq \sum_{k=1}^{t} k + \sum_{k=t+1}^{\infty} k \cdot p(X_1, \ldots, X_{k-1} \neq j) \\ &\leq \frac{t(t+1)}{2} + \sum_{k=t+1}^{\infty} k \cdot \left[1 - p(s_j) \prod_{l=1}^{t} \frac{|C_l|}{|C|}\right]^{k-t} \end{aligned}$$

Clearly, the first part is finite since t is finite. We show that the infinite summation is finite by employing the ratio test

$$\lim_{k\to\infty} \frac{(k+1)\cdot\left[1-p(s_j)\prod_{l=1}^{t}\frac{|C_l|}{|C|}\right]^{k+1-t}}{k\cdot\left[1-p(s_j)\prod_{l=1}^{t}\frac{|C_l|}{|C|}\right]^{k-t}} = \left[1-p(s_j)\prod_{l=1}^{t}\frac{|C_l|}{|C|}\right]\cdot\lim_{k\to\infty}\frac{k+1}{k} < 1 \qquad (12)$$

so the sum is finite and thus all states are positive recurrent. ■

III. Results

To demonstrate the power of our sampling methods we consider a family of WW domains—small proteins that assist in protein-protein interactions by binding to proline-containing targets. These proteins were the object of previous protein design studies by Ranganathan and colleagues [17], [19], who provided the following:

1) An input dataset of 42 wild-type WW domains multiply aligned to 39 residues;
2) A set *IC* of 43 new sequences designed by treating each residue position independently, sampling from the amino acid type distribution observed in the input WW dataset; and
3) A set *CC* of 43 new sequences designed by accounting for coupling in residue positions, by stochastically optimizing the sequences to match the coupling statistics from the input WW dataset.

Of the 86 new sequences, only 12—all from *CC*—were found to adopt the native fold [19].

Although our sampling methods will converge for any undirected graphical model of residue constraints, here we use models learned with our GMRC approach [20]. We first show that the learned graphical model captures significant constraints in the wild-type WW domains sequences and is effective in distinguishing folded from unfolded sequences. We then show that our sampling methods efficiently generate a wide range of high-likelihood putative WW domain sequences which may serve as hypotheses for further biological study.

A. Learning a Graphical Model

Using our learning algorithm [20] on the 42 wild-type WW domains generates a GMRC containing 35 statistically significant edges. The model consists of 33 cliques: seven 1-cliques (independent residues), twenty-one 2-cliques, and five 3-cliques. Fig. 2 illustrates some of the cliques, such as the 3-clique 1–11–13, and its neighbor 2-cliques 1–5 and 13–20. The model encodes many transitive coupling relationships, including a 23–37 coupling mediated by 35. Some residues, such as 14, remain independent of all other residues. Note that the model is not chordal since the cycle 1–5–20–13 has no chords. The model also contains four other non-chordal cycles (not shown).

A key advantage of graphical models is that they provide a mechanism for evaluating new sequences, by likelihood

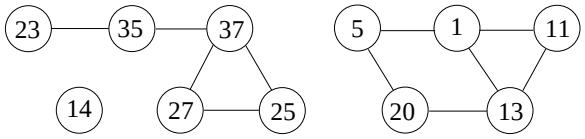

Fig. 2. Parts of the graphical model of residue coupling learned from a dataset of WW domains with 11 vertices and 11 edges. The complete model is provided in Tab. I in the Appendix, and contains 28 more vertices and 24 more edges.

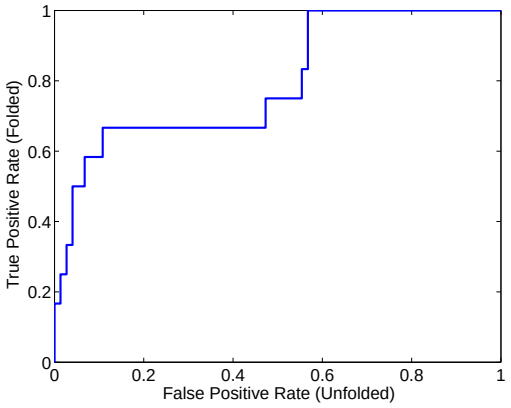

Fig. 3. ROC curve for classification of *IC* and *CC* sequences according to log likelihood under our graphical model. The false positive rate is the percentage of sequences predicted to be of native fold that are not; the true positive rate is the percentage of sequences predicted to be of native fold that are.

(Eq. 1). Since our model is not chordal, however, the likelihood equation in Eq. 1 does not strictly hold. We still use Eq. 1 in the following as an approximation to the true likelihood, to relatively assess sequences against each other and, in this manner, broadly explore the "good" regions of constrained sequence space.

To show that our likelihood approximation is useful, we examined, over the *IC* and *CC* sequences, the enrichment of folded sequences among the high-likelihood ones. It is important to note that predicting foldedness is not the goal of this paper; in fact, to do so perfectly would require solving the protein folding problem. However, since high-likelihood sequences should better satisfy the inferred constraints, we expect an improved "hit rate" of folded and functional ones among them. To formally quantify this result, we used the log likelihood of each sequence as a classifier, and generated an ROC curve (Fig. 3) by varying the threshold to separate folded from not. The power of this classifier (area under the ROC curve) is .80 (recall that power of 1 indicates perfect discrimination while power of .5 corresponds to random guessing). The fact that we can do this at all is especially impressive for this dataset since the sequences we are testing were designed to purposely "look like" the wild-type WW domains and therefore have many of the constraints on which our model is based.

These results suggest that our likelihood estimate is indeed a reasonable objective function. We now study how well the sampling methods are able to generate new, high-likelihood sequences.

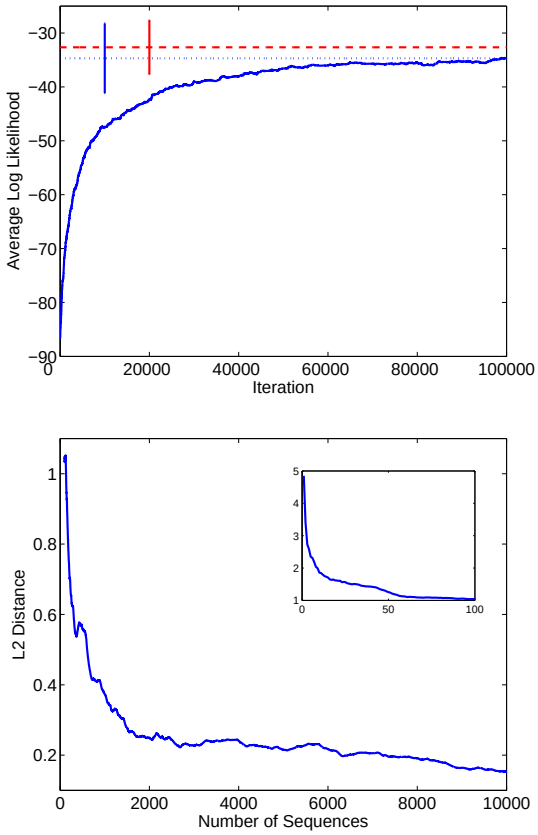

Fig. 4. Progress of our sampling algorithms. (Top) Constrained shuffling. Convergence of the average likelihood score of sequences generated by constrained shuffling. Blue dotted line: final value (-34.69 with standard deviation of 6.46); red dashed line: wild-type sequences (-32.65 with standard deviation of 4.93). (Bottom) Component sampling. Evolution of the total $L2$ distance (over all cliques) between the graphical model and the sequences generated by component sampling.

B. Sampling a Graphical Model

1) Convergence: We applied our sampling methods to sample new high-likelihood putative WW domain sequences from our model. For the constrained shuffling method (Algorithm 1), we generated a set of 42 new high-likelihood sequences with amino acid types shuffled column-wise from the input dataset. We ran our algorithm for 100,000 iterations. Each iteration corresponds to a set of 42 putative WW domain sequences. As our goal is to generate a set of high-scoring sequences, we evaluated the convergence of the algorithm in terms of the average log likelihood of the generated sequences at each iteration. Fig. 4(top) plots the convergence. Although the average log likelihood starts at only -86.52, after 100,000 iterations, it reaches -34.69 with a standard deviation of 6.46, shown by the dotted line. This value is very close to the average log likelihood of the wild-type sequences, -32.65 (with standard deviation 4.93), shown by the dashed line. After 100,000 iterations, constrained shuffling is able to generate a new set of WW sequences with log likelihoods similar to those of the wild-type WW domains.

For the component sampling method (Algorithm 2), after a burn-in series of 88 moves (the number of moves it took to sample each residue at least once) we generated a set of 10,000 new putative WW sequences. The algorithm got "stuck" (unable to make a move) 289 times; in these cases, the move was rejected and sampling continued by selecting another move. If sampling worked properly, we would expect the distributions of amino acid types for sets of residues in the generated sequences to match the distributions in the model. In contrast to component sampling, our goal here is to sample from the underlying distribution, and thus we measure convergence in terms of the $L2$-distances between clique distributions ($p_{\mathbf{c}}(\mathbf{r_c})$, Eq. 3) in the original sequences vs. the generated sequences. Fig. 5 empirically illustrates that three different residue sets from Fig. 2 have very similar distributions to the model distributions. For the 2-clique 35–37, the $L2$-distance is $.0125$, while for the 3-clique 1–11–13, it is $.0235$. Even for the transitive relationship between residues 23 and 37, component sampling still generates the correct distribution; the $L2$-distance is only $.0371$. To measure how quickly the sampling method converges to the model distribution, we monitored at each iteration the sum of the $L2$ distances for all cliques. Fig. 4(bottom) shows that the total $L2$ distance converges very quickly. After only 100 samples, the total $L2$-distance is 1.04; it improves to $.37$ after 1000 samples and $.15$ (approximately $.004$ per clique) after 10,000 samples.

In both cases, our Matlab implementation of the sampling algorithms required only minutes to converge.

2) Designed sequence likelihoods: For constrained shuffling, we found that the average log likelihood score converges to an average very close to those of the wild-type sequences. To further inspect the distribution of likelihood scores generated using constrained shuffling, we selected the MSA with the highest average log likelihood score (in this case the 100,000th MSA generated). These sequences provide a range of log likelihoods under the model, as shown in Fig. 6(a). These designed sequences have excellent scores: a mean of -34.69 (with standard deviation of 6.46), best of -23.68 and worst of -48.18. Similarly, component sampling generates high-likelihood sequences, as shown in Fig. 6(c). The designed sequences have excellent scores: a mean of -33.48 with a standard deviation of 5.02.

Even the least likely sequence generated by our methods (log likelihood -52.19, generated by component sampling) is more likely than all but one of the 86 *IC* and *CC* sequences (mean log likelihood -84.14 and standard deviation 15.12). These designed sequences with high likelihoods are predicted to be more likely to fold than their lower-scoring counterparts.

3) Designed sequence novelty: To ensure that our sampling methods aren't simply recreating the wild-type WW domains, we measured the sequence identity between the designed sequences and their closest natural neighbors. The sequence identity between two sequences is calculated as the percent of residue positions where they have the same amino acid. Thus, a sequence is 100% identical to itself. Fig. 6(b) and (d) show histograms of the designed sequence identities for constrained shuffling and component sampling, respectively. The designed sequences are quite different from the wild-type WW domains. The mean sequence identity for sequences generated using

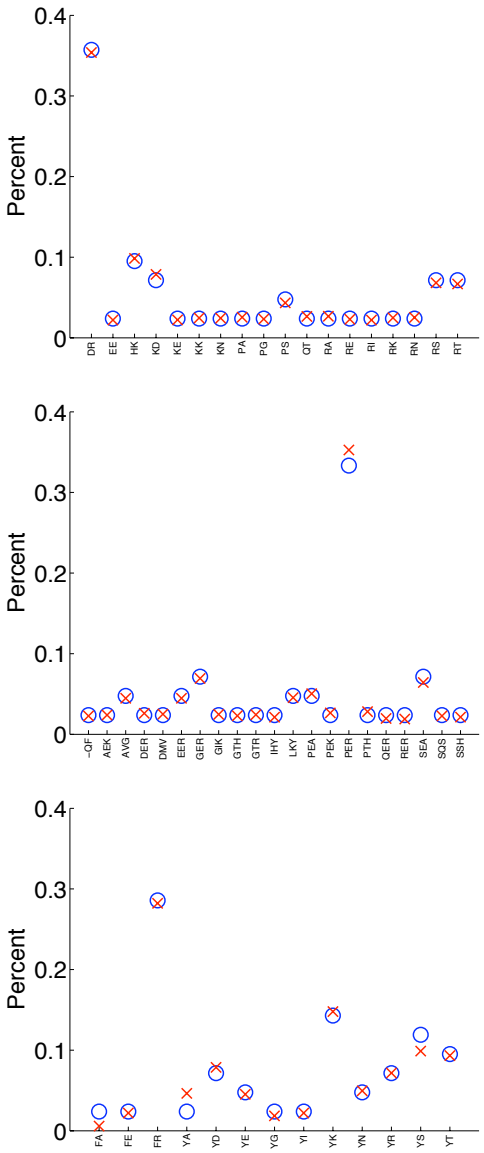

Fig. 5. Probability distribution over sets of amino acid types according to the graphical model (blue 'o's) and the sequences generated by component sampling (red '×'s) for the 2-clique 35–37 (top), the 3-clique 1–11–13 (middle), and the transitive relationship between residues 23 and 37 (bottom).

constrained shuffling to their nearest wild-type neighbor is 60.56% with a standard deviation of 7.70%. For component sampling, the identity is increased slightly with an average identity of 65.14% with standard deviation 8.06%. This degree of identity is similar to that of the *IC* and *CC* sequences, which have a mean of 60.41% and a standard deviation of 6.21%.

The sequence generated by our methods with the highest sequence identity to a wild-type WW domain has a sequence identity of 89.74% (generated by component sampling). If our methods had completely explored the space, we would have generated at least one of the wild-type sequences, indicating that there is still more sequence space unexplored. This is not surprising however, since the space of protein sequences (even those meeting all the constraints) is very large. To see this, we note that the most unique sequence (also generated by component sampling) has an identity of only 41.03% to its nearest wild-type WW sequence; this indicates that even sequences that meet the constraints can be very 'far' from those previously seen.

In addition to measuring the identity of designed sequences to their *nearest* wild-type sequences, we also measure their *average identity* to the wild-type sequences. For the sequences designed by constrained shuffling, the mean average identity to the wild-type sequences is 44.95% with a standard deviation of 4.92%. For the sequences designed by component sampling, the mean average identity to the wild-type sequences is 45.18% with a standard deviation of 5.29%. These levels of identity are again similar to those that we see in the *IC* and *CC* sequences, which have a mean of 43.75% and standard deviation of 4.09%.

Finally, by measuring the identity between designed sequences, we can see how diverse they are and how much of the constrained sequence space we have covered. For sequences designed by constrained shuffling, the average pairwise identity is 43.60% with a standard deviation of 9.69%. For sequences designed by component sampling, the average pairwise identity is slightly higher, 45.47% with a standard deviation of 11.17%. These measures are again similar to the level we see in the *IC* and *CC* sequences which have an average pairwise identity of 43.05% with a standard deviation of 8.10%. Thus we maintain the same level of novelty as previously designed sequences while increasing their likelihood under the model.

C. Closing the Loop

If the generated sequences were in fact representative of the wild-type sequences, we would expect them to generate a similar model. We call this "closing the loop"—learning a model of natural sequences and using it to generate new sequences, and then learning a model of the new sequences and using it to evaluate the original sequences. We chose to test our ability to close the loop with the sequences generated by constrained shuffling rather than by component sampling, as constrained shuffling has the same degrees of freedom (choices for amino acids) as the original MSA. Thus we generate exactly as many amino acids as we had before, but in different combinations, and we can evaluate how well the modeling and sampling methods have represented and satisfied the underlying sequence constraints.

We closed the loop by learning a model of the 42 sequences from the constrained shuffling MSA with the lowest average log likelihood score. The new model consists of 30 edges, of which 25 are the same as those in the original model. The 5 different edges in the new model were actually encoded as transitive relationships in the original model. The new model is also able to discriminate folded from unfolded *IC* and *CC* sequences, as shown in Fig. 7 (compare to Fig. 3). The power of the new classifier (area under the ROC curve) is .80, the same power as the original model. Thus the designed sequences capture the original sequence constraints sufficiently well that the graphical model learned from them is equally predictive of foldedness as the original model. While not

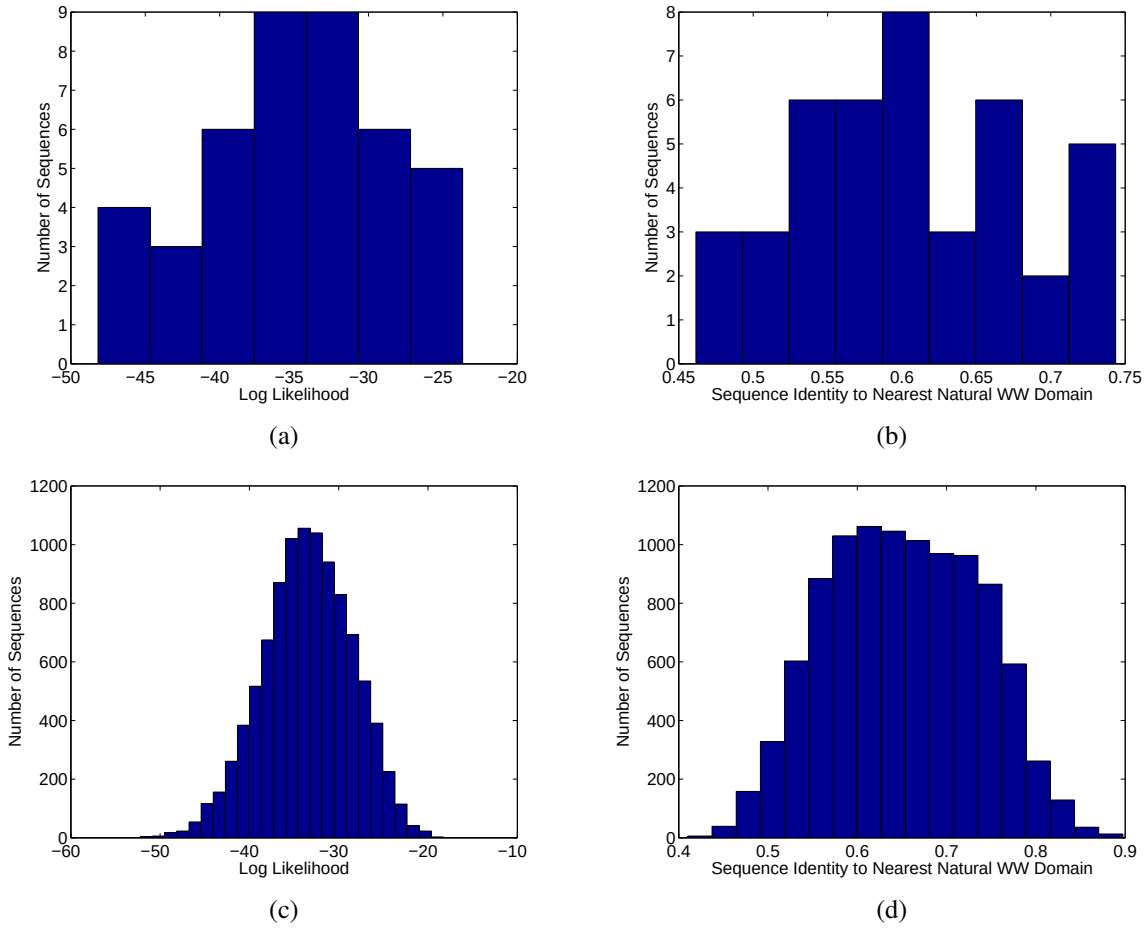

Fig. 6. The log likelihood distribution (a, c), and sequence identity to the nearest wild-type WW domains (b, d), for the 42 and 10,000 sequences generated by constrained shuffling and component sampling, respectively.

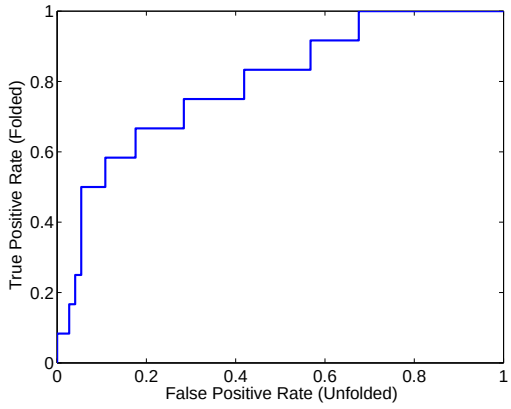

Fig. 7. The ROC curve of classification of *IC* and *CC* sequences according to log likelihood under the graphical model learned from the sequences designed by constrained shuffling.

providing any additional evidence for the validity of the model beyond the initial tests of Fig. 3, this closing-the-loop test does demonstrate that the sampling algorithm itself is correct in the sense that it is consistent with the learning algorithm.

IV. Discussion

We have formulated protein design in terms of first learning constraints on sequences in a protein family to capture "what it means" to be a member of that family, and then sampling sequence space as constrained by what we have learned about the family. The sampling methods presented converge regardless of the graphical model used—even when the graph is non-chordal. Here, we use one particular graphical model which we learn from the sequence record of a protein family. This graphical model factorizes conservation and coupling and we demonstrated that the estimated likelihood of a sequence under the learned model is predictive of foldedness even among sequences previously designed to "look like" wild-type sequences.

Our two sampling methods generate sequences that are of high likelihood under a model, but yet new and different. We showed the power of both approaches by learning and sampling graphical models for a family of WW domains. The sampling methods are complementary—component sampling explores broadly the high-likelihood portion of sequence space and can generate a large number of sequences, while constrained shuffling generates a fixed number of high-likelihood sequences that adhere to a provided diversity constraint. Another way to characterize these methods is that constrained

shuffling samples the MSA under constraint by the graphical model (via likelihoods), while component sampling directly samples the graphical model. Given a large enough MSA and a consistent learning algorithm, these should yield identical results. In the finite limit (here, only 42 sequences), they provide distinctly different results.

The sampling procedures depend on the model provided since the generated sequences will fit the constraints encoded in the model. For our model, the constraints are learned from a multiple sequence alignment and thus are dependent on the quality of the alignment. The WW alignment we use was manually aligned by experts. Unrepresentative alignments or misalignments may cause incorrect constraints to be found or crucial constraints to be omitted.

We reiterate that our work on protein design is neither meant to address, nor requires solving, the protein folding problem. Other works have used coupled residues to predict inter-residue contacts [13] or interfaces of protein-protein interactions [15]. Our results show that likelihood under our model is predictive of whether or not a sequence will be folded. Still, the sequences generated here will have to be subjected to further considerations of thermodynamics, stability, and kinetics. In fact, one direction for future work is to augment our models of evolutionary constraints with energetic constraints from structure-based models [4]. By combining potential terms our methods would be able to sample novel sequences that meet the evolutionary constraints and have favorable predicted free energies. Likewise, we can incorporate functional subclass information [20] in order to target the designed sequences for particular activities.

Instead of sampling from the model, one could instead seek to find the best designs, e.g., by using loopy belief propagation with max-marginals [21]. We decided here to seek a bigger picture of the constrained sequence space by sampling broadly from the high-likelihood part of the distribution. This is particularly appropriate for protein design, when the "best" design might not actually be active (due in part to the incompleteness of the model), and we must consider a diverse set of possibilities. While we didn't incorporate diversity directly into the sampling approach (as was done by Zheng et al. for recombination design [23], [24]), our results show that we do indeed obtain a diverse set of high-likelihood sequences.

Currently, we assume that a designed sequence will be constructed *ab initio*. One direction for future work is to develop sampling algorithms that work under experimental constraints such as site-directed recombination [14], [18], [22]. These experimental constraints restrict the degrees of freedom available in order to simplify the experimental process or to make it applicable on a larger scale.

Our software can be freely obtained for academic use by request from the corresponding authors.

ACKNOWLEDGMENTS

This work is supported in part by US NSF grants IIS-0444544 (CBK) and ITR-0428344 (NR). This work was inspired by conversations with Drs. Rama Ranganathan (UT Southwestern) and Alan Friedman (Purdue). We thank reviewers for a number of very helpful comments in characterizing our methods.

REFERENCES

[1] W.L. Buntine. Operations for learning with graphical models. *JAIR*, 2:159–225, 1994.

[2] B.I. Dahiyat and S.L. Mayo. De novo protein design: fully automated sequence selection. *Science*, 278(5335):82–87, Oct 1997.

[3] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter (eds.). *MCMC in Practice*. Chapman & Hall/CRC, London, UK, 1995.

[4] H. Kamisetty, E.P. Xing, and C.J. Langmead. Free energy estimates of all-atom protein structures using generalized belief propagation. In *Proc. RECOMB*, pages 366–380, Apr 2007.

[5] S. Kamtekar, J.M. Schiffer, H. Xiong, J.M. Babik, and M.H. Hecht. Protein design by binary patterning of polar and nonpolar amino acids. *Science*, 262(5140):1680–1685, Dec 1993.

[6] B. Kuhlman, G. Dantas, G.C. Ireton, G. Varani, B.L. Stoddard, and D. Baker. Design of a novel globular protein fold with atomic-level accuracy. *Science*, 302(5649):1364–1368, Nov 2003.

[7] S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, UK, 1996.

[8] J. Li, Z.-P. Yi, M.C. Laskowski, M. Laskowski Jr., and C. Bailey-Kellogg. Analysis of sequence-reactivity space for protein-protein interactions. *Proteins*, 58(3):661–671, Feb 2005.

[9] R.H. Lilien, B.W. Stevens, A.C. Anderson, and B.R. Donald. A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase A phenylalanine adenlytaion enzyme. *J. Comp. Biol.*, 12(6):740–761, July 2005.

[10] S.W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–299, Oct 1999.

[11] C. Loose, K. Jensen, I. Rigoutsos, and G. Stephanopoulos. A linguistic model for the rational design of antimicrobial peptides. *Nature*, 443(7113):867–869, Oct 2006.

[12] S.M. Lu, *et al.*, and M. Laskowski, Jr. Predicting the reactivity of proteins from their sequence alone: Kazal family of protein inhibitors of serine proteinases. *PNAS*, 98(4):1410–1415, Feb 2001.

[13] O. Olmea, B. Rost, and A. Valencia. Effective use of sequence correlation and conservation in fold recognition. *J. Mol. Biol.*, 293(5):1221–1239, Nov 1999.

[14] C.R. Otey, J.J. Silberg, C.A. Voigt, J.B. Endelman, G. Bandara, and F.H. Arnold. Functional evolution and structural conservation in chimeric cytochromes P450: Calibrating a structure-guided approach. *Chem. Biol.*, 11(3):309–318, Mar 2004.

[15] F. Pazos, M. Helmer-Citterich, G. Ausiello, and A. Valencia. Correlated mutations contain information about protein–protein interaction. *J. Mol. Biol.*, 271(4):511–523, Aug 1997.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[17] W.P. Russ, D.M. Lowery, P. Mishra, M.B. Yaffee, and R. Ranganathan. Natural-like function in artificial WW domains. *Nature*, 437(7058):579–583, Sep 2005.

[18] L. Saftalov, P.A. Smith, A.M. Friedman, and C. Bailey-Kellogg. Site-directed combinatorial construction of chimaeric genes: General method for optimizing assembly of gene fragments. *Proteins*, 64(3):629–642, Jun 2006.

[19] M. Socolich, S.W. Lockless, W.P. Russ, H. Lee, K.H. Gardner, and R. Ranganathan. Evolutionary information for specifying a protein fold. *Nature*, 437(7058):512–518, Sep 2005.

[20] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE Trans. Comp. Biol. and Bioinf.*, 5(2):183–197, Apr–Jun 2008.

[21] C. Yanover and Y. Weiss. Finding the *M* most probable configurations using loopy belief propagation. In *Proc. NIPS*, 2003.

[22] X. Ye, A.M. Friedman, and C. Bailey-Kellogg. Hypergraph model of multi-residue interactions in proteins: sequentially-constrained partitioning algorithms for optimization of site-directed protein recombination. *J. Comp. Biol.*, 14(6):777–790, Jul 2007.

[23] W. Zheng, A.M. Friedman, and C. Bailey-Kellogg. Algorithms for joint optimization of stability and diversity in planning combinatorial libraries of chimeric proteins. In *Proc. RECOMB*, pages 300–314, Apr 2008.

[24] W. Zheng, X. Ye, A.M. Friedman, and C. Bailey-Kellogg. Algorithms for selecting breakpoint locations to optimize diversity in protein engineering by site-directed protein recombination. In *Proc. CSB*, pages 31–40, Aug 2007.

John Thomas is a Ph.D. candidate in computer science at Dartmouth College. He holds an M.S. in computer science from Dartmouth College, and a B.S. in mathematics and B.A. in computer science from Gettysburg College. As a masters student, he worked in scheduling theory, proving the existence of schedules that are simulataneously near optimal for two optimality criteria. His current research interests are in bioinformatics, in particular identifying and modeling evolutionary constraints to assist in protein analysis, classification, and design.

Naren Ramakrishnan is a professor and associate head for graduate studies in the Department of Computer Science at Virginia Tech. He also serves as an adjunct professor at the Institute of Bioinformatics and Applied Biotechnology (IBAB), Bangalore, India. His research interests include problem solving environments, mining scientific data, and information personalization. He is an area editor for IEEE Computer and is a general chair for the Eighth IEEE International Conference on Data Mining (Pisa, Italy, Dec 2008). Ramakrishnan received his Ph.D. in computer sciences from Purdue University.

Chris Bailey-Kellogg is an associate professor of computer science at Dartmouth. His research focuses on integrated experiment planning and data analysis for protein structure determination and protein engineering. He earned a BS/MS from MIT and a PhD with Feng Zhao at Ohio State and Xerox PARC, and conducted postdoctoral research in computational biology with Bruce Donald at Dartmouth. He has received an NSF Career award and Alfred P. Sloan Foundation fellowship.

APPENDIX

Tab. I lists edges in order of selection by the greedy algorithm; earlier edges have a bigger impact on explaining the observed coupling.

TABLE I
COMPLETE EDGE LIST FOR THE GMRC LEARNED FROM A DATASET OF WW DOMAINS.

Edge number	Residue 1	Residue 2
1	35	37
2	24	26
3	1	5
4	11	13
5	2	3
6	27	37
7	15	39
8	21	39
9	5	20
10	6	32
11	24	32
12	13	20
13	26	30
14	1	29
15	25	27
16	12	17
17	17	34
18	12	19
19	6	19
20	24	31
21	20	23
22	23	35
23	1	11
24	22	28
25	28	33
26	10	30
27	1	13
28	24	29
29	3	32
30	3	6
31	15	22
32	29	32
33	24	25
34	25	37
35	25	29