Chapter 7

A Structure-Based Design Protocol for Optimizing Combinatorial Protein Libraries

Mark W. Lunt and Christopher D. Snow

Abstract

Protein variant libraries created via site-directed mutagenesis are a powerful approach to engineer improved proteins for numerous applications such as altering enzyme substrate specificity. Conventional libraries commonly use a brute force approach: saturation mutagenesis via degenerate codons that encode all 20 natural amino acids. In contrast, this chapter describes a protocol for designing "smarter" degenerate codon libraries via direct combinatorial optimization in "library space."

Several case studies illustrate how it is possible to design degenerate codon libraries that are highly enriched for favorable, low-energy sequences as assessed using a standard all-atom scoring function. There is much to gain for experimental protein engineering laboratories willing to think beyond site saturation mutagenesis. In the common case that the exact experimental screening budget is not fixed, it is particularly helpful to perform a Pareto analysis to inspect favorable libraries at a range of possible library sizes.

Key words Protein library design, Degenerate codon optimization, Rational mutagenesis, Saturation mutagenesis, Regression, Cluster expansion

1 Introduction

1.1 Expanding Computational Protein Design Horizons Using Regression

Algorithms for searching large conformational spaces tend to be iterative, evaluating one conformation at a time. Molecular dynamics simulations and conventional Monte Carlo protein structure prediction fall into this category, as do simulations that support the refinement of models to fit nuclear magnetic resonance spectroscopy or x-ray diffraction data. Even when each evaluation calculation is rapid, iterative methods are often unequal to the required conformational sampling tasks. Several grand-challenge problems in computational structural biology are intractable in part because of the inability of current methods to efficiently search through the space of protein conformations. For example, consider the problem of predicting the detailed structure of a protein, starting from the structure of a homologous protein that happens to be 2 Å root mean square deviation (rmsd) from the target structure. Even though the initial structure and the target are "close," it is difficult to find the target structure in part due to the vast number of similar protein conformations.

For some problems, including fixed-backbone protein design, it is feasible to limit the search to the combinatorial placement of discrete favored sidechain positions called rotamers [1]. In this case, finding the optimal combination is still a challenging (i.e. NP-hard) computational problem [2]. However, numerous powerful combinatorial optimization algorithms have been developed to optimize sidechain placement and protein design [3–8]. Predicting the energy for any given sequence requires a combinatorial rotamer optimization calculation.

Regression-based approximate models provide a powerful approach to circumvent this limitation. The basic strategy is to prepare an approximate model that can be used to rapidly guide more expensive search calculations to productive combinations. Much of the recent research that adopts this strategy has been elucidated and described as cluster expansion [9–14]. However, the use of regression approaches to model experimental protein library data belongs in the same category.

For example, Hahn et al. used regression to model experimental data for SH3 domains [13]. The ProteinGPS methodology of DNA2.0 also quantifies protein properties in terms of sequence variables [15]. Finally, the Arnold lab was repeatedly able to rationalize the thermostability of protein "chimeras" using only crude regression models that account for 1-body contributions from each sequence block [16–21]. A chimera is a protein composed of fragments of parent proteins joined at sequence junctions called crossover sites. Crossover sites are chosen with a variety of techniques that are designed to minimize the disruption of coherent and stable fragments within the protein [22–25]. The Arnold results suggest that protein fragments can make surprisingly modular contributions to the overall protein structure and stability. Johnson et al. recently provided an interesting exception to this trend, by characterizing a library of enzyme chimeras in which stability affects were decidedly cooperative rather than modular [25].

1.2 Protein Library Design

Whereas traditional computational protein design (CPD) calculations yield a single sequence (and structure) with minimal energy, the ultimate design target for this chapter is a library. Protein library design is a highly practical calculation with diverse protein engineering applications in industrial biotechnology, materials development, and the development of therapeutic biomolecules. To efficiently identify functional and/or optimized protein sequences, protein engineers commonly work at the level of libraries. Often, if structural information is absent and a suitable assay is available, these libraries consist of randomly mutated variants.

However, when a structure is available, there is a wide range of design options.

At one end of the continuum, a structure may be used simply to identify which residues are most likely to play an important role. Saturation mutagenesis refers to the practice of mutating such target residues to all possible amino acids. For example, a protein engineer working in the area of industrial biotechnology may seek to alter the specificity of a substrate-binding pocket. Alternately, a protein engineer working to optimize a therapeutic binding protein might wish to screen a library that diversifies the amino acids at the protein–protein interface.

At the other end of the continuum, conventional CPD methods combine explicit modeling of the structure with combinatorial optimization to predict a new low-energy sequence and structure thereof. There is interest in methods that combine the practical benefits of synthesizing a library of protein variants with benefits of structure-guided design. For example, Voigt et al. described a self-consistent mean field approach to identify low-energy amino acids for subtilisin E and T4 lysozyme [26]. There are numerous routes to merge these approaches. For example, the Arnold lab has often used structure-guided design to optimize libraries of synthetic enzymes derived via site-specific recombination [18].

Much of the effort has been to develop algorithms for the specific practical task of optimizing degenerate codons (*see* below). A variety of algorithms that have been developed use as an input a list of target sequences or a $20 \times n$ matrix that indicates the target frequency of each amino acid for each of the n design positions [27]. Enumeration, dynamic programming, and integer linear programming methods have all been described for the selection of degenerate codons to cover the desired sequence space [28–32].

Here, three such algorithms are described briefly. The LibDesign algorithm [28] begins with a set of aligned amino acid sequences and then identifies favorable degenerate codons independently for each position. A favorable degenerate codon encodes the specified amino acids with minimal degeneracy, avoiding stop codons if possible. Permutations of candidate codons are assessed via the resulting library size and the number of recovered sequences from the input alignment. Allen developed an algorithm called "Combinatorial Libraries Emphasizing and Reflecting Scored Sequences" (CLEARSS) that extends the conventional CPD approach [29]. CLEARSS begins with a list of fixed-backbone sequence designs. Possible degenerate libraries are sampled, given a list of allowed amino acids and a range of allowed library sizes, and are assessed using the ranked list of specific sequences. The overall score of a candidate library is the sum of scores for each design site, and the score for each design site is the sum of the Boltzmann weights of the sequences in the ranked list that contain a library-encoded amino acid. Finally, SwiftLib from the Kuhlman group uses dynamic programming to optimize the placement of multiple degenerate codons, obtaining very efficient libraries [32]. Notably, SwiftLib is presented as a highly accessible web server.

One limitation of such algorithms is the neglect of 2-body interactions. At the cost of significantly more difficult calculations (NP-hard optimization), this was addressed by Bailey-Kellogg and coworkers in the Optimization of Combinatorial Mutagenesis (OCoM) algorithm [30]. Another limitation is the use of a pre-calculated list of designs rather than a direct optimization in library space. It is not clear that pre-calculated lists of designs offer a balanced or thorough exploration of favorable sequence space; they may instead reflect a shallow exploration of sequence space, may feature diversity only at permissive sites, and could reflect systematic inaccuracies in the design potential. Treynor et al. performed combinatorial optimization in library space, but the 2-body potential between degenerate codons had significant drawbacks (amino acid:amino acid scores were obtained without rotamer optimization) [33]. The final relevant example is the Structure-based Optimization of Combinatorial Mutagenesis (SOCoM) algorithm reported in 2015 [14]. This last report is highly suggested reading as the SOCoM algorithm closely matches our independently developed approach.

There are several relevant figures of merit for candidate libraries. First, since these tools are intended to assist with actual experimental library design, the number of theoretical variants present in the encoded library is a key parameter. Theoretical library size is the starting point for selecting the number of clones that should be experimentally screened to obtain a target library coverage [34]. Another key parameter for a candidate library is the mean energy score (<E>) according to a design scoring function. Throughout this chapter the energy function is an all-atom Rosetta energy function [35]. Scores for protein structures are reported in Rosetta energy units (REU). One possible limitation of <E> is that the folding and functionality of protein sequences is not a graded response. Therefore, it may be more relevant to estimate the number of library members with $E < E_{cutoff}$, a threshold meant to flag library members at an elevated risk of not folding.

1.3 Degenerate Codon Libraries

A conventional approach to encode focused site diversity is to use a degenerate codon, in which the synthesized DNA primer consists of a mixture of nucleotides at particular positions. A single character analogous to the pure bases (A, T, C, G) represents each mixture of bases, with W→AT, S→CG, M→AC, K→GT, R→AG, Y→CT, B→CGT, D→AGT, H→ACT, V→ACG, and N→ACGT. A codon that includes at least one degenerate nucleotide is a degenerate codon. Common degenerate codons for site saturation mutagenesis are NNK and NNS, both of which encode all 20 amino acids (with varying codon and amino acid frequency). It is also important to note that 1/32nd of the codons that are physically realized from the NNK and NNS degenerate codons (assuming equimolar nucleotide mixtures) encode stop codons.

A key limitation of saturation mutagenesis is poor scaling to multiple residue targets, due to combinatorial explosion of the size of the resulting library. Fortunately, there are opportunities to improve; NNK is only one of many possible degenerate codons, the vast majority of which are underutilized. Since there are 15 possible nucleotide mixtures (*see* above) at each of the three positions making up a codon, there are 3375 legal degenerate codons. Ignoring codon usage considerations (organism codon preferences that are the usual target of codon optimization), there are 1482 degenerate codons that encode different ratios of amino acid (and stop codon) outcomes. To further simplify, degenerate codons that specify the same sets of amino acids (with varying amino acid probability) can be eliminated. Of these 840 degenerate codons, 115 can also be discarded since they encode sets of outcomes that are redundant with another degenerate codon except for the inclusion of stop codon outcomes. Thus, there are 725 degenerate codons that encode unique sense mixtures of amino acids. The specific computational challenge addressed by this chapter is to select which of these 725 options to pick for each site within a design problem (*see* **Note 1**).

Several groups have developed methods for site-specific libraries that rely on mixing primers rather than ordering standard degenerate oligonucleotides [32, 36–41]. By taking these alternate approaches, the precise set of desired amino acids can be encoded at each site. Can the computational design framework described here be useful in such scenarios? In theory, the method should apply equally well when selecting between arbitrary amino acid sets. The critical challenge is that the unconstrained library search space is much larger. Rather than the 725 mixtures of amino acids above, any combination of the 20 amino acids might be used. The number of possible amino acid sets is large enough ($2^{20} = 1{,}048{,}576$) that the current methods would likely be impractical due to memory limitations or combinatorial optimization performance limitations.

1.4 Regression and Energy Functions

Regression is a powerful tool to uncover the relationship between a dependent variable and one or more independent variables. In the current case, the dependent variable is the output value from a calculation (particularly a computationally expensive calculation) applied to a protein structure. Meanwhile, the independent variables correspond to the binary presence (1) or absence (0) of various mutually exclusive options. For example, in a protein design calculation, the task is to select exactly one amino acid at each design site. For a protein repacking problem, the task is to select exactly one sidechain rotamer position. Necessarily, the regression model only approximates the results from the more expensive calculation. The benefit is the dramatic increase in speed, since the predicted score for any discrete combination covered by the regression model can be computed nearly instantaneously [10]. Thus, if the regression model has sufficient accuracy, it can be used to effectively search enormous solution spaces.

Energy functions are used to evaluate structures and test them for plausibility. The Rosetta energy function is a well-known example, as are the energy functions employed by molecular dynamics simulations. Both of these are scoring functions, although there are important differences. Rosetta includes "knowledge-based" terms derived from protein structure statistics that are usually eschewed by the "force fields" that contain only physics-based, molecular mechanics terms. In either case, energy functions typically take the form of a sum of terms that approximate various interactions between atoms in a protein. The complexity of energy functions used for protein design is often immense. Many mathematical terms (e.g. electrostatic interactions, bond angles, solvation energy, etc.) may be combined in an effort to improve the accuracy of an energy function. Regression models can be used to approximate results obtained with these more expensive calculations.

1.5 Cluster Expansion

The use of regression to accelerate otherwise intractable protein calculations has been popularized in recent years by Grigoryan, Keating, and coworkers as cluster expansion [9, 10]. Cluster expansion is a regression-dependent method that was initially made to study alloys [42]. Cluster expansion techniques have now been used to generate useful approximations for a variety of protein-related problems. At heart, cluster expansion relies on regression to fit an expensive calculation (e.g. the stability of a protein evaluated via repacking calculations). The terms may be 1-body (e.g. is-residue-10-an-arginine), 2-body (e.g. are-both-arginine-10-and-glutamate-18-present), 3-body, or higher-order. Regression is used to determine the value of the terms. A key benefit is that the dependent variable, the expensive calculation, can be arbitrarily sophisticated.

Commonly, the expensive calculation includes combinatorial optimization. In the case of protein design, cluster expansion serves to "integrate out" the sidechain placement problem, providing a model that predicts the post-repack energy for any sequence. Given a model with only sequence variables, new design possibilities become feasible. For example, Grigoryan et al. used integer linear programming in conjunction with the cluster expansion model to directly incorporate negative design into the design of a family of coiled coils with orthogonal specificity [11]. In the current case (protein degenerate codon library design), a sequence-level model that predicts the energy of any sequence is converted into a library-level model that predicts the mean energy of any degenerate codon library (Fig. 1). A recent report from Verma et al. demonstrates an equivalent approach, direct optimization of degenerate codons via cluster expansion [14] (*see* **Note 2**).

One drawback of cluster expansion in particular, and regression in general, is the necessity of training the model with a large set of initial calculations; typical training sets for protein design

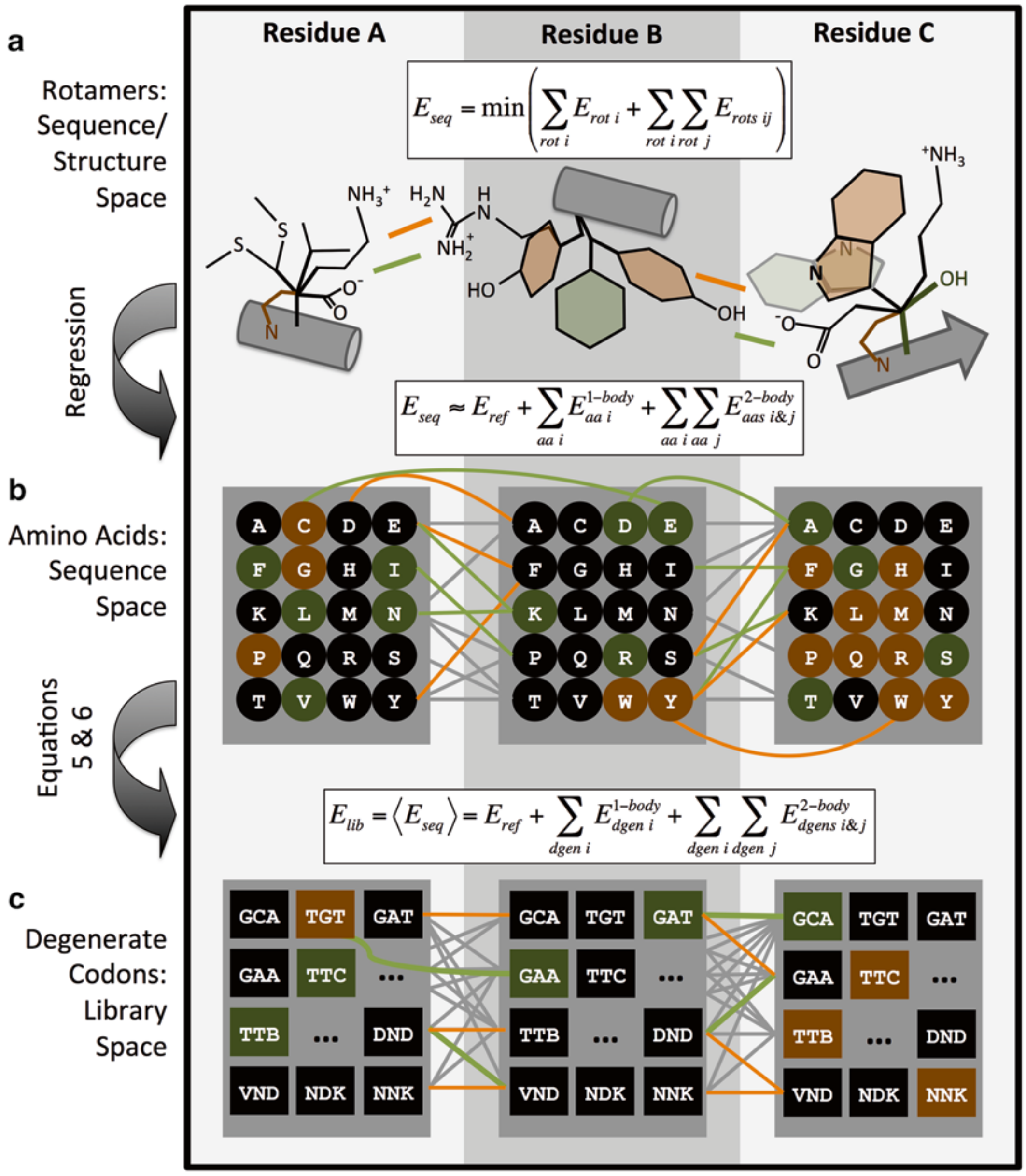
a
Rotamers: Sequence/ Structure Space
Regression
b
Amino Acids: Sequence Space
Equations 5 & 6
c
Degenerate Codons: Library Space
Residue A
Residue B
Residue C
$E_{seq} = \min\left(\sum_{rot\ i} E_{rot\ i} + \sum_{rot\ i}\sum_{rot\ j} E_{rots\ ij}\right)$
$E_{seq} \approx E_{ref} + \sum_{aa\ i} E^{1-body}_{aa\ i} + \sum_{aa\ i}\sum_{aa\ j} E^{2-body}_{aas\ i\&j}$
$E_{lib} = \langle E_{seq} \rangle = E_{ref} + \sum_{dgen\ i} E^{1-body}_{dgen\ i} + \sum_{dgen\ i}\sum_{dgen\ j} E^{2-body}_{dgens\ i\&j}$
A C D E F G H I K L M N P Q R S T V W Y
GCA TGT GAT GAA TTC ... TTB ... DND VND NDK NNK

Fig. 1 At each design site, the first layer (**a**) consists of a combined sequence/structure search space with discrete alternative positions for the sidechains (rotamers). Each rotamer gets a 1-body energy due to interactions with immobile groups and 2-body energy terms due to interactions with neighboring mobile groups. These might be particularly favorable (*green edges*) or unfavorable (*orange edges*). (**b**) Integrating out the structure degrees of freedom, we arrive at a regression model that only contains sequence variables. Favorable and unfavorable 1-body terms are represented with *green* or *red tint*, while 2-body terms are again represented as edges. Finally, by applying Eqs. 5 and 6, we can construct another energy graph (**c**) in which the numerous vertices correspond to degenerate codons. Depending on the constituent amino acids, the degenerate codons may be favorable (*green tint*) or unfavorable (*orange tint*). Edges may likewise carry favorable (*green*) or unfavorable (*orange*) effects

problems contain tens of thousands of calculations. The aggregate computational expense is significant but necessary. To save computational time, a training set of minimum feasible size is preferable, but large training sets are needed to avoid over-fitting large free parameter collections. Eventually, increasing the size of the training set will not lead to improved accuracy; at this point it has been saturated, and adding additional terms to the regression model is more likely to lead to an improvement.

Perhaps surprisingly, Apgar et al. found that a more expensive calculation (with a flexible backbone) was easier to approximate than a less expensive calculation (rigid backbone) [12]. It was suggested that allowing the backbone to move minimized steric clashes between individual residues. Removing steric clashes is helpful because such interactions are unlikely to be physically realistic, and because the large amplitude of such interactions can be difficult to fit.

Ng and Snow found that lower-order terms were sufficient for the prediction of multi-body energy function scores [43]. Specifically, the AMOEBA polarizable energy function [44], which is not pairwise decomposable, was approximated for combinatorial sidechain optimization. Lower-order (1-body, 2-body, and 3-body) terms were shown to be sufficient to accurately approximate the multi-body polarization effects. In addition, sets of lower-order terms could be used to predict which higher-order terms are relevant. If the 2-body terms for amino acids at three positions had significant magnitude, it was worth attempting to add a third-order term for those three amino acids. Snow and Ng's work revealed that one could filter out (i.e. ignore) almost 80 % of 3-body terms and thereby reduce the complexity of the regression with this simple check.

The generality of the regression/cluster expansion approach is a key feature. Many hard problems in CPD benefit from an accurate model that predicts energy directly from the sequence. The current chapter describes open-source, permissively licensed software for expanding the combinatorial optimization approach to problems that may not be pairwise decomposable. To take advantage of this flexibility, Python scripts are presented for computing regression-based approximate models with the robust combinatorial optimization capacity of the open source SHARPEN software platform [45].

2 Methods

2.1 Overview

In broad strokes, the steps to take to apply the regression tools are the same regardless of the exact goal. First, an initial set of discrete combinations is "instantiated," a process that varies depending on the problem but always includes an assessment or scoring of the combination in question. For the current case studies, the discrete combination is the protein sequence, and the instantiation consists of a combinatorial optimization of sidechain rotamer positions. The set of instantiated combinations is divided into two subsets: one to train the regression model and another to test the resulting approximation. The resulting trained regression model provides a rapid approximation to the more expensive instantiation operation.

The SHARPEN package [45] provides convenient data structures to store and apply regression approximations. Specifically, EnergyGraphs efficiently store 1-body and 2-body terms in a conventional graph structure consisting of nodes and edges (a thin wrapper around the underlying Boost Graph type). More unusually, SHARPEN also provides EnergyHyperGraph data structures that can also accommodate higher-order terms such as 3-body, 4-body, or N-body effects. Either EnergyGraphs or EnergyHyperGraphs can be used with a variety of independent combinatorial optimization routines for identifying favorable combinations. Therefore, it is easy to efficiently identify discrete combinations, "targets," that are predicted to minimize the instantiated score according to the current approximation.

The value of the entire scheme is predicated on the utility of the approximation to allow the combinatorial search process to more rapidly explore enormous swaths of a combinatorial search space, and to do so with enough accuracy to discover favorable combinations. Given the astronomical search size of typical combinatorial problems, and the rapidity of search methods using the regression approximation, accelerating the sampling is likely assured. The more challenging aspect is ensuring that the regression-based approximation is sufficiently accurate. Fortunately, this chapter illustrates that 2-body regression models appear to be largely sufficient to approximate the favorable portions of the combinatorial search space, and that such approximations can be used to facilitate the optimization of degenerate codons directly in library space.

The routines described below are implemented in a set of python scripts that use methods provided by a python module, **dgen_design**. These tools use the open source SHARPEN software, and are therefore provided via the www.sharp-n.org website wiki. Other useful scripts for practical protein design calculation, described by Johnson et al. [25], are also hosted on this site.

2.2 Instantiation: Combinatorial Optimization of Sidechain Positions

1. The only requirement for an instantiation method is that it accepts a combination and produces a score. Any algorithm that can be applied to a candidate protein and produces a number could be an instantiation method. Instantiation for this work involves combinatorial optimization of the sidechains ("repacking") for a particular sequence variant to minimize the model score according to an all-atom Rosetta energy function [35]. The outcome is the score E in Rosetta energy units

(REU). Structures with lower Rosetta energy scores are more plausible protein conformations.

From the standards of CPD, the case studies presented herein are small problems (Table 1). The FasterPacker combinatorial optimization object mimics the "singles" routine from the Desmet and Lasters FASTER algorithm [4]. The FasterPacker works somewhat like a traditional Monte Carlo trajectory, except that the moves that are accepted or rejected are "batch" moves. Candidate batch moves are generated by temporarily fixing a perturbing rotamer change and then sequentially relaxing interacting sidechains to their low energy rotamer.

2. FasterPacker typically yields optimal or near optimal solutions for problems of this size. To demonstrate, 600 sequences for case B.1 (*see* below) were solved to optimality using the mixed integer linear programming program CPLEX [46] via a CplexPacker wrapper provided by SHARPEN. Because these problems are reasonably small, the CplexPacker is able to identify the global minimum energy combination (GMEC) relatively quickly (an average time of 3.7 s). In 552 of 600 cases FasterPacker found a solution within 1E-6 REU of the GMEC, but did so in an average of only 0.14 s. Notably, CplexPacker was used to optimize the sidechain rotamer positions of the initial protein model prior to any other calculations.

Table 1
The reported best low-energy testset rmsd values ($rmsd_{LET}$) correspond to the lowest value encountered for varying training set sizes. Where applicable, the best-case exponential weight (τ) and regularization parameter (k) are also noted

	Site	Active design sites	Other mobile sites	Seq space size	Lib space size	Seq/str search size	Figures	Best LET rmsd	τ	k
A.1	Core	5, 30, 43	52, 54	8000	3.8E8	1.5E11	4,6,7,8	1.8	–	1e–7
A.2						4.9E7	4,6,13	1.7	–	0.01
A.3							5,6	1.5	125	1e–6
B.1	Core	5, 30, 43, 52, 54	3, 7, 16, 45	3.2E6	2.0E14	3.9E20	9	5.0	–	0.1
B.2								4.1	50	1e–7
B.3						2.5E13	10,13	14.4	–	1
B.4							10	8.4	75	1
C.1	Surf	2, 4, 6, 8, 13, 15, 17, 19, 42, 44, 46, 48, 49, 51, 53, 55	None	6.6E20	5.8E45	1.8E35	11,13, 14	9.9	–	10
C.2						6.5E45	12	4.8	–	1

3. A pool of random combinations is instantiated via FasterPacker at the outset of the campaign. This pool serves as the source of training and test combinations. A training batch is used to kick off the regression, while the test batch (all other members of the initial pool) is held in reserve to quantify regression model quality.

For the case study problems here, combinatorial optimization is quite rapid since there are a limited number of mobile residues (Table 1). For example, the 5-site library (case B), with the default (non-minimal) rotamer generation scheme has a structure-sequence search space size of 3.9×10^{20}. Only 20 min are required to instantiate 50,000 random sequences using a 2.8 GHz Intel Core i7 CPU.

2.3 Term Selection

1. The ability of regression to produce accurate approximations is predicated on the ability of terms to stand in for more complicated processes. It is desirable to be selective when adding terms, since adding an excessive number of terms will result in overfitting. To recapitulate most physical problems, it is necessary to include at least 1-body and 2-body terms. A free constant (i.e. a 0-order term) can also be helpful, allowing the remaining parameters to adopt smaller values without degrading the overall fit. Alternately, to shrink the absolute value of the free constant, a reference energy (*Eref*) can be subtracted from each element within the instantiated score vector (Υ).
2. In the particular case of approximating the energy of protein sequences, our general recommendation is to consider the wild-type (WT) or initial protein sequence as the reference state (with $E = Eref$). Then, each individual mutation at an active design site gets a 1-body parameter. WT amino acids at the design positions do not get parameters, as their contribution is subsumed within the reference state. Similarly, only interactions between two mutations serve as 2-body parameters, since a WT:mutant pair is already accounted for in the 1-body parameter for that mutant. Ideally, regression will drive the free constant parameter toward the score of the reference state. All of the problems described below include a free constant, all 1-body terms, and all possible 2-body terms (Fig. 1). For larger problems, it could become useful to skip 2-body terms that are not likely to correspond to physical effects. For example, one could require physical proximity or more direct evidence of energetic coupling between the particular sites before adding terms.
3. Similarly, it could also be useful to identify higher-order terms (i.e. 3-body terms) to improve the accuracy of the model in recapitulating low-energy combinations. A tricky aspect to this is that sizable 3-body effects for protein design can be "frustration" effects in which three pairs of amino acids can each coexist

nicely, but the combination of all three induces an unavoidable steric clash. Modeling this effect requires a large 3-body term, which breaks the typical approximation paradigm that higher-order effects will have lower magnitude than lower-order effects.

4. A more sophisticated (and lengthy) approach to term selection was described by Hahn et al. who developed an iterative feature selection scheme with rapid cross-validation [13]. Candidate terms are individually considered and included if they make a statistically significant improvement. For the degenerate codon design problem described here, one can avoid using 3-body terms and lengthy term selection procedures due to the sufficient accuracy of the 2-body models and the technical feasibility of modeling all possible 2-body terms.

2.4 Solving Large Regularized Regression Problems

1. After the training batch is instantiated, and fitting terms are selected, regression can proceed. The regression model will ascribe values to the fitting terms so that summing the appropriate terms can approximate any combination. For the case study problems here, there are thousands of one-body and two-body terms, and thousands of training set members.

2. Each of the training set members will have a relatively small number of applicable terms, depending on which amino acids (potential 1-body terms) and pairs of amino acids (potential 2-body terms) are present at the variable sites. To tackle the resulting large sparse regression problems, our approach relies on two solvers that work with sparse matrices and are convenient for use from Python. Specifically, the CVXopt software package [47] provides a sparse matrix structure, an interface to the Cholesky factorization routines of the CHOLMOD package [48], and functions for solving sparse sets of linear equations. Alternately, one can use the LSMR package [49], which is integrated into scipy [50]. For the following code snippets, Y is the vector of instantiation scores, k is a regularization parameter (*see* below, Eq. 1), and *spX* is a sparse matrix that encodes which terms apply to which training set combinations.

```
import scipy.sparse.linalg
results=scipy.sparse.linalg.lsmr(spX, Y, damp=k)
or
import cvxopt
from cvxopt import spmatrix, spdiag, cholmod
B=spX.T * cvxopt.matrix(Y)
XT_X=spX.T * spX
ridge=k * cvxopt.spdiag([1] * len(B))
cvxopt.cholmod.linsolve(XT_X+ridge, B)
```

3. Given the large number of fitting parameters that arise when 2-body or 3-body terms are included, overfitting is a serious concern. To combat the tendency for overfitting, use regularized regression. In both code snippets above the core calcula-

tion consists of regularized regression, also known as ridge regression (Eq. 1) or Tikhonov regression [51]. This technique penalizes terms that deviate from zero. The regularization parameter, *k*, serves to restrain the magnitude of the fitting parameters, *β*. The matrix *X* specifies which fitting terms contribute to each combination, with the vector *Y* holding instantiated scores and *I* as the diagonal identity matrix.

$$\left(X^{T}X + kI\right)\dagger = X^{T}Y \tag{1}$$

Ridge regression can be useful to suppress overfitting. It is important, however, to setup the problem so that the value of the terms should indeed be small numbers.

2.5 Weighted Regression

1. Weighting is a useful optional strategy to increase the accuracy of the regression model for some of the combinations. Typically, the performance of the approximation is much more important for favorable combinations than unfavorable combinations. It is recommended to sacrifice the *overall* fit in favor of higher accuracy for the *favorable* combinations. A matrix W has weights along the diagonal, w_i, that are selected using the following scheme intended to resemble Boltzmann weighting. The adjustable parameter *τ* sets the energy scale that defines the favorable sequences of interest.

$$X^{T}WX\dagger = X^{T}WY \tag{2}$$

$$w_i = \min\left[\mathrm{e}^{-\frac{(y_i - \min(y))}{\tau}}, 0.02\right] \tag{3}$$

2. If the exponential weighting parameter $\tau = 100$, then training set combinations that are 10, 25, 50, 100, and 200 REU less favorable than the minimum REU combination in the training set will have weights of 0.90, 0.78, 0.61, 0.37, and 0.14. Equation 3 assumes that the more important combinations have the lower scores. If necessary, the sign of the scores can be flipped. The use of a minimum weight (0.02 above) ensures that the regression model cannot entirely neglect high-energy combinations.

2.6 Quantifying Regression Model Performance

1. To quantify the performance of a regression model, one can compute the root mean square deviation (rmsd) between the predicted E scores for the testset with the actual repacked E scores. However, also consider the possibility that the regression model predictions may have a systematic bias (e.g. a slope of 1.5 or a non-zero intercept). If such a bias is consistent, it could be corrected by fitting a line. Therefore, before computing rmsd one should correct systematic deviations using the scipy.stats.linregress function to compute the slope and intercept.

2. It is not recommended to equally favor all combinations. The explicit goal is to maximize accuracy for the more favorable, low-energy combinations. Rough prediction of high-energy combinations is sufficient; clashes need not be precisely quantified if they can be avoided. Accordingly, one might quantify accuracy for the favorable members of the test set, with $E_i < \min(E) + 100$ REU. Hereafter, this figure of merit will be termed rmsd_{LET}, the rmsd for the low-energy testset. One may also use rmsd to quantify the extent to which library <E> predictions match directly sampled <E> values.

2.7 Using the Approximation to Select Targets

1. Once a regression model has been trained, it can be used to efficiently identify combinations that are predicted to be favorable upon instantiation. Such combinations are termed "targets." If only 1-body terms are present in the model, optimal targets are trivially easy to identify; one need only select the best score for each mutually exclusive choice. More generally, a low-scoring target combination is found using combinatorial optimization routines, typically the FasterPacker described above, or the SimulatedAnnealingPacker. A SimulatedAnnealingPacker implements a Monte Carlo trajectory over combinations with a gradually reducing temperature value.
2. These combinatorial optimization methods are generally intended to find individual favorable combinations, possibly the GMEC. When target diversity is critical, the output combination from the SimulatedAnnealingPacker or FasterPacker can serve as the initial combination for subsequent MonteCarloPacker sampling. To ensure a diverse pool of target combinations, one can generate multiple Monte Carlo trajectories at escalating temperature. Temperature is increased in repeated Monte Carlo runs until a minimum number of distinct combinations are found. This method of target selection is still effective if higher-order terms are present in the regression model. Also, the use of an escalating temperature should render this protocol somewhat robust when applied to different problems with varying intrinsic energy scales.
3. A related strategy can come into play at the outset of a learning process. In the case of building a training set to use regression to approximate the AMOEBA energy function (Ng, 2011), the model was trained using rotamer combinations that were highly diverse, but not the purely random combinations that usually result in van der Waal clashes. To do so, an initial approximate energy model that included only strong van der Waal clashes was built. Then, to generate a maximally diverse pool of combinations, excluding unrealistic high-energy combinations, Ng ran a series of Monte Carlo trajectories with the temperature set to zero. These trajectories began from random

combinations and executed a downhill walk, thereby preserving maximal diversity while attempting to avoid unphysical clashes. This allowed the regression process to "learn" about more interesting effects than the steric clashes.

2.8 Calculating the Properties of Degenerate Codon Libraries

1. For small degenerate codon libraries (e.g. case A with three design sites), it is feasible to enumerate the predicted E values for each of the encoded sequences. Then, the expectation value of the Rosetta energy for any library can be readily calculated by computing the mean energy, <E>, of the constituent sequences. However, for library design problems with more sites, precise calculation of <E> can be overly time-consuming. Therefore, our code instead estimates <E> by sampling the value of n random combinations drawn from the library. One can compute the standard error of the mean ($\sigma\langle_{\mathrm{E}}\rangle$) given the standard deviation of the E values in the sample (σ), using the finite sample correction for libraries with N members:

$$\sigma_{\mathrm{E}} = \frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}} \tag{4}$$

2. Relatively modest samples ($n=400$) are sufficient to assess the correlation between the predicted library <E> from the regression model and the sampled <E>. To estimate the number of library combinations with E below a threshold, compute the fraction below the threshold for the sample and multiply by the library size. The library size, N_{lib}, refers to the theoretical number of distinct sequences and is calculated as the geometric product of the number of amino acids Naa^{site} encoded at each design site: $N_{\mathrm{lib}} = \prod_{\text{design sites}} N_{aa}^{\mathrm{site}}$.

2.9 Combinatorial Design in Degenerate Codon Library Space

1. The technical details above cover the preparation, tuning, and validation of regression models that map specific protein sequences to predicted post-instantiation Rosetta energy scores. However, a larger goal for this chapter is to demonstrate how such models can be used to efficiently select degenerate codon libraries. Specifically, the goal is to execute a search for favorable degenerate codon libraries directly in "library space." To enable library design via the various combinatorial optimization algorithms provided by SHARPEN (e.g. the Packers mentioned above), one need only prepare an EnergyGraph or EnergyHyperGraph in which the nodes no longer correspond to mutually exclusive amino acid choices, but instead correspond to mutually exclusive degenerate codon choices (Fig. 1).
2. The regression models described above make this possible. In the notation below, $E_i^{1-\mathrm{body}}$ is the 1-body regression term for amino acid outcome i, and $E_{ij}^{2-\mathrm{body}}$ is the 2-body regression term for the simultaneous selection of amino acids i and j. First, com-

pute 1-body terms for each degenerate codon by computing the expectation 1-body term for the constituent sense amino acids according to the regression model (*see* **Note 3**). The relative frequency of the constituent amino acids, *pi*, can serve as weights to compute the expectation value. Alternately, *pi* values can be set to model equally probable amino acid outcomes. The latter approach is adopted here under the assumption that all variants within the library might be isolated and characterized (ignoring the different frequencies of encountering these variants).

$$E_{\text{dgen A}}^{\text{1-body}} = \sum_{\text{A } aa\, i} p_i \cdot E_i^{\text{1-body}} \tag{5}$$

3. Similarly, compute the expectation value for the 2-body interaction between two degenerate codons (A and B):

$$E_{\text{dgens A and B}}^{\text{2-body}} = \sum_{\text{A } aa\, i} \sum_{\text{B } aa\, j} p_i \cdot p_j \cdot E_{ij}^{\text{2-body}} \tag{6}$$

4. For a closer look at the calculation of an EnergyGraph that embodies the library design landscape, *see* the *M_score_dgen_codon_sets.py* script.

2.10 Sampling Diverse Libraries via Combinatorial Optimization

1. Library size is a key consideration when it is time to select a library for experimental testing. However, the feasible experimental library size is rarely in practice a strict cutoff. Instead, it is valuable to illustrate what candidate libraries look like as a function of library size before making final decisions.

All other factors being equal, use libraries with a lower <E>, since those libraries are the most likely to be highly folded, stable, and functional. However, the global minimum energy library (GMEL) will have exactly one sequence and that sequence will correspond to the GMEC. This is true since each individual amino acid is an option within the 725 amino acid sets that can be encoded. If libraries are sampled thoroughly, the libraries with minimal <E> are going to contain relatively few sequences. The more useful task is to identify libraries with minimal <E> for every library size. This will allow the protein engineers involved to select the library size that offers the best <E> yet fits within the assay screening budget. Therefore, when performing combinatorial optimization directly in library space, an explicit bias favoring larger libraries may help to sample diverse library options.

2. To sample larger libraries, implement a bias favoring degenerate codons that encode more amino acids. First, compute log(Naa^{site}) for each degenerate codon. Then, when assessing candidate degenerate codon combinations, the total library size is $e^{\left[\sum_{\text{design sites}} \log\left(N_{aa}^{\text{site}}\right)\right]}$. Each candidate degenerate codon choice contributes additively to the predicted library E via its 1-body

and higher-order regression terms, and contributes additively to the library size (N_{lib}) via the log(Naa^{site}). Iteratively perform combinatorial optimization according to the energy model, but in each round r_i increment a cumulative 1-body bias favoring degenerate codons that encode more amino acids: $r_i \cdot \varepsilon \cdot \log\left(N_{aa}^{\text{site}}\right)$, where ϵ is a small weight factor. For a closer look at this iterative library sampling scheme, *see* the *P_iter-lib_sample.py* script.

3. Manually adjust ϵ so that libraries of the largest interesting size are sampled by the end of 100 rounds of combinatorial optimization. The largest interesting library size is problem-dependent. For three design sites, full NNK saturation may be worth considering. Given more numerous design sites, the maximum interesting library size will likely be limited by the screening capacity. Even in vitro methods such as mRNA display or ribosome display have limits (e.g. 10^{14} variants) [52].

2.11 Selecting Advantageous Degenerate Codon Libraries via Pareto Analysis

1. Seek to identify libraries that are Pareto optimal [53] for minimal <E> and large library size. In other words, if candidate library 1 has a higher <E> and a smaller size than candidate library 2, then candidate library 1 can be discarded from consideration. To accelerate this process, our code uses a divide and conquer approach. *See* the *Q_calc_pareto_stats.py* script for more details.

Due to threshold protein stability effects, a library with 90 % favorable sequences and 10 % very unfavorable sequences may be preferable to a library consisting entirely of mediocre sequences. Unfortunately, the <E> could be lower for the latter library. Therefore, a preferable Pareto analysis scheme identifies libraries that have the greatest (predicted) number of sequences with scores below a threshold, while otherwise having the smallest total library size. Generally, this threshold should be set to a value such that combinations exceeding the threshold would be at risk of not being functional.

2. After either Pareto analysis is complete, the remaining set of libraries (the Pareto front) includes only the libraries that are most worthy of consideration. Inspection of the resulting plots should help when weighing the tradeoffs between selecting small libraries with favorable energy statistics and larger libraries with less favorable statistics.

3 Example Tests and Results

3.1 Model Design Problems

This section describes results for several illustrative degenerate codon design problems (Fig. 2) using protein G (pdb entry 1pgb). Table 1 defines which amino acids are design positions and which other amino acids that are allowed to move.

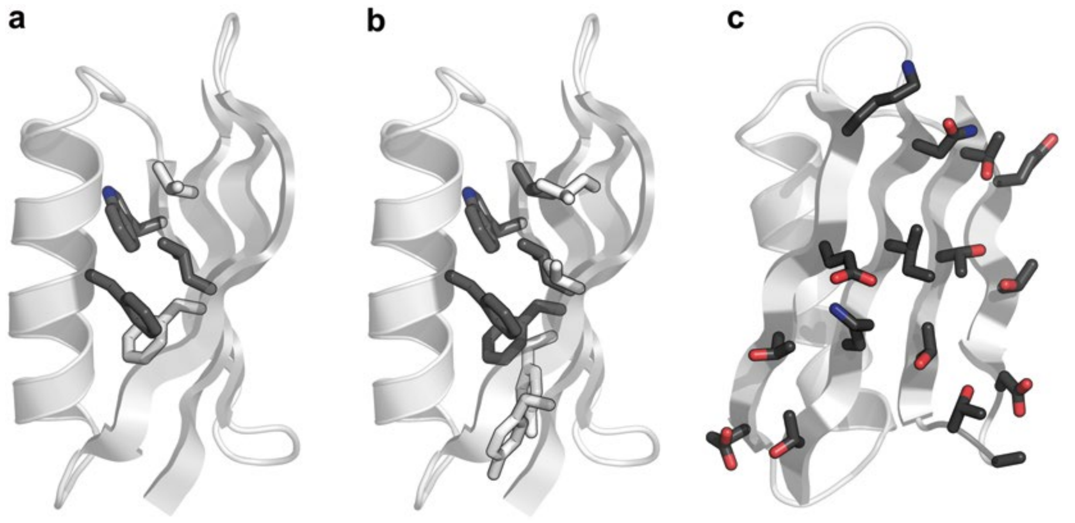

Fig. 2 Case study design problems. Design position (*gray*) and mobile sidechains (*white*) are shown in sticks. (**a**) Case A has three design sites in the hydrophobic core. (**b**) Case B has five design sites in the hydrophobic core. (**c**) Case C has 16 design sites on the exposed surface of the beta sheet

1. **Case A** is intended to provide the smallest possible interesting problem (Fig. 2a). In this case, by limiting the number of design positions to 3, it is possible to optimize and score each of the 8000 possible sequences via sidechain optimization. Note, however, that even very rapid calculations become time-consuming when applied to 381 million candidate degenerate codon libraries (725^3).
2. **Case B** is intended to provide an example of a realistic use scenario for these tools (Fig. 2b). For many experimental assays, it would be impractical to experimentally screen a site saturation library at 5 design positions. However, by using tailored degenerate codons it may be possible to obtain a library that is small enough to screen, and will consist of a higher fraction of favorable sequences. Thus, by degenerate codon design, one could make the most of the available screening capacity (e.g. ten 96-well plates). For example, consider a size constrained hydrophobic site. Rather than using all 20 amino acids, the degenerate codon "VTM" would provide just Ile, Leu, and Val, and would help reduce combinatorial explosion of the library size.
3. **Case C** is intended to demonstrate performance when applying the approach to a larger design problem (*see* **Note 4**). Case C constitutes the redesign of an entire surface face of a beta sheet (Fig. 2c). Saturation mutagenesis of such a 16-site library is out of reach for experimental screening, but tailored codons might be used to identify favorable libraries small enough to be screened for binding properties via a high-throughput approach (e.g. fluorescence-activated cell sorting).

4. All three of these cases are suitable for conventional CPD. The case A GMEC (wild-type Leu5, Phe30, and Trp43) was found using SimulatedAnnealingPacker, FasterPacker, or CplexPacker in 0.6, 13, or 87 s respectively. For case B.1, design results in a double mutation W43T, V54I. The GMEC was found using SimulatedAnnealingPacker, FasterPacker, or CplexPacker in 1.8, 104, or 795 s respectively. Finally, FasterPacker and CplexPacker found the GMEC for case C.1, which had 15 surface mutations (T2R, K4E, I6E, N8R, K13E, E15R, T17Y, E19W, E42L, T44R, A48N, T49R, T51R, T53I, and T55I). In this last case, the FasterPacker and CplexPacker required 5 and 803 s, respectively.
5. To illustrate the performance determinants for the presented methods, variant calculations were performed (Table 1) to assess the effects of rotamer density and the use of weighting. Specifically, for cases A.2, A.3, B.3, B.4, and C.1 rotamers were reduced to base Dunbrack rotamer options [54]. For these cases, the sequence/structure search space size was reduced (Table 1) and combinatorial sidechain optimization was more rapid. Cases A.3, B.2, and B.4 use weighted regression. Table 1 indicates which subsequent figures apply to each case and highlights the best-case prediction accuracy (rmsd_{LET}).

As described above, the first step for each of these model design problems is to perform thousands of combinatorial sidechain optimization calculations (instantiation) for random sequences. For case A, all 8000 variant sequences were instantiated. For cases B and C, 50,000 and 80,000 combinations were instantiated, respectively. For all three cases, a large fraction of the sequence space achieves a low score when optimized via FasterPacker. As expected, the cases with more generous rotamer provisioning (Fig. 3abc) reach lower energy values.

The score for the wild-type sequence with fully optimized sidechain rotamers is −112.5 REU. For case A.1 the mean (median) E is −70.9 (−99.6) REU. For case B.1 the mean (median) E is −75.8 (−104.6) REU. For case C.2 the mean (median) E is -58.7 (−93.0) REU. The median values are lower than the mean values due to the outsized influence of high-energy sequences on the mean. Given these values, sequences with a predicted E above −105 REU were flagged as having an elevated risk of being unfolded.

3.2 Case A: High Accuracy Approximation of a 3-Site Library

1. Despite the close physical interaction of the design site residues for case A, it was possible to very accurately fit the post-repacking energy of the 8000 possible sequences via regression (Fig. 4a–c). There are 1141 fitting parameters in this case, consisting of 1 free constant, 57 one-body terms (the 19 possible mutations for each of the three sites), and 1083 two-body terms (double mutations). If all 8000 sequences are fit, the rmsd is only 1.9 REU. A better test, however, is to train the regression

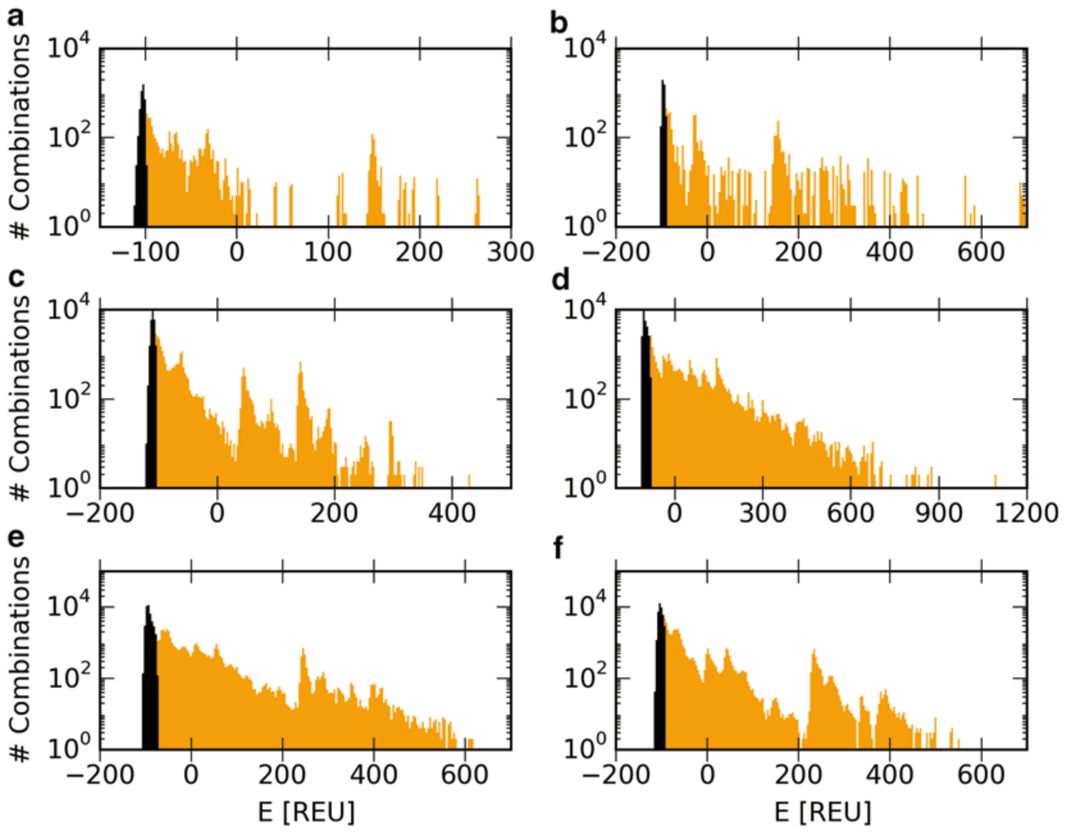

Fig. 3 Instantiated scores for random combinations. *Black* (*orange*) *bars* are combinations with E lower (higher) than the median. (**a**) All 8000 sequences for case A.1. (**b**) All 8000 sequences for case A.2 (minimal rotamers). (**c**) 50,000 random sequences for case B.1. (**d**) 50,000 random sequences for case B.3 (minimal rotamers). (**e**) 80,000 random sequences for case C.2. (**f**) 80,000 random sequences for case C.1 (minimal rotamers)

model using portions of the 8000-sequence pool and to assess the quality of the resulting approximate energy model using the remaining sequences as a test set. To illustrate the effect of training set size and regularization parameter, Fig. 4a shows how approximation accuracy depends on these parameters.

2. For case A.1, regularization was not critical. Scanning the training set size and the regularization parameter (Fig. 4a), the best rmsd$_{\text{LET}}$ was an impressively low 1.8 REU. The best performance came when using a 7500-member training set with the smallest test regularization parameter (k = 1E−7). Running regression without regularization produced the same results.

Surprisingly, reducing the number of rotamers (case A.2) does not reduce the performance of the regression model. Instead, rmsd$_{\text{LET}}$ actually decreased from 1.8 to 1.7 REU. One difference between case A.1 and case A.2 comes for small training sets (approximately 1000 combinations) and low regularization penalty (k < 1E−3). The slight shoulder in the case A.1 parameter scan

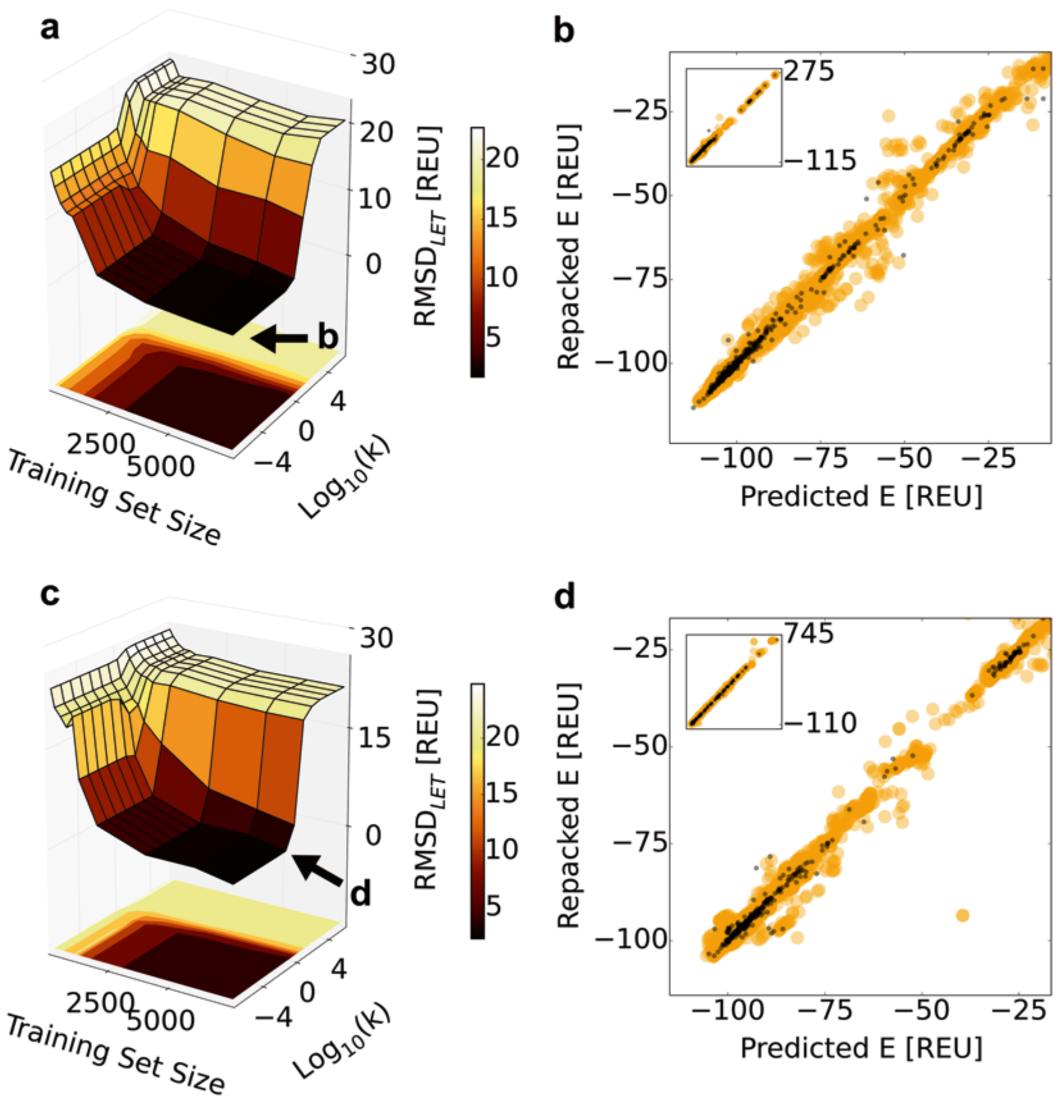

Fig. 4 Case A.1 and A.2 approximation performance. Training set combinations are partial transparent orange points while test set combinations are black points. (**a**) rmsd_{LET} versus training set size and the regularization parameter for **case A.1**. The best performance from this scan is shown in (**b**), where a random training set (7500 combinations) was used to fit 1141 parameters with regularization ($k = 1\text{e}-07$) resulting in training set recapitulation (rmsd = 1.8). Performance for low-energy combinations was excellent ($\text{rmsd}_{\text{LET}} = 1.8$ REU) for the 468 test set combinations within 100 REU of the minimum test set combination (–113.1 REU). The entire test set (500 combinations) was predicted with rmsd = 3.0 REU (inset). (**c**) rmsd_{LET} versus training set size and the regularization parameter for **case A.2** (minimal rotamers). (**d**) A random training set (7500 combinations) was used to fit 1141 parameters with regularization ($k = 0.01$) resulting in training set recapitulation (rmsd = 4.2 REU). Performance for low-energy combinations was excellent ($\text{rmsd}_{\text{LET}} = 1.7$ REU) for the 390 test set combinations within 100 REU of the minimum test set combination (–103.9 REU). The entire test set (500 combinations) was predicted with rmsd = 2.9 REU (inset)

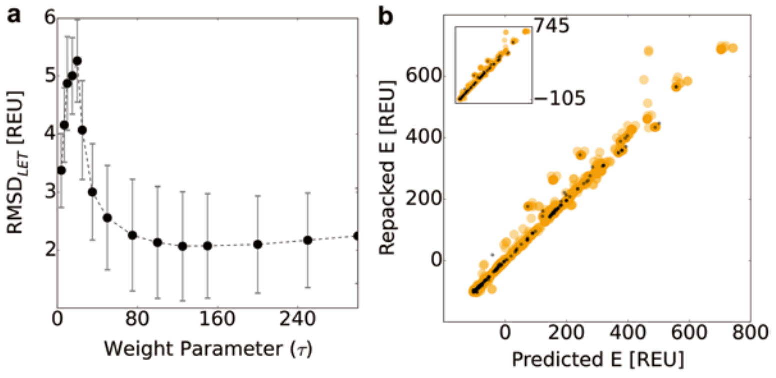

Fig. 5 Case A.3. (a) Scanning the expweight parameter (τ in Eq. 3). *Error bars* reflect the standard deviation from 20 trials with random 7500-member training sets. (**b**) With $\tau = 125$, a random training set (7500 combinations) was used to fit 1141 parameters with regularization ($k = 1e-06$) resulting in training set recapitulation (rmsd = 11.0). Performance for favorable test set combinations was good (**rmsd$_{LET}$ = 1.5 REU**) for the 410 combinations within 100 REU of the minimum test set combination (–103.7 REU). The entire test set (500 combinations) was predicted with rmsd = 8.7 REU (inset)

surface (Fig. 4a) becomes a distinct peak for case A.2 (Fig. 4c). This peak represents a counterintuitive result; *decreased* prediction performance for a *larger* training set. This result will be discussed below in the Overfitting Trends section.

3. Case A.3 attempts to improve the case A.2 performance with weighting. Keeping the training set and regularization parameter fixed (7500 training set members and $k = 1E{-}6$), the exponential weighting parameter τ was varied to determine which value gave the lowest rmsd$_{LET}$ (Fig. 5a). $\tau = 125$ was most effective (Fig. 5a). Compared to the non-weighted case A.2 (Fig. 4d), Fig. 5b demonstrates slightly improved rmsd$_{LET}$ ($1.7 \rightarrow 1.5$ REU), with a significant concomitant sacrifice of global fit rmsd ($2.9 \rightarrow 8.7$ REU).

3.3 Predicting <E> for Case A Libraries

1. The regression models described above predict the instantiated Rosetta energy for any sequence within the 8000-sequence search space. For each of the cases above, 1000 random degenerate codon libraries were selected. For each degenerate codon library, the <E> was computed using the pre-calculated energies for constituent sequences.

2. The library <E> predictions are quite accurate (Fig. 6), with <E> prediction rmsd values lower than the rmsd for the prediction of E for individual sequences. The rotamer-rich case A.1 accuracy was good globally (rmsd = 0.38 REU) and for the 559 libraries with predicted <E> within 30 REU of the –105.5 REU minimum (rmsd = 0.37 REU). The case A.2 accuracy was comparable (0.5 REU globally, 0.35 REU for the 256 libraries with predicted <E> within 30 REU of the –98.7 REU mini-

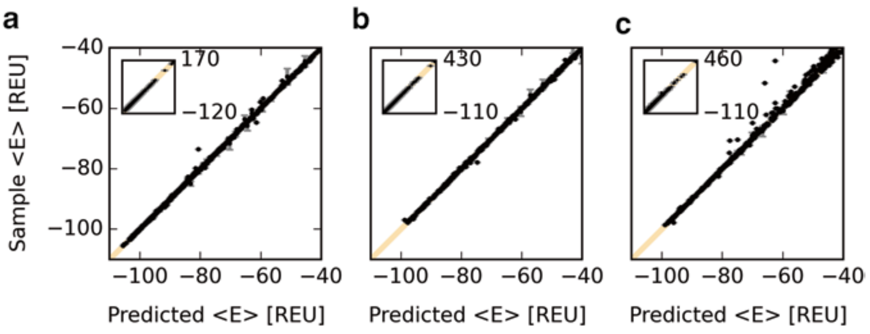

Fig. 6 Predicted library <E> versus instantiated sample <E>. *Vertical error bars* reflect $\sigma\langle E\rangle$ (Eq. 4). An identity line is orange. Global (or best 30 REU) rmsd values for (**a**) Case A.1, (**b**) Case A.2, and (**c**) Case A.3 were 0.38 (0.37), 0.5 (0.35), and 2.6 (0.73) REU respectively

mum). Finally, weighting (case A.3) degraded the <E> prediction, with 2.6 REU rmsd for the global library <E> prediction, and 0.73 REU for the 240 libraries within 30 REU of the –99.0 REU minimum. Despite the counterproductive effect of weighting, these levels of precision for library <E> prediction performance are encouraging. It was particularly gratifying that the minimal rotamer case A.2 performed so well, since all 8000 sequences can be instantiated in less than 4 s in this case.

3.4 Sampling Case A Libraries

1. Random sampling is neither a systematic nor a satisfying solution for efficiently identifying libraries that are maximally appealing for experimental testing. A systematic approach is preferable. Since 1-body and 2-body scoring terms for the degenerate codons are stored in a SHARPEN EnergyGraph, a variety of combinatorial optimization algorithms are readily available to assist with sampling.

For case A, the total library search space of 381 million is small enough for enumeration, albeit via a relatively expensive calculation (approximately 26 min and 8 Gb memory). Therefore, for a 3-site library the BruteForcePacker object from SHARPEN is feasible. The BruteForcePacker can be configured to retain a ranked queue of the best combinations encountered. For case A, a large priority queue was needed to retain the 4.6E5 libraries with a predicted <E> < –105 REU (Fig. 7). For a closer look at this enumeration-based sampling scheme, *see* the *sample_libs*_via_*enum.py* script.

2. For larger libraries, enumeration is not going to be a practical option. Instead, it would be better to identify the potentially numerous low-<E> libraries with an inexpensive calculation. Another example script launches and pools parallel Monte Carlo trajectories to rapidly collect a set of unique libraries. Almost 4E5 libraries predicted to be low energy were collected in less than 5 min (Fig. 7b). For a closer look at this Monte Carlo sampling scheme, *see* the *sample_libs*_via_*MC.py* script.

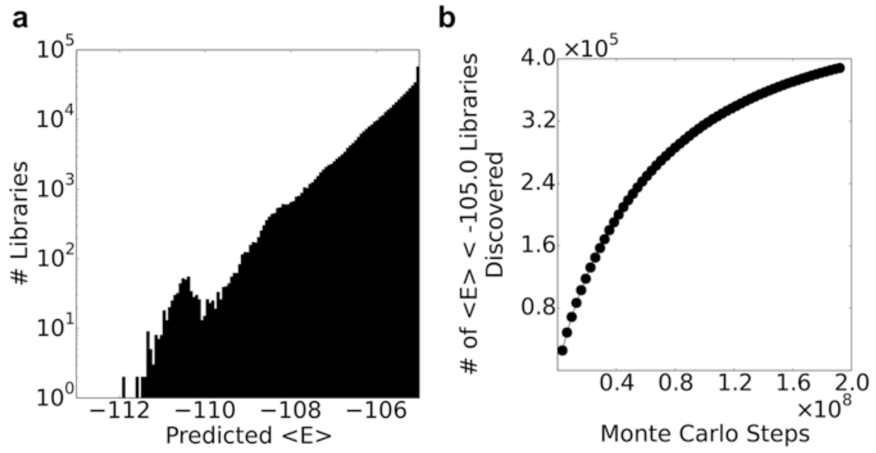

Fig. 7 High density of low <E>-prediction libraries for Case A.1. (**a**) Distribution of predicted <E> for the 3-site libraries with predicted <E> < –105 REU. (**b**) Discovery of libraries with predicted <E> -105 REU using parallel MonteCarloPacker trajectories

3. With so many candidate libraries there is a clear need for effective methods for identifying the most favorable options at a range of library sizes. Iterative library design with an increasing bias favoring larger libraries was used to compile a thorough list of favorable case A.1 libraries. For each library, tabulated energy values for all 8000 possible case A sequences were used to compute the library <E> and the number of library members with E < –105 REU.

3.5 Selection of Case A Libraries via Pareto Analysis

1. Pareto analysis helps identify interesting library candidates that are worth consideration given two or more competing quality metrics. For a given library size, libraries with lower <E> are preferable. For a given <E>, libraries with larger size are preferable. Several illustrative example libraries are described in Table 2.
2. In the absence of a high-throughput assay, a library that is highly enriched for stable sequences may be a superior option. In this scenario, a strong case A candidate library consists of the degenerate codons CTG:DVC:NNK (Table 2). These encode a Leu for residue 5, Ala/Cys/Asp/Gly/Asn/Ser/Thr/Tyr for residue 30, and all 20 amino acids (and a stop codon) for residue 43. This library is a nice example of how the design approach can end up providing suggestions that are quite different from traditional saturation mutagenesis; only residue 43 gets full amino acid diversity while residue 30 gets a tailored amino acid palette and residue 5 is left as the wild-type. This library has $1 \times 8 \times 20 = 160$ sense outcomes and a total library size of 168. The library <E> is –106 REU, and 136 of the sequences have E < -105 REU. The 24 less favorable (E > –105) variants include Cys 30 paired with (Lys, Arg, Ile, Gly, His, Cys, Asn, Asp, or Leu 43), Gly 30 paired with (His,

Table 2
Library size refers to all distinct outcomes including genes that include stop codons. Stop codons are encoded by DND, NNK, THW, and NDK

Degenerate codons	<E>, REU	#E < −105 REU	# Sense	Library size	Amino acid outcomes
CTG:DVC:NNK	−106	136	160	168	Just L: ACDGNSTY : All 20
VND:DND:NNK	−90.2	665	5440	6048	All but CFWY: All but QHP: All 20
NNK:NNK:NNK	−70.9	761	8000	9261	All 20: All 20: All 20
TTA:THW:NDK	−108	68	68	90	Just L: FLSY: All but APT
VND:TTC:VND	−94.7	143	256	256	All but CFWY: F : All but CFWY

Asn, Cys, Gly, Asp, or Leu 43), Thr 30 with Leu 43, and all 8 variants with Pro 43.

3. Rather than asking what fraction of a library is predicted to be low-energy, it may be helpful to turn the question around and ask what fraction of the low-energy sequence space the library captures. For case A, low-energy sequences could be identified exactly via enumeration. The CTG:DVC:NNK library captures only 18 % of the 761 case A sequences with instantiated E < −105 REU.
4. To capture a larger share of the low-energy sequence space, be willing to test libraries with a larger risky sequence fraction. In this latter scenario, the Pareto analysis can still provide guidance. For example, the library encoded by VND:DND:NNK (Fig. 8a) is likely a better choice than the next library on the Pareto front (NNK:DHN:NNK), which only excludes Arg, Cys, Gln, Gly, His, Pro, and Trp30 from full saturation mutagenesis.
5. A second illustrative example of a tailored library is TTA:THW:NDK (Fig. 8c). This library has a low <E> of −108, and all 68 of its sense outcomes are low-energy variants (Table 2). As with the other low-energy library described above, this one uses most of the "diversity budget" for residue 43, fixes residue 5 as Leu, and uses a more tailored degenerate codon for residue 30.

The second Pareto analysis (Fig. 8b) identifies libraries with the largest predicted number of sub-threshold sequences (E < −105 REU for case A.1) for a given library size. Full saturation mutagenesis is required to capture all 761 of the low-energy sequences (Fig. 8b). A close inspection of the smaller libraries (Fig. 8d) shows that the fraction of the library that consists of low-energy sequences drops dramatically as the library size exceeds 256. The VND:TTC:VND library is an appealing option since 56 % of its members consist of low-energy sequences and there are no stop codons (Table 2).

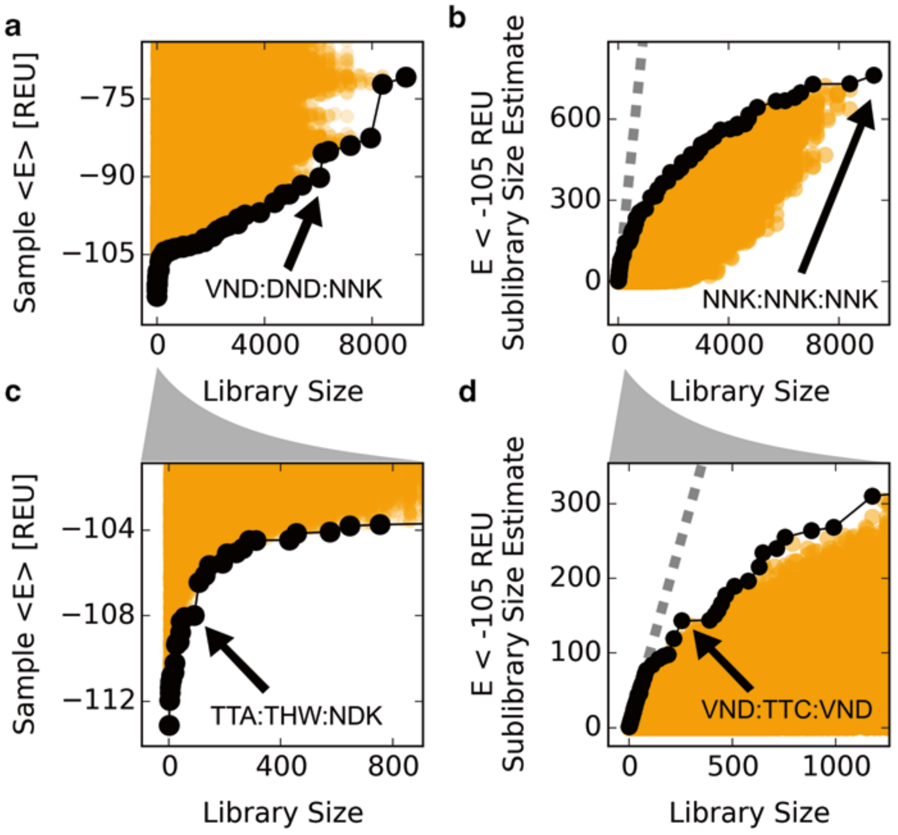

Fig. 8 Case A.1 library selection Pareto analysis. All 450838 libraries sampled by the iterative bias scan are shown as *orange dots*. Pareto optimal libraries are *black dots*. (**a**) The Pareto front for those libraries that have a low <E> and a high library size. (**b**) The Pareto front for libraries with a high estimated no. of sequences with $E < -105$ REU and a low total library size (including nonsense members). (**c**) Small library close inspection of panel a. (**d**) Small library close inspection of panel b

6. Note that the calculated <E> values for these libraries weight all constituent sequences equally, rather than reflecting the statistical likelihood of observing each sequence (*see* **Note 5**).

3.6 Case B: The 5-site Core Redesign Library

1. The five-site library is intended to serve as a realistic model for the kind of library where degenerate codon optimization is valuable. Thorough sampling of five-site full saturation mutagenesis libraries is beyond the reach of most experimental assays. Thus, codon tailoring is advisable prior to undertaking experimental construction and characterization of a library.

Several trends are consistent with the 3-site library (Table 1). The best rmsd_{LET} (4.1 REU) was for case B.2 with plentiful rotamers and optimized exponential weighting ($\tau = 50$, Eq. 3). The choice of regularization penalty (k) was somewhat important when using smaller training data sets (Fig. 9c). The best case B.2 rmsd_{LET} was obtained (Fig. 9d) when using a large training set (32,000

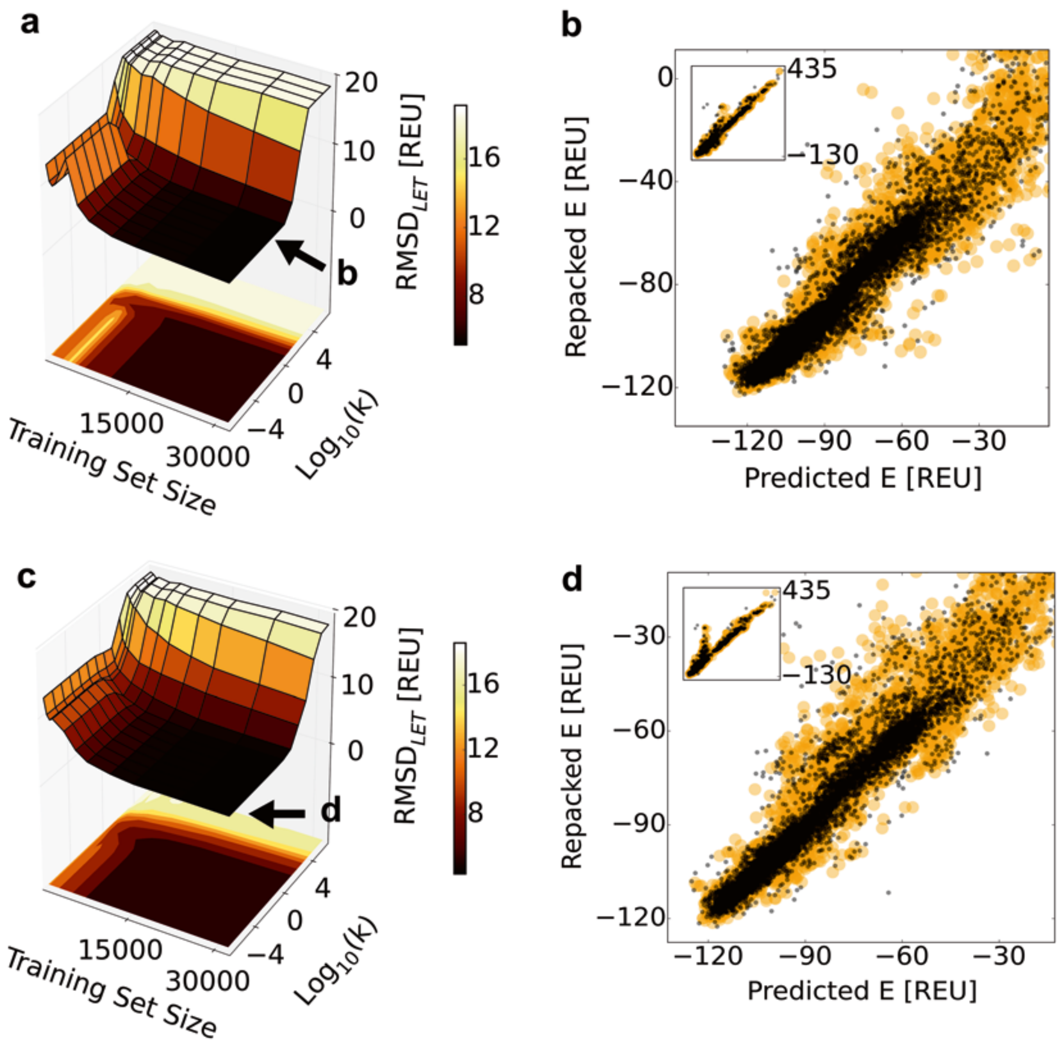

Fig. 9 Case B.1 and B.2 approximation performance. Training set combinations are partial transparent orange points while test set combinations are black points. (**a**) **Case B.1** rmsd$_{LET}$ versus training set size and the regularization parameter *k*. The best prediction was (**b**), when a random training set (32,000 combinations) was used to fit 3706 parameters with regularization ($k = 0.1$) resulting in training set recapitulation (rmsd = 5.95 REU). Performance for favorable test set combinations was reasonable (**rmsd$_{LET}$ = 5.0 REU**) for the 15,984 combinations within 100 REU of the minimum test set combination (–122.5 REU). The entire test set (18,000 combinations) was predicted with rmsd = 7.5 REU (inset). (**c**) rmsd$_{LET}$ versus training set size and the regularization parameter *k* for **Case B.2** (weighted regression with a tuned $\tau = 50$ REU). The best prediction was (**d**), when a random training set (32,000 combinations) was used to fit 3706 parameters with regularization ($k = 1e-07$) resulting in training set recapitulation (rmsd = 7.36 REU). Performance for favorable test set combinations was reasonable (**rmsd$_{LET}$ = 4.1 REU**) for the 15,984 combinations within 100 REU of the minimum test set combination (–122.5 REU). The entire test set (18,000 combinations) was predicted with rmsd = 8.3 REU (inset)

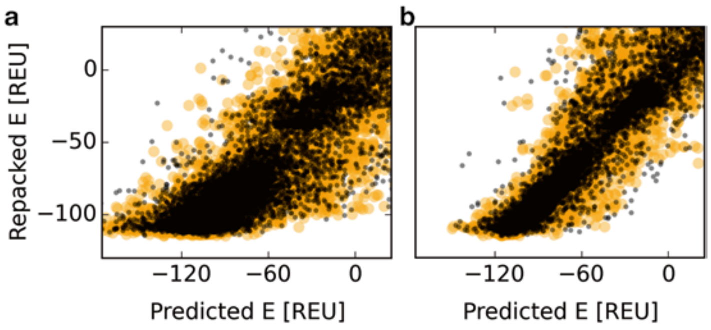

Fig. 10 Benefits of weighted regression. (**a**) **Case B.3** versus, (**b**) **Case B.4**. Using weighted regression (with an optimized parameter $\tau = 75$) cut rmsd_{LET} almost in half, from 14.4 REU (case B.3) to 8.4 REU (case B.4)

random combinations) with minimal regularization parameter ($k = 1\text{E}{-}7$).

2. Without the exponential weighting (case B.1), performance is still good but the regularization parameter (k) is more important still (Fig. 9a). The lowest rmsd_{LET} (5.0 REU, Fig. 9b) was obtained when using a large training set (32,000 random combinations) with a significant regularization parameter ($k = 0.1$).
3. Pruning rotamers (cases B.3 and B.4) results in significant performance degradation. Given minimal rotamers, the effect of exponential weighting was more dramatic. Without exponential weighting (case B.3), rmsd_{LET} was fairly high (14.4 REU, Fig. 10a). However, with minimal rotamers and exponential weighting (case B.4), rmsd_{LET} was greatly improved (8.4 REU, Fig. 10b). This result highlights the potential utility of weighting.

3.7 Case C.1: A 16-Site Surface Library

1. Case C has 16 design positions. Unlike cases A and B, these amino acids are on the protein surface which may significantly change the roughness of the design energy landscape (*see* Fig. 3f versus d). Two specific scenarios are illustrated. In case C.1, the 16 surface sites are provided only with base Dunbrack rotamers for all 20 possible amino acids resulting in a combinatorial search problem of 1.8E35 sequence/structures. In case C.2, provision of standard rotamers increases the search size to 6.5E45.
2. Broadly speaking, the regression performance trends are similar to the trends from the smaller libraries. Case C.1 prediction performance suffers for overly large regularization parameter (Fig. 11a, $k \gg 10$), but a modest penalty of ten yields the best prediction performance ($\text{rmsd}_{\text{LET}} = 9.9$ REU). An eyecatching feature of the regression model training performance plot (Fig. 11a) is the large peak (poor prediction performance)

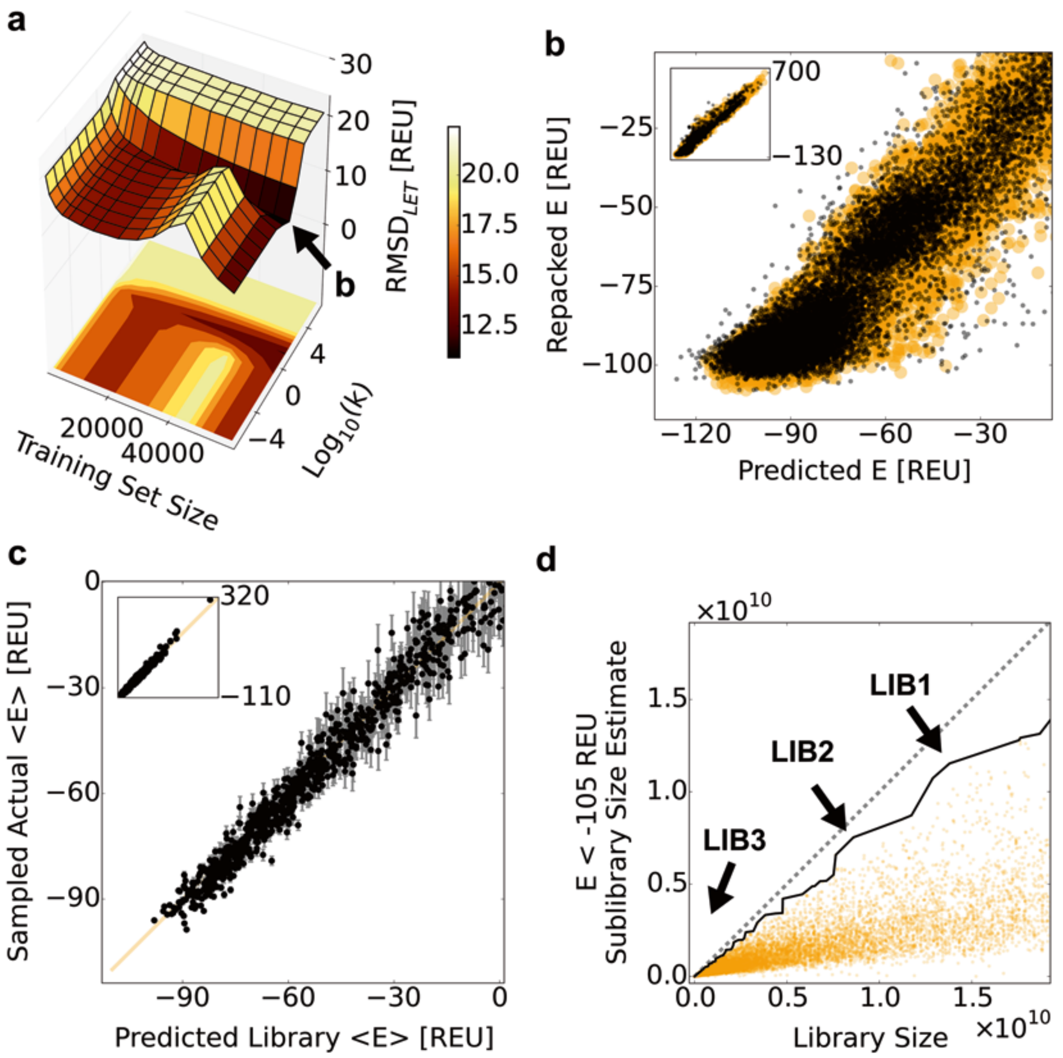

Fig. 11 Case C sequence and library approximation performance. (**a**) Test set rmsd versus training set size and the regularization parameter *k*. (**b**) A random training set (60,000 combinations) was used to fit 43,625 parameters with regularization ($k = 10$) resulting in training set recapitulation (rmsd = 10.46 REU). Performance for favorable test set combinations was reasonable (**$\text{rmsd}_{\text{LET}} = 9.9$ REU**) for the 15,265 combinations within 100 REU of the minimum test set combination (–107.9 REU). The entire test set (20,000 combinations) was predicted with rmsd = 16.1 REU (inset). (**c**) 1000 random degenerate codon libraries were selected. From each, either the full library or a sample of 400 sequences were optimized via sidechain repacking and scored. The estimated <E> values of the sample sequences were fairly well predicted. *Error bars* reflect $\sigma\langle E\rangle$ (Eq. 4). The full range (inset) was predicted with rmsd = 5.5 REU, while the 257 libraries with predicted <E> within 30 REU of the minimum (–98.1) had **rmsd = 3.6 REU.** (**d**) Pareto analysis of libraries sampled via the iterative combinatorial optimization with an escalating large-library bias. *See* below for library descriptions

when using a low regularization parameter ($k < 0.01$) and fairly large training sets (30,000–50,000 combinations). This apparent overfitting pathology is present to varying degree in the previous cases (Figs. 4ac and 9ac). This phenomenon is investigated below in the Overfitting Trends section.

3. Pareto analysis (Fig. 10d) suggests that it is easy to find large case C libraries that are largely composed of low-energy members ($E < -105$ REU). For example, the libraries that are marked LIB1, LIB2, and LIB3 all are predicted to have an 83 % or greater low-energy fraction (Table 3). Notably, the predicted <E> for LIB1 is on par with the original wild-type sequence (-112 REU). LIB2 is slightly smaller (8.6E9 total variants) and has a higher low-energy fraction (88 %). To get a library that is predicted to fall entirely below the threshold, much smaller libraries are necessary (i.e. LIB3 with 8.96E6 sequences). The predicted <E> is well below the original wild-type sequence (-127 REU) thanks to accrued mutations that Rosetta assesses as stabilizing. Six sites are fixed while other sites have up to eight amino acids. The encoded amino acid sets for LIB3 are:

```
L:NT:G:R:GILRSV:DGHNRS:FSY:IKLMQR:FV:EGKMRV:F:CDGHNRSY
:H:DEGHKNQRS:S:EGIKLQRV
```

As the library size decreases further, it becomes easy to find libraries that are predicted to fall entirely below the −105 REU threshold. Almost all of the 288 sampled libraries with size below two million sequences fall into this category. In principle, to differentiate between these candidates it could be helpful to introduce another evaluation criterion such as <E>, or to assess the fraction of constituent sequences that meets a more stringent stability threshold (e.g. $E < -120$ REU).

3.8 Case C.2: With Additional Rotamers

1. The largest combinatorial problem for this chapter is case C.2 (Table 1). As above, the additional rotamers make a significant improvement in performance (Fig. 12). It may be surprising to note that the rmsd_{LET} is lower (4.8 REU) than the comparable calculation for case B ($\text{rmsd}_{\text{LET}} = 5.0$). This can be rationalized by noting that the design positions for case C are all surface-exposed sites, where it is easier for amino acid combinations to avoid clashes (given sufficient rotamer flexibility). Thus, there are fewer legitimate higher-order frustration effects encountered in this scoring landscape (Fig. 3), and it is possible to obtain a high accuracy fit.

Presumably, this fit could be further improved using weighting or perhaps the introduction of 3-body terms. However, the current level of accuracy seems quite sufficient to assist with the design of degenerate codon libraries (Fig. 12c). The rmsd between regression model <E> predictions and directly sampled <E> estimates was only 4.1 (or 1.7) REU for all (or <E> < −76 REU) libraries. When chasing high precision, it is important to recall that the underlying scoring function is itself a fairly crude approximation of the biophysical effects in play.

Table 3
Example libraries of interest for Case C. Library size refers to all distinct outcomes including genes that include stop codons

Case	Name	Codons	Library size	<E> [REU]	% with E < –105 REU
C.1	LIB1	CNT:KYC:GGT:CDD:KYA:ASC:TBS:ABG:RYG:VDA:MDK:VDS:YDY:SAA:VBB:MWA	1.4E10	–112	84
C.1	LIB2	WTG:DSG:GGT:VDG:WYW:HDT:SDR:WSG:RYG:KNT:TTC:VDW:GVA:GVA:VDG:CRS	4.7E9	–113	88
C.1	LIB3	TTA:AMC:GGT:AGG:VKY:VRT:THY:MDR:KTC:RDG:TTC:NRC:CAT:VRS:AGC:VDA	9.0E6	–127	100
C.2	LIB4	BBG:TAC:WYS:NKB:ABG:HKY:SAA:MWA:VKY:RNR:VRS:CWR:VDS:CDG:RHR:ADB	1.1E11	–110	88
C.2	LIB5	BBG:TAC:WYS:NKB:ABG:SRA:SAA:MWA:VKY:RNR:VRS:CWR:VDS:CDG:RHR:ADB	7.3E10	–111	89
C.2	LIB6	BTT:AGG:GTA:AGG:VAK:DYR:RHS:ARS:AHH:AAD:RHS:AMC:AGG:RBH:RDG:VKY	1.6E8	–119	100
C.2	SAT	NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK:NNK	6.6E20	–59	15

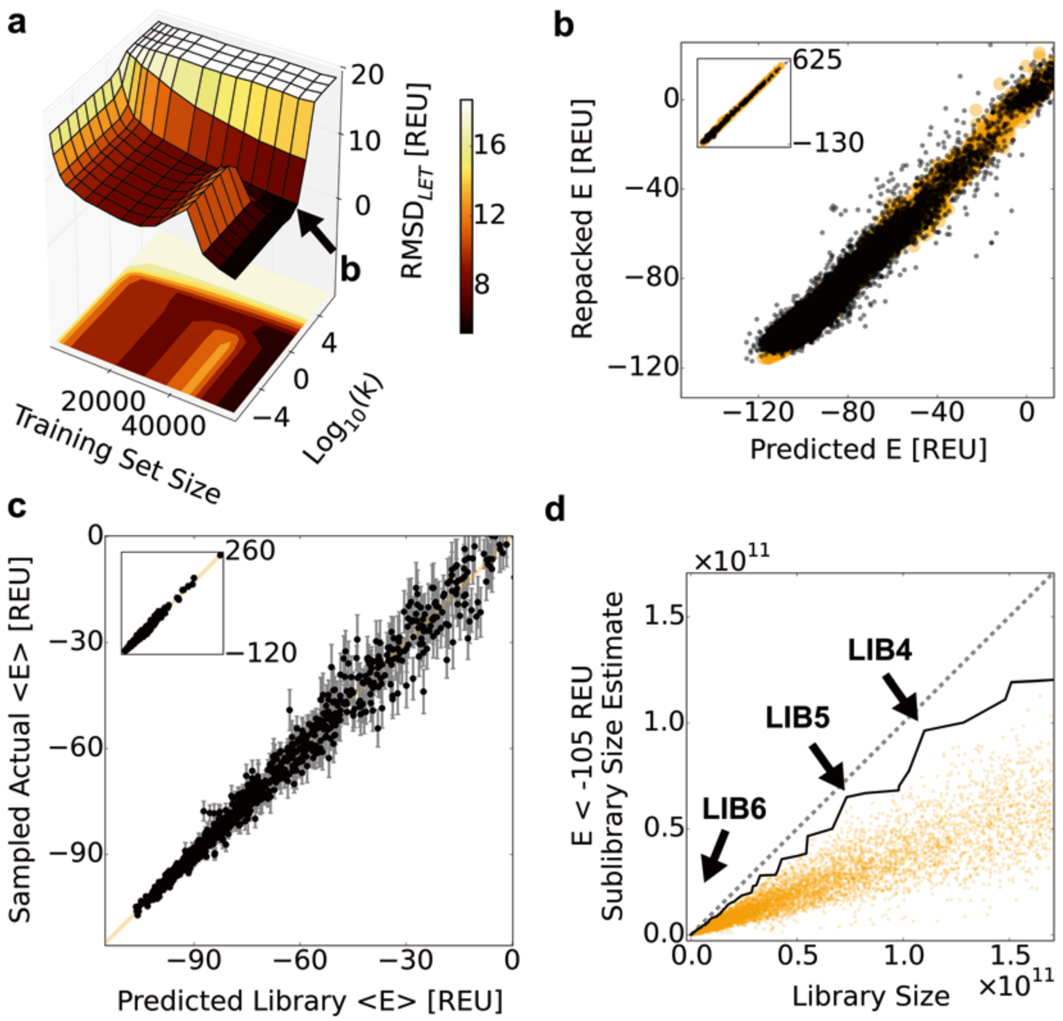

Fig. 12 Case C.2 sequence and library approximation performance. (**a**) Test set rmsd versus training set size and the regularization parameter *k*. (**b**) A random training set (60,000 combinations) was used to fit 43,625 parameters with regularization ($k = 1$) resulting in training set recapitulation (rmsd = 1.7 REU). The test set (20,000 combinations) was predicted with rmsd = 6.6 REU. Performance for favorable combinations was reasonable (**rmsd$_{\mathbf{LET}}$ = 4.8 REU**) for the 16,996 test set combinations within 100 REU of the minimum test set combination (–117 REU). **(c)** 1000 random degenerate codon libraries were selected. From each, either the full library or a sample of 400 sequences were optimized via sidechain repacking and scored. The estimated <E> values of the sample sequences were fairly well predicted (rmsd = 4.1 REU over the full range, inset). *Error bars* reflect $\sigma\langle E\rangle$ (Eq. 4). Performance for the 457 libraries with predicted <E> within 30 REU of the minimum (–106.5 REU) was better still (**rmsd = 1.7 REU**). (**d**) Pareto analysis of libraries sampled via the iterative combinatorial optimization with an escalating large-library bias. *See* below for LIB descriptions

2. As above, use iterative bias sampling to collect optimized libraries of varying size and proceed with Pareto analysis. Illustrative Pareto analysis of 755,912 candidate case C.2 libraries (Fig. 12d) suggests that it is even easier to find large case C.2 libraries that are highly enriched for low-energy members (Table 3).

This is not surprising since the additional rotamers result in more highly optimized structures with lower Rosetta scores. LIB6 is the largest library that is predicted to have 100 % low-energy constituents. The encoded amino acid sets for LIB6 are:

```
FLV:R:V:R:DEHKNQ:AILMSTV:ADEIKMNTV:KNRS:IKNT:KN:ADEIKM
NTV:NT:R:AGIRSTV:EGKMRV:GILRSV
```

These libraries compare favorably to a brute force saturation mutagenesis approach. For one, library size can be matched to the available transformation and screening capacity whereas the size of the NNK library exceeds feasible screening size. Also, the NNK library has a high <E> (−59 REU) and a large fraction of the constituent sequences are unfavorable (85 % have E > −105 REU).

3.9 Overfitting Trends

1. The purpose of this section is to investigate the counterintuitive overfitting behavior noted above. Inferior prediction performance for models derived from larger training sets occurred repeatably for varying input training sets and using either the lsmr or cvxopt regression tools. This effect was most prominent for three regression model variants (cases A.2, B.3, and C.1) with negligible regularization ($k = 1E{-}7$) (Fig. 13a). The effect was largest for case C.1. Therefore, to illustrate the fitting pathology two regression models can be compared: the fit for case C.1 with training sets of 24,000 (Fig. 13b) versus 45,000 (Fig. 13c).

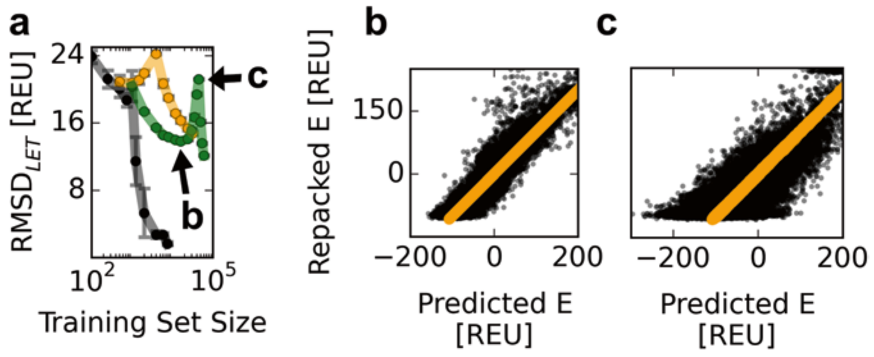

Fig. 13 Overtraining. (**a**) Apparent overfitting with "free" regression (regularization $k = 1E{-}7$) was prominent for training case A.2 with ~1000 combinations (black), case B.2 with ~4000 combinations (*orange*), and case C.1 with ~45,000 combinations (*green*). *Error bars* reflect the standard deviation among 5 random training set replicates. We use case C.1 to further illustrate the degradation of prediction performance: **(b)** Prediction performance (*black*) is reasonable (rmsd$_{LET}$ = 13.9 REU) when trained with 24,000 random combinations. The training set (*orange*) is recapitulated exactly (rmsd = 0.0). **(c)** Prediction performance (*black*) is degraded (rmsd$_{LET}$ = 21.2 REU) when trained with 45,000 random combinations. The training set (*orange*) is still recapitulated nearly exactly (rmsd = 1.4 REU)

2. First, a random training set (24,000 combinations) is used to fit 43,625 parameters with regularization ($k = 1e{-}07$). Given the over-abundance of fitting parameters, it was not surprising that the training set was recapitulated *exactly* (rmsd = 0.00, orange dots). In contrast, the entire test set was predicted with rmsd = 20.7 REU (Fig. 13b). The 42,745 test set combinations within 100 REU of the minimum test set combination (−107.9 REU) could be predicted with $\text{rmsd}_{\text{LET}} = 13.9$ REU (Fig. 13b).
3. For comparison, a larger random training set (45,000 combinations) was used to fit 43,625 parameters with regularization ($k = 1e{-}07$) resulting in *near exact* training set recapitulation (rmsd = 1.4 REU, orange dots). In contrast, the entire test set was predicted with rmsd = 42.4 REU (Fig. 13c). The 26,756 test set combinations within 100 REU of the minimum test set combination (-107.8 REU) were only predicted with $\text{rmsd}_{\text{LET}} = 21.2$ REU.

3.10 Recap the Pertinent Observations

1. Scenarios with fewer rotamers (e.g. cases A.2, B.3, and C.1) have a greater tendency to experience overfitting.
2. Overfitting can be suppressed by regularization (e.g. Fig. 11a)
3. The best regularization parameter seems to grow with the problem size ($k = 0.01$, 1, and 10 for cases A.2, B.3, and C.1).
4. Prediction performance is most degraded when the fit has a certain number of training examples: ~1000 for A.2, ~4000 for B.3, and ~45,000 for C.1 (Fig. 13a). These numbers are similar to the number of fitting parameters: respectively 1141, 3706, and 43,625.
5. Despite training the case C.1 model with 45,000 combinations, the training set combinations (orange points) are still clearly being overfit (compare to Fig. 11)
6. Given these observations, it may be that having a small number of training instances (relative to the number of fit parameters) serves to restrain the magnitude of the fit parameters, much as the regularization process penalizes large fit parameter magnitude. When the number of training instances is comparable to the number of parameters, fit parameters are more likely to adopt large magnitude values to fit the training set data. Meanwhile, minimal rotamer cases make the energy landscape rougher (compare Fig. 3bdf to Fig. 3acd). The rougher training set energy landscape will also result in more extreme fit parameters that degrade the testset prediction performance. The effectiveness of regularization, which attempts to keep parameters near 0, supports the idea that the fitting pathology is tied to the magnitude of the fitting terms.

To investigate, examine the fit parameters for the two case C1 regression models (Fig. 13bc). The superior model trained with

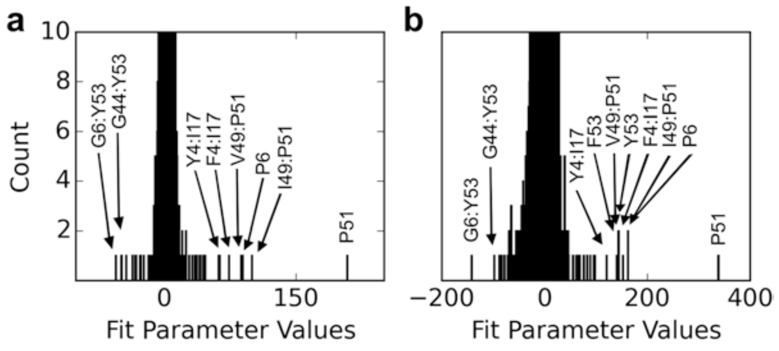

Fig. 14 Comparison of Case C.1 fit coefficients. (**a**) The superior model was trained with 24,000 sequences (*see* also Fig. 13b) and had parameters of lesser magnitude, while (**b**) the inferior model was trained with 45,000 combinations (*see* also Fig. 13c) and had fit parameters of greater magnitude

24,000 sequences has a sum of absolute parameter values of 60,831, while the inferior model trained with 45,000 combinations has a sum of absolute parameter values of 301,104.

7. The dramatic fivefold shift in the magnitude of the fit parameters also appears in histograms of the fitting parameter values (Fig. 14). There is some qualitative consistency between the two models. For example, the most unfavorable term for both models is the 1-body effect of T51P (unsurprising since T51 is within a beta strand), and the most favorable term is the 2-body effect of I6G:T53Y which can be rationalized as a bump/hole interaction between these adjacent residues on the beta sheet surface. However, it is important to note that the actual values of the coefficients are not stable; attempts to glean additional insight from inspection of the fit coefficients may be problematic.

One take home lesson is that regression model performance can be difficult to anticipate (and may be strongly dependent on regularization) unless the training set size significantly exceeds the number of fitting parameters. Any scientist preparing a regression model of this type should carefully scan the training set size and regularization parameter to ensure optimal model quality (*see* **Note 6**).

3.11 Computational Time

The three design problems discussed here can be framed as conventional CPD calculations. Searching the sequence-structure space directly, optimal solutions can be found using either CPLEX or FasterPacker (*see* above). However, the 725 possible mixtures of amino acids possible at each site dwarfs the 20 possible amino acids. Sampling in library space, therefore, is significantly more challenging. With the exception of case A, it is impractical to tabulate the instantiated energy of all the sequences encoded by each library. A brute force search instantiating all sequence combinations (assuming generous rotamers) requires only 38 seconds for case A.1 to a projected 8E11 years for case C.2.

Table 4
This Table summarizes the Python scripts used for the calculations. Time represents the elapsed wall clock time in seconds necessary to complete the calculations on a single CPU

Time (s)	Script name	Script purpose
3	A_prep.py	Process the input PDB model and save as a CHOMP System
219	B_fill_energy_graph.py	Setup the design problem, fill, and save the EnergyGraph
10	C_pick_initial_set.py	Select a set of combinations to serve as an initial pool
3077	D_score_initial_set_multi.py	Instantiate: run repacking calculations on the initial pool
5	E_split_to_training_samples.py	Randomly divide the initial pool into training and test sets
nd	*F_try_regression.py*	*Do a quick regression test to ensure things are working so far*
2190	G_scan_training_params.py	Repeatedly run regression varying the training set size and regularization parameter
nd	*H_gen_figure_trainsize_vs_ridgeparam_vs_testrmsd.py*	*Illustrate test set performance* versus *training parameters*
nd	*I_gen_fig_example_train_test.py*	*Illustrate the training set and test set fit for the best case parameters*
1978	M_score_dgen_codon_sets.py	Convert the amino acid level regression model to a degenerate codon level model
13,563	N_sample_random_libraries.py	Randomly select a set of random degenerate codon libraries and perform the requisite instantiation (repacking)
nd	*O_lib_predictE_vs_actualE.py*	*Illustrate the correlation between predicted and actual (sampled) <E> for each library*
1490	P_iterlib_sample.py	Collect libraries by 100 rounds of combinatorial optimization for low <E> with an escalating bias favoring larger libraries.
22,918	Q_calc_pareto_stats.py	First lookup or predict the energy for a sequence sample from each candidate library. Then calculate <E> and the expected number of variants with E < a threshold. Finally, use a divide and conquer approach to compute the two Pareto fronts of interest.
nd	R_plot_pareto_ok.py	Plot the Pareto front (black) and other libraries (orange)

In comparison, the aggregate calculation time for the steps described above is attractive. All reported calculations could be performed on a single 2.8 GHz Intel Core i7 machine over several days (*see* Table 4 for case C.2 calculation time table). Most of the time-consuming calculations were parallelized across 8 threads using the Python multiprocessing module. Distributing bottleneck calculations beyond the cores of a single CPU could easily further reduce wall time.

One easy way to limit the CPU time while retaining the power of the library-space optimization approach would be to prune the set of degenerate codons considered at each design site. One pragmatic approach might be to design a limited number of amino acid sets, guided by biophysical intuition (e.g. hydrophobic, large hydrophobic, small, large, charged, aromatic, etc.). Selecting several hundreds of these useful amino acid sets would make the library design code more efficient than the current search over 725 possibilities. Similarly, with repeated use of the current 725-member design palette, it may be possible to identify which degenerate codons are rarely useful and eliminate them from consideration.

4 Notes

1. The presented methods are flexible, and amenable for modification. One such modification that might be particularly desirable would be to enable optimization of amino acid bias. At the outset, the degenerate codon search space was defined to be the 725 degenerate codons that produce unique sense mixtures of amino acids. It is worthwhile to note, however, that the formalism presented here would also work if the design palette consists of the 1439 degenerate codons that produce unique sense *ratios* of amino acids. If outcome amino acid probabilities are included when creating the degenerate codon energy model (Eqs. 5 and 6), the resulting optimization target <E> will reflect the expectation REU score for clones pulled at random from the experimental library. This additional level of design could prove useful. Optimizing amino acid frequencies could further increase library fitness by decreasing <E>. For example, given a particular site that favors leucine over phenylalanine, combinatorial library optimization might select a degenerate codon like YTD that encodes a 5:1 ratio of Leu to Phe rather than TTB that encodes a 1:2 ratio of Leu to Phe.
2. The illustrative examples presented in this chapter provide another example of regression-based approximations successfully capturing more expensive calculations with sufficient accuracy to guide an otherwise infeasible search problem. By "integrating out" the structure variables, and providing an essentially instantaneous lookup of the predicted energy for

any given sequence, it becomes feasible to execute a combinatorial search directly in "library space." This approach was recently reported in the context of cluster expansion [14]. Readers of this chapter who are preparing to design a codon library are therefore encouraged to review Verma et al.

3. In principle, a penalty could also be levied for stop codons by giving stop codon outcomes a large 1-body energy term. The goal would be to ensure that non-sense outcomes have large unfavorable scores commensurate with other likely unfolded sequences.

4. It is worth noting that there are certain technical challenges to performing library-space optimization for case C. With 725 possible degenerate codons and 16 design sites, the library search space has 725^{16} combinations, or 5.8E45. Building a graph of the codon:codon scores (Fig. 1) required 30 minutes. Storing the graph in binary form on disk requires nearly a gigabyte.

5. In practice, some sequences will be more frequent than others. For example, the MKD degenerate codon yields an arginine 5 times more frequently than a serine. If desired, it would be easy to instead calculate the expectation value <E> for sequences drawn from the library with the actual amino acid frequency weights (Eqs. 5 and 6) rather than assuming equal representation. The former approach may be preferable if the planned approach is to build a large experimental library and characterize only a random subset thereof.

6. Additional caution and careful regularization parameter tuning is recommended if pursuing high-accuracy regression models including 3-body terms, since the high number of possible 3-body terms may make it difficult to prepare models with a large excess of training data.

References

1. Ponder JW, Richards FM (1987) Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequences for different structural classes. J Mol Biol 193: 775–791
2. Pierce NA, Winfree E (2002) Protein design is NP-hard. Protein Eng 15:779–782
3. Desmet J, De Maeyer M, Hazes B, Lasters I (1992) The dead-end elimination theorem and its use in protein side-chain positioning. Nature 356:539–542
4. Desmet J, Spriet J, Lasters I (2002) Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins 48:31–43
5. Canutescu AA, Shelenkov AA, Dunbrack RL (2003) A graph-theory algorithm for rapid protein side-chain prediction. Protein Sci 12:2001–2014
6. Kingsford CL, Chazelle B, Singh M (2005) Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics 21:1028–1039
7. Allen BD, Mayo SL (2006) Dramatic performance enhancements for the FASTER optimization algorithm. J Comput Chem 27: 1071–1075
8. Hallen MA, Keedy DA, Donald BR (2012) Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm

with continuous sidechain and backbone flexibility., Proteins

9. Zhou F, Grigoryan G, Lustig SR et al (2005) Coarse-graining protein energetics in sequence variables. Phys Rev Lett 95:148103
10. Grigoryan G, Zhou F, Lustig SR et al (2006) Ultra-fast evaluation of protein energies directly from sequence. PLoS Comput Biol 2, e63
11. Grigoryan G, Reinke AW, Keating AE (2009) Design of protein-interaction specificity gives selective bZIP-binding peptides. Nature 458: 859–864
12. Apgar JR, Hahn S, Grigoryan G, Keating AE (2009) Cluster expansion models for flexible-backbone protein energetics. J Comput Chem 30:2402–2413
13. Hahn S, Ashenberg O, Grigoryan G, Keating AE (2010) Identifying and reducing error in cluster-expansion approximations of protein energies. J Comput Chem 31(6):2900–2914
14. Verma D, Grigoryan G, Bailey-Kellogg C (2015) Structure-based design of combinatorial mutagenesis libraries. Protein Sci 24: 895–908
15. Liao J, Warmuth MK, Govindarajan S et al (2007) Engineering proteinase K using machine learning and synthetic genes. BMC Biotechnol 7:16
16. Otey CR, Landwehr M, Endelman JB et al (2006) Structure-guided recombination creates an artificial family of cytochromes P450. PLoS Biol 4, e112
17. Li Y, Drummond DA, Sawayama AM et al (2007) A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. Nat Biotechnol 25: 1051–1056
18. Heinzelman P, Snow CD, Wu I et al (2009) A family of thermostable fungal cellulases created by structure-guided recombination. Proc Natl Acad Sci U S A 106:5610–5615
19. Heinzelman P, Snow CD, Smith MA et al (2009) SCHEMA recombination of a fungal cellulase uncovers a single mutation that contributes markedly to stability. J Biol Chem 284:26229–26233
20. Heinzelman P, Komor R, Kanaan A et al (2010) Efficient screening of fungal cellobiohydrolase class I enzymes for thermostabilizing sequence blocks by SCHEMA structure-guided recombination. Protein Eng Des Sel 23:871–880
21. Smith MA, Rentmeister A, Snow CD et al (2012) A diverse set of family 48 bacterial glycoside hydrolase cellulases created by structure-guided recombination. FEBS J 279:4453–4465
22. Silberg JJ, Endelman JB, Arnold FH (2004) SCHEMA-guided protein recombination. Methods Enzymol 388:35–42
23. Endelman JB, Silberg JJ, Wang ZG, Arnold FH (2004) Site-directed protein recombination as a shortest-path problem. Protein Eng Des Sel 17:589–594
24. Pantazes RJ, Saraf MC, Maranas CD (2007) Optimal protein library design using recombination or point mutations based on sequence-based scoring functions. Protein Eng Des Sel 20:361–373
25. Johnson LB, Huber TR, Snow CD (2014) Methods for library-scale computational protein design. Methods Mol Biol 1216: 129–159
26. Voigt CA, Mayo SL, Arnold FH, Wang ZG (2001) Computational method to reduce the search space for directed protein evolution. Proc Natl Acad Sci 98:3778
27. Wang W, Saven JG (2002) Designing gene libraries from protein profiles for combinatorial protein experiments. Nucleic Acids Res 30:e120
28. Mena MA, Daugherty PS (2005) Automated design of degenerate codon libraries. Protein Eng Des Sel 18:559–561
29. Allen BD, Nisthal A, Mayo SL (2010) Experimental library screening demonstrates the successful application of computational protein design to large structural ensembles. Proc Natl Acad Sci 107:19838–19843
30. Parker AS, Griswold KE, Bailey-Kellogg C (2011) Optimization of combinatorial mutagenesis. J Comput Biol 18:1743–1756
31. Chen TS, Palacios H, Keating AE (2013) Structure based re-design of the binding specificity of anti-apoptotic Bcl-xL. J Mol Biol 425:171–185
32. Jacobs TM, Yumerefendi H, Kuhlman B, Leaver-Fay A (2015) SwiftLib: rapid degenerate-codon-library optimization through dynamic programming. Nucleic Acids Res 43:e34
33. Treynor TP, Vizcarra CL, Nedelcu D, Mayo SL (2007) Computationally designed libraries of fluorescent proteins evaluated by preservation and diversity of function. Proc Natl Acad Sci U S A 104:48–53
34. Patrick WM, Firth AE, Blackburn JM (2003) User-friendly algorithms for estimating completeness and diversity in randomized protein-encoding libraries. Protein Eng 16:451–457
35. Rohl CA, Strauss CEM, Misura KMS, Baker D (2004) Protein structure prediction using Rosetta. Methods Enzymol 383:66–93

36. Hughes MD, Nagel DA, Santos AF et al (2003) Removing the redundancy from randomised gene libraries. J Mol Biol 331:973–979
37. Tang L, Gao H, Zhu X et al (2012) Construction of "small-intelligent" focused mutagenesis libraries using well-designed combinatorial degenerate primers. Biotechniques 52:149–158
38. Kille S, Acevedo-Rocha CG, Parra LP et al (2013) Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. ACS Synth Biol 2:83–92
39. Ashraf M, Frigotto L, Smith ME et al (2013) ProxiMAX randomization: a new technology for non-degenerate saturation mutagenesis of contiguous codons. Biochem Soc Trans 41:1189–1194
40. Tang L, Wang X, Ru B et al (2014) MDC-Analyzer: a novel degenerate primer design tool for the construction of intelligent mutagenesis libraries with contiguous sites. Biotechniques 56:301–302, 304, 306–308, passim
41. Nov Y, Segev D (2013) Optimal codon randomization via mathematical programming. J Theor Biol 335:147–152
42. Sanchez JM, Ducastelle F, Gratias D (1984) Generalized cluster description of multicomponent systems. Physica A 128:334–350
43. Ng AH, Snow CD (2011) Polarizable protein packing. J Comput Chem 32:1334–1344
44. Ponder JW, Wu C, Ren P et al (2010) Current status of the AMOEBA polarizable force field. J Phys Chem B 114:2549–2564
45. Loksha IV, Maiolo JR 3rd, Hong CW et al (2009) SHARPEN-systematic hierarchical algorithms for rotamers and proteins on an extended network. J Comput Chem 30: 999–1005
46. IBM ILOG CPLEX Optimization Studio 12.6.2. IBM
47. Andersen MS, Dahl J, Vandenberghe L (2013) CVXOPT: a python package for convex optimization, version 1.1.6
48. Davis TA (2009) User guide for CHOLMOD: a sparse Cholesky factorization and modification package
49. Fong D, Saunders M (2011) LSMR: an iterative algorithm for sparse least-squares problems. SIAM J Sci Comput 33:2950–2971
50. Jones E, Oliphant T, Peterson P, others (2001) SciPy: open source scientific tools for Python
51. Levine HA (1979) Review: A. N. Tikhonov and V. Y. Arsenin, solutions of ill posed problems. Bull Am Math Soc 1:521–524
52. Amstutz P, Forrer P, Zahnd C, Plückthun A (2001) In vitro display technologies: novel developments and applications. Curr Opin Biotechnol 12:400–405
53. He L, Friedman AM, Bailey-Kellogg C (2012) A divide-and-conquer approach to determine the Pareto frontier for optimization of protein engineering experiments. Proteins 80: 790–806
54. Dunbrack RL Jr, Karplus M (1993) Backbone-dependent rotamer library for proteins. J Mol Biol 230:543–574