Chapter 7

Methods for Library-Scale Computational Protein Design

Lucas B. Johnson, Thaddaus R. Huber, and Christopher D. Snow

Abstract

Faced with a protein engineering challenge, a contemporary researcher can choose from myriad design strategies. Library-scale computational protein design (LCPD) is a hybrid method suitable for the engineering of improved protein variants with diverse sequences. This chapter discusses the background and merits of several practical LCPD techniques. First, LCPD methods suitable for delocalized protein design are presented in the context of example design calculations for cellobiohydrolase II. Second, localized design methods are discussed in the context of an example design calculation intended to shift the substrate specificity of a ketol-acid reductoisomerase Rossmann domain from NADPH to NADH.

Key words Protein library design, Codon selection, Protein engineering, Computational protein design, Consensus analysis, Recombination, SCHEMA

1 Introduction

Library-scale design includes many divergent methods, ranging from random mutagenesis (e.g., error-prone PCR) to computational protein design. Library-scale design methods strive to achieve three goals: produce many diverse solutions, maintain folding and functionality in the majority of variants, and maximize ease of interpretation.

In practice, directed evolution (DE) is a remarkably effective library design method; many protein engineering challenges are readily solved via the stepwise accumulation of random mutations [1, 2]. Moreover, whereas a structure is typically a prerequisite for computational protein design (CPD), no special insight into the structure and function of the protein is required for DE methods. However, the search space of all random mutations is enormous; even a high-throughput assay can only sample a tiny fraction of the sequences within a few mutations from a parent sequence. Interpreting randomly accumulated mutations can also be difficult. Only in rare instances can favorable mutations be rationalized from available structural models.

Compared to DE, CPD methods can consider an astronomical number of candidate sequences, including sequences that vary significantly from the initial sequence. CPD can result in impressive changes to the stability [3], aggregation-resistance [4], specificity [5], or enzymatic activity [6, 7], to name a few examples. Despite these successes, the foundation of CPD relies upon approximate models of protein structure and stability. Deficiencies in the scoring function or in the sampling of potential conformations can result in unfolded or inactive design variants [8]. Experimental testing of CPD sequences provides a referendum on the underlying CPD methodology; however, in practice it is difficult to learn from the success or failure of a single design attempt. An unfolded design variant indicates a model deficiency, but usually does not reveal an unambiguous remedy.

The philosophies behind these different methods are divergent: a pure DE scheme can be effective in the absence of protein structure and function information, while an accurate model of protein structure and function is the foundation and goal of CPD. Despite these philosophical differences, the gap between DE and CPD can be quite small in practice. For instance, a design cycle might start by using CPD to identify a low-energy sequence and progress to DE methods [9–11]. Combining the rational methods of CPD with DE screening methods balances search size with diversity. Rather than a search based on a large number of blind guesses (random mutations), one can formulate a search over a discrete set of hypotheses. Library-scale computational protein design (LCPD) methods combine rational and random methods to create a discrete set of hypothesized variants. Ideally, LCPD results in interpretable libraries that (1) are enriched for improved variants and (2) provide useful information for predicting sequence-structure-function relationships.

The appropriate choice of method will depend on the design goal at hand. Our first example discusses LCPD strategies and tools suitable for altering delocalized protein properties. Delocalized properties, such as stability or solubility, are the result of numerous amino acid interactions across a protein. Our second example focuses on protein properties that are localized to a distinct region. Significant variation within localized regions, such as binding pockets or protein interfaces, can create libraries with varying substrate specificity or enzymatic activity.

2 Materials

All computational scripts mentioned in this example are available at www.sharp-n.org. SHARPEN is an open-source C++/Python software library intended to facilitate the development of new algorithms for protein modeling and design.

3 Example I: Delocalized Design Libraries

Diverse libraries sample a broad range of sequence space, farther afield from an initial sequence, and are therefore more likely to contain significant variants of interest. However, when constructing a library of protein sequences, a trade-off is established between sequence diversity and library stability. Library stability is reflected in the properties of the individual sequences in two ways. First, a stable library will have few unfolded sequences. Second, the individual folded sequences within the library will be stable and functional. While allowing a wide range of mutations within a library greatly increases diversity, many mutations will decrease library stability [12]. When available, structural models can guide the selection of stable sequences by providing insight into which mutations are likely to be destabilizing [13].

Current library design methods use sequence and structure information to predict potentially stabilizing mutations. Hecht and co-workers have demonstrated the ability to design de novo proteins with binary patterning of polar and nonpolar amino acids [14–16]. Alternatively, the palette can be designed to ensure that mutations are compatible with the neighboring amino acids, considering multiple amino acid properties, such as volume, charge, and hydrophobicity [17]. Furthermore, information from multiple sequence alignments can be used to identify tolerated or favored mutations at each site [18–21]. Structural models can still be useful in conjunction with sequence-based design methods. For example, a structure can be used to refine ambiguous portions of the alignment (i.e., insertion/deletion sites) and to determine if certain residues (e.g., Pro, Trp) are likely to be incompatible with the protein backbone or the neighboring amino acids.

If detailed structural models are available, combinatorial CPD methods can be used. These methods provide each amino acid with multiple side chain conformations (rotamers) and provide each design site with multiple candidate amino acid identities [22, 23]. The design calculation is thus reduced to the combinatorial optimization problem of finding a rotamer combination of low energy. This problem can be solved using stochastic methods such as simulated annealing. The identification of the global minimum energy combination can also be achieved using methods such as dead-end elimination [24].

To obtain a library of designed sequences, a simple expedient is to repeatedly execute a stochastic design method or to design combinatorial mutation libraries to capture the sequence variation found within the pool of design solutions [25–27]. Such a library, however, will vary largely at sites that the CPD methods are found to be of marginal importance. Furthermore, if the CPD method confidently selects an unfavorable mutation (a systematic error), the poor choice could be present in all members of a library. Such an error could cause the entire library to be unfolded or nonfunctional. For example, a CPD algorithm with insufficient weight for van der Waal interactions might "overpack" the protein core, resulting in a molten globule sequence.

In contrast, an interpretable library of CPD variants could be designed to explicitly uncover and overcome systematic errors. A typical CPD scoring function assesses amino acid interactions as a series of contributions from hydrogen bonding, hydrophobic packing, van der Waals interactions, salt-bridge interactions, and other terms. A favorable design library would serve as a training set suitable for "learning" the weights associated with these different types of interactions. Whereas a CPD method might predict a stabilizing surface salt-bridge, a good library design would test this hypothesis. For example, if the library contains variants with the wild-type interaction, variants with the proposed salt-bridge, and variants with only one partner substituted, there is the possibility of determining the effective contribution of the salt-bridge. The concept is similar to the idea of a double-mutant cycle, although in this case the interaction is assessed in the presence of potentially confounding background sequence variation. If the library contains many such examples, the energy function could be trained to better predict the importance of salt-bridge interactions.

Recombination can be used to generate libraries that reduce the trade-off between library diversity and library stability; sequences generated through recombination are much more likely to retain stability than comparably diverse sequences generated through mutagenesis [28]. Similar to DNA shuffling [29, 30], site-directed recombination diversifies a library by substituting sequence blocks that contain multiple mutations. Recombination of homologous wild-type sequences has proven to be an effective library design method for a variety of protein folds including beta-lactamase [31], cytochrome P450 [32], arginase [33], and several cellulase families [34–37].

Recombination need not be limited to natural sequences; homologous parent sequences identified from directed evolution or CPD methods could also be recombined to create a diverse library. In the example below, we recombine one wild-type parent with two computationally designed sequences. Incorporating computational designs into a recombination library allows the designer to specifically target a library property of interest (e.g., stability at low pH). Energy scoring functions attempt to incorporate many global stability factors, including hydrogen bonds, hydrophobic interactions, packing efficiency, and conformational strain. By searching through a large sequence space, computational designs may identify improved variants that have never occurred in nature. As discussed above, CPD variants are likely to include design errors. Recombining blocks from CPD variants with a wild-type sequence will allow the dissection of stabilizing and destabilizing sequence motifs. Constructing a chimera sequence that incorporates successfully designed blocks from the CPD sequence and leaves out blocks corresponding to CPD failures is likely to result in chimeras that meet the design goals.

In the example below, we demonstrate how LCPD methods might be applied with a model target, cellobiohydrolase II (CBHII) from *Humicola insolens* (PDB entry 1OCN). In the first two sections, parent sequences are designed using CPD. We then recombine the parents to form a chimera library. The final section discusses how information from library screening could be used to enhance subsequent designs. We will not discuss the experimental construction of chimera libraries because protocols for site-directed chimeragenesis are thoroughly described in earlier reports [38, 39].

3.1 Phase I: Create a Design Palette

To begin a protein design problem we define a design palette: the set of candidate amino acids for each design position. Ideally, the design palette should be limited in size so that the resulting sequence space can be computationally searched in a reasonable time frame. Early zinc finger protein design work by Dahiyat and Mayo demonstrated the value of specifying a carefully selected design palette [40]. In this case the palette was restricted to alanine and hydrophilic residues (Ala, Ser, Thr, His, Asp, Asn, Glu, Gln, Lys, and Arg) at surface sites, hydrophobic residues (Ala, Val, Leu, Ile, Phe, Tyr, and Trp) at buried sites, and hydrophilic or hydrophobic residues at boundary sites. Furthermore, two sites with φ angles greater than 0° were restricted to Gly only. Even with this reduced design palette, the small 30-residue protein had a search space of 1.9×10^{27} possible sequences, or 1.1×10^{62} unique conformational variations. Modern computers and search algorithms can effectively search combinatorial solution spaces of this astounding size [24, 41], but such a diverse palette would not be feasible for proteins with hundreds of residues. One reason to use a design palette is to avoid buried hydrophilic amino acids and exposed hydrophobic amino acids. However, Kuhlman and co-workers recently reported a method for avoiding hydrophobic surface patches without eliminating them from the design palette altogether [42].

At the outset of a design challenge it can be difficult to calibrate the design palette. A conservative design palette would consist of relatively few design sites, and would avoid any mutations that are a priori likely to be disruptive. While a non-conservative palette may facilitate the design of a superior sequence, it will also allow more mutations, lead to a diverse library, and may result in a largely unfolded library. The balance between diversity and stability motivates an iterative approach; if the desired library "phenotype" and library stability are not achieved in the first round of library design, the palette can be adapted in subsequent iterations to be more or less conservative.

In this example, we design a very conservative palette intended to engender a largely folded library. First, prevalent mutations are identified from homologous multiple sequence alignments. While mutations commonly seen in consensus alignments are not guaranteed to be stabilizing, the selective pressure of evolution strongly suggests that these mutations are not destabilizing. Second, folding free energy changes are estimated for every point mutation. In principle, excluding mutations predicted to have unfavorable folding free energy changes will result in a smaller, conservative palette that is less likely to harbor destabilizing mutations.

1. Identify sequences with a high sequence identity to the query sequence.
   We used the Basic Local Alignment Search Tool (blast.ncbi.nlm.nih.gov) to identify similar sequences and retained alignments with sequence identity greater than 35 %. For the CBHII consensus analysis, 175 sequences met this cutoff criterion.
2. Identify common amino acids at each site and save in a consensus design palette.
   BLAST results were parsed using *run_alignment.py*. A cumulative approach was used that retained the most common amino acid, second most common amino acid, etc. at each site until 90 % of the sequences had been included.
3. Calculate predicted folding free energy changes ($\Delta\Delta G$) for each point mutation.
   Preparatory steps and FoldX calculations were executed using *run_foldx_multi.py*. All 20 amino acids were considered at each site. *See* **Note 1** for more information.
4. Combine all favorable mutations ($\Delta\Delta \leq 0$) in a secondary palette.
   FoldX outputs were parsed using *run_foldx_analysis.py*.
5. Repeat **steps 3** and **4** with alternate backbone scaffolds to account for slight variations in structure.
   Potential backbone scaffolds 1BVW, 2BVW, 1GZ1, and 1OC5 were identified by BLAST searching against the Protein Data Bank (PDB). Each chain from within a structural model was considered a unique backbone scaffold. *See* **Note 2**.
6. For a conservative design, reduce the design palette to the intersection between the consensus analysis and the multiple structural modeling palettes (Table 1).
   The script *run_consensus_foldx.py* was used to identify the intersection between multiple design palettes. We chose to include mutations allowed in the consensus palette and by the majority of the folding free energy palettes (arbitrarily defined as 2/3).

Table 1
Potential CBHII mutations at selected sites

WT A.A.	Consensus palette	FoldX palette	Intersection palette	Chosen A.A.	Rationale
N103	ANPSK	NP	NP	P	Allows h-bond between Y100 and E154 (Fig. 2a)
R123	AIKNRV	IKLMQRTV	IKRV	I	Computational model predicts favorable energy interactions (Fig. 2b)
Q361	GKLQSV	ILMQV	LQV	Q	Mutating Q361 loses side chain backbone h-bond (Fig. 2c)
K366	AEIKLNQST	FKLY	KL	K	K366L mutation introduces an unfavorable nonpolar surface residue (Fig. 2d)

3.2 Phase II: Select Parent Sequences

There are a few basic principles to consider when selecting parent proteins for recombination. First, parent proteins must have similar structure in order to remain folded upon recombination. If structural models are unavailable, sequence identity can be used to estimate structural similarity. Parent sequences with high sequence identity (60–80 % identity) generally have similar structure [43] and recombination will result in a high fraction of folded chimeras. In contrast, parent sequences with low sequence identity (<40 % identity) are much more likely to engender unfolded chimeras [32, 44, 45]. Second, critical residues should be conserved in each parent. Catalytic active site residues may be considered critical, since variants that do not conserve these amino acids are very unlikely to retain enzymatic activity. Other sites that may be considered critical include disulfide residues, sites for posttranslational modification (e.g., glycosylation sites), and sites that could affect the protein folding mechanism such as *cis*-prolines.

Combinatorial optimization software, such as SHARPEN (www.sharp-n.org), can be used to search for low-energy sequences that meet these criteria [46, 47]. Alternate side chain conformations (rotamers) are included from the backbone-dependent Dunbrack rotamer library [48]. Numerous algorithms exist for finding low-energy sequences and conformations. SHARPEN allows users to easily try a variety of stochastic algorithms (e.g., FasterPacker, SimulatedAnnealingPacker). Because these algorithms may yield different results for each repetition, repeated trials are useful for identifying mutations that are strongly or weakly preferred by the scoring function.

7. Given a design palette, search for low-energy sequences.
   We used the FasterPacker search algorithm in SHARPEN to identify low-energy candidates according to an all-atom Rosetta energy function [49]. This combinatorial optimization routine mimics the single-residue perturbation/relaxation method within the original description of the FASTER algorithm [41]. Briefly, this method systematically attempts to surmount local minima during optimization by temporarily fixing a single side chain in a particular conformation, and then assessing the effect of that perturbation combined with the relaxation/optimization of the neighboring side chains. Design calculations were performed using *run_conservative_design.py*. Separate calculations were run for the conservative and consensus design palettes. A total of 100 repetitions were performed for each design.
8. Sort candidate designs to identify the lowest energy design (Fig. 1).
   The list of designed protein models generated by SHARPEN was sorted using *run_sort_by_energy.py*. For the conservative design, the lowest energy sequence was 38 Rosetta energy

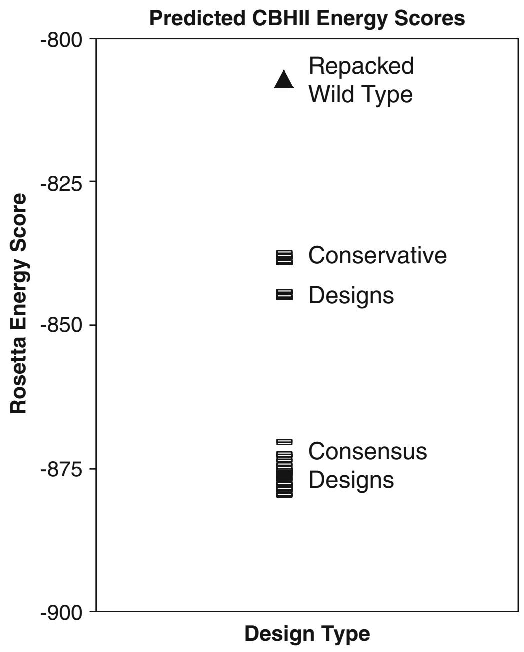


Fig. 1 Using a stochastic search algorithm in a design problem yields variants of differing energies. The distribution of potential low-energy candidates was sampled by performing 100 repetitions for each design palette. The starting energy score of 1OCN.pdb was −501 Rosetta energy units (REU). After repacking to optimize side chain conformations, the energy score was reduced to −807 REU (*triangle*). Searches based on the conservative design palette (intersection of consensus and FoldX methods) achieved an energy reduction of 38 REU (*rectangles*), while the larger consensus palette allowed an energy reduction of 72 REU (*rectangles*)

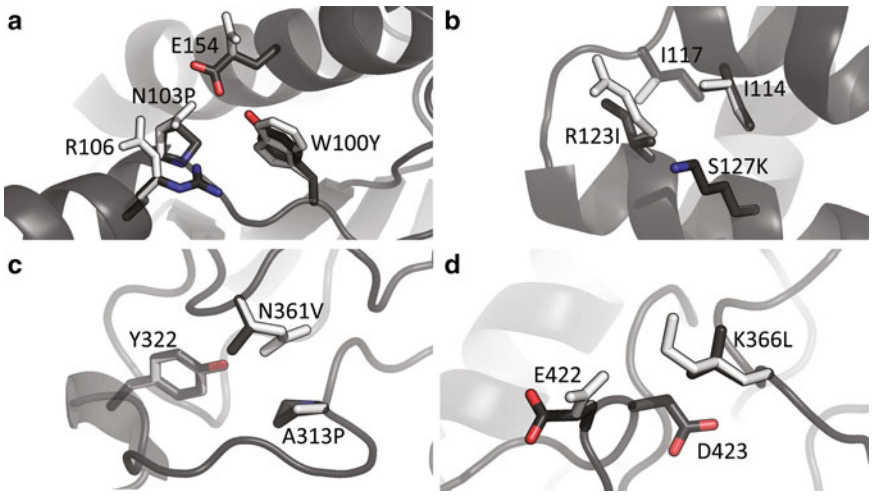


Fig. 2 Visual inspection of potential mutations. Example mutations include (**a**) W100Y and N103P, (**b**) R123I and S127K, (**c**) A313P and N361V, and (**d**) K366L. *See* Table 1 for discussion regarding which mutations were kept or reverted back to wild type

units lower than the wild-type sequence. The larger consensus design palette allowed a slightly more favorable energy change of 73 energy units.

9. Inspect designs to identify common mutations and stabilizing features (Fig. 2a–d).
   Given the limitations of contemporary sampling and scoring in CPD methods, visual inspection of the prospective mutations can provide an additional opportunity to ensure a reasonable design. The script *master.py* incorporates many analysis functions, including multiple sequence alignments (*run_multiple_sequence_alignment.py*), global energy comparisons (*run_compare_pdbs.py*), and amino acid polarity comparisons (*run_polarity_of_mutations.py*). *See* **Note 3** for more information.

The final conservative design contained a total of 58 mutations (84 % sequence identity to wild-type sequence), whereas the consensus design contained 120 mutations (66 % sequence identity to wild-type sequence).

3.3 Phase III: Recombine Parent Sequences to Form a Library

After parent sequences have been finalized, one must select the number of blocks to recombine. Block size influences library interpretability and library size. We define library interpretability as the extent to which it is possible to (1) rationalize the functionality of the library members in terms of structural detail and (2) deploy the experimental data to construct an improved, more predictive model for future designs. Small blocks can greatly improve library interpretability. Namely, small blocks have fewer mutations per block, allowing interesting changes in the protein fitness to be tracked to the responsible mutations. For example, Heinzelman et al. were able to isolate an individual stabilizing mutation C404S from recombined CBH II parents because the cognate block contained only ten other mutations [50]. However, dividing a parent sequence into small blocks can greatly increase library size. Library size can be determined from the number of blocks and the number of parent sequences; for three parent sequences divided into four blocks each, the resulting library size will be 3^4 or 81 chimeras. If the aim is experimental screening of all library members, the library should be sized according to the screening capacity. Recombining more blocks, of smaller size, will increase the library size exponentially.

Given a range of desired block sizes, various structure-guided methods can be used to determine ideal recombination sites. Methods such as SCHEMA [51], SIRCH [52], and OPTCOMB [17] aim to minimize the number of disruptive amino acid contacts in recombined chimeras. Using a slightly different method, FamClash combines clash detection with protein family sequence data to maximize chimera functionality [53]. The protocol in this chapter is built around the recombination as a shortest path problem (RASPP) method [54]. That said, the presented protocol could readily be adapted to incorporate an alternative method.

SCHEMA aims to maximize the number of folded library members by minimizing the number of novel amino acid interactions (not found in parent proteins) [31, 55]. Interactions are defined as heavy atom pairs (excluding backbone O and N and all H) within 4.5 Å in a parent protein. A predictive SCHEMA energy score "E" is assigned to represent the number of novel contacts within each chimera. A diversity parameter "m" specifies the number of mutations between each chimera and the closest parent. The average SCHEMA energy and mutation level of all chimeras within a library are denoted <E> and <m>, respectively. While considering <m> does provide a means of favoring diverse libraries, it does not lend itself to the design of interpretable libraries. We therefore propose a third metric H_{sbs}^{max}, which is the maximum Hamming number for a single block substitution. If a candidate library is dominated by one or a few large blocks H_{sbs}^{max} will be large and the library will be less interpretable because the effect of changing the large blocks will include the aggregate effect of many mutations. A small H_{sbs}^{max} indicates that any block substitution that is found to be important is less likely to have an obscure origin. Multi-scale enzymology, tracing an important block effect to the role of individual mutations, will be more feasible for such a library.

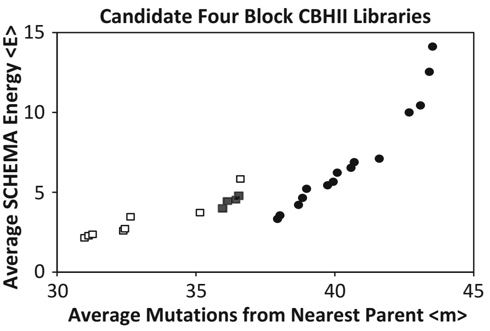
The library containing the minimum number of nonnative amino acid interactions can be determined by formulating the library optimization as a dynamic programming problem [54]. By weighting edges of a graph according to a SCHEMA penalty, a shortest path can be chosen that contains optimum block crossover sites. Further restrictions can be placed on the search space, such as limiting crossover sites to locations where three or four nucleotides

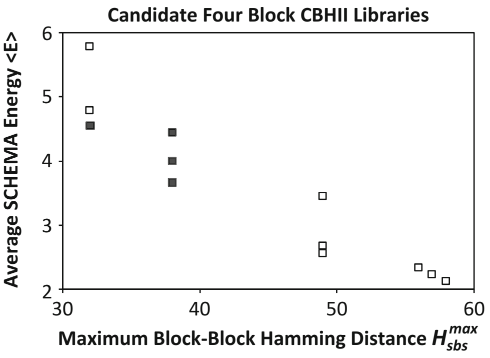
Table 2
RASPP settings used for CBHII recombination

Parameter	Value	Description
pdbfile	"1ocn.A.pdb"	Name of pdb file used to identify native contacts
Cutoff	4.5	Distance cutoff used to identify native contacts (Å)
Skipatoms	['N', 'O', 'H']	Atoms to be skipped when identifying native contacts (skips N and O in backbone only)
Numxo	3	Number of crossover sites (three crossover sites generate four blocks)
Overhang	3	Number of conserved nucleotides required at crossover sites (can be 0 if overhangs are unnecessary for library construction)
Min_lengths	Range (5,7)	Range of minimum block lengths

are preserved in all parent sequences. Conserving nucleotides at crossover sites allows blocks to be recombined using type II restriction enzymes [39].

10. Create a sequence alignment file based on the parent sequences. A number of programs are available for generating sequence alignment files; we used ClustalOmega (www.ebi.ac.uk/Tools/msa/clustalo/) and converted the format using *run_convert_msa_format.py*.
11. Specify the library design parameters (Table 2).
    Block size is the primary parameter in recombination problems. However, the provided code is engineered for flexibility. The user can specify how amino acid contacts are defined (minimum cutoff distances, and the heavy atoms considered), and which sites are feasible crossover locations (e.g., the number of nucleotides in overlap regions). These parameters can also be modified in the settings file *raspp_config.py*.
12. Identify potential crossover sites. For each candidate set of crossover sites, calculate <E>, <m>, and H_{sbs}^{max}.
    Running *run_raspp_curve.py* generated a list of optimum crossover sites. <E>, <m>, and H_{sbs}^{max} were saved in an output file pareto.csv for each set of sites.
13. Select a candidate library corresponding to a set of crossover sites.
    Ideally, the selected library will have low <E>, high <m>, and low H_{sbs}^{max}. Four potential CBHII libraries were identified from a plateau region on the <E>/<m> Pareto front (Fig. 3a). The <E>/ H_{sbs}^{max} Pareto front (Fig. 3b) allowed us to discriminate between these four candidate libraries and select a design that optimized diversity and interpretability. The final design features recombination sites 167, 244, and 345.

a


b


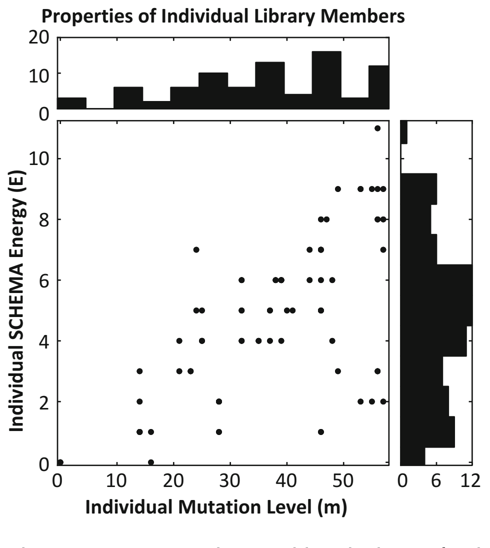
c


Fig. 3 Multiple design parameters can be considered when selecting a candidate library. (**a**) A Pareto front for four-block CBHII recombination with one wild-type and two conservatively designed parents (*squares*) has a similar average mutation level <m> and average SCHEMA energy <E> as four-block recombination with three wild-type parent sequences *Humicola insolens*, *Chaetomium thermophilum*, and *Hypocrea jecorina* (*filled circles*). Promising candidate libraries have low <E> and high <m> (*filled squares*). (**b**) Maximum block-block hamming distance H_{sbs}^{max} quantifies the interpretability of each library. Four similar candidate libraries from the <m> Pareto front (*filled squares*) are easily distinguished by the H_{sbs}^{max} Pareto curve. (**c**) Within the library featuring crossover sites 167, 244, and 345, chimeras have a distribution of mutation level "m" and SCHEMA energy "E"

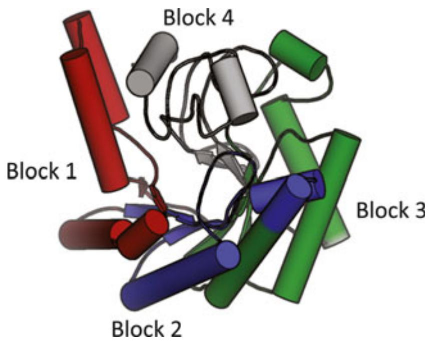


Fig. 4 Structural blocks identified using RASPP methods. Blocks are defined as follows: Block 1—residues 91–166 (*red*), Block 2—residues 167–243 (*blue*), Block 3—residues 244–344 (*green*), Block 4—residues 345–450 (*grey*)

14. Inspect the prospective design.
    (a) Generate histograms of chimera properties (Fig. 3c).
    Do outliers skew the library average properties? Do the distributions show that most library members have acceptable diversity and predicted disruption? Distributions can range from normal to multimodal, depending on the parent proteins. Our selected library showed an approximately normal <E> distribution and a slightly skewed <m> distribution.
    (b) Inspect the crossover sites and structural features of each block (Fig. 4).
    A candidate library design can be inspected using PyMOL (www.pymol.org). First load the parent pdb, and then run the corresponding showcontacts.pml script by typing *@showcontacts.pml.*[*recombination sites*] into the PyMOL command line. Blocks are colored based on selected crossover sites.
    (c) Verify that the library is constructible.
    Are the block sizes compatible with construction? Small DNA fragments could be difficult to purify using gel purification. If using a restriction enzyme-based construction protocol, ensure that the design produces the correct overhangs. If the candidate splice sites are not orthogonal, can alternate codons be used? If necessary, select a new candidate library from the Pareto front.

Another approach for selecting recombination crossover sites is to ignore the protein sequence and select sites solely on the basis of one protein structure. This alternate approach could be useful for preliminary library designs where CPD parent sequences have not yet been determined. In principle, structure-based crossover sites could be selected using a variety of approaches similar to domain detection algorithms [56]. However, to build a recombination library experimentally, the blocks should consist of contiguous residues.

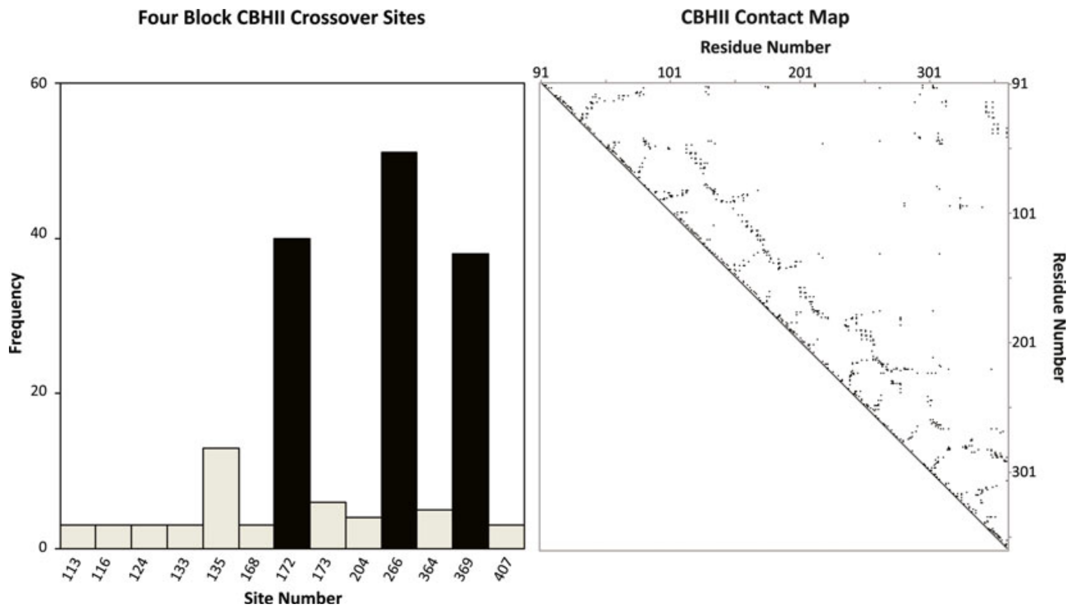


Fig. 5 (**a**) Scanning over a range of minimum block sizes from 5 to 90 creates a range of optimum block recombination sites. To identify preferred sites, all sites occurring in more than two unique libraries are considered. Recombination sites 172, 266, and 369 are preferred for a four-block CBHII library. (**b**) The contact map for 1OCN.pdb shows contacts characteristic of alpha helices and beta sheets. Ideal recombination blocks maximize intra-block contacts and minimize inter-block contacts

Therefore, a simple expedient is to reuse the dynamic programming approach of RASPP, but to replace the SCHEMA penalty matrix with a simple binary contact map. The resulting crossover sites will be those that minimize the number of inter-block contacts (and therefore maximize the number of intra-block contacts). We demonstrate this alternative method using *run_pick_modules.py*.

15. Identify crossover sites for a range of minimum block lengths. We used *run_pick_modules.py* to create a histogram of potential block crossover sites.
16. Select a set of preferred crossover sites and inspect structural blocks.

    In our example, sites 172, 266, and 369 were frequently chosen as crossover sites (Fig. 5a). *Pick_modules.py* strongly favors certain crossover sites that minimize the number of inter-block contacts. Notably, these sites are not obvious from inspection of the protein structure or the contact map (Fig. 5b).

3.4 Phase IV: Evaluate the Library

In any design cycle, the final step involves constructing and experimentally verifying the designs. Selected chimeras can be synthesized using traditional molecular biology techniques [39] or via gene synthesis and assayed to determine the extent of folding or

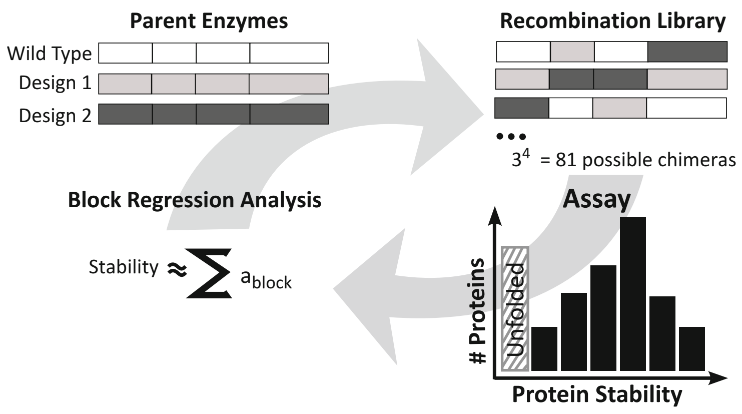


Fig. 6 Library design is an iterative cycle that consists of parent selection, block recombination, experimental testing, and validation of biophysical models

retained activity. In the CBHII example, an activity assay such as the Nelson-Somogyi reducing sugar assay could be performed at a variety of temperatures to test chimera function and stability [34].

Experimental verification of large libraries can be costly and time consuming. One approach is to experimentally screen a small percentage of the library and attempt to use the initial screening data to derive a predictive stability model applicable to the remainder of the library. Simple regression methods that model the stability of each chimera as the sum of contributions from each block have been found to be predictive [57]. The surprising additivity of block contributions to stability can be attributed to sequence conservation among the parents and the partitioning of epistatic interactions into structural modules [45].

To complete the iterative library design cycle, knowledge gained from experimental testing can be incorporated into subsequent designs (Fig. 6). In addition to predicting the fitness of library members, a trained regression model can also guide the refinement of the CPD methodology. For example, if a particular sequence block from one of the CPD design variants was found to be highly destabilizing, the deficiency in the CPD model can be investigated by reexamining the mutations that comprise that block.

4 Example 2: Localized Protein Design Libraries

4.1 Introduction

Many properties of a protein depend critically on a subset of the amino acids. Protein-protein binding, cofactor binding, enzyme specificity, and catalysis are all properties for which structural models can enable hypothesis-driven engineering of specific residues. The applications for focused protein library design are nearly limitless. Below, we briefly survey a selection of such applications before presenting an example protocol.

4.1.1 Protein-Protein Interface Design

Protein-protein interactions (PPIs) are fundamental to many of the biomolecular recognition events that drive biological processes. However, understanding PPIs is difficult because they typically involve many weak noncovalent bonds over large surfaces. The biophysical principles (e.g., extent of buried nonpolar surface area) underpinning protein-protein interfaces vary [58], and not all participating amino acids will contribute equally to the binding affinity [59].

Just as understanding PPIs plays a key role in molecular biology, the ability to control PPIs is key for engineering new therapeutic biomolecules. Baker and co-workers demonstrated an effective protocol de novo protein inhibitor design with a protein that binds an influenza virus stalk site [60, 61]. Engineering new PPIs as a CPD problem extends the methods deployed for monomeric CPD [10, 62]. Combinatorial optimization routines are applied to the interfacial amino acids to optimize a scoring function that includes van der Waals, hydrogen bonding, and electrostatic interactions with the partner protein. Notably, alternate approaches to engineer interactions can circumvent the need to engineer large complementary surfaces. Examples include the addition of shared metal-binding sites [63] or disulfide bonds [64].

An improved understanding of PPIs could also be useful for downstream problems in biotherapeutic development. For example, prevention of aggregation is important to extending the shelf life of therapeutic proteins. Unwanted PPIs could be destabilized through site-specific mutations of existing complementary interfaces or electrostatic repulsion via supercharging [4, 62, 65].

4.1.2 Binding Small Molecules

Binding of metals and small organic molecules is necessary for many proteins to function. Mutations of amino acids in the hydrophobic protein core can result in new cavities for small molecules to bind. For small nonpolar molecules, it is desirable to create a hydrophobic local environment around the cavity. Hecht and co-workers demonstrated that the simple truncation of Phe to Ala in the de novo protein S-824 resulted in the ability to bind small aromatic compounds [66]. Binding polar molecules and metals is more challenging because it requires the installation of complementary electrostatic interactions and hydrogen bonds. Notably, Matthews and co-workers have created cavities in T4 lysozyme that can bind the polar ligands pyridine, phenol, and aniline [67].

4.1.3 Changing Cofactor Specificity

Engineering organisms to produce higher yields of products via knockouts of competing metabolic pathways can create cofactor imbalances. Shifting cofactor specificity may resolve this problem by substituting the limiting cofactor with one that is in excess. For example, in attempts to anaerobically produce isobutanol in *Escherichia coli* via the Ehrlich pathway, NADPH-dependent

Table 3
Translation of degenerate codon base to nucleotides

Degenerate base	Actual base
N	A or C or G or T
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
K	G or T
M	A or C
R	A or G
S	C or G
W	A or T
Y	C or T

enzymes were engineered to shift the specificity preference to NADH. The best variant of the final library exhibited a specificity of 185:1 for NADH to NADPH, a 54,000-fold change from the original variant. By completely removing the dependence on NADPH, isobutanol titres at 100 % theoretical yield were achieved [68].

4.1.4 Degenerate Codons

A widely used approach for introducing amino acid diversity at a particular site is through the use of degenerate codons [69]. Degenerate codons are sets of oligonucleotides that code for multiple amino acids. The standard degenerate codon naming convention used in this text is presented in Table 3. Routine site saturation mutagenesis protocols often employ the degenerate codon NNK, which codes for all 20 amino acids [70]. While site saturation mutagenesis is simple and efficient, combinatorial explosion limits the number of sites that may be targeted. As the number of saturation sites increases, it rapidly becomes infeasible to transform, isolate, and thoroughly screen the resulting library. Even with a very-high-throughput screen, allowing for screening of 10^8–10^{11} targets [71], site saturation mutagenesis can only be performed on eight residues. Furthermore, NNK encodes the amino acids unevenly (Fig. 7). The resulting bias against rare amino acid combinations increases exponentially with the number of sites.

The limitations of site saturation mutagenesis motivate the development of more efficient methods that eschew brute force search. Table 4 illustrates various useful degenerate codon alternatives to NNK. These sets allow for introduction of diversity at a

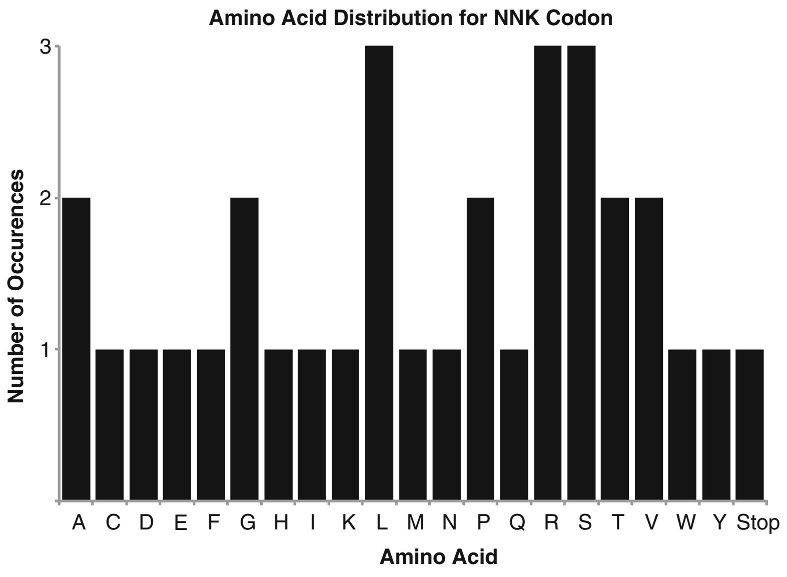


Fig. 7 Selection of the codon NNK unevenly encodes the amino acids. NNK also encodes for a stop codon, which will result in a nonfunctional variant

Table 4
Examples of degenerate codon to amino acid subset

Codon	Type	Amino acids	Stop codons	Unique codons
NNK	All 20 A.A.	All 20	TAG	32
DVT	Hydrophilic	A,C,D,G,N,S,T,Y	None	9
NVT	Charged, hydrophilic	C,D,G,H,N,P,R,S,T,Y	None	12
VVC	Hydrophilic	A,D,G,H,N,P,R,S,T	None	9
NTT	Hydrophobic	F,I,L,V	None	4
TDK	Hydrophobic	C,F,L,W,Y	TAG	6
TTN	Hydrophobic	F,L	None	4
(DSC/DST/DSY)	Small	A,C,G,S,T	None	5
GMT	Single-mutation alanine scanning	A,D	None	2
GMA		A,E	None	2
GST		A,G	None	2
SCA		A,P	None	2
KCC		A,S	None	2
RCT		A,T	None	2
GYT		A,V	None	2

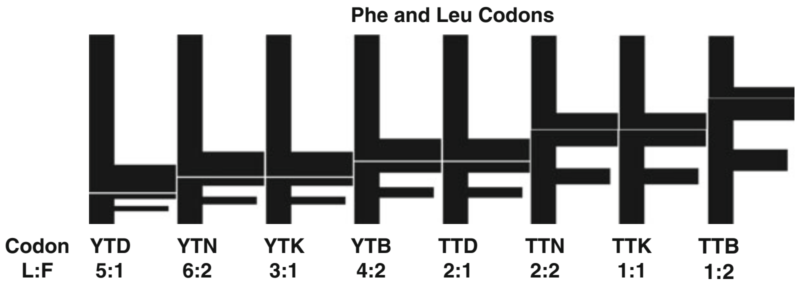


Fig. 8 Visualization of amino acid bias for codons exclusively encoding for L and F. Codon optimization must be performed to discriminate between codons encoding for similar amino acid ratios (i.e., 2:4 and 1:2)

site, but limit the mutations to a set of hypotheses. The foremost factor when considering a degenerate codon is the resulting set of amino acids. A secondary factor to consider is bias. For example, Fig. 8 illustrates how the set of amino acids containing exclusively Phe and Leu can be encoded by eight degenerate codons, with varying bias. Only the degenerate codons TTK and TTN encode Phe and Leu in equal proportions.

4.2 Approach

Due to the problems associated with site saturation mutagenesis, semi-rational methods have been developed for "intelligent" picking of codons to optimize either library size or amino acid ratios [72, 73]. We present a method below that uses an interactive python script (*codons.py*) for selecting site-specific degenerate codons. *Codons.py* allows a user to consider how alternate degenerate codons will drive the distribution of amino acids at a particular site, and to consider the library size and screening requirements that result from degenerate codons at multiple sites. To focus the discussion we will consider an illustrative example consisting of the cofactor switch of NADPH to NADH in ketol-acid reductoisomerase (KARI) [68, 74].

4.3 Phase I: Identification of Site Mutations

We will assume the availability of a structural model. The identification of specific target residues (e.g., active site, cofactor-binding site) can be accomplished via visual inspection in PyMOL [75] or via computational algorithms [76, 77]. Once the positions are selected, continued visual inspection will inform the decision of whether to use site saturation mutagenesis or a more limited subset of amino acids. Just as with site selection, the amino acid design palette can be based on calculations [78] or through biophysical intuition alone. The PyMOL mutation tool is an excellent prospective modeling technique for inspecting candidate mutations, since it allows rotamer sampling and indicates steric clashes. Thus visual inspection of candidate mutations may elucidate amino acids that are too large for the site or cannot avoid a detrimental interaction with existing amino acids/cofactors. Alternately, the scan of potential mutations and the conformations thereof can be automated.

Regardless, a list of favorable and unfavorable amino acids should be tabulated prior to use of *codons.py*. Under most circumstances, it is highly recommended that the wild-type amino acid be included in the design palette. One benefit is practical: including the wild-type amino acid will increase the fraction of the library that retains structure and function. Another benefit is philosophical: if the wild-type amino acid is an option for each design position, then the wild-type sequence should be a member of the library. With sufficient screening, such a library should yield this positive control.

In our cofactor switch example, the structure of the IlvC *E. coli* (without cofactor) was aligned to that of KARI spinach bound with NADPH. Five residues were identified for mutagenesis through proximity to the homologous NADPH position: R68, A71, R76, S78, and Q110. Residues R68, A71, R76, and S78 were selected due to their interactions with the 2′ phosphate group of the bound NADPH. Q110 was selected for its potential to orient the cofactor through interaction with the adenine moiety.

A strategy for shifting specificity from NADPH to NADH is to disrupt the salt bridge between positively charged residues interacting with the NADPH 2′ phosphate by mutations to negatively charged aspartic or glutamic acids [79]. Figure 9 illustrates how unfavorable interactions with NADPH could be formed via mutagenesis of each of the identified residues to Asp. Introducing negatively charged side chains can lower NADPH affinity and is sometimes sufficient to switch the cofactor specificity in favor of NADH [80]. However, improved specificity for NADH is often accompanied by loss of activity. As seen in Fig. 9, mutating S78 not only disrupts the salt bridge of NADPH 2′ phosphate, but it also might create a favorable hydrogen bond with the NADH hydroxyl group. We will use sites R68, R76, and S78 as examples in the degenerate codon design protocol (Table 5).

4.4 Phase II: Codons. py

Codons.py is a user-friendly tool for interactively selecting degenerate codons. The primary function of *codons.py* is to rank all prospective codons according to user-provided design goals. A set of required, taboo, preferred ("good"), and penalized ("bad") amino acids are provided either as arguments or interactive inputs to the main function. Subsequently, simple scoring methods rank the candidate codons that best fulfill the design objectives. Predefined amino acid sets can easily be specified and incorporated into the code. Aliphatic, hydrophobic, hydrophilic, acidic, and basic amino acid sets are predefined and can be input in place of individual amino acids. As outlined in detail below, scoring function options include the number of preferred amino acids encoded by a codon, the number of unique preferred amino acids encoded by a codon, and percentage of preferred amino acids out of amino acids in distribution encoded by a codon. Despite the simplicity of the scoring functions, the results nonetheless facilitate the sifting of many codon possibilities.

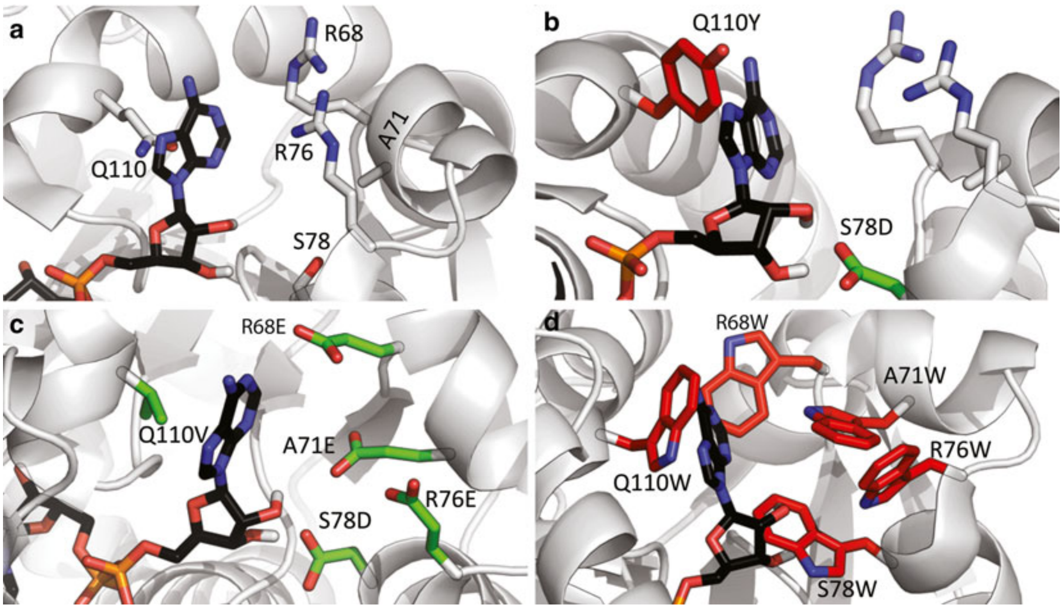


Fig. 9 Visualization of structural alignment between *E. coli* IlvC and Spinach KARI with NADPH bound in the active site. (**a**) Identification of R68, A71, R76, S78, and Q110 as potential residues for mutation in IlvC. (**b**) Depiction of favorable interaction created by S78D mutation and steric clashes created by Q110Y mutation. (**c**) Depiction of potential favorable mutations. (**d**) Depiction of unfavorable interactions from mutations to large residues

The scoring functions could be easily adapted if a more sophisticated scoring scheme is desired. The method for running *codons.py* is as follows:

1. Run *codons.py* interactively by entering the following into the command line: python codons.py (if manual usage is desired, enter python –i codons.py manual; *see* **Note 4** for examples of manual input).
2. The program will interactively ask for arguments (the help screen can be accessed at anytime by entering "?"):
   (a) Enter required amino acids.
   Set of amino acids that *must* be encoded. The wild-type amino acid is highly recommended for this set.
   (b) Enter good amino acids.
   Set of amino acids that give a positive score if encoded.
   (c) Enter bad amino acids.
   Set of amino acids that give a negative score if encoded.
   (d) Enter taboo amino acids.
   Set of amino acids that are not allowed to be encoded. For example, stop codons (denoted by an underscore) are typically designated as taboo.

Table 5
Hypotheses for NADPH cofactor switch example

Target Residue	Required	Preferred	Rationale
R68	R	E,D	Unfavorable interaction with 2′ phosphate group NADPH, potential hydrogen bonding with NADH
A71	A	E,D	Unfavorable interaction with 2′ phosphate group NADPH, potential hydrogen bonding with NADH
R76	R	E,D	Unfavorable interaction with 2′ phosphate group NADPH, potential hydrogen bonding with NADH
S78	S	E,D	Unfavorable interaction with 2′ phosphate group NADPH, potential hydrogen bonding with NADH
Q110	Q	–	No clear preference. Q110 mainly provides steric interaction
Target residue	**Taboo**	**Disfavored**	**Rationale**
R68	Stop	H,K F,W,Y	Favorable interaction with 2′ phosphate on NADPH Size
A71	Stop	P,G,S,T F,W,Y	Disfavored in alpha helix Size
R76	Stop	H,K F,W,Y	Favorable interaction with 2′ phosphate on NADPH Size
S78	Stop	H,K F,W,Y	Favorable interaction with 2′ phosphate on NADPH Size
Q110	Stop	P,G,S,T F,W,Y	Disfavored in alpha helix Size

(e) Enter desired scoring function.

Specify how to score the codons. The default scoring scheme is “distribution.”

- *Set*: In this mode, candidate degenerate codons will be assessed using the unique set of encoded amino acids. Scoring is performed by adding 1 if the amino acid is in the preferred set and subtracting 1 if the amino acid is in the bad set. A penalty of −1,000 is included if the amino acid is taboo or if a required amino acid is not encoded by the codon.
- *Distribution*: In this mode, candidate degenerate codons will be assessed using the distribution of amino acids encoded by each codon rather than just the set of unique amino acids. Each amino acid in the codon outcome distribution is scored. Scoring is performed by adding 1 if the amino acid is in the preferred set and subtracting 1 if the amino acid is in the bad set. If the codon includes a taboo amino acid or lacks a required amino acid, the score decreases by 1,000.

- *Percent*: In this mode, candidate degenerate codons will be assessed by scoring the percentage of the outcome amino acids (including the distribution bias) that appear in the "good" set. If required amino acids are not included in the distribution or if taboo amino acids are included a penalty of −1,000 is added.

(f) Specify output cutoff (integer).
Option that only prints codons that score above a value.

(g) Specify the output file name.
Option that prints output to specified file name.

3. Following user input, a table of the ten highest scoring codons will be displayed. If more results are desired, answer "y" to the prompt and type in the desired number of results.
4. After analysis of the table, the user is prompted to select a degenerate codon. Guidelines for selecting degenerate codons are presented in Phase III below.
5. Once a codon is selected for the site, the program asks if another site is desired. If selection of a degenerate codon for another site is desired, answer "y" and **steps 1–4** will be repeated.
6. As the user selects degenerate codons for multiple sites, a multi-site library is defined. Key parameters for a multi-site library include the number of unique variants and the bias in the amino acid distributions at the design positions. The screening (number of random clones) necessary to experimentally observe most of the library (e.g., 95 %) can be estimated using random sampling with the function library_sampling defined within *codons.py*.

Codons.py was run for each mutation site identified in Phase I using hypotheses discussed in Table 5. Sample output from running codons.py for site A71 is represented in Table 6.

4.5 Phase III: Codon Selection

Although the script *codons.py* is interactive, the final selection of a particular codon is manual. On the first attempt at selecting a codon, the ranked candidates should be inspected to determine the frequency of preferred amino acids to non-preferred amino acids (*see* **Note 5**). If the preferred amino acids do not appear frequently enough in the codons, consider rerunning *codons.py* with the preferred amino acid in the required list. The opposite is true as well; if a "bad" amino acid is appearing at too high of a frequency, consider moving that amino acid to the taboo list. Another key aspect of the interactive codon selection is the process of refining the design criteria in the light of the candidate codons. Typically, the candidate list will include codons that result in larger and smaller sets of amino acids, leading naturally to questions of screening capacity. Also, by considering the list of candidates, other tradeoffs are likely to surface. Potentially, one might be selecting between a panel of amino acids that includes all of the desired

Table 6
Sample *codons.py* output for NADPH cofactor switch example

Score	Amino acid distribution	Codons
4	AAAADDEEVVVV	GHN
4	AAAADDEE	GMN
3	AAADEE	GMD/GMV
3	AAADDEVVV	GHB/GHH
3	AAADEEVVV	GHD/GHV
3	AAADDE	GMB/GMH
2	AAEEVV	GHR
2	AAAADDEEKKNNTTTT	RMN
2	AADD	GMY
2	AAADDEKNNTTT	RMB/RMH
2	AADDVV	GHY
2	AADDIINNTTVV	RHY
2	AADEVV	GHK/GHM/GHS/GHW
2	AAADDEIIIKNNTTTVVV	RHH
2	AADDNNTT	RMY
2	AAAADDEEIIIKKMNNTTTTVVVV	RHN
2	AAEE	GMR
2	AADE	GMK/GMM/GMS/GMW
2	AAADDEIIKMNNTTTVVV	RHB
1	ADNT	RMC/RMT

amino acids but also includes an amino acid that is likely to be incompatible with the protein conformation. The user must decide if that codon is preferable to an alternative that avoids the destabilizing option but covers fewer of the favored amino acids. At this stage, it is worth reconsidering how the amino acids that appear in favored codons, but were neither assigned as "good" or "bad," are likely to perform. We suggest evaluating amino acids interactively in PyMOL using the Mutagenesis wizard with the following checklist in mind:

1) Does the mutation clash with the protein backbone?
2) Does the mutation clash with existing side chains?
    (a) If there is a clash with a neighboring side chain, can the neighbor move?

3) Does the mutation clash with an existing water molecule?
   (a) Can the water molecule be displaced without the loss of favorable interactions?
4) Does the mutation clash with a bound substrate?
5) If favorable interaction with a bound substrate is a design criterion, can a candidate mutation make favorable interaction(s) considering size and hydrogen bonding geometry?
6) If an unfavorable interaction with a bound substrate is a design criteria, can a candidate mutation avoid making the unfavorable interactions?

After running *codons.py* for each mutation site, a list of optimum codons was identified (Table 7). Given the availability of a high-throughput screen to determine NADPH/NADH binding (fluorescence of NADPH/NADH) [68], codons encoding high diversity at sites R68 and R76 were allowed. While A71 can accommodate many mutations, the palette was restricted to favor diversity at the neighboring design positions. As a result, codons encoding only the hypothesized residues and the WT were selected. Considering the strong preference for the S78D mutation, D was included in the required set for *codons.py*. The "percent" scoring function was used to identify codons that provided the highest percent coverage of D and E in the resulting amino acid distributions. Finally, due to lack of clear hypotheses for site Q110, only

Table 7
Favorable codons for NADPH cofactor switch example

Site	Candidate codons	Amino acid distribution
R68/R76	VDN	DDEEGGGGHHIIIKKLLLLMNNQQRRRRRRSSVVVV
	RRN	DDEEGGGGKKNNRRSS
	RNN	AAAADDEEGGGGIIIKKMNNRRSSTTTTVVVV
A71	GMN	AAAADDEE
	GMK/GMM/GMS/GMW	AADE
	GMD/GMV	AAADEE
S76	RNN	AAAADDEEGGGGIIIKKMNNRRSSTTTTVVVV
	RRK,RRM,RRS,RRW	DEGGKNRS
	RRC,RRT	DGNS
Q110	VWN	DDEEHHIIIKKLLLLMNNQQVVVV
	VWH	DDEHHIIIKLLLNNQVVV
	VWR	EEIKKLLMQQVV

Table 8
Final codon selection for NADPH cofactor switch example

Site	Final codon	Distribution	Rationale
R68/R76	RNN	AAAADDEEGGGGIIIK KMNNRRSSTTTTVVVV	(1) Introduction of diversity to these sites with good representation of preferred mutations (11 % frequency). (2) No large amino acids included in set. (3) Small frequency of bad amino acids
A71	GMK,GMM, GMS,GMW	AADE	(1) Lowest A:D:E ratio that encodes exclusively for A,D,E. (2) Limited diversity at this site is not unfavorable due to high diversity at other sites. (3) WT contributes to 50 % of encoded distribution—potentially helpful due to high diversity at other sites which might require A71 to avoid steric clashes
S78	RRK,RRM, RRS,RRW	DEGGKNRS	Favorable interaction with hydroxyl group appears at a high frequency (25 %)
Q110	VWN	DDEEHHIIIKKLLLLMN NQQVVVV	(1) Good diversity of smaller amino acids. (2) No large amino acids included in set

disfavored amino acids were specified. Codons at Q110 were thus ranked highly if they encoded high diversity and excluded large residues.

From the selection of the top candidates, final codons were selected as presented in Table 8. Depending on the degenerate codon candidates, codon optimization for the selected expression system (e.g., avoiding rare codons) could help discriminate between candidates that result in different distributions of the same amino acids (e.g., AAAALL and AAL) [81].

4.6 Phase IV: Experimental Synthesis

Commercial oligonucleotide providers (e.g., Integrated DNA Technologies, IDT) can synthesize primers with a mixture of wild-type and non-wild-type nucleotides. If only a single-mutation site or multiple-mutation sites in close proximity are desired, introduction of a degenerate codon can be accomplished with a single PCR [82]. However, if the desired sites are distant from one another, more extensive protocols must be used [70]. If the desired distribution of amino acids is not possible by a degenerate codon, mixing oligonucleotides is an alternative option [78].

5 Notes

1. Numerous programs exist for estimating folding free energy change, including FoldX, I-Mutant2.0, Eris, and sMMGB [83–86]. We chose the commonly used, semiempirical FoldX

for CBHII calculations. Estimating folding free energy changes for 20 amino acids at 358 sites created a computationally intensive calculation. Computing all FoldX calculations for six different backbones took approximately 3 days on a 2.6 GHz CPU.

2. Traditional CPD relies on fixed-backbone combinatorial optimization of side chain positions and amino acid identity. However, small differences in the backbone position can make a large difference in the ability of amino acids to be favorably placed at a given design position. Using an ensemble of reasonable backbone models provides a more realistic approximation of the protein backbone flexibility. This strategy is a partial substitute for true flexible-backbone design algorithms [24, 87].

3. In inspecting the designs, we checked for the loss of hydrogen bonds or the addition of questionable nonpolar surface mutations. Detailed pairwise energy comparisons (*run_evaluate_mutations.py*), combined with visual inspection, constituted the additional analysis of each proposed mutation. If we could not identify the rationale for a mutation chosen by SHARPEN, we performed a secondary search with additional rotamers near the questionable residue (*run_questionable_mutations.py*). All rotamers with chi angles within two standard deviations of the default Dunbrack rotamer library angles were included. Mutations that were still considered favorable in this secondary search were included in the final design. Otherwise, we used *run_mutate_to_wt.py* to revert mutations back to the wild-type amino acid variant.

4. Manual input of arguments in codons.py can be accessed by typing the following into the command line: *python –i codons.py manual*. Manual input allows quick and easy iterations for experienced users. Example inputs are given below for the identification of small replacements for leucine:

   pickcodons(good='AGVLIST', bad='WYFHRKED', taboo='_', required='L', scoring='percent', outfile=codons.txt')

   librarysize=compute_library_size('stringofcodons')

5. As a general rule, it is best to start with soft constraints on required and taboo mutations (i.e., only include WT in required and "_" in taboo). After evaluation of results, if a suitable distribution is not located, or visual inspection merits further discrimination for or against certain amino acids, then mutations may be moved into the required or taboo categories.

6. If a desired amino acid distribution can be encoded by multiple codons, codon optimization can be performed to discriminate between codons that encode for similar ratios.

References

1. Romero PA, Arnold FH (2009) Exploring protein fitness landscapes by directed evolution. Nat Rev Mol Cell Biol 10:866–876
2. Tracewell CA, Arnold FH (2009) Directed enzyme evolution: climbing fitness peaks one amino acid at a time. Curr Opin Chem Biol 13:3–9
3. Kuhlman B, Dantas G, Ireton GC, Varani G, Stoddard BL, Baker D (2003) Design of a novel globular protein fold with atomic-level accuracy. Science 302:1364
4. Miklos AE, Kluwe C, Der BS, Pai S, Sircar A, Hughes RA et al (2012) Structure-based design of supercharged, highly thermoresistant antibodies. Chem Biol 19:449–455
5. Grigoryan G, Reinke AW, Keating AE (2009) Design of protein-interaction specificity gives selective bZIP-binding peptides. Nature 458: 859–864
6. Röthlisberger D, Khersonsky O, Wollacott AM, Jiang L, DeChancie J, Betker J et al (2008) Kemp elimination catalysts by computational enzyme design. Nature 453:190–195
7. Privett HK, Kiss G, Lee TM, Blomberg R, Chica RA, Thomas LM et al (2012) Iterative approach to computational enzyme design. Proc Natl Acad Sci U S A 109:3790–3795
8. Dantas G, Kuhlman B, Callender D, Wong M, Baker D (2003) A large scale test of computational protein design: folding and stability of nine completely redesigned globular proteins. J Mol Biol 332:449–460
9. Chica RA, Doucet N, Pelletier JN (2005) Semi-rational approaches to engineering enzyme activity: combining the benefits of directed evolution and rational design. Curr Opin Biotechnol 16:378–384
10. Karanicolas J, Com JE, Chen I, Joachmiak LA, Dym O, Peck SH et al (2011) A de novo protein binding pair by computational design and directed evolution. Mol Cell 42:250–260
11. Khersonsky O, Kiss G, Röthlisberger D, Dym O, Albeck S, Houk KN et al (2012) Bridging the gaps in design methodologies by evolutionary optimization of the stability and proficiency of designed Kemp eliminase KE59. Proc Natl Acad Sci U S A 109:10358–10363
12. Bloom JD, Labthavikul ST, Otey CR, Arnold FH (2006) Protein stability promotes evolvability. Proc Natl Acad Sci U S A 103:5869–5874
13. Gromiha MM (2007) Prediction of protein stability upon point mutations. Biochem Soc Trans 35:1569–1573
14. Kamtekar S, Schiffer JM, Xiong H, Babik JM, Hecht MH (1993) Protein design by binary patterning of polar and nonpolar amino acids. Science 262:1680
15. Bradley LH, Thumfort PP, Hecht MH (2006) De novo proteins from binary-patterned combinatorial libraries. Methods Mol Biol 340: 53–69
16. Bradley LH, Wei Y, Thumfort P, Wurth C, Hecht MH (2007) Protein design by binary patterning of polar and nonpolar amino acids. Methods Mol Biol 352:155–166
17. 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
18. Steipe B, Schiller B, Plückthun A, Steinbacher S (1994) Sequence statistics reliably predict stabilizing mutations in a protein domain. J Mol Biol 240:188–192
19. Lehmann M, Kostrewa D, Wyss M, Brugger R, D'Arcy A, Pasamontes L et al (2000) From DNA sequence to improved functionality: using protein sequence comparisons to rapidly design a thermostable consensus phytase. Protein Eng 13:49–57
20. Amin N, Liu A, Ramer S, Aehle W, Meijer D, Metin M et al (2004) Construction of stabilized proteins by combinatorial consensus mutagenesis. Protein Eng Des Sel 17:787
21. Kono H, Wang W, Saven JG (2007) Combinatorial protein design strategies using computational methods. Methods Mol Biol 352:3–22
22. Dunbrack RL Jr (2002) Rotamer libraries in the 21st century. Curr Opin Struct Biol 12: 431–440
23. Shetty RP, De Bakker PIW, DePristo MA, Blundell TL (2003) Advantages of fine-grained side chain conformer libraries. Protein Eng 16:963–969
24. 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 81:18–39
25. Mena MA, Treynor TP, Mayo SL, Daugherty PS (2006) Blue fluorescent proteins with enhanced brightness and photostability from a structurally targeted library. Nat Biotechnol 24:1569–1571
26. 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 U S A 107:19838–19843

27. Chica RA, Moore MM, Allen BD, Mayo SL (2010) Generation of longer emission wavelength red fluorescent proteins using computationally designed libraries. Proc Natl Acad Sci U S A 107:20257–20262
28. Drummond DA, Silberg JJ, Meyer MM, Wilke CO, Arnold FH (2005) On the conservative nature of intragenic recombination. Proc Natl Acad Sci U S A 102:5380
29. Stemmer WPC (1994) Rapid evolution of a protein in vitro by DNA shuffling. Nature 370:389–391
30. Harayama S (1998) Artificial evolution by DNA shuffling. Trends Biotechnol 16:76–82
31. Meyer MM, Silberg JJ, Voigt CA, Endelman JB, Mayo SL, Wang ZG et al (2003) Library analysis of SCHEMA-guided protein recombination. Protein Sci 12:1686–1693
32. Otey CR, Landwehr M, Endelman JB, Hiraga K, Bloom JD, Arnold FH (2006) Structure-guided recombination creates an artificial family of cytochromes P450. PLoS Biol 4:e112
33. Romero PA, Stone E, Lamb C, Chantranupong L, Krause A, Miklos AE (2012) SCHEMA designed variants of human arginase I & Ii reveal sequence elements important to stability and catalysis. ACS Synth Biol 1:221–228
34. Heinzelman P, Snow CD, Wu I, Nguyen C, Villalobos A, Govindarajan S et al (2009) A family of thermostable fungal cellulases created by structure-guided recombination. Proc Natl Acad Sci U S A 106:5610–5615
35. Heinzelman P, Komor R, Kanaan A, Romero P, Yu X, Mohler S 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
36. Komor RS, Romero PA, Xie CB, Arnold FH (2012) Highly thermostable fungal cellobiohydrolase I (Cel7A) engineered using predictive methods. Protein Eng Des Sel 25:827–833
37. Smith MA, Rentmeister A, Snow CD, Wu T, Farrow MF, Mingardon F et al (2012) A diverse set of family 48 bacterial cellulases created by structure-guided recombination. FEBS J 279:4453–4465
38. Hiraga K, Arnold FH (2003) General method for sequence-independent site-directed chimeragenesis. J Mol Biol 330:287–296
39. Farrow MF, Arnold FH (2010) Combinatorial recombination of gene fragments to construct a library of chimeras. Curr Protoc Protein Sci Chapter 26, Unit 26.2
40. Dahiyat BI, Mayo SL (1997) De novo protein design: fully automated sequence selection. Science 278:82–87
41. 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
42. Jacak R, Leaver-Fay A, Kuhlman B (2012) Computational protein design with explicit consideration of surface hydrophobic patches. Proteins 80:825–838
43. Chothia C, Lesk AM (1986) The relation between the divergence of sequence and structure in proteins. EMBO J 5:823–826
44. Meyer MM, Hiraga K, Arnold FH (2006) Combinatorial recombination of gene fragments to construct a library of chimeras. Curr Protoc Protein Sci Chapter 26, Unit 26.2
45. Romero PA, Arnold FH (2012) Random field model reveals structure of the protein recombinational landscape. PLoS Comput Biol 8: e1002713
46. Loksha IV, Maiolo JR 3rd, Hong CW, Ng A, Snow CD (2009) SHARPEN-systematic hierarchical algorithms for rotamers and proteins on an extended network. J Comput Chem 30: 999–1005
47. Leaver-Fay A, Tyka M, Lewis SM, Lange OF, Thompson J, Jacak R et al (2011) ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol 487:545–574
48. Dunbrack RL Jr, Cohen FE (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 6:1661–1681
49. Rohl CA, Strauss CEM, Misura KMS, Baker D (2004) Protein structure prediction using Rosetta. Methods Enzymol 383:66–93
50. Heinzelman P, Snow CD, Smith MA, Yu X, Kannan A, Boulware K et al (2009) SCHEMA recombination of a fungal cellulase uncovers a single mutation that contributes markedly to stability. J Biol Chem 284:26229–26233
51. Voigt CA, Martinez C, Wang ZG, Mayo SL, Arnold FH (2002) Protein building blocks preserved by recombination. Nat Struct Mol Biol 9:553–558
52. Moore GL, Maranas CD (2003) Identifying residue–residue clashes in protein hybrids by using a second-order mean-field approach. Proc Natl Acad Sci U S A 100:5091
53. Saraf MC, Horswill AR, Benkovic SJ, Maranas CD (2004) FamClash: a method for ranking the activity of engineered enzymes. Proc Natl Acad Sci U S A 101:4142
54. 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

55. Silberg JJ, Endelman JB, Arnold FH (2004) SCHEMA-guided protein recombination. Methods Enzymol 388:35–42

56. Ingolfsson H, Yona G (2008) Protein domain prediction. Methods Mol Biol 426:117–143

57. Li Y, Drummond DA, Sawayama AM, Snow CD, Bloom JD, Arnold FH (2007) A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. Nat Biotechnol 25:1051–1056

58. Jones S, Thornton JM (1996) Principles of protein–protein interactions. Proc Natl Acad Sci U S A 93:13–20

59. Grosdidier S, Fernández-Recio J (2008) Identification of hot-spot residues in protein-protein interactions by computational docking. BMC Bioinformatics 9:447

60. Fleishman SJ, Whitehead TA, Ekiert DC, Dreyfus C, Corn JE, Strauch EM et al (2011) Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. Science 332:816–821

61. Whitehead TA, Chevalier A, Song Y, Dreyfus C, Fleishman SJ, De Mattos C et al (2012) Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing. Nat Biotechnol 30: 543–548

62. Kortemme T, Baker D (2004) Computational design of protein–protein interactions. Curr Opin Chem Biol 8:91–97

63. Salgado EN, Radford RJ, Tezcan FA (2010) Metal-directed protein self-assembly. Acc Chem Res 43:661–672

64. Ballister ER, Lai AH, Zuckermann RN, Cheng Y, Mougous JD (2008) In vitro self-assembly of tailorable nanotubes from a simple protein building block. Proc Natl Acad Sci U S A 105:3733–3738

65. Lawrence MS, Phillips KJ, Liu DR (2007) Supercharging proteins can impart unusual resilience. J Am Chem Soc 129: 10110–10112

66. Das A, Wei Y, Pelczer I, Hecht MH (2011) Binding of small molecules to cavity forming mutants of a de novo designed protein. Protein Sci 20:702–711

67. Liu L, Baase WA, Michael MM, Matthews BW (2009) Use of stabilizing mutations to engineer a charged group within a ligand-binding hydrophobic cavity in T4 lysozyme. Biochemistry 48:8842–8851

68. Bastian S, Liu X, Meyerowitz JT, Snow CD, Chen MM, Arnold FH (2011) Engineered ketol-acid reductoisomerase and alcohol dehydrogenase enable anaerobic 2-methylpropan-1-ol production at theoretical yield in *Escherichia coli*. Metab Eng 13:345–352

69. Tang L, Gao H, Zhu X, Wang X, Zhou M, Jiang R (2012) Construction of "small-intelligent" focused mutagenesis libraries using well-designed combinatorial degenerate primers. Biotechniques 52:149–158

70. Georgescu R, Bandara G, Sun L (2003) Saturation mutagenesis. Methods Mol Biol 231:75–83

71. Denault M, Pelletier JN (2007) Protein library design and screening: working out the probabilities. Methods Mol Biol 352:127–154

72. Mena MA, Daugherty PS (2005) Automated design of degenerate codon libraries. Protein Eng Des Sel 18:559–561

73. Patrick WM, Firth AE (2005) Strategies and computational tools for improving randomized protein libraries. Biomol Eng 22:105–112

74. Bastian S, Arnold FH (2012) Reversal of NAD(P)H cofactor dependence by protein engineering. Methods Mol Biol 834:17–31

75. Schrödinger L (2010) The PyMOL molecular graphics system, version 1.3r1

76. Fischer JD, Mayer CE, Söding J (2008) Prediction of protein functional residues from sequence by probability density estimation. Bioinformatics 24:613–620

77. Sankararaman S, Sha F, Kirsch JF, Jordan MI, Sjölander K (2010) Active site prediction using evolutionary and structural information. Bioinformatics 26:617–624

78. Chen MMY, Snow CD, Vizcarra CL, Mayo SL, Arnold FH (2012) Comparison of random mutagenesis and semi-rational designed libraries for improved cytochrome P450 BM3-catalyzed hydroxylation of small alkanes. Protein Eng Des Sel 25:171–178

79. Scrutton NS, Berry A, Perham RN (1990) Redesign of the coenzyme specificity of a dehydrogenase by protein engineering. Nature 343: 38–43

80. Rane MJ, Calvo KC (1997) Reversal of the nucleotide specificity of ketol acid reductoisomerase by site-directed mutagenesis identifies the NADPH binding site. Arch Biochem Biophys 338:83–89

81. Fuglsang A (2003) Codon optimizer: a freeware tool for codon optimization. Protein Expr Purif 31:247–249

82. Chiang LW, Kovari I, Howe MM (1993) Mutagenic oligonucleotide-directed PCR amplification (Mod-PCR): an efficient method for generating random base substitution mutations in a DNA sequence element. PCR Methods Appl 2:210–217

83. Guerois R, Nielsen JE, Serrano L (2002) Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. J Mol Biol 320:369–387
84. Capriotti E, Fariselli P, Casadio R (2005) I-Mutant2. 0: predicting stability changes upon mutation from the protein sequence or structure. Nucleic Acids Res 33:W306–W310
85. Yin S, Ding F, Dokholyan NV (2010) Computational evaluation of protein stability change upon mutations. Methods Mol Biol 634:189–201
86. Zhang Z, Wang L, Gao Y, Zhang J, Zhenirovskyy M, Alexov E (2012) Predicting folding free energy changes upon single point mutations. Bioinformatics 28: 664–671
87. Mandell DJ, Kortemme T (2009) Backbone flexibility in computational protein design. Curr Opin Biotechnol 20:420–428