Protein Design by Mining and Sampling an Undirected Graphical Model of Evolutionary Constraints

Dartmouth Computer Science Technical Report TR2007-587

John Thomas
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755
jthomas@cs.dartmouth.edu

Naren Ramakrishnan
Dept. of Computer Science
Virginia Tech
Blacksburg, VA 24061
naren@cs.vt.edu

Chris Bailey-Kellogg
Dept. of Computer Science
Dartmouth College
Hanover, NH 03755
cbk@cs.dartmouth.edu

ABSTRACT

Evolutionary pressures on proteins to maintain structure and function have constrained their sequences over time and across species. The sequence record thus contains valuable information regarding the acceptable variation and covariation of amino acids in members of a protein family. When designing new members of a protein family, with an eye toward modified or improved stability or functionality, it is incumbent upon a protein engineer to uncover such constraints and design conforming sequences. This paper develops such an approach for protein design: we first mine an undirected probabilistic graphical model of a given protein family, and then use the model generatively to sample new sequences. While sampling from an undirected model is difficult in general, we present two complementary algorithms that effectively sample the sequence space constrained by our protein family model. One algorithm focuses on the high-likelihood regions of the space. Sequences are generated by sampling the cliques in a graphical model according to their likelihood while maintaining neighborhood consistency. The other algorithm designs a fixed number of high-likelihood sequences that are reflective of the amino acid composition of the given family. A set of shuffled sequences is iteratively improved so as to increase their mean likelihood under the model. Tests for two important protein families, WW domains and PDZ domains, show that both sampling methods converge quickly and generate diverse high-quality sets of sequences for further biological study.

1. INTRODUCTION

Data mining techniques are now firmly established in biological data analysis, especially data resulting from high-throughput screens such as microarrays and genome-wide deletion screens [4]. From simple clustering of gene expression profiles [6], researchers are now able to generate system-wide perspectives on complex diseases such as cancer [22]. It is a natural step to move from mining existing data, to using mined models to guide subsequent experiments so as to clarify ambiguities, resolve inconsistencies, and so forth [9]. Perhaps an even more exciting step is to use mined knowledge *generatively*, designing new biological and biochemical entities according to constraints mined from existing data.

This paper develops methods to design new proteins by first mining a graphical model that captures constraints in a dataset of existing related proteins, and then sampling from the model new sequences that satisfy the constraints. To understand the difficulty of our problem, it is helpful to briefly review the basics of protein sequence, structure, and function. A protein is chiefly defined by its *primary sequence*, i.e., a string of *amino acids* (loosely, *residues*). Different amino acid types have different biochemical and biophysical properties; consequently, different primary sequences adopt different three-dimensional structures and perform different functions (e.g., catalyzing different reactions). The goal of protein design is to produce a primary sequence meeting desired characteristics (e.g., specified structure [5, 11] or catalytic activity [16, 14, 19]). This is a difficult problem due to the complex relationship between the available degrees of freedom (choices of amino acid types) and their impact on structure and function—scientists don't have a good set of "rules" mapping between sequence and folding, participation in binding, etc.

Our approach takes as input a *family* of related proteins (e.g., the same protein from different organisms) in order to uncover the sequence constraints underlying them (Fig. 1). The constraints include both amino acid *conservation* (common amino acid types at particular positions in the given sequences) as well as *coupling* (common pairs of amino acid types at pairs of positions). The importance of coupling information was convincingly demonstrated by Ranganathan and colleagues, who used it to design new, stably folded [23] and functional [20] WW domains, and showed that satisfying the coupling constraints was to some extent both necessary and sufficient for viability. Most other such work focuses on conservation alone, e.g., designing antimicrobial peptides by identifying a regular expression grammar underlying naturally occurring peptides [17], or designing alpha-helical folds by simple polar-nonpolar patterning of amino acid types [10]. Our graphical models naturally encode both types of constraints, with a compact representation provid-

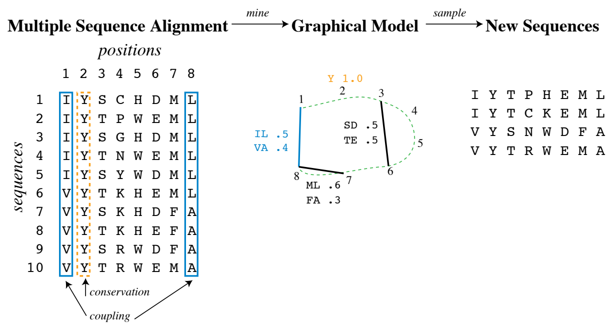

Figure 1: Mining and sampling a protein family model. We are given a set of sequences from a protein family, in a multiple sequence alignment (i.e., each row is a sequence, aligned so that each column has corresponding residues from the different sequences). Coupling and conservation constraints are mined and summarized into a graphical model. Through its edges, the model captures conditional independence constraints (e.g., that residue positions 1 and 8 are coupled, as are 8 and 7, and that these two couplings serve to relate 1 and 7). Through its clique potentials (not shown here), the model captures probability distributions for subsets of residues. Such a graphical model is then sampled to yield new sequences that obey the underlying constraints of the protein family.

ing a probabilistic semantics.

Although graphical models are an active area of KDD research (e.g., see [2]), our work is novel in two critical aspects: the input to the mining algorithm and the uses for the mined patterns. We have been developing the first methods for learning undirected graphical models from multiple sequence alignments of proteins [24]. Secondly, traditional applications of undirected graphical models, e.g., in computer vision, remote sensing, and scene modeling, have not emphasized sampling from these models, and we develop new methods for the same.

To summarize, our specific contributions are as follows:

1. Formalizing protein design as a two-step process: mining undirected graphical models capturing the essential constraints underlying a protein family, and using the models *generatively* to produce new sequences.

2. Formulating a probabilistic basis (via our graphical models) for evaluating the quality of new sequences with respect to satisfaction of protein family constraints. We demonstrate the ability of this approach to predict foldedness of a set of previously designed sequences.

3. Developing two new sampling techniques, *component sampling* and *constrained shuffling*, for navigating the constrained sequence space represented by the mined models. With case study application to two different protein families, we demonstrate that the algorithms converge quickly and generate high-likelihood sequences, reflective of (but different from) the original sequences.

2. MINING A GRAPHICAL MODEL OF RESIDUE COUPLING

A *multiple sequence alignment* (MSA) organizes a set of related sequences into a matrix $\mathcal{S}$, such that each row of $\mathcal{S}$ is a sequence, and each column has corresponding residues (Fig. 1). 'Gap' characters are inserted into the sequences to make the columns line up appropriately. An MSA both relies upon and reveals correspondence between residues in the related proteins. We can see conservation constraints in the columns of $\mathcal{S}$—some columns have particularly biased distributions of amino acid types. For example, column 2 in the example MSA in Fig. 1 is constrained to only be Y (Tyrosine). These constraints form the basis for most existing models of protein families, such as Hidden Markov Models. We can also see coupling constraints in pairs of columns of $\mathcal{S}$—for some pairs, not all combinations of amino acid types (factoring in degree of conservation) are equally likely. For example, columns 1 and 8 in the MSA of Fig. 1 are coupled. As discussed in the introduction, these pairwise terms are critical in designing new proteins that are folded and functional. While there are experimental methods available for probing both conservation constraints (e.g., Alanine-scanning mutagenesis) and correlation constraints (e.g., double mutants cycles), we focus here on mining them from an MSA.

Mining an *MSA* $\mathcal{S}$ of a given protein family can be viewed as a problem of modeling categorical data [7]. In particular, we can cast key protein family constraints as instances of functional dependencies (FDs) in such categorical data. For instance a strict conservation constraint implies that the FD

$$\rightarrow (r_2 = \text{`Y'})$$

holds, i.e., two tuples (sequences) from relation $\mathcal{S}$ unconditionally will agree on the right side of the FD, namely that the second residue is a 'Y.' Similarly, coupling constraints, at the bare minimum, obey the pair of FDs

$$r_1 \rightarrow r_8, r_8 \rightarrow r_1$$

which states that if two sequences agree on residue r_1 (or r_8), then they also agree on the other.

One approach to modeling coupling and conservation in an MSA is to find all sets of (exact and approximate) FDs using association rule mining or a correlation mining algorithm (e.g., as done in [25]). However, our goal is not just to find pairs but to factorize them into a core set of dependencies, and to be able to use the constraints generatively to sample from the induced space of sequences. Hence a probabilistic model is called for. Toward this end, we model $\mathcal{S}$ by factorizing its amino acid distribution into an undirected *graphical model of residue coupling (GMRC)*, $G = (V, E)$. The vertices V are random variables, one for each column of $\mathcal{S}$. A vertex captures the observed frequency of each of the 20 amino acids at a position. The edges E encode independence relationships—a vertex is conditionally independent of all other vertices, given its immediate neighbors. G thereby defines a pdf $P_G(R)$ on residue types R for its vertices V, computed by combining scores ("potentials") for the cliques in the graph.

$$P_G(R) = \frac{1}{Z} \prod_{C \in \text{cliques}(G)} \phi_C(R_C) \tag{1}$$

Here, the subscript C restricts to the set of vertices (and corresponding amino acid types) of clique C. Z is a normalizing factor, and the ϕ_C are potential functions, such that

$$\prod_{C \,\in\, \text{cliques}(G)} \phi_C(R_C) = \frac{\prod_C P_C(R_C)}{\prod_{A \,\in\, \text{cliqueadj}(G)} P_A(R_A)} \qquad (2)$$

Notice that the potentials are given by the product of marginals defined over the cliques divided by the product of marginals defined over the clique adjacencies A, which could be nodes, edges, or general subgraphs. Thus each potential is either a conditional or a joint marginal distribution. Since pointwise as well as joint probabilities are represented, the model generalize traditional conservation-based approaches to characterizing protein sequences.

We adopt the following estimator for $P_C(R_C)$, the probability of a set of amino acid types R_C at a clique.

$$P_C(R_C) = \frac{f_C(R_C) + \frac{\rho|\mathcal{S}|}{21^{|C|}}}{|\mathcal{S}|(1+\rho)} \qquad (3)$$

Here $f_C(R_C)$ is the frequency, in $\mathcal{S}$, of the set of amino acid types, $|\mathcal{S}|$ is the total number of sequences, $|C|$ is the cardinality of the clique, and ρ is a parameter that weights the importance of missing data. Notice that even when a particular clique value does not appear in the MSA, it still has a positive (but small) probability, thereby enabling proper factorization according to the Hammersley-Clifford theorem [12].

The equations above give us the ability to compute sequence likelihoods under a given model, and we can now turn our attention to mining these models. There are two broad classes of algorithms: score-based and constraint-based. In the former, we sequentially search over the space of possible edge additions, greedily adding edges that improve the score. In the latter, we exploit graphical properties of probabilistic networks, such as d-separation, to first identify a set of conditional independencies that hold in the dataset, and then proceed to find a network that obeys these independencies. We adopt the former approach due to the sparse data contexts that underly protein design.

To help pick edges for consideration by our search algorithm, we look for those that cause good decouplings, i.e., those whose inclusion helps render sets of residues (conditionally) independent. A direct way to seek decouplers is to estimate conditional mutual information:

$$MI(v_i, v_j \mid v_k) =$$

$$\sum_{R_k \in \mathcal{A}^*} P_k(R_k) \sum_{R_i \in \mathcal{A}} \sum_{R_j \in \mathcal{A}} P_{ij}(R_{ij}|R_k) \log \frac{P_{ij}(R_{ij}|R_k)}{P_i(R_i|R_k) P_j(R_j|R_k)} \quad (4)$$

where we estimate the conditionals by subsetting residue k to its most frequently occurring amino acid types ($\mathcal{A}^* \subset \mathcal{A}$), defined as those that appear in at least 15% of the original sequences in the subset. As discussed [15], such a bound is required in order to maintain fidelity to the original MSA and allow for evolutionary exploration. We also ensure that $P_k(R_k)$ distributes probability mass of 1 among just these indices, in proportion to the number of sequences in each subset, so that $\sum_{R_k \in \mathcal{A}^*} P_k(R_k) = 1$.

Our algorithm greedily and incrementally grows a graph by, at each step, selecting the best edge—the one that most reduces the coupling in the graph and is statistically significant. If, while growing the model, we find that an edge is statistically insignificant, we can exclude that edge from consideration and look for other ways to factorize the relationships. We evaluate an edge's decoupling effect by comparing the total coupling of the graph without the edge vs. with the edge, scoring each graph as follows.

$$\text{Score}(G) = \sum_{v \in V} \sum_{u \notin \text{neighbors}(v)} MI(u, v \mid \text{neighbors}(v)) \quad (5)$$

We evaluate the statistical significance of an edge according to a p-value test for independence:

$$\chi^2 = \sum_{a \in \mathcal{A}_i} \sum_{b \in \mathcal{A}_j} \frac{\left(f_{\{i,j\}}(\{a,b\}) - \frac{f_{\{i\}}(\{a\}) \cdot f_{\{j\}}(\{b\})}{|\mathcal{S}|^2}\right)^2}{\frac{f_{\{i\}}(\{a\}) \cdot f_{\{j\}}(\{b\})}{|\mathcal{S}|^2}} \quad (6)$$

Here i and j are the vertices of the edge, and $f_C(R_C)$ is, as in Eq. 3, the number of occurrences of residue types R_C at positions C. The first term in the numerator is the actual number of pairs observed; the second term is the expected number, if the two residues were independent.

Since the model grows iteratively, the runtime of the algorithm is determined by how many calculations are performed in each iteration. A naïve implementation of the algorithm would compute the score for every possible edge at every iteration. Since there are potentially $O(n^2)$ edges and each edge requires $O(n)$ MI calculations, this implementation requires $O(n^3)$ computations per iteration. However, by caching edge scores from previous iterations and only recomputing when conditioning contexts change (which can happen to at most $O(n)$ edges per iteration), each iteration requires only $O(n^2)$ MI computations, yielding a speedup of $O(n)$.

3. 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 [3]. Sampling from an undirected graphical model is more difficult, however, due to the presence of cycles. We present two approaches for sampling an undirected graphical model of residue coupling. The first method, *component sampling*, samples new sequences according to their likelihood under the model. The problem of cycles is mitigated by randomly ordering the cliques in a connected component of the graphical model, and then sampling from the clique distributions in order, conditioning each sample on the values chosen for preceding cliques. Component sampling allows for an essentially unlimited number of sequences to be generated from the model. The second method, *constrained shuffling*, on the other hand, seeks to design a fixed number of sequences drawn from a fixed pool of amino acids for the residue positions. Using a Monte Carlo simulation, the amino acids are permuted column-wise in an attempt to improve the overall average likelihood of the generated sequences. We discuss each of these sampling methods in turn.

At an abstract level, our first algorithm, component sampling, is a Gibbs sampler [8] 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

Algorithm 1 ComponentSampling(G)

```
Input: graphical model of residue coupling $G$
Output: set of sequences sampled from $G$
 1: $S \leftarrow \emptyset$
 2: while not converged do
 3:   $C \leftarrow$ random clique from $G$, with equal probability
 4:   $S_C \leftarrow$ random AA types $R_C$, with probability $P_C(R_C)$
 5:   $Q \leftarrow$ queue initialized as random permutation of cliques neighboring $C$ in $G$
 6:   while $Q$ is not empty do
 7:     $D \leftarrow$ dequeue from $Q$
 8:     $A \leftarrow$ vertices already assigned in $D$
 9:     if $A \neq D$ then
10:       $S_{D-A} \leftarrow$ random AA types $R_{D-A}$ with probability $P_{D-A}(R_{D-A}|R_A)$
11:       enqueue onto $Q$ a random permutation of cliques neighboring $D$ in $G$
12:     end if
13:   end while
14:   output $S$
15: end while
```

get stuck in local minima 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.

Algorithm 1 provides the details of our component sampling algorithm. 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 distribution is sufficiently good.

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). Thus 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 do not cause large deviations from the true distribution.

The motivation for our second algorithm, constrained shuffling, arises from the desire to study experimentally a small number of new sequences. One option would be to generate a large number of sequences using component sampling and study only the highest scoring sequences. However, these sequences may be highly similar to each other (and thus somewhat redundant to test). The goal of constrained shuffling is to generate a small set of high-likelihood new sequences that have the same amino acid composition as the original sequences. That is, each column of the MSA of the new sequences is a permutation of the corresponding column of the original MSA. Constrained shuffling is a modification of an approach used by Ranganathan and colleagues [23]. Here, we adopt their move strategy, but develop an objective function targeting sequences of high likelihood under a graphical model.

Algorithm 2 ConstrainedShuffling(G, $\mathcal{S}$)

```
Input: graphical model of residue coupling $G$ mined from MSA $\mathcal{S}$ (size $m$ sequences of $n$ residues)
Output: MSA $\mathcal{S}$' of sampled sequences
 1: $\mathcal{S}' \leftarrow$ column-wise permutation of $\mathcal{S}$
 2: $v \leftarrow \text{average}_{s\in\mathcal{S}'} P_G(s)$
 3: while not converged do
 4:   $c \leftarrow$ random column $\in [1, n]$
 5:   $s, t \leftarrow$ random rows $\in [1, m]$ s.t. $s \neq t$
 6:   swap $\mathcal{S}'[s, c]$ and $\mathcal{S}'[t, c]$
 7:   $v' \leftarrow \text{average}_{s\in\mathcal{S}'} P_G(s)$
 8:   if $v' \geq v$ or with probability $e^{v'-v}$ then
 9:     accept $\mathcal{S}$'
10:   else
11:     undo the swap
12:   end if
13: end while
```

Algorithm 2 provides the algorithm. The procedure begins by independently shuffling the columns of an MSA. Then an MCMC process is begun, making moves that swap the amino acids in two random sequences at a single random column. Notice this move preserves the same amino acid composition as the original MSA. The move is accepted if the average log likelihood of the new sequences is improved. Otherwise, the move 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.

4. RESULTS

Our primary case study is 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 a previous protein design studies by Ranganathan and colleagues [23, 20], who provided the following:

1. An input dataset of 42 natural 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

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 [23].

We show here that our mining method learns a graphical model that captures significant constraints in the natural

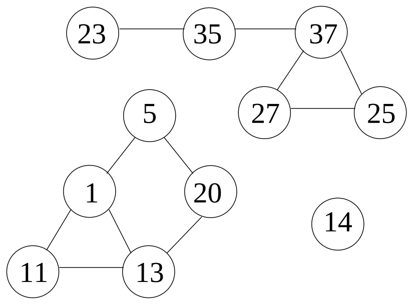

Figure 2: Parts of the graphical model of residue coupling mined from a dataset of WW domains with 11 vertices and 11 edges. The remainder of the model not shown contains 28 more vertices and 24 more edges.

sequences and is able to classify the new *IC* and *CC* sequences according to foldedness. We then demonstrate that our sampling methods efficiently generate a wide range of high-likelihood putative WW domain sequences. To further validate our method, we illustrate its ability to efficiently generate new putative members of another protein family—PDZ domains. The designed putative WW and PDZ sequences serve as hypotheses for further biological study.

4.1 Mining a Graphical Model

Our mining algorithm (Sec. 2) learns a graphical model of residue coupling 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. The model contains several non-clique cycles, such as 1–5–20–13.

A key advantage of our probabilistic models is that they provide a mechanism for evaluating new sequences, by likelihood (Eq. 1). We tested the ability of the likelihood to predict foldedness of the *IC* and *CC* sequences (Fig. 3). For the most part, sequences that adopt the native fold tend to score higher than those that do not. To quantify this result, we used the log likelihood as a classifier, and generated an ROC curve (Fig. 4, blue solid line) 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). In contrast, a classifier based on conservation alone (Fig. 4, red dashed line) has a power of only .68. By "conservation alone," we mean that each residue is independent; the model has no edges.

It is important to note that the classification results are for sequences that were designed in some sense (conservation for *IC* and coupling for *CC*) to represent the natural WWs. Thus the model extracts even more information useful in predicting foldedness. Since the results show that likelihood under our graphical model is highly predictive of foldedness, we are justified in applying our sampling algorithms to design new proteins accordingly.

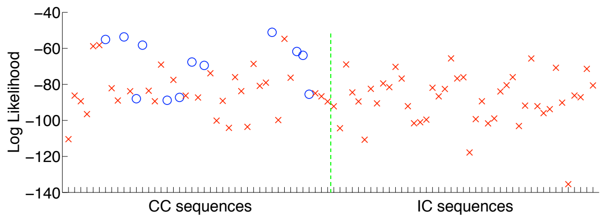

Figure 3: Log likelihood, under our graphical model, of the 86 *IC* and *CC* sequences. Blue 'o's and red '×'s indicate those proteins found by Ranganathan and colleagues to be folded and unfolded, respectively.

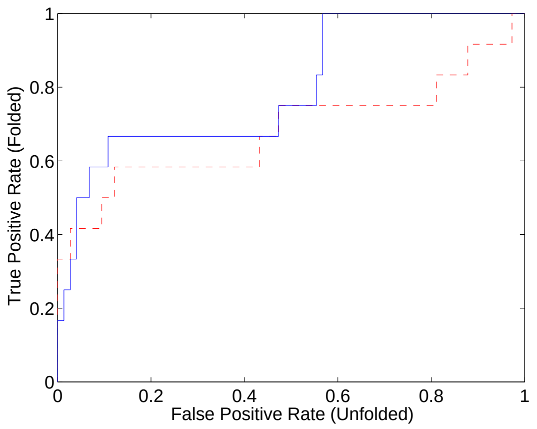

Figure 4: ROC curves for classification of *IC* and *CC* sequences according to log likelihood under our graphical model (blue solid line) and by conservation alone (red dashed line). The false positive rate is the percentage of sequences predicted to be of native fold that are not; the true positive rate is the percentage of sequences predicted to be of native fold that are.

4.2 Component Sampling

We applied our component sampling method (Algorithm 1) to generate new putative WW sequences. After a burn-in series of 88 moves (the number of moves it took to sample each residue at least once) we generated 10000 new 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. We measure the extent of agreement as the $L2$ distance between the distributions. Fig. 5 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

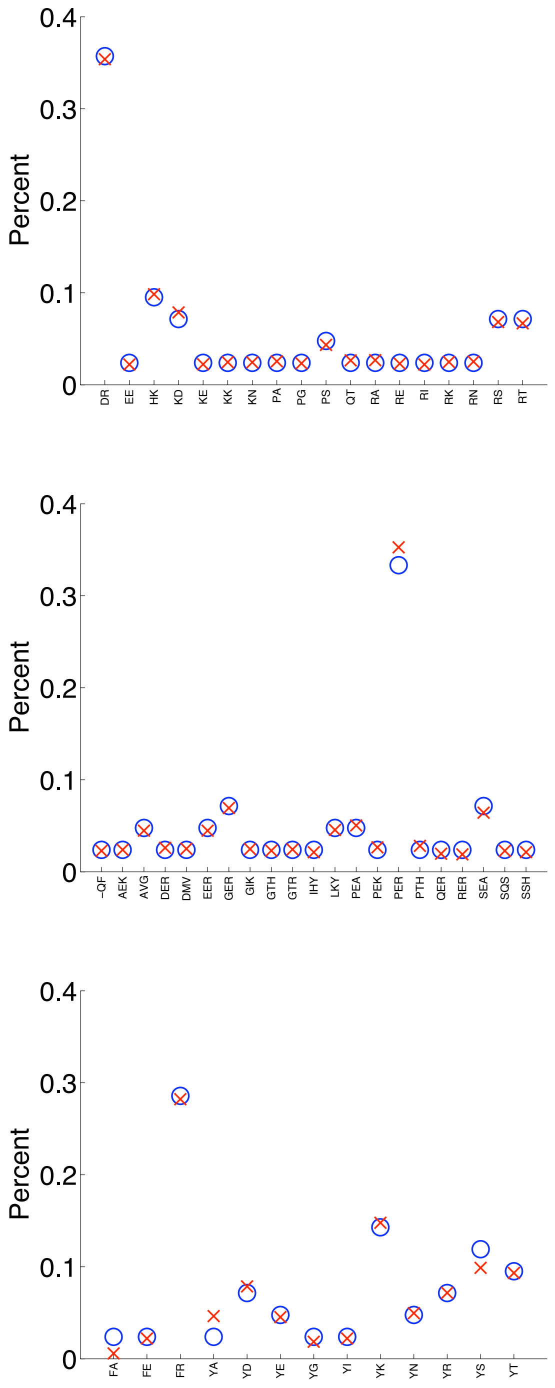

Figure 5: Probability distribution over sets of amino acid types according to the graphical model (blue 'o's) and the sequences generated by component sampling (red '×'s) for the 2-clique 35–37 (top), the 3-clique 1–11–13 (middle), and the transitive relationship between residues 23 and 37 (bottom).

to the model distribution, we monitored at each iteration the $L2$ distances for all cliques. Fig. 6 shows that the total

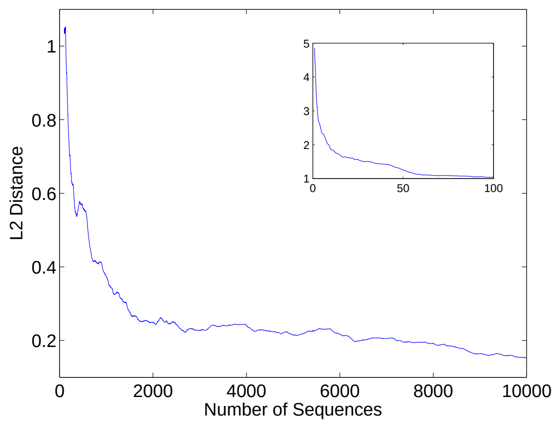

Figure 6: Evolution of the total $L2$ distance (over all cliques) between the graphical model and the sequences generated by component sampling.

$L2$ 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 10000 samples.

In addition to sampling from the model distribution, component sampling generates sequences of high likelihood. Fig. 7 shows a histogram of the log likelihood; the mean is -33.48 and the standard deviation 5.02. By comparison to Fig. 3, we see that even the least likely sequence generated by component sampling (log likelihood -52.19) is more likely than all but one of the 86 *IC* and *CC* sequences (mean log likelihood -84.14 and standard deviation 15.12).

The sequences generated by component sampling are also substantially different from the natural WW domain sequences used to learn the model—we aren't simply re-generating the input, but in fact exploring the space constrained by the model. Fig. 8 shows a histogram of the sequence identity for the new sequences to their most similar natural sequences. The mean sequence identity is 65.14% and the 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%. Of the sequences generated using component sampling, the sequence most similar to a natural WW domain has a sequence identity of 89.74% while the designed sequence with the lowest identity has only 41.03% identical residues.

Another way to measure the diversity of the new sequences is their average sequence identity to the natural WW domains and to each other. Fig. 9 shows a histogram for the average sequence identity to the natural WW domains. We again see that the new sequences are quite different from the natural ones, with a mean average sequence identity of 45.18% and standard deviation of 5.29%. This level of identity is similar to that of the *IC* and *CC* sequences, which have a mean of 43.75% and standard deviation of 4.09%. The new sequences are also different from each other, with an average pairwise identity of 45.47% with standard deviation 11.17%. This is again comparable to the values in *IC* and *CC* of 43.05% and 8.10%. Thus we maintain the level of novelty of sequences, while increasing their likelihood under

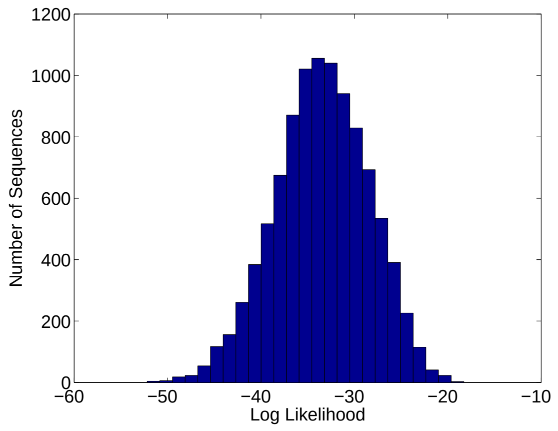

Figure 7: Log likelihood distribution for the 10000 sequences generated by component sampling.

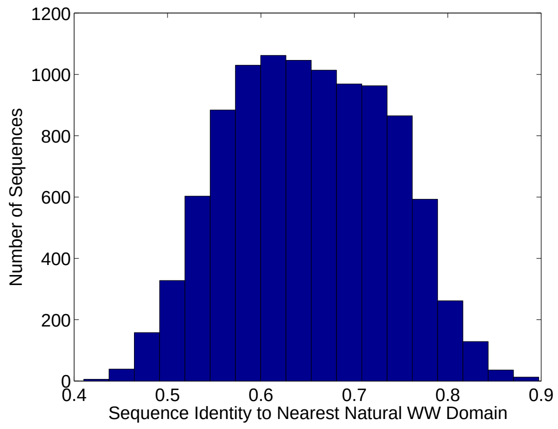

Figure 8: Distribution of the sequence identity to the nearest natural WW domain of the 10000 sequences generated by component sampling.

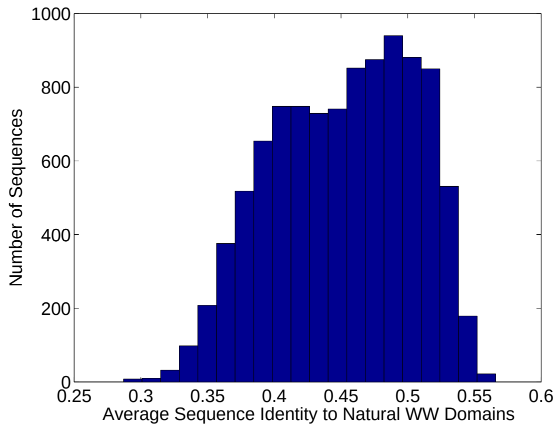

Figure 9: Distribution of the average sequence identity to the natural WW domains of the 10000 sequences generated by component sampling.

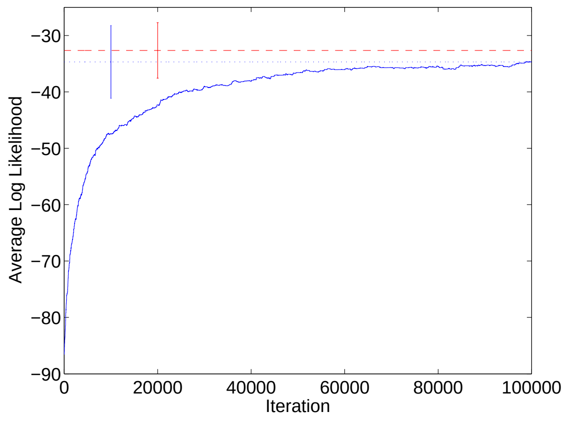

Figure 10: Convergence of the average likelihood score of sequences generated by constrained shuffling. Blue dotted line: final value (-34.69 with standard deviation of 6.46); red dashed line: natural sequences (-32.65 with standard deviation of 4.93).

the model (and, by correlation, the expected likelihood of folding).

4.3 Constrained Shuffling

We applied the constrained shuffling method to sample from the graphical model a set of 42 new high-likelihood sequences with amino acid types shuffled columnwise from the input dataset. We ran our algorithm for 100000 iterations. Fig. 10 shows the convergence of the average log likelihood of the generated sequences at each iteration. Although the average log likelihood starts at only -86.52, after 100000 iterations, it reaches -34.69 with standard deviation 6.46, shown by the dotted line. This value is very close to the average log likelihood of the natural sequences, -32.65 (with standard deviation 4.93), shown by the dashed line. After 100000 iterations, constrained shuffling is able to generate a new set of WW sequences with log likelihoods similar to those of the natural WW domains.

The sequences generated by constrained shuffling provide a range of log likelihoods under the model, as shown in Fig. 11. As was the case for component sampling, the 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. Our sampling method is once again able to produce higher scoring sequences than the reference *IC* and *CC* sequences.

To ensure that constrained shuffling isn't simply recreating the natural WW domains, we measured the sequence identity between the designed sequences and their closest natural neighbors. Fig. 12 shows a histogram of the sequence identities. As was the case for component sampling, the designed sequences are quite different from the natural WW domains. The mean sequence identity for the generated sequences is 60.56% with a standard deviation of 7.70%. Of the sequences generated by constrained shuffling, even the sequence most similar to the natural WW domain sequences is only 74.36% identical, while the least similar sequence is only 46.15% similar.

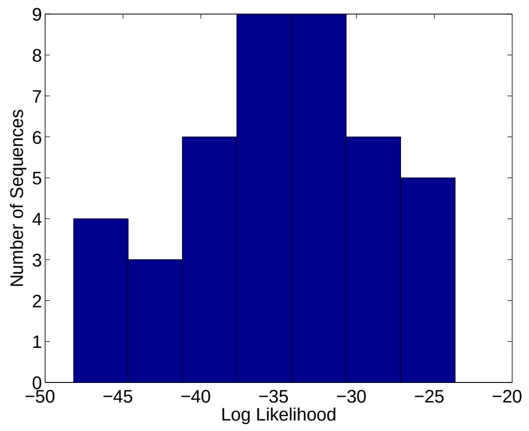

Figure 11: Log likelihood distribution for the 42 sequences generated by constrained shuffling.

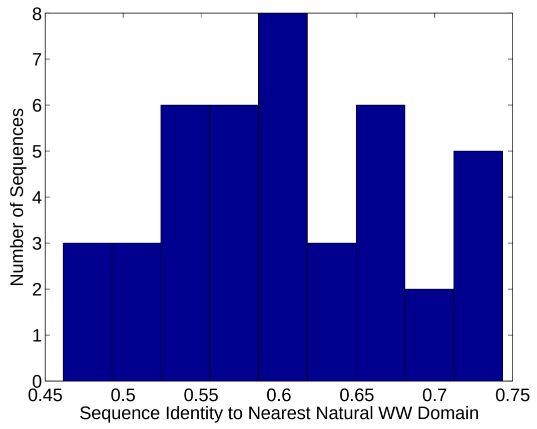

Figure 12: Distribution of the sequence identity to the nearest natural WW domain of the 42 sequences generated by constrained shuffling.

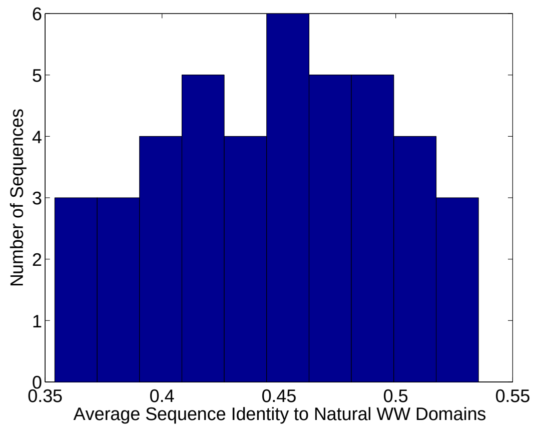

Figure 13: Distribution of the average sequence identity to the natural WW domains of the 42 sequences generated by constrained shuffling.

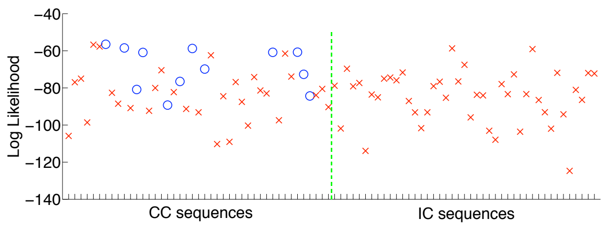

Figure 14: Log likelihood, under the model mined from constrained shuffling sequences, of the 86 *IC* and CC sequences.

As we did with the component sampled sequences, we can measure the average identity of these new sequences to the natural WW domains and to each other. Fig. 13 shows the histogram of the average identity of the new sequences to the natural WW domains. Again, we find that the designed sequences to be different from the natural sequences, having a mean average identity of 44.95% and a standard deviation of 4.92%. The sequences are different from each other, with an average pairwise identity of 43.60% (standard deviation 9.69%).

4.4 Closing the Loop

If the generated sequences were in fact representative of the natural ones, we would expect them to generate a similar model. We call this "closing the loop"—mining a model of natural sequences, using it to generate new sequences, and mining a model of the new sequences. To test our ability to close the loop, we mined a model of the 42 sequences generated by constrained shuffling. 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. 14 (compare Fig. 3) and Fig. 15 (compare Fig. 4). The power of the new classifier (area under the ROC curve) is .80, the same power as the original model, demonstrating a successful closing of the loop.

4.5 PDZ Domains

As further validation of our methods, we applied them to a family of PDZ domains, which, like WW domains, are small proteins that assist in protein complex formation. We obtained from PDZBase [1] an MSA of 80 class I PDZs aligned to 80 residues (class I PDZs recognize ligands with C-terminal sequences of the form S/T-X-[hydrophobic residue]). Our mining algorithm learned a graphical model consisting of 85 statistically significant edges forming 96 cliques. The details of the model are omitted due to lack of space.

Unlike with WWs, we do not have a negative control (i.e., putative PDZs that did not fold). However, since we demonstrated above with WWs that the likelihood under a model was highly predictive of foldedness, we hypothesize that new sequences sampled from the PDZ model merit further biological study. We focus here on the computational effectiveness of our sampling algorithms.

We first applied our component sampling method (Al-

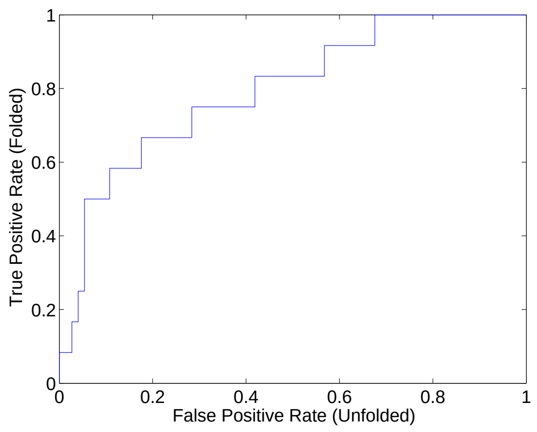

Figure 15: The ROC curve of classification of IC and CC sequences according to log likelihood under the graphical model mined from the sequences designed by constrained shuffling.

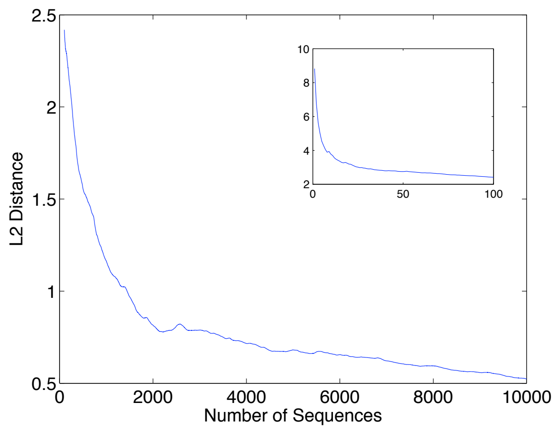

Figure 16: Evolution of the total $L2$ distance (over all cliques) between the graphical model and the sequences generated by component sampling for class I PDZ domains.

gorithm 1) to generate new putative class I PDZ domain sequences. After a burn-in series of 223 moves (by which point each residue was sampled at least once), we generated 10000 new sequences. The algorithm got stuck, and consequently unwound the samples of a partially completed move, 1060 times. As with the WW domains, we monitored the convergence of the algorithm in terms of the total $L2$ distance, over all cliques, between the model and sampled sequences. Fig. 16 shows that the total $L2$ distance converges very quickly: it is 2.42 after only 100 samples, 1.16 after 1000 samples, and .5232 (approximately .0054 per clique) after 10000 samples.

We also applied the constrained shuffling method (Algorithm 2) to sample a set of 80 new high-likelihood sequences. We ran the constrained shuffling algorithm for 250000 iterations. We performed more iterations than for the WWs to account for the larger MSA; our WW MSA is of size 42 sequences $\times$ 39 positions, while our PDZ MSA is of size 80 sequences $\times$ 80 positions. Nonetheless, Fig. 17 shows that the average log likelihood for the generated sequences converges quickly. Although the average log likelihood starts at only -190.01, after 250000 iterations it reaches -101.82 with a standard deviation of 9.08. This value is very close to the average log likelihood of the natural class I PDZ domains, -93.90 (with a standard deviation of 18.55), and thus we expect the new proteins to be of high quality biologically.

5. CONCLUSION

We have formulated protein design in terms of first mining sequences in a protein family to learn "what it means" to be a member of that family, and then sampling sequence space as constrained by what we've learned about the family. Our mining algorithm identifies and factorizes conservation and coupling constraints within family sequences, and our two sampling methods generate sequences that are of high likelihood (but yet new and different) under a model. 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. We showed the power of our approach by mining and sampling graphical models for two protein family case studies, WW domains and class I PDZ domains.

We assume that a designed sequence will be constructed *ab initio*, i.e., essentially "printing" the specified amino acid chain. This is in contrast to, say, employing site-directed mutagenesis for a small number of selected positions [14, 13], or recombining fragments from existing proteins [19, 21, 26], both of which restrict the available degrees of freedom in order to simplify the experimental process or make it applicable on a larger scale. It is interesting future work to constrain our sampling methods to generate sequences that can be constructed by these experimental techniques.

Purely sequence-based design is one approach to protein design; other approaches incorporate additional or alternative information. Structure-based approaches (e.g., [5, 11, 16, 14]) employ sophisticated optimization techniques and biophysical models that predict the energies of possible amino acid substitutions. Data-driven methods (e.g., [18, 13]) guide design based on experimental measurements of the effects of amino acid choices. We expect our graphical models to allow integration of detailed structural and experimental information and are pursuing that possibility.

Our sampling methods are readily adaptable to focus on only a subset of residue positions (e.g., near an active site). In component sampling, treat the non-focus positions as fixed, and sample only the cliques involving focus positions. If a particular sequence is desired for the non-focus positions, then condition the focus clique values accordingly; otherwise, marginalize out the non-focus positions. In constrained shuffling, shuffle and make moves only for the focus columns.

Overall, the mining and sampling methods presented here should constitute a valuable tool in the biologist's toolkit for computational protein design, and highlight an important and fertile area for future KDD research.

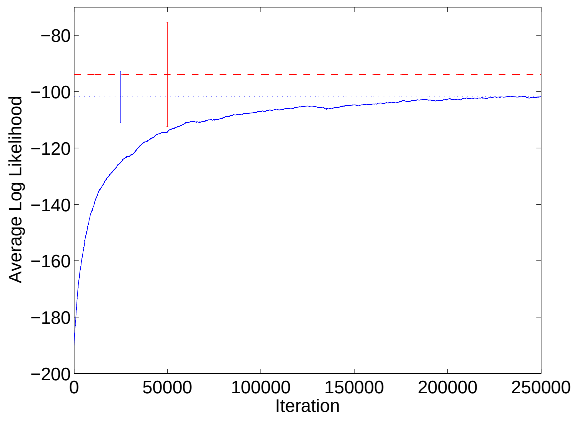

Figure 17: Convergence of the average likelihood score of PDZ sequences generated by constrained shuffling. Blue dotted line: final value (-101.82 with standard deviation of 9.08); red dashed line: natural sequences (-93.90 with standard deviation of 18.55).

6. REFERENCES

[1] T. Beuming, L. Skrabanek, M.Y. Niv, P. Mukherjee, and H. Weinstein. PDZBase: A protein-protein interaction database for PDZ-domains. *Bioinformatics*, 21:827–8, 2005.

[2] F. Bromberg, D. Margaritis, and V. Honavar. Efficient markov network structure discovery using independence tests. In *Proc SIAM Data Mining*, 2006.

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

[4] A.E. Carpenter and D.M. Sabatini. Systematic genome-wide screens of gene function. *Nat Rev Genet*, 5:11–22, 2004.

[5] B.I. Dahiyat and S.L. Mayo. De novo protein design: fully automated sequence selection. *Science*, 278:82–7, 1997.

[6] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 96:14863–8, 1998.

[7] D. Gibson, J.M. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. *VLDB J*, 8:222–36, 2000.

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

[9] T. Ideker, V. Thorsson, J.A. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, R. Aebersold, and L. Hood. Integrated genomic and proteomic analyses of a systematically pertrubed metabolic network. *Science*, 292:929–34, 2001.

[10] 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*, 262:1680–5, 1993.

[11] 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*, 302:1364–8, 2003.

[12] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[13] 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*, 58:661–71, 2005.

[14] 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. In *Proc RECOMB*, pages 46–57, 2004.

[15] S.W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286:295–9, 1999.

[16] L. Looger, M. Dwyer, J. Smith, and H. Hellinga. Computational design of receptor and sensor proteins with novel functions. *Nature*, 423:185–90, 2003.

[17] C. Loose, K. Jensen, I. Rigoutsos, and G. Stephanopoulos. A linguistic model for the rational design of antimicrobial peptides. *Nature*, 443:867–9, 2006.

[18] 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*, 98:1410–5, 2001.

[19] 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*, 11:309–18, 2004.

[20] W.P. Russ, D.M. Lowery, P. Mishra, M.B. Yaffee, and R. Ranganathan. Natural-like function in artificial WW domains. *Nature*, 437:579–83, 2005.

[21] 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*, 64:629–42, 2006.

[22] E. Segal, N. Friedman, D. Koller, and A. Regev. A module map showing conditional activity of expression modules in cancer. *Nat Genet*, 36:1090–8, 2004.

[23] M. Socolich, S.W. Lockless, W.P. Russ, H. Lee, K.H. Gardner, and R. Ranganathan. Evolutionary information for specifying a protein fold. *Nature*, 437:512–8, 2005.

[24] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of residue coupling in protein families. In *Proc BIOKDD*, pages 1–9, 2005.

[25] H. Xiong, S. Shekhar, P.-N. Tan, and V V. Kumar. TAPER: A two-step approach for all-strong-pairs correlation query in large databases. *IEEE TKDE*, 18:493–508, 2006.

[26] 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. In *Proc RECOMB*, pages 15–29, 2006.