DE NOVO PROTEINS DESIGNED FROM EVOLUTIONARY PRINCIPLES

Timothy Michael Jacobs

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Bioinformatics and Computational Biology.

Chapel Hill
2015

Approved by:

Brian Kuhlman

Jack Snoeyink

Alexander Tropsha

Edward Collins

Jane Richardson

ABSTRACT

Timothy Michael Jacobs: DE NOVO PROTEINS DESIGNED FROM EVOLUTIONARY PRINCIPLES
(Under the direction of Brian Kuhlman)

Protein engineering has rapidly developed into a powerful method for the optimization, alteration, and creation of protein functions. Current protein engineering methods fall into the category of either high-throughput directed evolution techniques, or engineering through the use of computational models of protein structure. Despite significant innovation in both of these categories, neither is capable of handling the most difficult and desirable protein engineering goals. The combination of these two categories is an area of active research, and the development and testing of combination methods is the focus of this dissertation. Chapters 2 and 3 describe the development of a computational framework for de novo protein design called SEWING (**S**tructural **E**xtension **WI**th **N**ative-fragment **G**raphs). In contrast to existing methods of de novo design, which attempt to design proteins that match a designer-supplied target topology, SEWING generates large numbers of diverse protein structures. We show that this strategy is highly effective at creating diverse helical backbones. Experimental characterization of SEWING designs shows that the experimental structures match the design models with sub-angstrom root mean square deviation (RMSD). Chapter 3 extends this methodology to the creation of protein interfaces. Using this method, several de novo designed proteins are created that bind their designated target. Chapter 4 describes the combination of directed evolution and computational modeling through the improvement of directed evolution techniques. In this chapter, a web tool called SwiftLib is developed, which allows rapid generation of degenerate codon libraries. SwiftLib allows protein engineers to determine optimal degenerate codon primers for the incorporation of desired sequences, such as sequence profiles generated from computational modeling and evolutionary data. Together, these chapters

outline the creation of tools for the engineering of protein functions, and provide additional evidence that computational modeling and evolutionary principles can be combined for the improvement of protein engineering methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The focus of this dissertation is the development of computational tools that aid in various aspects of protein engineering. This introduction will cover the goals and challenges related to protein engineering, and review various protein engineering methods along with their strengths and limitations.

1.1 Introduction to Protein Engineering

Proteins are responsible for a diverse array of cellular functions, including chemical catalysis, intra- and inter- cellular signaling, molecular scaffolding, and immune response. Protein engineering encompasses the optimization, alteration, and creation of protein structures and functions. The earliest studies in protein engineering involved the solid-state synthesis of peptide fragments derived from naturally occurring enzymes[1, 2]. These studies, along with crystal structures of naturally occurring proteins, helped elucidate the basic determinants of protein structure and function, and led quickly to the first design of a functional synthetic peptide[3]. The subsequent development of modern molecular biology techniques allowed researchers to test modifications to full-length proteins, and initial efforts resulted in the reprogramming of enzyme specificity, the selection of a catalytic antibody, and the development of entirely novel sequences that adopt well-folded tertiary structures[4, 5, 6]. These successes demonstrated important proof that engineered proteins could be made to adopt a variety of new structures and functions, and set the stage for an impressively rapid advancement in protein engineering efforts. Today, the implications of protein engineering are ubiquitous: pharmaceuticals, such as engineered insulin analogs, are developed to improve pharmacoki-

netic and pharmacodynamic properties; therapeutic antibodies for the treatment of cancers and autoimmune diseases are frequently engineered to exhibit improved binding affinity and reduced antigenicity[7, 8]. Outside the clinic, engineered proteins have found extensive use as research tools, allowing live-cell imaging, light-mediated cellular probing, and a wide variety of other commonly used biochemical techniques[9, 10, 11, 12, 13].

1.2 Methods and Strategies for Protein Engineering

1.2.1 Directed Evolution

Directed evolution is a widely used experimental strategy that broadly describes the generation of DNA libraries, and the selection of encoded proteins based on a chosen phenotype. DNA libraries can be generated by strategies that vary widely in their ease of use, level of control, and cost. The earliest described techniques utilize processes that mimic natural protein evolution, and offer the lowest level of control. DNA shuffling is one such technique, in which homologous DNA sequences are digested and recombined[14, 15]. Another evolutionarily inspired technique is error-prone polymerase chain reaction (PCR), in which specific buffer conditions and non-proofreading polymerases are used to increase the error rate of PCR reactions[16]. With both techniques, little control over the specific location and identity of inserted mutations is possible. Despite this limitation, the low barrier to entry and the ability to incorporate mutations throughout an entire gene has led to widespread use. The several successes resulting from libraries generated using these mechanisms demonstrate the power of natural evolutionary processes in the generation of functional diversity, a principle that will be applied to computational design in Chapters 2 and 3

The rapid development of DNA synthesis techniques over the past two decades has led to a number of possibilities for DNA library generation. Specifically, synthesis technologies have evolved to allow the addition of nucleotide mixtures at specific positions in short oligonucleotide sequences. The result of this technique is PCR oligonucleotides that contain degenerate codons, which introduce a subset of amino-acid mutations at desired positions. In this way, the diversity of DNA libraries can be restricted to favor desired amino acid sequences. These non-random

libraries allow researchers to impart external information, such as computational prediction and evolutionary information into generated libraries. Unfortunately, the degeneracy of the genetic code and the limitations of current screening strategies complicate this process, which is further discussed in Chapter 4. It is noteworthy that more exacting control of DNA diversity can be achieved through mixing of specific trinucleotide phosphoramidite primers, but the resultant combinatorial expansion in required primers restricts the use of such libraries to highly specialized applications[17].

DNA library generation can easily create libraries that are much larger than can be reasonably screened by existing high-throughput techniques, such as phage-, yeast-, mRNA- and ribosome-display, which range in their screening capacity from approximately 10^7 to 10^{13} DNA sequences[18]. Screening of protein sequences involves the coupling of protein function to the DNA sequence that encodes it. In the case of phage and yeast display, DNA library members are combined with plasmids that allow the presentation of the functional protein on the surface of a yeast cell or bacteriophage. In the case of mRNA and ribosome display, in-vitro translation techniques are used to couple the protein-encoding RNA to the fully translated protein.

The applications of directed evolution are restricted to engineering problems in which the desired phenotype can be readily screened for. The most common use cases for directed evolution are the generation and optimization of binding affinity for a target molecule, and the generation and optimization catalytic activity. In the case of evolving binding affinity this selection is often achieved through panning, a technique in which the target substrate is immobilized on a surface and protein-displaying molecules are applied and washed with increasing stringency. While effective, panning is susceptible to biases introduced by variable cellular display and translation. This limitation can be partially mitigated by single-cell screening techniques, such as flow cytometry, in which the binding affinity can be normalized by the amount of surface-bound protein[19, 20]. The successes in directed evolution of protein binders are numerous and impressive. In favorable cases, protein sequences have been selected that bind their target molecule with femtomolar dissociation constants[21, 22]. However, directed evolution of binding is most often successful if a basal affinity is already established,

and is therefore not ideally suited for creation of de novo binders[23]. Additionally, in most cases, it is difficult or impossible to dictate the binding orientation and binding location on the target molecule, which is a critical attribute for many functions, such as steric agonism and antagonism.

The second common use case for directed evolution is the optimization or modification of catalytic activity. The ability to generate novel and improved enzymes has implications in the generation of many high-value molecules and holds great promise for the production of biofuels, materials, and pharmaceuticals[24]. The most common goal in the evolution of protein catalysts is the improvement of enzymatic activity in a specified environment, such as at elevated temperature or in the presence of non-organic solvents. However, studies have demonstrated that other catalytic goals, such as modified substrate specificity and even de novo function, can be achieved[25, 26].

Unlike protein-binding, evolution for catalytic function is often complicated by the lack of a general and straightforward screening method. Frequently, the development of a selection scheme requires a clever and system-dependent solution such as the use of substrate analogs with fluorescent leaving groups[27], the creation of a covalent modification[28], or the coupling of enzyme function to viability[29, 30]. Evolved binding to transition-state analogs is an attractively general screening strategy, but studies have shown that increased affinity does not translate to increased catalytic activity and transition state analogs are not available for all substrates[31]. Additionally, even in favorable cases experimentally evolved enzymes fail to approach the astounding rate enhancements of 10^{19} achieved by some naturally occurring enzymes[32, 33]. Most successful catalysts have been evolved from existing enzyme backbones, and the mediocre rate enhancements achieved for novel reactions may indicate limits in adapting existing protein backbones for novel reactions. One potential solution to this problem is usage of highly diverse pools of novel backbones, such as those generated by the methods described in Chapter 2.

The protein engineering applications of directed evolution extend beyond the evolution of binding and catalysis, with results encompassing the creation of novel fluorescent proteins, the increasing of protein thermostability, and the generation of novel and well-folded protein

structures[34, 35, 36, 37]. These efforts, though fewer in comparison, demonstrate that directed evolution, through ingenuity in library generation and screening strategies, can be applied to many useful and difficult protein engineering tasks.

1.2.2 Computational Protein Design

Computational protein design, in the context of this dissertation, entails the usage of computational models of protein structure. Computational design protocols typically employ a search algorithm, which samples the sequence and conformational space available to the polypeptide chain, a score function, which evaluates the predicted energy of a given conformation. The challenges of computational protein design are two-fold: the conformational space available to a polypeptide chain is exceedingly large, and accurate evaluation of protein energetics is difficult to calculate efficiently. The issue of large conformational space was famously described in a paper by Cyrus Levinthal in 1969 in which he postulated that a conservative estimate for the number of conformations available to a 100 residue protein was 3^{300}[38]. This value will increase significantly if the amino-acid composition of the protein is not fixed, as in the case of protein design. Given the enormity of this value, it is unfeasible for computational design protocols to exhaustively sample the available conformational space. Additionally, the energy landscape of protein folding has many local minima and maxima, which hinder the ability of gradient based optimization methods to find global minimal. These local minima render continuous physics-based sampling strategies, such as molecular dynamics, inefficient for sampling of large-scale protein motions. A common strategy to address these difficulties is reducing the degrees of freedom available during a conformational search algorithm. The most prevalent method to accomplish this task is a technique called protein redesign, in which design simulations start from the experimentally solved structure of a protein sequence. In many cases, protein redesign is conducted with a completely fixed backbone conformation, limiting the search space tremendously. If backbone motion is allowed, it is common practice to fix the bond angles and lengths and sample backbone conformations only through modification of phi and psi internal coordinates. Even with these simplifications, conformational space remains very large and the level of sampling required for many protein engineering problems

necessitates the use of large computing clusters, which have only become available in recent years. The second challenge in computational protein design is the energetic evaluation of a given conformational state. These energetics, though understood in principle, are extremely difficult to compute efficiently and with high accuracy[39]. Quantum mechanical simulations, for example, are highly accurate, but the required computational expense limits these simulations to systems with a very small numbers of atoms[40]. On the opposite end of the spectrum, coarse-grained representations of atomic systems can be used to approximate the energetics of a protein structure[41]. Coarse models have the advantage of rapid calculation and smoother energy landscapes, but often fail to capture the level of intricacy needed to achieve most modern protein engineering goals. Currently, the predominant methods of evaluating the energetics of protein structure lie between these two extremes, with a mixture of physical force fields rooted in classical mechanics and empirical potentials derived from observation of known protein structures. These force fields allow energetic evaluations to be conducted on physiologically large protein structures and retain much of the accuracy granted by more precise equations. One such force field is implemented in the Rosetta molecular modeling suite, which was used for all of the computational modeling tasks contained in this text[42]. The Rosetta force-field is composed of approximately 15 score terms, which aim to capture the major determinants of protein energetics. Importantly, the Rosetta force-field is pairwise decomposable, meaning the total energy of a given conformational state can be represented as the sum of residue-pair energies. This simplification affords rapid re-evaluation of similar conformational states, which is a common occurrence in many sampling strategies. A simplified version of the current default Rosetta score function, is detailed in Equation 1.1.

$$E_{tot} = \sum \Theta_{vdw} + \Theta_{solv} + \Theta_{elec} + \Theta_{rot} + \Theta_{rama} \tag{1.1}$$

In this equation, the Θ_{vdw} term captures the van der Waals forces between atoms through a Lennard-Jones potential and Θ_{solv} is the Lazaridus-Karplus implicit solvation term. Θ_{elec} is a simplification of several hydrogen bonding terms and a coulombic electrostatic potential with

a distance-dependant dielectric. Θ_{rot} and Θ_{rama} encompass emperically derived potentials to capture sidechain rotameric preferences, backbone Φ/Ψ preference, and amino-acid identitites.

Development and improvement of force-fields for protein evaluation is a highly active area of research and many of the existing challenges, such as accurate hydrogen bond design, have experienced recent development[43, 44]. Continued efforts in force-field and score-function improvements are numerous and outside the scope of this dissertation.

1.2.3 Protein Redesign

Protein redesign is a common engineering strategy in which design begins from an experimental structure of an existing protein. The potential of an existing protein backbone to accommodate multiple sequences was eloquently demonstrated by Lim, *et al.* through the introduction of random mutations into the core of λ-repressor[45]. Following this publication, computational algorithms designed to place distinct amino-acid sidechain conformations, or rotamers, onto known protein backbones were used to redesign protein cores of many existing proteins[46, 47, 48, 49]. In one particularly demonstrative example, Dantas *et al.* applied these methods to redesign of nine globular proteins[50]. A serendipitous result from this study was that the thermostability of several redesigned proteins was increased relative to the native sequence. This result is particularly illustrative of the way in which protein redesign can be applied to difficult and valuable protein engineering tasks. In a related study, Murphy *et al.* incorporated local backbone flexibility in the redesign of a 4-helix bundle protein. Strikingly, despite a backbone RMSD of only 1.7A from the starting crystal structure, the redesigned protein was stabilized by more than 12 kcal/mol. This result was particularly impressive due to the fact that of the 38 residues designed, all were mutated from their wild-type identity, demonstrating both the ability of computational redesign methods to identify unique packing solutions and the ability of similar protein structures to accommodate different sequences. Redesign methods have also been applied to the task of engineering protein conformational switches, as in the case of a designed sequence that could adopt either a coiled-coil or zinc-finger like fold dependent on the pH and presence of transition metals[51]. Conformational switching is a critical component of many biological functions, such as viral infection and transcriptional

regulation, and therefore represents an important engineering accomplishment[52, 53]. The design of protein interfaces represent another important application of protein redesign. Unlike protein cores, which are composed predominantly of hydrophobic amino acids, protein interfaces contain many polar and charged amino acids that serve the dual purpose of promoting solubility, and generating affinity and specificity through inter-chain hydrogen bonds. Initial applications of protein redesign for the engineering of protein interfaces were aimed at the re-engineering of existing protein-protein interfaces. In these studies, homo-oligomeric proteins were redesigned to change oligomeric state, a monomeric protein was designed into a domain swapped dimer, and specificity inducing interactions were incorporated to generate orthogonal interface pairs[54, 55, 56, 57, 58, 59]. These successes supported the use of protein redesign towards the engineering of novel, or de novo protein interfaces. De novo interface design represents a significantly more difficult challenge for the computational design of interfaces, due to the fact that one must determine both a rigid body orientation and an appropriate sequence for the binding pair. For this reason, the de novo design of protein interfaces has remained a difficult and largely unsolved problem. A comprehensive review of protein interface design from 2013 reported that only 5 designs out of a total 158 tested demonstrated strong binding in a verifiably correct manner[60]. Importantly, this statistic represents an optimistic view of the field, given that unsuccessful design projects are rarely published. In a few cases of de novo interface design, binders have been generated that bind the target protein very weakly, or bind in a manner inconsistent with the design model, highlighting the challenge of generating both affinity and orientation specificity[61, 62]. In these cases predominantly hydrophobic residues were utilized in the design model, which can lead to broad specificity. However, successful redesigns tend to be more hydrophobic than most native interactions. This result is likely caused by both score-function inaccuracies and the difficulty of sampling complex, multi-residue polar networks between amino acid sidechains[60]. A successful strategy used to account for these interactions has been the use of properly defined or regular hydrogen bonding anchors to impart specificity, a topic I recently reviewed in detail[63]. In one case, Stranges *et al.* utilize the regular hydrogen bonding pattern of beta-strand pairing to design a highly accurate homodimeric protein (fig 1A)[64]. In another study, metal coordination was used in a similar

fashion to dictate the binding orientation of a designed homodimer (Figure 1.1B)[65]. In both of these studies, symmetry across the dimeric axis allows each amino acid mutation to impart twice the effect. This concept of utilizing symmetry in the design of protein-protein interfaces has recently been used in spectacular fashion towards the design of protein nanocages[66]. In the absence of symmetry, de novo interface design has been particularly challenging. In one case, a 16 residue extension was built onto an existing peptide interface (Figure 1.1C). In this study backbone flexibility was incorporated into the design simulation, allowing the design of helix packed tightly into a groove, a strategy commonly used in natural protein interfaces. The use of so-called hot spot residues to serve as a framework for selecting appropriate protein backbones for redesign was applied to the design of two de novo binders to influenza hemagglutinin. In this study, nearly 80 designs were tested, highlighting the difficulty in generating accurate design models. The usage of portions of natural interfaces as templates has been successful in the redesigning of natural interfaces, and work has been conducted to apply these methods to de novo interfaces[57, 67]. Relatedly, investigators have shown that the grafting of antigen epitopes from antibody interfaces allows for robust antibody re-elicitation, and holds significant promise for the design of vaccines against difficult targets[68, 69]. These studies have involved the grafting of protein segments into existing protein structures; an alternative approach is the custom design of de novo backbones to these accommodate interface motifs, a strategy explored in Chapter 3 of this document. These successes in computational interface engineering demonstrate significant potential. However, the affinities of most de novo interfaces are relatively weak and often necessitate the use of directed evolution for improvement to functionally relevant binding. This combination strategy for interface engineering leverages the benefits of both directed evolution and computational design and serves as inspiration for the methods described in later chapters.

In addition to the concrete engineering examples, protein redesign methods have been invaluable in the continued improvements to computational design methods. By limiting conformational sampling to the local space around an existing structure, rigorous sampling of sidechain and backbone conformations can be achieved. These studies, coupled with careful analysis of existing protein structures, have given rise to many advanced sampling strategies

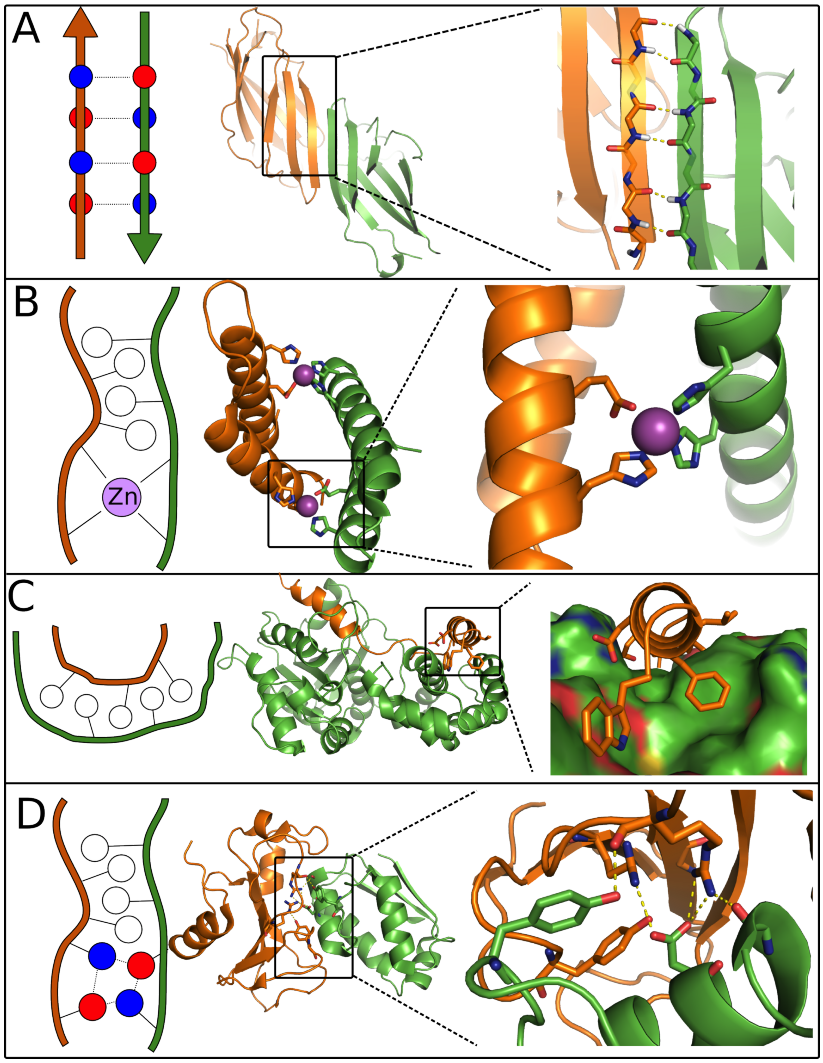

Figure 1.1: Anchor motifs found in natural and designed protein interfaces. Polar residues illustrated in red and blue, hydrophobic residues in white. (A) β-strand pairing across a designed protein interface (PDB code 3ZY7). (B) Designed metal-mediated protein interface (PDB code 3V1E). (C) Designed helix-in-groove interface (PDB code 2XNS). (D) Naturally occurring interface mediated by hydrogen bonds (PDB code 1BGS).

commonly used for protein engineering tasks, including loop design and prediction[70, 71, 72, 73] and local backbone sampling[74, 75, 76]. In sum, protein redesign is a well-studied protein engineering technique capable of addressing many engineering goals with atomic level precision.

1.2.4 De Novo Protein Design

De novo protein design is an alternative approach to computational protein engineering in which no starting experimental structure is used. The lack of a starting structure allows de novo designed proteins to incorporate structural features not found in naturally occurring structures. The ability to accurately engineer these novel structural details would expand the landscape of designable structures, which may be necessary to realize many complex protein functions. Unlike protein redesign where it is possible to hold the backbone degrees of freedom fixed, de novo design requires explicit backbone sampling. Protein backbones designed from scratch lose the assurance of designability afforded by existing protein structures and therefore serve as a rigorous test of protein design force-fields and sampling methods. Commonly, the task of de novo design is split into the generation of large numbers of novel backbones and the subsequent design of sidechains onto these backbones via the previously described redesign techniques. To address the task of backbone sampling, several methods have been developed.

For α-helical coiled-coil proteins, which form a regular and repetitive structure, Francis Crick famously derived a parameterization of the conformational space[77]. This parameterization was utilized in an early example of de novo design to generate backbone conformations for a variety of right-handed coiled-coil structures, many of which had not yet been observed in the set of known experimental structures[78]. More recently, similar methodologies were applied to the design of helix-bundle proteins that demonstrate incredible thermodynamic stability[79] and coiled-coils with water-soluble pores[80]. In all cases, design models matched experimental structures with atomic-level accuracy, demonstrating the precision afforded by parametric methods. Additionally, parametric methods benefit from the ability to easily produce large number of unique protein backbones through simple iteration of the available parameter space. To date, the applications of parametric design methods have been restricted to the engineering of proteins with repetitive sequences, which, similar to many of the interface design methods discussed above, benefit from the use of symmetry. In certain cases, such as the engineering of protein nanomaterials, symmetric structural repetition can be a powerful engineering advantage, allowing structures to be readily extended to arbitrary length. However, the regularity of parametrized backbones is also a disadvantage for certain engineering tasks, as it precludes the

incorporation of backbones outside of the parameter space, which is a a common occurrence in protein functional sites[81]

Another successful strategy for the de novo design of protein backbones is the use of fragment libraries derived from the protein data bank (PDB). Design simulations of this type typically starts from an elongated chain of set length. Backbone torsion angles from PDB fragments are applied to random windows of the protein backbone in order to fold the chain into a globular structure. Often, a coarse-grained representation of atomic structure is used during this process to smooth the energy landscape and focus sampling on the generation of gross structural features such as secondary structure and a hydrophobic core[82]. The application of fragment-based methods to de novo design gave rise to the first de novo protein not composed of symmetric elements[83]. The designed protein, called Top7, adopted a fold not yet observed in the PDB and matched the design model with atomic level accuracy. This promising initial result let to additional fragment-based design studies, which resulted in the accurate design of several canonical protein folds[84, 85]. The authors attribute the success of these studies to the adherence of several rules for designing ideal proteins, or proteins composed entirely of regular structural features commonly observed in the PDB. The proteins designed in this fashion are often highly thermostable, consistent with parametrically designed proteins, but similarly lack structural idiosyncrasies such as long loops, bulges, and cavities that are often needed for protein function[86, 79]. Existing methods for de novo design universally start with the definition of a target topology: In the case of parametric design, topologies are dictated by the parameterization space, whereas in fragment folding methods, restraints on secondary structure and residue-pair distances are applied to guide backbone sampling to a desired topology. This dependence on a target topology restricts the space of designed molecules to those explicitly defined by the designer and may hinder the ability to create the novel topologies required for desired function. However, these proof-of-concept successes in protein design provide valuable evidence that current force-fields used in computational design are able to accurately identify physically realistic and designable protein backbones.

1.3 Opportunities for Improved Engineering Methods

The protein engineering techniques outlined here have been successfully employed towards the design and optimization of numerous proteins. Nevertheless, each of these techniques is limited, and the ability to reliably engineer many desired protein properties and functions remains elusive. The focus of this dissertation is the development of novel computational tools that build on the successes of these existing strategies. Directed evolution allows diverse sampling for a given function but lacks the fine-grained structural control vital for many applications. Conversely, existing computational design allows atomic-level control but is limited to the adaptation of existing structures or the generation of a provided topology, which may or may not be able to accommodate the desired function. The goal of this work is to address the methodological gap between the function-first attributes of directed evolution and the structure-first attributes of computational design. The following chapters describe tools designed to bridge this gap through the incorporation of design principles from one method to the other. Chapter 2 describes a novel computational method for the de novo design of large numbers of highly diverse protein backbones without the use of a target fold. A fundamental tenant of this method is the computational adaptation of the evolutionary principles used to great effect in directed evolution techniques. Chapter 3 extends this method towards the design of protein-protein interfaces bolstered by naturally occurring interface templates. Finally, Chapter 4 focuses on the efficient incorporation of computational predictions into directed evolution techniques.

REFERENCES

1. Gutte, B. (1975) A synthetic 70-amino acid residue analog of ribonuclease s-protein with enzymic activity. J. Biol. Chem. **250**, 889–904

2. Bärwald, K. R., Reid, R. E., and Gutte, B. (1975) Formation and enzymic properties of dimeric RNase P. FEBS Lett. **60**, 423–426

3. Gutte, B., Däumigen, M., and Wittschieber, E. (1979) Design, synthesis and characterisation of a 34-residue polypeptide that interacts with nucleic acids. Nature **281**, 650–655

4. Napper, A. D., Benkovic, S. J., Tramontano, A., and Lerner, R. A. (1987) A stereospecific cyclization catalyzed by an antibody. Science **237**, 1041–1043

5. Clarke, A. R., Atkinson, T., and John Holbrook, J. (1990) FROM ANALYSIS TO SYNTHESIS: NEW LIGAND BINDING SITES ON THE LACTATE DEHYDROGENASE FRAMEWORK. PART II. In Proteins: Form and Function, 39–44. Elsevier

6. Hecht, M. H., Richardson, J. S., Richardson, D. C., and Ogden, R. C. (1990) De novo design, expression, and characterization of felix: a four-helix bundle protein of native-like sequence. Science **249**, 884–891

7. Pandyarajan, V. and Weiss, M. A. (2012) Design of non-standard insulin analogs for the treatment of diabetes mellitus. Curr. Diab. Rep. **12**, 697–704

8. Riechmann, L., Clark, M., Waldmann, H., and Winter, G. (1988) Reshaping human antibodies for therapy. Nature **332**, 323–327

9. Wu, Y. I., Frey, D., Lungu, O. I., Jaehrig, A., Schlichting, I., Kuhlman, B., and Hahn, K. M. (2009) A genetically encoded photoactivatable rac controls the motility of living cells. Nature **461**, 104–108

10. Ai, H.-W., Baird, M. A., Shen, Y., Davidson, M. W., and Campbell, R. E. (2014) Engineering and characterizing monomeric fluorescent proteins for live-cell imaging applications. Nat. Protoc. **9**, 910–928

11. Gautier, A., Juillerat, A., Heinis, C., Corrêa, I. R., Jr, Kindermann, M., Beaufils, F., and Johnsson, K. (2008) An engineered protein tag for multiprotein labeling in living cells. Chem. Biol. **15**, 128–136

12. Möglich, A. and Moffat, K. (2010) Engineered photoreceptors as novel optogenetic tools. Photochem. Photobiol. Sci. **9**, 1286–1300

13. Binz, H. K. and Plückthun, A. (2005) Engineered proteins as specific binding reagents. Curr. Opin. Biotechnol. **16**, 459–469

14. Crameri, A., Raillard, S. A., Bermudez, E., and Stemmer, W. P. (1998) DNA shuffling of a family of genes from diverse species accelerates directed evolution. Nature **391**, 288–291

15. Stemmer, W. P. (1994) Rapid evolution of a protein in vitro by DNA shuffling. Nature **370**, 389–391

16. Zhou, Y. H., Zhang, X. P., and Ebright, R. H. (1991) Random mutagenesis of gene-sized DNA molecules by use of PCR with taq DNA polymerase. Nucleic Acids Res. **19**, 6052

17. Virnekäs, B., Ge, L., Plückthun, A., Schneider, K. C., Wellnhofer, G., and Moroney, S. E. (1994) Trinucleotide phosphoramidites: ideal reagents for the synthesis of mixed oligonucleotides for random mutagenesis. Nucleic Acids Res. **22**, 5600–5607

18. Leemhuis, H., Kelly, R. M., and Dijkhuizen, L. (2009) Directed evolution of enzymes: Library screening strategies. IUBMB Life **61**, 222–228

19. Boder, E. T. and Wittrup, K. D. (1997) Yeast surface display for screening combinatorial polypeptide libraries. Nat. Biotechnol. **15**, 553–557

20. Boder, E. T. and Wittrup, K. D. (2000) Yeast surface display for directed evolution of protein expression, affinity, and stability. Methods Enzymol. **328**, 430–444

21. Boder, E. T., Midelfort, K. S., and Wittrup, K. D. (2000) Directed evolution of antibody fragments with monovalent femtomolar antigen-binding affinity. Proc. Natl. Acad. Sci. U. S. A. **97**, 10701–10705

22. Vasserot, A. P., Dickinson, C. D., Tang, Y., Huse, W. D., Manchester, K. S., and Watkins, J. D. (2003) Optimization of protein therapeutics by directed evolution. Drug Discov. Today **8**, 118–126

23. Bolon, D. N., Voigt, C. A., and Mayo, S. L. (2002) De novo design of biocatalysts. Curr. Opin. Chem. Biol. **6**, 125–129

24. Turner, N. J. (2009) Directed evolution drives the next generation of biocatalysts. Nat. Chem. Biol. **5**, 567–573

25. Yano, T., Oue, S., and Kagamiyama, H. (1998) Directed evolution of an aspartate aminotransferase with new substrate specificities. Proc. Natl. Acad. Sci. U. S. A. **95**, 5511–5515

26. Buchholz, F. and Stewart, A. F. (2001) Alteration of cre recombinase site specificity by substrate-linked protein evolution. Nat. Biotechnol. **19**, 1047–1052

27. Luis Briseño Roa, ., Jim Hill, ., Stuart Notman, ., David Sellers, ., Andy P. Smith, ., Christopher M. Timperley, ., *, Janet Wetherell, ., Nichola H. Williams, ., Gareth R. Williams, ., Alan R. Fersht, ., and Griffiths, A. D. (2006) Analogues with fluorescent leaving groups for screening and selection of enzymes that efficiently hydrolyze organophosphorus nerve agents. J. Med. Chem. **49**, 246–255

28. Cesaro-Tadic, S., Lagos, D., Honegger, A., Rickard, J. H., Partridge, L. J., Blackburn, G. M., and Plückthun, A. (2003) Turnover-based in vitro selection and evolution of biocatalysts from a fully synthetic antibody library. Nat. Biotechnol. **21**, 679–685

29. Chin, J. W., Martin, A. B., King, D. S., Wang, L., and Schultz, P. G. (2002) Addition of a photocrosslinking amino acid to the genetic code of escherichia coli. Proceedings of the National Academy of Sciences **99**, 11020–11024

30. Tao, H. and Cornish, V. W. (2002) Milestones in directed enzyme evolution. Curr. Opin. Chem. Biol. **6**, 858–864

31. Baca, M., Scanlan, T. S., Stephenson, R. C., and Wells, J. A. (1997) Phage display of a catalytic antibody to optimize affinity for transition-state analog binding. Proc. Natl. Acad. Sci. U. S. A. **94**, 10063–10068

32. Wolfenden, R. and Snider, M. J. (2001) The depth of chemical time and the power of enzymes as catalysts. Acc. Chem. Res. **34**, 938–945

33. Yuan, L., Kurek, I., English, J., and Keenan, R. (2005) Laboratory-directed protein evolution. Microbiol. Mol. Biol. Rev. **69**, 373–392

34. Song, J. K. and Rhee, J. S. (2001) Enhancement of stability and activity of phospholipase A1 in organic solvents by directed evolution. Biochimica et Biophysica Acta (BBA) - Protein Structure and Molecular Enzymology **1547**, 370–378

35. Eijsink, V. G. H., Gåseidnes, S., Borchert, T. V., and van den Burg, B. (2005) Directed evolution of enzyme stability. Biomol. Eng. **22**, 21–30

36. Sawano, A. and Miyawaki, A. (2000) Directed evolution of green fluorescent protein by a new versatile PCR strategy for site-directed and semi-random mutagenesis. Nucleic Acids Res. **28**, E78

37. Wei, Y., Kim, S., Fela, D., Baum, J., and Hecht, M. H. (2003) Solution structure of a de novo protein from a designed combinatorial library. Proc. Natl. Acad. Sci. U. S. A. **100**, 13270–13273

38. Levinthal, C. (1969) How to fold graciously. Mossbauer spectroscopy in biological systems 22–24

39. Dill, K. A. (1990) Dominant forces in protein folding. Biochemistry **29**, 7133–7155

40. Hatfield, M. P. D. and Lovas, S. (2014) Conformational sampling techniques. Curr. Pharm. Des. **20**, 3303–3313

41. Tozzini, V. (2005) Coarse-grained models for proteins. Curr. Opin. Struct. Biol. **15**, 144–150

42. Leaver-Fay, A., Tyka, M., Lewis, S. M. S. M., Lange, O. F., Thompson, J., Jacak, R., Kaufman, K. W., Renfrew, P. D., Smith, C. A., Sheffler, W., Davis, I. W., Cooper, S., Treuille, A., Mandell, D. J., Richter, F., Ban, Y.-E. A., Fleishman, S. J., Corn, J. E., Kim, D. E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J. J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J. J., Kuhlman, B., Baker, D., and Bradley, P. (2011) Rosetta3: An Object-Oriented software suite for the simulation and design of macromolecules. Methods Enzymol. **Volume 487**, 545–574

43. O'Meara, M. J., Leaver-Fay, A., Tyka, M., Stein, A., Houlihan, K., DiMaio, F., Bradley, P., Kortemme, T., Baker, D., Snoeyink, J., and Kuhlman, B. (2015) A combined Covalent-Electrostatic model of hydrogen bonding improves structure prediction with rosetta. J. Chem. Theory Comput. **11**, 609–622

44. Leaver-Fay, A., O'Meara, M. J., Tyka, M., Jacak, R., Song, Y., Kellogg, E. H., Thompson, J., Davis, I. W., Pache, R. A., Lyskov, S., Gray, J. J., Kortemme, T., Richardson, J. S., Havranek, J. J., Snoeyink, J., Baker, D., and Kuhlman, B. (2013) Scientific benchmarks for guiding macromolecular energy function improvement. Methods Enzymol. **523**, 109–143

45. Lim, W. A. and Sauer, R. T. (1989) Alternative packing arrangements in the hydrophobic core of lambda repressor. Nature **339**, 31–36

46. Dahiyat, B. I. (1997) De novo protein design: Fully automated sequence selection. Science **278**, 82–87

47. Desjarlais, J. R. and Handel, T. M. (1995) De novo design of the hydrophobic cores of proteins. Protein Sci. **4**, 2006–2018

48. Malakauskas, S. M. and Mayo, S. L. (1998) Design, structure and stability of a hyperthermophilic protein variant. Nat. Struct. Biol. **5**, 470–475

49. Nauli, S., Kuhlman, B., and Baker, D. (2001) Computer-based redesign of a protein folding pathway. Nat. Struct. Biol. **8**, 602–605

50. Dantas, G., Kuhlman, B., Callender, D., Wong, M., and 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

51. Ambroggio, X. I. and Kuhlman, B. (2006) Computational design of a single amino acid sequence that can switch between two distinct protein folds. J. Am. Chem. Soc. **128**, 1154–1161

52. Beckett, D. (2004) Functional switches in transcription regulation; molecular mimicry and plasticity in protein-protein interactions. Biochemistry **43**, 7983–7991

53. Zheng, A., Yuan, F., Kleinfelter, L. M., and Kielian, M. (2014) A toggle switch controls the low ph-triggered rearrangement and maturation of the dengue virus envelope proteins. Nat. Commun. **5**, 3877

54. Grueninger, D., Treiber, N., Ziegler, M. O. P., Koetter, J. W. A., Schulze, M.-S., and Schulz, G. E. (2008) Designed protein-protein association. Science **319**, 206–209

55. Kuhlman, B., O'Neill, J. W., Kim, D. E., Zhang, K. Y., and Baker, D. (2001) Conversion of monomeric protein L to an obligate dimer by computational protein design. Proc. Natl. Acad. Sci. U. S. A. **98**, 10687–10691

56. Kortemme, T., Joachimiak, L. a., Bullock, A. N., Schuler, A. D., Stoddard, B. L., and Baker, D. (2004) Computational redesign of protein-protein interaction specificity. Nat. Struct. Mol. Biol. **11**, 371–379

57. Potapov, V., Reichmann, D., Abramovich, R., Filchtinski, D., Zohar, N., Ben Halevy, D., Edelman, M., Sobolev, V., and Schreiber, G. (2008) Computational redesign of a Protein–Protein interface for high affinity and binding specificity using modular architecture and naturally occurring template fragments. J. Mol. Biol. **384**, 109–119

58. Lewis, S. M., Wu, X., Pustilnik, A., Sereno, A., Huang, F., Rick, H. L., Guntas, G., Leaver-Fay, A., Smith, E. M., Ho, C., Hansen-Estruch, C., Chamberlain, A. K., Truhlar, S. M., Conner, E. M., Atwell, S., Kuhlman, B., and Demarest, S. J. (2014) Generation of bispecific IgG antibodies by structure-based design of an orthogonal fab interface. Nat. Biotechnol. **32**, 191–198

59. Kapp, G. T., Liu, S., Stein, A., Wong, D. T., Reményi, A., Yeh, B. J., Fraser, J. S., Taunton, J., Lim, W. a., and Kortemme, T. (2012) Control of protein signaling using a computationally designed GTPase/GEF orthogonal pair. Proc. Natl. Acad. Sci. U. S. A.

60. Stranges, P. B. and Kuhlman, B. (2013) A comparison of successful and failed protein interface designs highlights the challenges of designing buried hydrogen bonds. Protein Sci. **22**, 74–82

61. Karanicolas, J., Corn, J. E., Chen, I., Joachimiak, L. a., Dym, O., Peck, S. H., Albeck, S., Unger, T., Hu, W., Liu, G., Delbecq, S., Montelione, G. T., Spiegel, C. P., Liu, D. R., and Baker, D. (2011) A de novo protein binding pair by computational design and directed evolution. Mol. Cell **42**, 250–260

62. Jha, R. K., Leaver-Fay, A., Yin, S., Wu, Y., Butterfoss, G. L., Szyperski, T., Dokholyan, N. V., and Kuhlman, B. (2010) Computational design of a PAK1 binding protein. J. Mol. Biol. **400**, 257–270

63. Jacobs, T. M. and Kuhlman, B. (2013) Using anchoring motifs for the computational design of protein-protein interactions. Biochem. Soc. Trans. **41**, 1141–1145

64. Stranges, P. B., Machius, M., Miley, M. J., Tripathy, A., and Kuhlman, B. (2011) Computational design of a symmetric homodimer using β-strand assembly. Proc. Natl. Acad. Sci. U. S. A. **108**, 1–6

65. Der, B. S., Machius, M., Miley, M. J., Mills, J. L., Szyperski, T., and Kuhlman, B. (2012) Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer. J. Am. Chem. Soc. **134**, 375–385

66. King, N. P., Sheffler, W., Sawaya, M. R., Vollmar, B. S., Sumida, J. P., André, I., Gonen, T., Yeates, T. O., and Baker, D. (2012) Computational design of self-assembling protein nanomaterials with atomic level accuracy. Science **336**, 1171–1174

67. Lewis, S. M. and Kuhlman, B. A. (2011) Anchored design of protein-protein interfaces. PLoS One **6**, e20872

68. Azoitei, M. L., Correia, B. E., Ban, Y.-E. A., Carrico, C., Kalyuzhniy, O., Chen, L., Schroeter, A., Huang, P.-S., McLellan, J. S., Kwong, P. D., Baker, D., Strong, R. K., and Schief, W. R. (2011) Computation-guided backbone grafting of a discontinuous motif onto a protein scaffold. Science **334**, 373–376

69. Azoitei, M. L., Ban, Y.-E. A., Julien, J.-P., Bryson, S., Schroeter, A., Kalyuzhniy, O., Porter, J. R., Adachi, Y., Baker, D., Pai, E. F., and Schief, W. R. (2012) Computational design of high-affinity epitope scaffolds by backbone grafting of a linear epitope. J. Mol. Biol. **415**, 175–192

70. Das, R. (2013) Atomic-Accuracy prediction of protein loop structures through an RNA-Inspired ansatz. PLoS One **8**, e74830

71. Canutescu, A. A. and Dunbrack, R. L. (2003) Cyclic coordinate descent: A robotics algorithm for protein loop closure. Protein Sci. **12**, 963–972

72. Mandell, D. J., Coutsias, E. a., and Kortemme, T. (2009) Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling. Nat. Methods **6**, 551–552

73. Stein, A. and Kortemme, T. (2013) Improvements to robotics-inspired conformational sampling in rosetta. PLoS One **8**, e63090

74. Davis, I. W., Arendall, W. B., Richardson, D. C., and Richardson, J. S. (2006) The backrub motion: how protein backbone shrugs when a sidechain dances. Structure **14**,

265–274

75. Tyka, M. D., Jung, K., and Baker, D. (2012) Efficient sampling of protein conformational space using fast loop building and batch minimization on highly parallel computers. J. Comput. Chem. **33**, 2483–2491

76. Lauck, F., Smith, C. A., Friedland, G. F., Humphris, E. L., and Kortemme, T. (2010) RosettaBackrub—a web server for flexible backbone protein structure modeling and design. Nucleic Acids Res. **38**, W569–W575

77. Crick, F. H. C. (1953) The packing of α-helices: simple coiled-coils. Acta Crystallogr. **6**, 689–697

78. Harbury, P. B. (1998) High-Resolution protein design with backbone freedom. Science **282**, 1462–1467

79. Huang, P.-S., Oberdorfer, G., Xu, C., Pei, X. Y., Nannenga, B. L., Rogers, J. M., DiMaio, F., Gonen, T., Luisi, B., and Baker, D. (2014) High thermodynamic stability of parametrically designed helical bundles. Science **346**, 481–485

80. Thomson, a. R., Wood, C. W., Burton, a. J., Bartlett, G. J., Sessions, R. B., Brady, R. L., and Woolfson, D. N. (2014) Computational design of water-soluble -helical barrels. Science **346**, 485–488

81. Brown, J. H., Cohen, C., and Parry, D. A. (1996) Heptad breaks in alpha-helical coiled coils: stutters and stammers. Proteins **26**, 134–145

82. Rohl, C. A., Strauss, C. E., Misura, K. M., and Baker, D. (2004) Protein structure prediction using rosetta. Methods Enzymol. **383**, 66–93

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

84. Koga, N., Tatsumi-Koga, R., Liu, G., Xiao, R., Acton, T. B., Montelione, G. T., and Baker, D. (2012) Principles for designing ideal protein structures. Nature **491**, 222–227

85. Murphy, G. S., Sathyamoorthy, B., Der, B. S., Machius, M. C., Pulavarti, S. V., Szyperski, T., and Kuhlman, B. (2014) Computational de novo design of a four-helix bundle protein - DND_4HB. Protein Sci. **00**, 1–12

86. Murphy, G. S., Mills, J. L., Miley, M. J., Machius, M., Szyperski, T., and Kuhlman, B. (2012) Increasing sequence diversity with flexible backbone protein design: the complete redesign of a protein hydrophobic core. Structure **20**, 1086–1096

Chapter 2

DESIGN OF STRUCTURALLY UNIQUE PROTEINS USING STRATEGIES INSPIRED BY EVOLUTION

2.1 Introduction

Most efforts in de novo protein design have been focused on creating idealized proteins composed of canonical structural elements. Examples include the design of coiled-coils, repeat proteins, TIM barrels and Rossman folds[1, 2, 3, 4]. These studies are excellent for studying the minimal determinants of protein structure, but they do not aggressively explore new regions of structure space. Additionally, idealized structures may not always be the most effective starting points for engineering novel protein functions. Functional sites in proteins are often created from structural idiosyncrasies, such as kinks, pockets and bulges. The lack of these structural elements from de novo designed proteins highlights a key difference between natural protein evolution and current design methods. Specifically, that protein design methods universally begin with a target structure in mind. Therefore, the space of designable structures that can accommodate these non-ideal protein elements is limited by the imagination of the designer. In contrast, natural evolution is uncaring of the structure being evolved so long as the result accomplishes the necessary function. This lack of a predetermined target fold is a powerful feature of protein evolution that holds significant potential for the design of novel structures and functions. In an effort to tap this potential we sought to develop a method of computational protein design inspired by mechanisms found in natural protein evolution. Nature uses gene duplication and homologous recombination to mix and match elements of protein structure to give rise to novel structures and functions[5, 6, 7]. This phenomenon is most evident at the level of independently folding protein domains [8, 9, 10], but recent studies have shown that these same principles function at a smaller scale during the evolution of distinct, globular protein

folds[11]. Insertions, deletions and replacement of secondary and supersecondary structural elements are used by nature to sample alternative tertiary structures[12]. Our design strategy, called SEWING (Structure Extension With Native-substructure Graphs), computationally mimics this process by building new protein structures from pieces of naturally occurring protein domains. The process is not dictated by the need to adopt a specific target fold, but rather aimed at creating large sets of alternative structures that satisfy predefined design requirements. One of the strengths of this approach is that it ensures that all of the structural elements of the protein are inherently designable, while at the same time allowing for the incorporation of structural oddities unlikely to be found in idealized proteins. Here, we apply SEWING to the design of helical proteins. We show that designed structures are highly diverse, and contain structural features absent from alternative design strategies.

2.2 Results

2.2.1 Overview of the SEWING Method

SEWING begins with the extraction of small structural motifs, or substructures, from existing protein structures. These serve as the basic building block for all generated models. We aimed to identify substructures that were both large enough to carry information regarding structural preference, yet small enough to allow combinations that can generate novel globular structures. Ultimately, we chose to extract two distinct types of substructures. The first type of substructure is composed of contiguous stretches of protein structure that encompass two secondary structural elements separated by a loop (Figure 2.1). These substructures capture the relative orientation between adjacent secondary structure elements and maintain local packing interactions. Additionally, there is evidence that substructures of this size adopt a relatively limited number of conformations that have been sampled exhaustively in known protein structures [12]. The second type of substructure is composed of groups of 3-5 secondary structural elements, where each element makes make van der Waals contacts with every other but are not necessarily contiguous in primary sequence (Figure 2.1, Supplementary Methods).

Non-adjacent, or discontiguous substructures maintain longer-range tertiary interactions that provide valuable stability, and are often conserved during protein evolution [13].

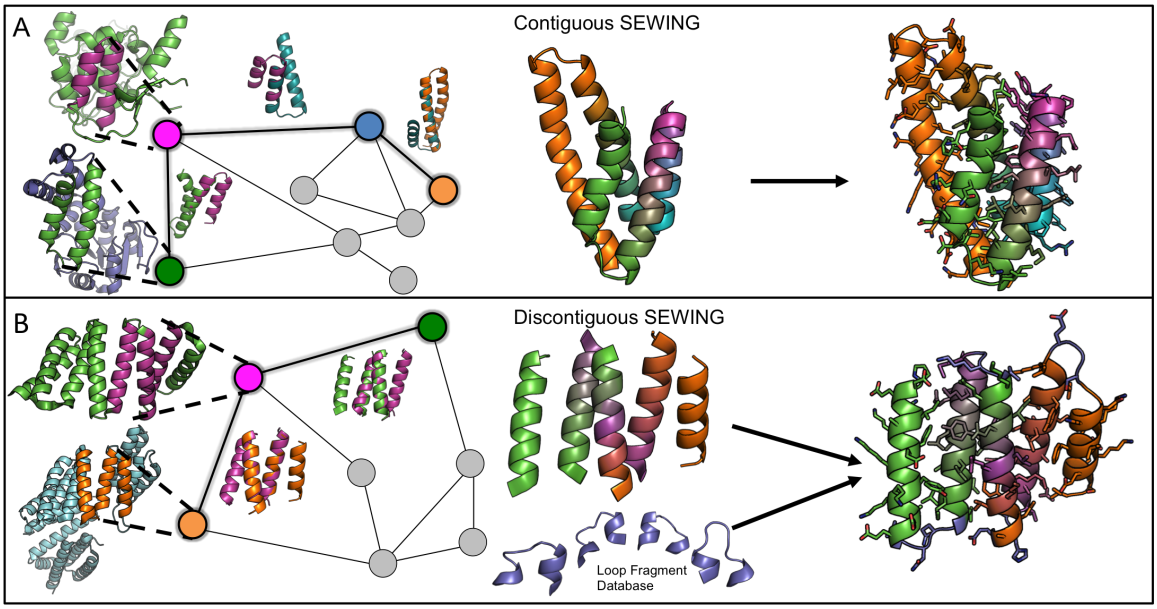

Figure 2.1: Overview of the SEWING method. (A) Contiguous SEWING workflow for CA01. (B) Discontiguous SEWING workflow for DA05. Each panel, from left to right: parental PDBs with extracted substructures; Graph schematic colored nodes indicate substructures contained in final design model, superimposed structures show structural similarity indicated by adjacent edges; Design model before sequence optimization and loop design; Final design models.

The goal of SEWING is to combine and modify these extracted components in order to develop new tertiary structures. In nature, homologous recombination guides the formation of new protein chimeras, in which sequence similarity between DNA strands leads to combination of the genetic material. This process enriches for proteins that are more likely to be well-folded and functional, as sequence similarity filters for segments that are structurally compatible. In the case of SEWING, we know the three dimensional structures of the building blocks, and therefore, we can directly use structural information to probe which substructures are well suited for combination. During SEWING, contiguous substructures are eligible for combination if the c-terminal region of one substructure shares high structural similarity with the n-terminal region of another substructure, and superposition of the two regions does not create any steric clashes between other regions in the two substructures. This type of superposition ensures that the three-dimensional spacing between all pairs of secondary structural elements

adjacent in primary sequence is similar to that observed in the PDB. During discontiguous SEWING, combinations are created by superimposing two elements (helices in this study) from one substructure with two elements from another substructure. For both contiguous and discontiguous SEWING, structure similarity is identified using a fast geometric hashing approach that ensures that the regions of interest can align with low-RMSD[14]. Once pairwise structural similarity is calculated between all substructures, these data are used to generate two large graphs: a contiguous graph and a discontiguous graph (Figure 2.1). The nodes in each graph represent the substructures, and the edges indicate a level of structural similarity that allows recombination. Novel structures are generated from each graph by traversing a path, where each followed edge adds new structural elements to the design model. In this way, many unique structures can be efficiently generated by randomly sampling paths. The number of edges included in the sampled paths can control the approximate size of the generated structures. Importantly, unlike previously described methods of de novo backbone generation, no target structure is required, and output structures span a diverse set of globular folds. Similar to natural evolution, the design models created by recombination alone are suboptimal, and require additional refinement through mutagenesis. This optimization step was achieved using iterative steps of sidechain packing and backbone minimization available in the Rosetta molecular modeling suite [15]. Preference for the amino acid sequence present in the parental substructure was used to better preserve the structural interactions inherent to the parent substructures. To test SEWING, we designed a diverse set of helical proteins using graphs composed of contiguous substructures or discontiguous substructures. Contiguous substructures were extracted from a non-redundant dataset of approximately 8000 high-quality crystal structures [16]. Discontiguous substructures were extracted from a non-redundant subset of structures from the protein data bank (PDB), generated using the Pisces server [17]. In total, 33928 contiguous substructures, and 4584 discontiguous substructures were extracted.

2.2.2 Contiguous SEWING Design and Characterization

Design models from the contiguous graph were generated using 3-edge paths, and were therefore composed of substructures extracted from 4 existing structures from the PDB (Fig-

ure 2.1). Initially, 10,000 alternative tertiary structures were created and used as templates for rotamer-based sequence optimization and energy minimization. These models were filtered and sorted using metrics that evaluate predicted energy (normalized by sequence length), sidechain packing, buried polar groups, and sequence/structure agreement[18]. The top 1,000 of design models were then further refined with 3 rounds of the same sequence optimization protocol. In total, 11 designs based on contiguous SEWING were selected for experimental characterization. 8 of these designs expressed well in E. coli and were readily purified from the soluble fraction of 1L cultures. 4 of the 8 proteins were monomeric via Size Exclusion Chromotagraphy/Multi-angle light scattering, had a circular dichroism (CD) spectra characteristic of a helical protein and unfolded cooperatively upon thermal denaturation (Figure 2.2, S2.6). Two of the designed proteins are hyperthermophiles and require high concentrations of chemical denaturant in order to observe thermal unfolding (Figure 2.2B). For one design, CA01, several thermodynamic parameters were determined by fitting a modified Gibbs-Hemholtz equation to the thermal and chemical denaturation surface (Figure 2.3B, Supplemental Methods). The extrapolated melting temperature of 126 °C places it among the top .01% of values in the ProTherm database of protein stabilities [19]. The crystal structure of CA01 was solved to 2.2Åand shows excellent agreement with the design model, with a Cα RMSD of 0.7$\mathring{A}$. Similarly, the side-chain packing of the protein core is nearly identical between the design model and experimental structure (Figure 2.3).

The structural variety in the design models for the well-folded proteins is of particular note (Figure 2.2). The SEWING generated models often include non-ideal elements such as kinked or highly curved helices, long loops, and near perpendicular helix-crossing angles (Figure 2.2). The topologies of SEWING models are also quite variable, particularly when compared to previously designed alpha-helical proteins, which are restricted entirely to coiled-coils, repeat proteins and up-down four helix bundles (Figure 2.2C). To further examine the structural diversity of SEWING models, we searched for structurally similar domains using the DALI server [20]. Surprisingly, despite being derived from elements of existing structures, SEWING models adopt conformations that differ considerably from any known structure (Figure S2.5). This

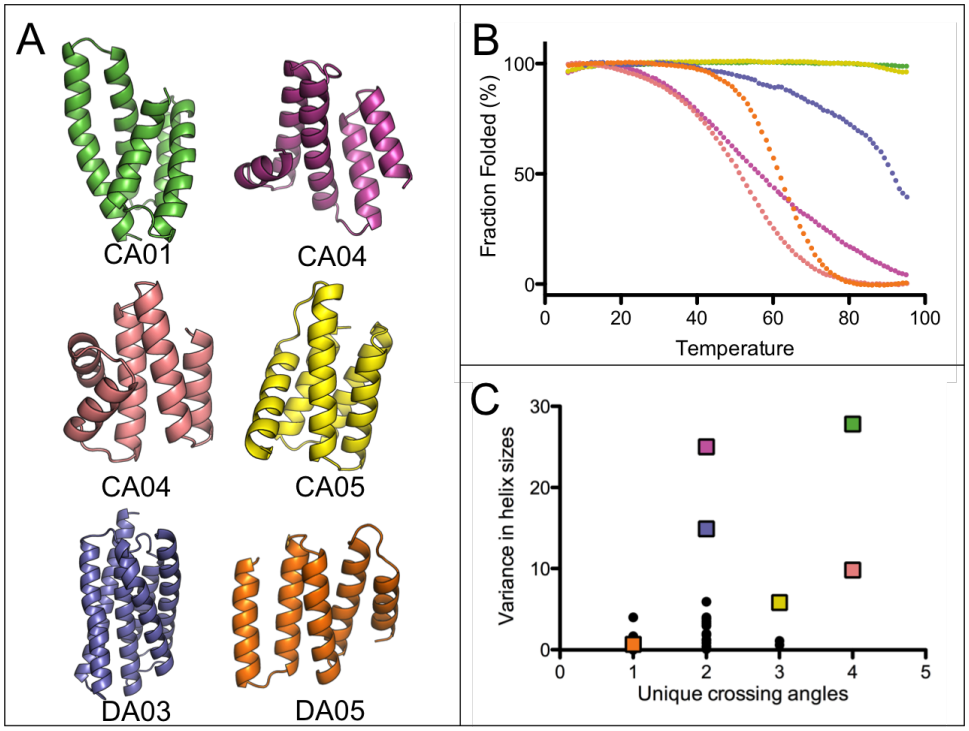

Figure 2.2: Well folded SEWING designs. (A) Final design models for contiguous and discontiguous SEWING. (B) Temperature denaturation curves for well-folded SEWING designs, colored to match design models. (C) A comparison of previously design helical structures (black dots), to SEWING models (colored squares) demonstrates the structural complexity of SEWING designs. Crossing angles were counted as unique if they differed by >20 degrees from each other.

serendipitous result further demonstrates the potential of recombination for the production of novel protein structures.

2.2.3 Discontiguous SEWING Characterization

To test discontiguous SEWING, models were generated from 2-edge paths, and thus were composed of structural elements from 3 parent structures. The variable number of helical elements in the discontiguous substructures therefore allowed design models to be composed of between 5 and 11 helices. Unlike models from the contiguous-substructure graph, discontiguous models require the addition of loops between consecutive helices. Loops were designed using a database of fixed-length fragments from the PDB, hashed on the geometric transform from the first to last residue of the fragment [21]. For each designed loop, fragments were selected

from the fragment-database based on the transform between the two disconnected helices. Each fragment was then superimposed onto the design structure and optimized using gradient-based minimization in Cartesian space. The best-scoring loop fragment was selected for use in the final design model. Any path that created structures for which no loop-fragment could be found was eliminated from the set of designs. Design models were filtered and optimized in the same way as models from the contiguous graph. In total, 10 were selected for experimental characterization.

Of these 10 designs, 2 expressed at levels sufficient for purification. Both purified proteins were helical and folded as evidenced by CD (Figure 2.2). Similar to the results from the contiguous designs, one discontiguous design demonstrated very high thermostability, requiring

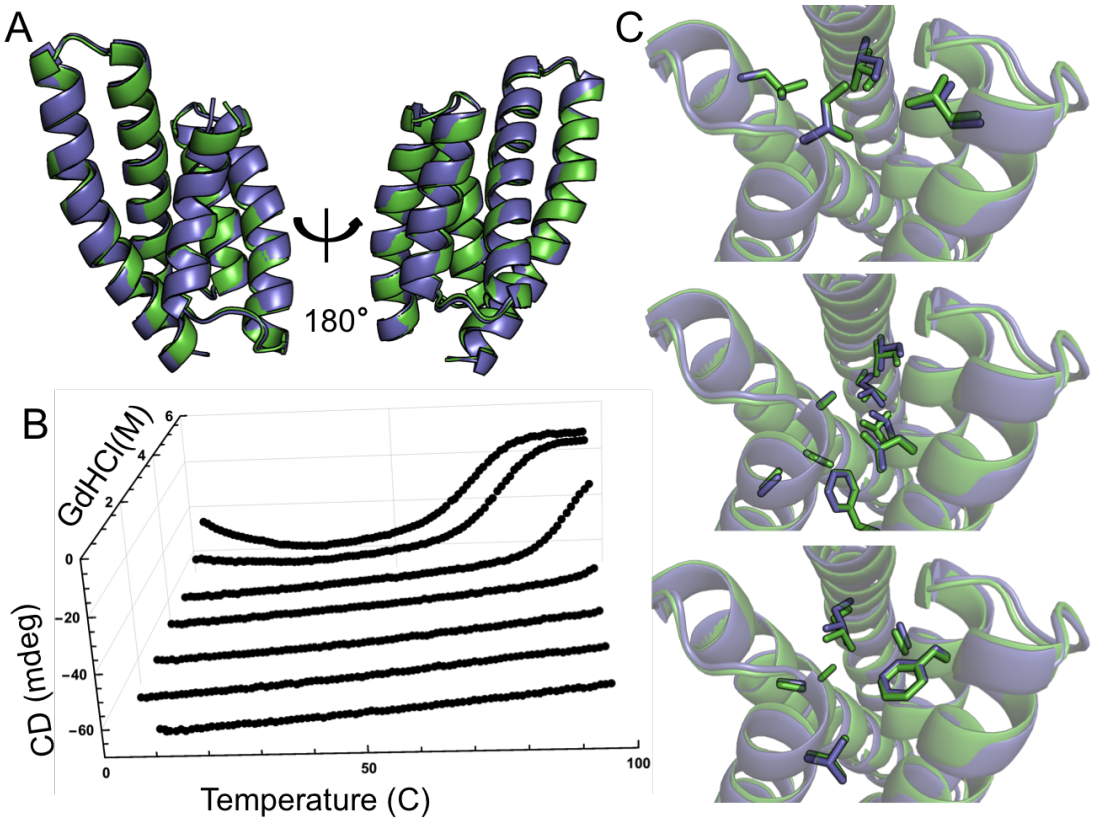

Figure 2.3: Structural and biophysical analysis of contiguous design CA01. (A) Backbone superimposition of CA01 design model (green) and crystal structure (blue). (B) Chemical and temperature denaturation of CA01. A sharp unfolding transition is observed prior at 5M GdmCl and 350 K (C) Comparison of sidechain packing between design model (green) and crystal structure (blue) at three different layers of the structure.

high levels of denaturant to completely unfold(Figure S2.6). The structure of DA05 was solved using nuclear magnetic resonance (NMR) spectroscopy. The first 4 helices of the experimental structure match the design model very closely, with a Cα RMSD of 0.9$\mathring{A}$ (Figure 2.4). However, the NMR data indicate the final helix of the protein is disordered in solution. In an effort to identify the errors in the design model that led to the unstructured region, structural preference for the designed sequence was evaluated with fragment analysis as described previously [2]. Interestingly, the fragments extracted for the unstructured region showed especially poor preference for the designed helical structure (Figure S2.3). In light of this result, fragment analysis was conducted on the remaining discontiguous design models. In nearly all cases, the sequences for the designed loop regions showed poor agreement with the designed structure. This discrepancy suggests that errors in the loop design protocol could account for the relatively lower success rate of the discontiguous designs.

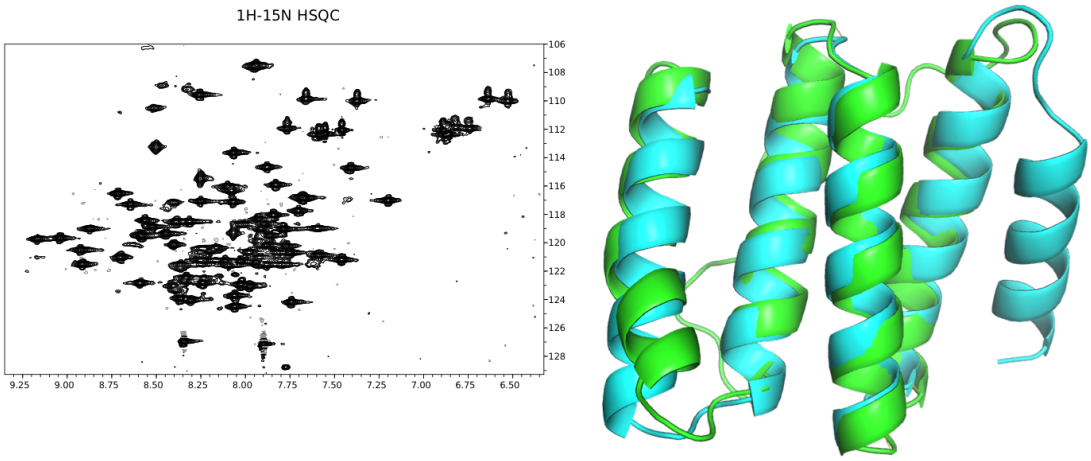

Figure 2.4: Structural analysis of discontiguous design DA05 via NMR reveals that the first four helices of the structure match well with the design model. However, the 5th helix of the structure is disordered in solution (Supplemental Methods).

2.2.4 Discussion

Our results show that computational adaptations of basic evolutionary principles, such as recombination and mutation, can be used to accurately and rapidly design a diverse set of helical protein structures. Additionally, these methods allow incorporation of the idiosyncratic

structural elements that have proven difficult to incorporate into de novo designed structures. A surprising result from this study is that despite the incorporation of these non-ideal protein elements, several of the designed proteins exhibit high levels of thermostability, consistent with proteins designed using solely ideal structures[1, 2]. This result suggests that proteins designed for specific function have the potential to retain some of the remarkable thermodynamic properties achieved by modern protein design methods. The structure diversity displayed by the SEWING designs demonstrates the power of recombination to generate novel structures. In creating the designs, only a small portion of the SEWING graph was sampled. The graph for contiguous SEWING contains over 30,000 nodes with 345 million edges, allowing an estimated 7×10^{16} unique backbones that could be created by following three consecutive edges (Supplementary Methods). The diversity will further increase when alternative types of supersecondary structure are included, such as $\beta\alpha$ motifs and β-hairpins, and discontiguous and contiguous SEWING are merged. We anticipate that this structural diversity will be advantageous for functional design, as every backbone generated with SEWING has unique surface and pocket features that provide potential binding sites for ligands or macromolecules. Additionally, SEWING offers a direct approach for stitching together functional motifs from naturally occurring proteins, an approach frequently used by Nature to generate multi-functional proteins and allosteric systems.

2.3 Supplemental Methods

2.3.1 Computational Modeling

Extraction of Substructures

Contiguous and discontiguous substructures are composed of elements of secondary structure defined by the DSSP implementation in Rosetta[22]. The Rosetta implementation used here simplifies secondary structure classification to helices (H), loops (L), and β-strands (E). Contiguous substructures are composed of 3 adjacent elements of secondary structure that follow the pattern HLH. Discontiguous substructures are composed of 3-5 helical (H) elements, in which each helix makes favorable Leonard Jones contacts with every other (a clique). Contiguous substructures were extracted from the Top8000 dataset using the SmotifFeaturesReporter built into the Rosetta Features framework. Discontiguous substructures were extracted using the ModelFeaturesReporter on a non redundant set of the PDB generated from the Pisces web server[17]. A complete set of inputs and commands can be found in the Supplemental Material.

Geometric Hashing of Substructures

The geometric hashing algorithm used to calculate an all-against-all comparison of extracted substructures is an implementation of the algorithm described by Nussinov et. al[14]. The computational implementation is distributed as part of the Rosetta molecular modeling suite. The current implementation differs from the described algorithm in that noncollinear triplets (RS) were restricted to N/Cα/C backbone atoms of the same residue. For contiguous substructures two RS that place 10 atoms of the same Rosetta AtomType from a single secondary structure element into identical or adjacent quarter-angstrom bins were considered sufficient to infer structurally similarity. For discontiguous substructures, RS pairs were required to place 10 atoms of the same Rosetta AtomType from two elements secondary structure. To prevent clashes during recombination, any two RS pairs were rejected if they 1) placed any two atoms of different Rosetta AtomTypes in the same or adjacent bins, or 2) placed atoms for

additional elements of secondary structure in the same or adjacent bins. Rosetta commands for running the geometric hasher can be found in the Supplemental Material.

Graph Generation and Computational Design of Novel Structures

The extracted substructures were used to create nodes in a large graph. Edges were drawn between any two substructures that are deemed structurally similar by the geometric hashing metrics described previously. In total, the completed graphs contained 33,928 nodes and 345,700,224 edges for the contiguous substructures, and 4,584 nodes and 31,045 edges for the discontiguous substructures. The following equations were used to estimate the number of k-edge assemblies in a given graph:

n =number of nodes

e =number of edges

$$den(n,e) = \frac{2 \times e}{n(n-1)} \tag{2.1}$$

$$\text{number of k-edge assemblies } = n \times n \times den(n,e) \times n \times den(n,e)^2 \ldots den(n,e)^k \tag{2.2}$$

$$= n^{k+1} \times den(n,e)^{\frac{k(k+1)}{2}} \tag{2.3}$$

De Novo backbones were created from the graphs by random traversal of edges and structural recombination. Recombination of contiguous substructures entails an RMSD superimposition of all matched atoms from the geometric hashing algorithm, followed by the creation of a chimeric secondary structure element in which the atoms N-terminal to the midpoint of aligned region come from one substructure, and atoms C-terminal to the midpoint of the aligned region come from the other. Discontiguous substructures were recombined by aligning

the substructures on the two overlapping helices and removing the two helices from one of the parents (chosen randomly).

Sequences were designed onto recombined backbones by several rounds of Monte Carlo rotamer optimization followed by gradient-based energy minimization in Cartesian space. Early rounds of optimization were conducted with a dampened repulsive term, which was iteratively increased during the course of the optimization. Throughout the simulation, an energy bonus of 1 Rosetta Energy Unit (REU) was given to rotamers with the same amino-acid as the same position in the parental substructures. Loops were designed using a LoopHash-derived loop creation protocol[21]. A RosettaScript implementation of the graph-traversal and subsequent sequence optimization is provided in the Supplemental Materials[23].

2.3.2 Experimental Materials and Methods

Cloning and Expression

Linear gene fragments (gBlocks) encoding each protein were synthesized by Integrated DNA Technologies (Coralville, Iowa), and subcloned into either in-house pQE-HT and/or pCDB24 expression vectors. pQE-HT was constructed by inserting a TEV protease site between the hexa-histidine tag and BamHI restriction cut site. The final constructs were transformed into chemically competent BL21 Star E. coli cells and grown overnight at 37 C, and 225 r.p.m. in Luria-Bertani (LB) medium supplemented with 50μg/mL Ampicillin. The following day, 10mL of the overnight culture were used to inoculate 1.5L cultures of LB broth. Cells were grown at 37 C until an optical density (O.D.) of approximately 0.8 was reached. Protein expression was induced using 0.66mM of isopropyl thio-β-D-galactoside (IPTG), and cells continued to grow at 25 C. Cells were harvested after approximately 15 hours of protein expression and collected by centrifugation for 15 minutes at 4500 r.p.m.

For pCDB24 constructs, genes were subcloned using Gibson Assembly after vector linearization with BamHI. Protein was expressed in the same way as pQE-HT constructs.

Protein Purification

Cells were resuspended by vortexing in 25mL of wash buffer (20mM Tris-HCl, 300mM NaCl, 80mM Imidazole, pH 8.0). Resupsended cells were lysed by sonication at 4°C for 5 minutes. Cell debris was cleared by centrifugation at 20,000 r.p.m. at 4°C for 60 minutes. Cleared supernatant was filtered with a 5 micron filter and applied to an Econo-Pac gravity chromatography column (Bio-Rad) previously loaded with 5mL of His60 Ni Superflow Resin (Clontech) and equilibrated with 20mL of wash buffer. The column was washed with 40mL of wash buffer and bound protein was eluted in 15mL of elution buffer (20mM Tris-HCl, 300mM NaCl, 50mM Imidazole, pH 8.0). Purified protein was dialyzed overnight into 4 liters of wash buffer prepared without imidazole. Dialyzed samples were cleaved with TEV (pQE-HT constructs) or ULP1 (pCDB24 constructs) protease overnight at 4°C or for 1 hour at room temperature respectively. Cleaved samples were applied again to nickel chromatography columns equilibrated with dialysis buffer and the flow-through was collected as the final sample. Protein purity was assessed via SDS-PAGE electrophoresis.

Circular Dichroism (CD)

CD measurements were conducted on a Jasco J-815 CD spectrometer. All measurements were made in a buffer composed of 25mM sodium phosphate, 50mM NaCl, pH 7.0. Protein concentrations ranged from 10-150μM. All wavelength scans were collected at 20°C. Wavelength scans for contiguous designs were recorded using a 0.1mm cuvette with readings every 1nm from 280nm to 190nm. Measurements for discontiguous designs were recorded using a 1mm cuvette with readings every 1nm from 260 to 200nm (S2.1). Temperature denaturation was recorded at 222nm from 4 to 95°C for contiguous design, and from 20 to 95°C for discontiguous designs. Measurements were taken every 1nm, and temperature was increased 3°C every minute. Fraction folded was calculated as:

x=temperature

$u(x)$=linear fit of post-transition CD data

$f(x)$=linear fit of pre-transition CD data

$sm(x)$=CD data after rectangular smoothing with window of size 5

$$\frac{sm(x) - u(x)}{f(x) - u(x)} \times 100 \tag{2.4}$$

The free energy surface for CA01 was calculated using a Gibbs-Hemholtz equation modified to account for denaturant concentration[24].

Size Exclusion Chromatography/Multi-Angle Light Scattering (SEC/MALS)

SEC/MALS experiments were conducted using a Wyatt DAWN HELEOS II light scattering instrument interfaced to an Agilent FPLC System equipped with a Superdex S200 10/300 and a Wyatt T-rEX refractometer. Prior to sample loading, the equipment was equilibrated with a buffer composed of 25mM sodium phosphate, 250mM NaCl, 0.05% sodium azide, pH 7.0.

Crystallography

Samples of CA01 used for crystallographic studies were subject to size exclusion chromatography on a Superdex S75 16/600 equilibrated with 100mM Ammonium Acetate. Protein containing fractions were pooled and concentrated to an A280 of 12.8 (approximately 15mg/mL). Crystals were grown in 24-well hanging drop format in a buffer composed of 0.1M Tris:HCL pH 8.5, 1.0M Ammonium Phosphate Dibasic. Crystals were flash frozen in liquid nitrogen and data collected at the Advanced Photon Source (APS) BM line official name. The structure was solved by molecular replacement using the Rosetta model and PHASER. Models were refined using iterative rounds of Phenix refine and manual refinement in COOT[25, 26, 27].

Nuclear Magnetic Resonance (NMR) Spectroscopy of DA05

Good signal dispersion was observed in 1D 1H NMR spectra recorded for unlabeled DA05 and subsequently also in heteronuclear resolved 2D NMR experiments recorded for [5% 13C,U-15N]-labeled[28] DA05 and [U-13C,15N]-labeled DA05, which confirmed the finding inferred

from CD that the designed protein is well folded. Moreover, DA05 turned out to be highly soluble indicating that NMR-based structure determination appeared to be feasible. Hence, we acquired a comprehensive set of higher-dimensional NMR experiments for resonance assignment and structure determination (see Methods section).

NMR Solution Structure of DA05

Protein DA05 was nominated as a PSI:Biology community outreach target assigned to the Northeast Structural Genomics Consortium (http://www.nesg.org; NESG target ID OR626). The 2D [15N, 1H]-HSQC spectrum of DA05 (Figure 2.4) shows that a homogeneous NMR sample containing well-folded DA05 was obtained. Furthermore, the correlation time for isotropic reorientation estimated from average 15N spin relaxation times ($\tau_c =$ 6 ns at 25 °C) is monomeric in solution, as seen previously by size exclusion chromatography. A high-quality NMR solution structure was obtained (Table S2.3) and will be deposited into the protein data bank. Comparison of the DA05 NMR structure and the computationally predicted structure is the most rigorous test of the success of our design. We compared the predicted structure and the experimental structure by calculating several metrics: root mean square deviation (RMSD) values for backbone heavy atoms N, Cα, and C, by comparing ϕ, ψ, χ1 dihedral angles, and by identifying NOE-derived 1H1H upper-distance limit constraints which are violated in the design model. The RMSD value calculated for all backbone heavy atoms N, C and C of the first four α-helices (residues: 4-20, 24-41, 44-60, and 65-81) between the DA05 design model and the mean coordinates of the 20 conformers is 1.1$\mathring{A}$. Notably, the first and last turn of helix 1 exhibit increased flexible disorder. In contrast to the first four α-helices, α-helix 5 of the design model was not observed in NMR structure. The comparison of ϕ and ψ dihedral angles of the design model with the corresponding range observed in the 20 conformers representing the NMR solution structure likewise documents the high accuracy of the design model. 94% of ϕ angles and 96% of ψ angles in the design model are within $\pm 15^\circ$ of the corresponding angles of the regular secondary structure elements in the NMR solution ensemble. However, χ1 and χ2 angles in the NMR ensemble are not in similarly good agreement with the corresponding angles in the model mostly because they are not well defined.

Detailed NMR Methods

In order to acquire heteronuclear 13C/15N-resolved NMR spectra, designed protein DA05 was grown and purified as described above, except that cells were harvested by centrifugation at OD600 of 0.6 and then washed and transferred to minimal media with uniformly labeled 13C glucose and 15N ammonium chloride. Subsequently, protein over-expression was induced by adding 0.5 mM IPTG. NMR samples of [U-13C,15N]-labeled DA05 and biosynthetically-directed fractionally [5% 13C,U-15N]-labeled[28] DA05 were prepared at concentrations of 0.6 mM and 0.2mM in 90% H2O/10% D2O containing 25 mM sodium phosphate 50 mM NaCl (pH 6.5), 0.02% NaN3, and 0.5mM PMSF. An isotropic overall rotational correlation time of 6 ns was inferred from averaged 15N spin relaxation times, indicating that DA05 is monomeric in solution. The following spectra were recorded for [U-13C, 15N]-DA05 at 25 °C on Varian INOVA 750 spectrometer (with total measurement time of 6 days) equipped with a cryogenic 1H13C,15N probe: 2D [15N,1H] HSQC, aliphatic and aromatic 2D constant-time [13C,1H] HSQC, 3D HNCO, 3D HN(CA)CO, (4,3)D HNNCαC$\alpha\beta$, (4,3)D C$\alpha\beta$Cα(CO)NHN, aliphatic and aromatic (4,3)D HCCH, [29, 30], 3D aliphatic and aromatic (H)CCH COSY, 3D H(CC-TOCSY-CO)NHN[31], simultaneous 3D 15N/13Caliphatic/13C aromatic-resolved [1H, 1H]-NOESY (mixing time 70 ms, measurement time 2 days)[32], and long-range 2D [15N,1H] HSQC for His side chains [33]. For [5% 13C, U-15N]-DA05, aliphatic 2D constant-time [13C,1H]-HSQC spectra (with constant time delays of 28, 42 and 56ms) were acquired as described[34] (measurement time: 13.5 hours) in order to obtain stereo-specific assignments for Val and Leu isopropyl groups[28]. All NMR spectra were processed using PROSA[35] and analyzed using CARA[36]. Sequence-specic backbone (HN, N, Cα, Hα, and CO) and Hβ/Cβ resonance assignments were obtained by using the program AutoAssign[37, 38]. Resonance assignment of side-chains was accomplished using (4,3)D HCCH, 3D H(CC-TOCSY-CO)NH, and simultaneous 3D 15N/13Caliphatic/13Caromatic-resolved [1H, 1H]-NOESY. Overall, for residues 1-100, sequence-specific resonance assignments were obtained for 95% of backbone and 99% of side-chain resonances assignable with the NMR experiments listed above (Table S2.3). Chemical shifts will be deposited in the BioMagResBank soon. 1H1H upper distance limit constraints for structure calculation were obtained from simultaneous 3D 15N/13Caliphatic/13Caromatic-

resolved [1H,1H]-NOESY, and backbone dihedral angle constraints for residues located in well-defined regular secondary structure elements were derived from chemical shifts using the program TALOS N[39]. Automated NOE assignment was performed iteratively with CYANA[40, 41, 42], and the results were verified by interactive spectral analysis. Stereo-specific assignments of methylene protons were performed with the GLOMSA module of CYANA, and the final structure calculation was performed with CYANA followed by renement of selected conformers in an explicit water bath [43]using the program CNS[44]. Validation of the 20 rened conformers was performed with the Protein Structure Validation Software (PSVS) server[45]. The NMR structure will be deposited in the PDB.

2.4 Supplemental Figures

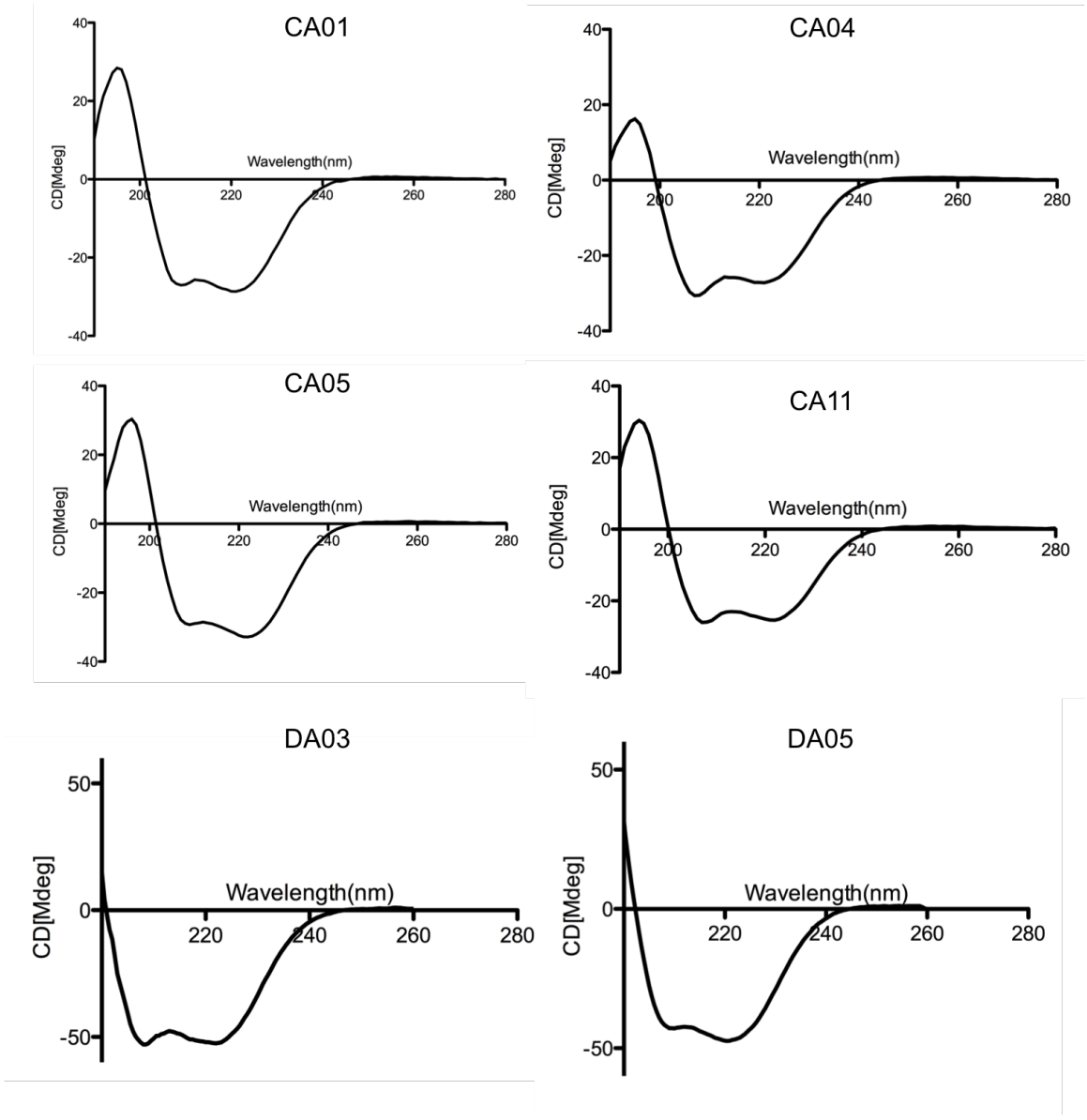

Figure S2.1: CD Spectra for folded SEWING designs. All models demonstrate peaks at 222nm and 208nm, characteristic of a helical protein.

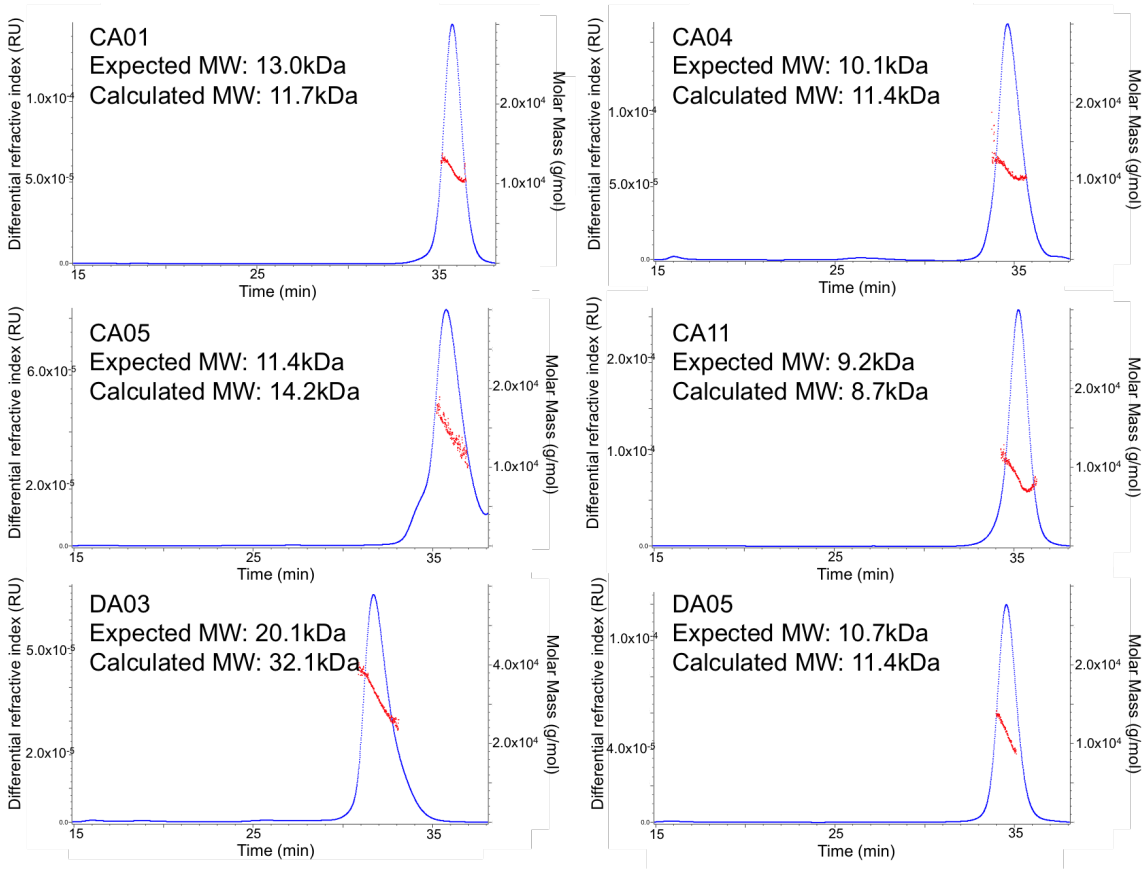

Figure S2.2: Size-exclusion chromatography/multi-angle light scattering data for SEWING designs. Experimentally fitted values for CA01, CA04, CA05, CA11, and DA05 suggest designs are monomeric in solution. The fitted value of DA03, despite a single homogenous peak, lies between a monomer and dimer.

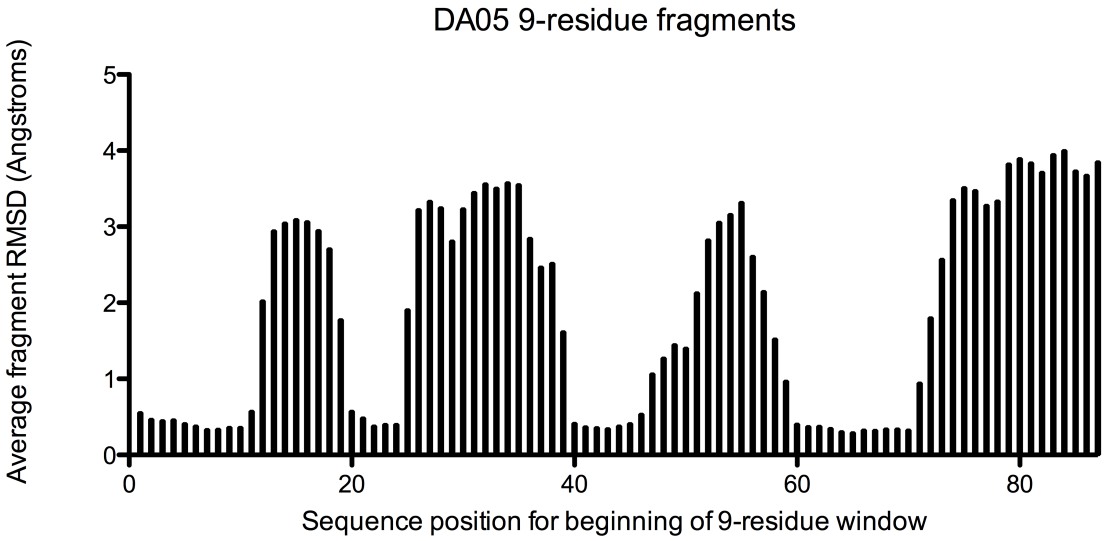

Figure S2.3: Fragment analysis for design DA05. The sequence of DA05 was used to select 200 9-residue fragments for each 9-residue window in the structure(10). The average RMSD/window is plotted, indicating a poor sequence/structure agreement in loop regions and in the disordered 5th helix.

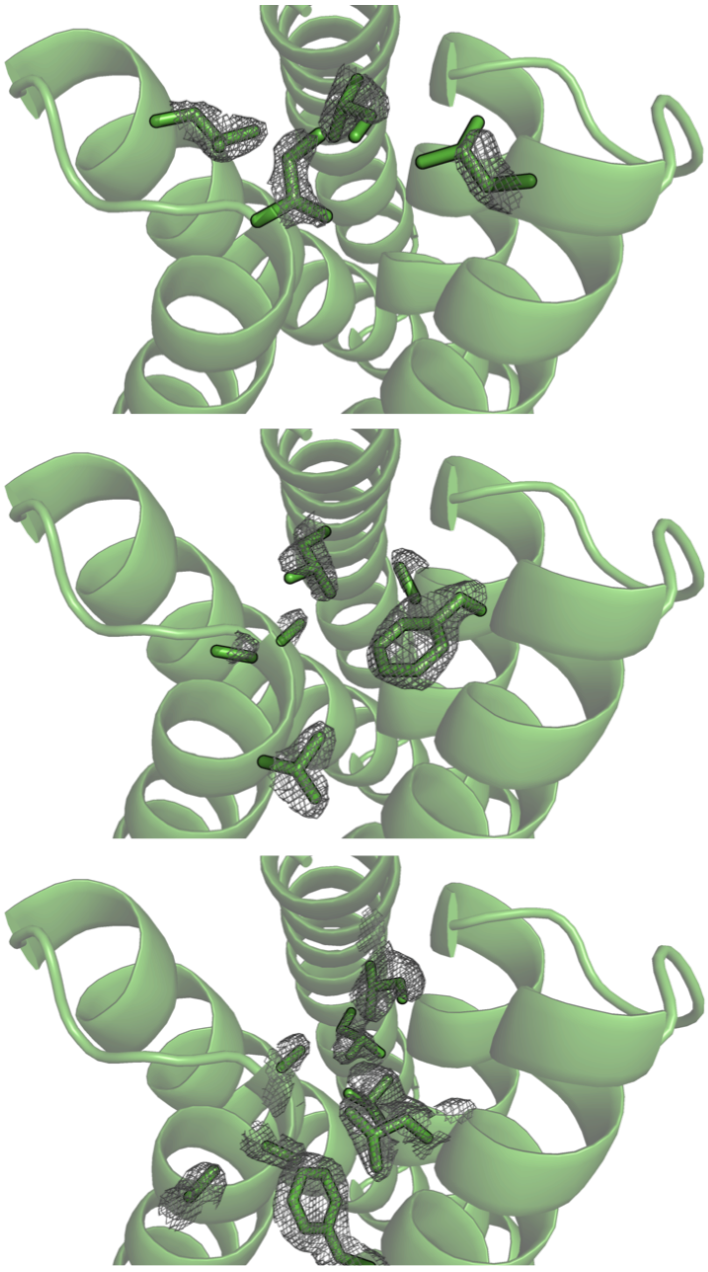

Figure S2.4: Design model CA01 superimposed with experimental electron density.

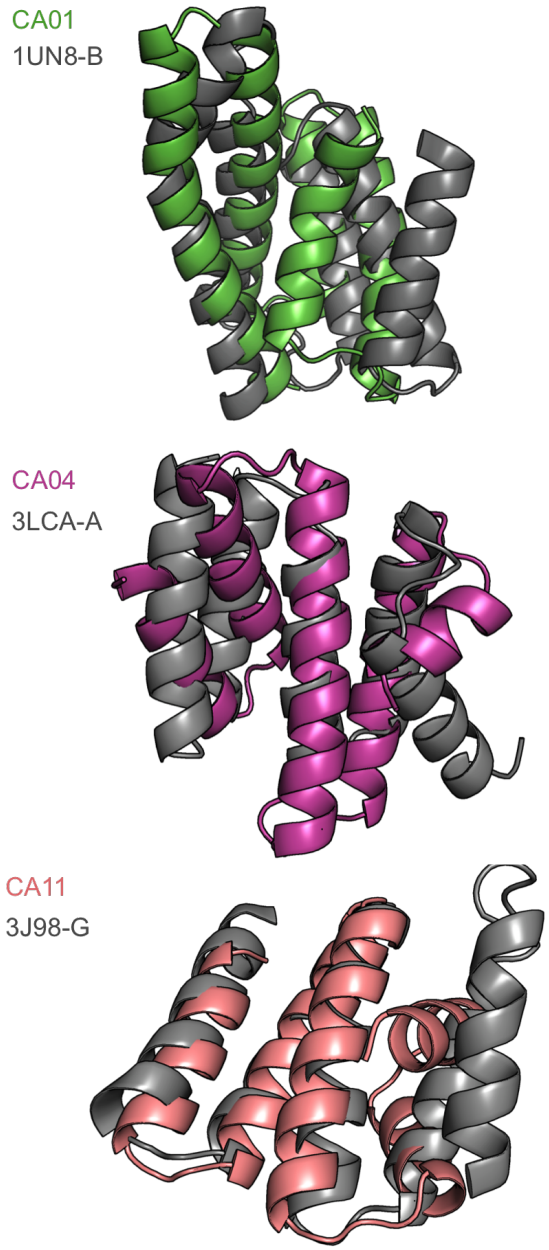

Figure S2.5: The top three design models determined by DALI Z-score, superimposed with closest structural homolog (Table S1)

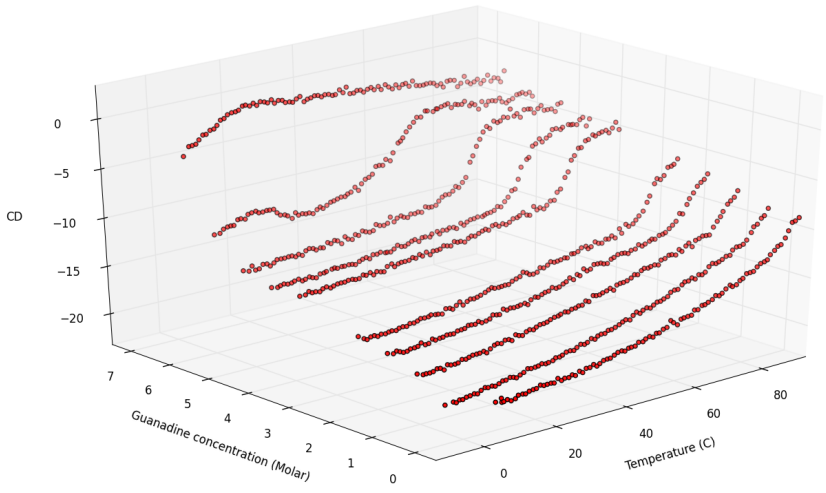

Figure S2.6: Full chemical and thermal denaturation of discontiguous design DA03 shows that the protein is highly thermostable, requiring high levels of denaturant to unfold.

2.5 Supplemental Tables

Design	Closest natural structural homolog	Z-score	RMSD
CA04	4G2V-A	6.8	5.2
CA03	2hsb-A	7	3.6
CA01	1un8-B	7	3.5
2LTA	3s3t-A	7	2.3
2LVB	2ofw-A	8.8	2.7
1YQS	3epb-A	8.9	2.1
CA11	3j98-G	9.1	2.6
2LV8	3gx1-B	9.5	2.6
2LN3	3zie-B	10.1	2.6
2KL8	5aj3-J	10.6	1.8
4UOS	4x0r-A	12.4	4.8
CA05	3dfb-A	12.4	1.1
4TGL	2wcd-T	14.5	5.9
DA05	4jhr-A	15.7	1.3
DA03	3dby-M	20.8	0.7

Table S2.1: DALI z-score comparison of De Novo designed proteins. SEWING design models are shown in red. Other designs are represented by 4-letter PDB code.

CA01	PEQMAEEIRQALEKILKQLENEIEIARNAGDDEREDRYRIAYLAALEAYRLLAEGVRIPEAVQ RAAAYLASMGYPHYAELFRAKGEELVKRLLEGKVTGEEFARQLVFYPAQAV
CA02	GVAKKLAIEAKQKGNLDAAAAFEDAQRALEMGHKDGAIAALLGALYLATGDERYRLEAEK MEADARKQLKTDGQSQEAHEALKGAQLG
CA03	VSIQEQEKKKMELIIEELKRRLLTNGYDNESAEALAWAILALVYMKQGDGARAATAAAMAA LMAQDPQKANEAAEAAKRLNTNDTTENRKIALQKAMEGANEALKYVTEMRK
CA04	EVEKKLAIKQQQAGNFDLALEYLKRELLTNGYDNTEAEAAAAAQIARDAKVNEDLGLMSE EEAERLRQRMIKVLLKLGKPLKKIQQMID
CA05	PFSEIGKFLGLAAKELQRGNLANAAVYIEKAAKEALKAGFKPMADYMNLAAERAKQGDYE TACKELYKAILEMFKISTKEGKQAPPDGVEGFKILQDAHRSWN
CA06	VGDAESMALLAALLGGAPLDVALEVVSKVKNVAAKVEEGANQGGMGENARAAIAMIAGA AAGAMAAAKKALHQGRKQEAEALERAARIALQQLDLLRKLGELGAEEQAKIALSLAAKAA ELARSEAARH
CA07	PLERAIREGRAEDVARILLNLGVRATRAAIEAERAALEGGMSQKAADLAKLAAIKLAQGNEE EAERLGRKALEV
CA08	AQELKEQAKRALKEGKGPEALRLIAKALKDEGLDRAAAKAEELAQRAEEGKVTLEEALKR ALRQALQERGEESKFDEELNKIK
CA09	AGDAKMGALIAGLEANKPLEKVERVANEVAEDAKKIEELLAHGKKELAREELKKLAEKLK DGPEGIRALIELIEARGNGDAERIDKHQKEARS
CA10	TEVMDRVRQALEDNNDTPQAKKILELLLEKLKRVLETNGYDNTSAEGEAAAQIGREAKVA EDLGLLSPEAAEKIRQEMIKIALKLDFPLKKIQQMLD
CA11	VKADRDEGQRQLEMESYDQAIEELLRAANKAQEEGNHDEAAKDVAQAILAAKELGKHPAQ IRQELEQKGISREAIEKGMREA
DA01	PRDAFERAMEMYKNGKYDRAIEYFRAVFSYGTTHEHAGRAMKYLGSAALGAGRTEEAIEA YLKAFAMLPNETETQKNLKIAQKLASL
DA02	SDDVLKQMKKAIKKGDYKTAVKLGEKEVRLDPNNTDALYLAALAMFQLGANAKKDDDRL SYLDAARRLVEVAGLTDTSKMINDALKKAKEKADK
DA03	PEDEARKEHAYWLAVLTLHAQLVLKALHDSEKTDILMAQEFVETFTRLLKRGTDYGLASQ AAAVAAAAFKLFLMIIIMKQKAGKIIINLSIEQLNKMVKEVEKYIEVLKDLKKSGTGTDDGEA AERMLGKALKWCEKANRANKGLLEADDAEQAAIEWAIYAAKFAKELSNRAQKNADATG
DA04	PTDDIAAQCLKECAEALRGAQAAVQQLYETALRQAKNGKGEGGEKLARFAKAFKELLRRI ESIKDNVNTPLGGLVLLASVLGAVAGVAGEFGREIAESEFPGEGAKVLIGAAARAPDLLKTL IAAITQEGTFAYGKWHAEITTLLKIIKLVLEALLQAVKIAAGGGVSDEMKDIIEDIKKAQERIE KAI
DA05	PDENIAKFEKAYKKAEELNQGELMGRALYNIGLEKNKMGKVKEAIEYFLRAKKVFDAEHD TDGARRAAKSLSEAYQKVEGSGDKGKIFQKEGESI
DA06	NRVELKILEKIAAVNADQAVKVSLQIAKDNPTEAREGAGNAAVAVGLARAALEMAEKAAIG FGGTEVAAILAKMRKMVEIARKAAQIAGKAHEGYKAREQASIAEETATKAAKQVTADAVQ VMGDDGY
DA07	VDDLIKKATEQIKEGRTEDAEKTAQKVLEADPSAADGHLNRGIALYLGGRYDLAAEDLKRA AALQQSDEAKKNLENALRAS
DA08	LTIAGQAVIDMAQLSAIMTSQIVKYLTSIATDPDLIGRYKTASEEFARAIDYLEQARDALIKG DYDSLNIYASAAKDGARNAELSFTGPVYIPEALHQAAQWLEALCDIVLIISKTTQPVTDIVAE LLKASKWARKMLEITAKQLVYETDDVKKYAIDGAKALDAMEGDLRALSQGVSGGLYWLK RAKEYL
DA09	PVDEWLKRAAKYLNNGQTKMAELEAAKVLAKEKDNAEALKRRAKARFLAGKLPEAIEDA LLARARTDETEAKALELLIRKQIKG
DA10	PDDLLAKARKSYNNGRAADVVRVVAKLVIKHSNNTDARQLAADALEELAKQAKTDDGRK MYMKGAYLLRYGADFTDDLAKAMKEAARFADN

Table S2.2: Sequences of designed proteins

Space group	C 2 2 21
Average mosaicity	0.17
Rwork/Rfree	20/25%
Resolution	92.16 to 2.20
Rmerge	0.069
Mean((I)/sd(I))	21.9
Completeness	99.6
Multiplicity	7.1

Table S2.3: Crystallography statistics for CA01

Completeness of resonance assignments (%)	
Backbone/side chain	94.9/98.6
Completeness of stereospecific assignments (%)	
Val and Leu isopropyl (%)	100
Conformationally-restricting distances constraints	
Intraresidue [i = j]	322
Sequential [—i - j— = 1]	230
Medium range [1 ¡ —I - j— ¡ 5]	290
Long range [—I - j— 5]	263
Dihedral angle constraints (ϕ/ψ)	65/65
Average number of constraints per residue	13.4
Average number of long range distance constraints per residue	2.9
CYANA target function ($Å^2$)	0.008± 0.003
Average number of distance constraint violations per conformer	
0.2-0.5 $Å$	4.1
>0.5$Å$	0
Average number of dihedral angle constraint violations per conformer	
>10°	0
Average RMSD from mean coordinates ($Å$)	
Regular secondary structure elements, backbone heavy atoms	0.47
Regular secondary structure elements, all heavy atoms	0.98
Ordered residues, backbone heavy atoms	0.5
Ordered residues, all heavy atoms	1
Global quality scores (raw/Z-score)	
PROCHECK G-factor (ϕ and ψ)	0.62/2.75
PROCHECK G-factor (all dihedral angles)	0.45/2.66
MOLPROBITY clash score	10.38/-0.26
Verify 3D	0.32/-2.25
ProsaII	0.91/1.08
RPF scores	
Recall/precision/F-measure	0.993/0.835/0.907
DP-score	0.822
Ramachandran plot summary (%)	
Most favored	96
Additionally allowed	4
Generously allowed	0
Disallowed	0

Table S2.4: NMR statistics for DA05

2.6 Command Lines and Input Files

2.6.1 Inputs for the Extraction of Features from a Set of Structures Into a Features Database

Command

```
rosetta_scripts.linuxgccrelease @flags_report
```

RosettaScript

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
    </SCOREFXNS>
    <TASKOPERATIONS>
    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
        <ReportToDB name=features database_name=scores.db3>
            <feature name=ResidueFeatures />
            <feature name=PoseConformationFeatures />
            <feature name=ResidueConformationFeatures />
            <feature name=ProteinResidueConformationFeatures />
            <feature name=ResidueSecondaryStructureFeatures />
            <feature name=SecondaryStructureSegmentFeatures />
            <feature name=ModelFeatures />
        </ReportToDB>
    </MOVERS>
    <APPLY_TO_POSE>
    </APPLY_TO_POSE>
    <PROTOCOLS>
        <Add mover_name=features />
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

Flag File

```
-parser:protocol extract_features.xml
-l inputs.txt
-ignore_unrecognized_res true
```

```
-delete_old_poses true
```

2.6.2 Inputs for generating SEWING nodes from Features database

Command

```
sewing_hasher.linuxgccrelease @flags
```

Flags File (contiguous)

```
-sewing:mode generate
-sewing:assembly_type discontinuous
-model_file_name continuous.models
-inout:dbms:database_name scores.db3
```

Flags File (discontiguous)

```
-sewing:mode generate
-sewing:assembly_type discontinuous
-model_file_name discontinuous.models
-inout:dbms:database_name scores.db3
```

2.6.3 Inputs for Running Geometric Hasher to Generate Edges

Inputs for the Generation of Plain-Text Edge File

```
sewing_hasher.linuxgccrelease @flags
```

Inputs for the Generatation of Plain-Text Edge File from Contiguous Substructures

```
-model_file_name continuous.models
-score_file_name continuous.scores
-sewing:mode hash
-sewing:num_segments_to_match 1
-sewing:min_hash_score 10
-sewing:max_clash_score 0
```

Inputs for the Generatation of Plain-Text Edge File from Discontiguous Substructures

```
-model_file_name discontinuous.models
-score_file_name discontinuous.scores
-sewing:mode hash
-sewing:num_segments_to_match 2
-sewing:min_hash_score 10
-sewing:max_clash_score 0
```

Command for the converstion of text-file to binary

```
sewing_hasher.linuxgccrelease @flags
```

Flags for the converstion of text-file to binary for contiguous edges

```
-model_file_name continuous.models
-score_file_name continuous.scores
-sewing:mode convert
```

Flags for the converstion of text-file to binary for discontiguous edges

```
-model_file_name discontinuous.models
-score_file_name discontinuous.scores
-sewing:mode convert
```

2.6.4 Inputs for Generating SEWING backbones

Command

```
rosetta_scripts.linuxgccrelease @flags
```

RosettaScript

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
    </SCOREFXNS>
```

```
    <TASKOPERATIONS>
    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
        <MonteCarloAssemblyMover
            name=assemble
            cycles=1000
            min_segments=5
            max_segments=5
            add_probability=0.05
            delete_probability=0.005
        />
    </MOVERS>
    <PROTOCOLS>
        <Add mover_name=assemble />
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

Flags

```
-nstruct 1
-s dummy.pdb

-sewing:model_file_name continuous.models
-sewing:score_file_name continuous.scores
-sewing:skip_refinement true
-parser:protocol generate.xml

-mh:match:ss1 true
-mh:match:ss2 true
-mh:match:aa1 false
-mh:match:aa2 false

-mh:score:use_ss1 true
-mh:score:use_ss2 true
-mh:score:use_aa1 false
-mh:score:use_aa2 false

-mh:path:motifs xsmax_bb_ss_AILV_resl0.8_msc0.3.rpm.bin.gz
-mh:path:scores_BB_BB xsmax_bb_ss_AILV_resl0.8_msc0.3

-mh:gen_reverse_motifs_on_load false
```

2.6.5 Inputs for the Optimization of SEWING Designs

Command

```
rosetta_scripts.linuxgccrelease @flags
```

RosettaScript

```
<ROSETTASCRIPTS>
    <TASKOPERATIONS>
    </TASKOPERATIONS>
    <SCOREFXNS>
        <talaris_cart weights="talaris2013_cart" />
    </SCOREFXNS>
    <FILTERS>
        <CavityVolume name="cav_vol" />
        <ExposedHydrophobics name="ex_hy" threshold=-1 />
        <SSPrediction name="sspred"
            threshold=1
            use_probability=true
            use_svm=true
        />
        <PackStat name="packstat" threshold=0 chain=0 repeats=1 />
        <SSShapeComplementarity name="ss_sc" />
        <BuriedUnsatHbonds2 name=bunsat
            scorefxn=talaris_cart
            cutoff=20
            generous_hbonds=true
            sasa_burial_cutoff=0.01
            AHD_cutoff=90
            dist_cutoff=3.0
            hxl_dist_cutoff=3.5
            sulph_dist_cutoff=3.3
            metal_dist_cutoff=2.7 />
    </FILTERS>
    <MOVERS>
        <FastRelax name=fastrelax
            repeats=1
            disable_design=false
            scorefxn=talaris_cart
            cartesian=1
        >
            <MoveMap>
                <Chain number=1 chi=1 bb=1 />
```

```
            </MoveMap>
        </FastRelax>
        <AssemblyConstraintsMover name=ACM
            native_rotamers_file="%%rotfile%%"
            native_bonus=1.0
        />
    </MOVERS>
    <PROTOCOLS>
        <Add mover_name="ACM"/>
        <Add mover_name="fastrelax"/>
        <Add filter_name="cav_vol" />
        <Add filter_name="ex_hy" />
        <Add filter_name="packstat" />
        <Add filter_name="sspred" />
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

Flags

```
-use_input_sc
-linmem_ig 10
-parser:protocol refinement.xml
-constrain_relax_to_start_coords
-relax:default_repeats 3
```

REFERENCES

1. Huang, P.-S., Oberdorfer, G., Xu, C., Pei, X. Y., Nannenga, B. L., Rogers, J. M., DiMaio, F., Gonen, T., Luisi, B., and Baker, D. (2014) High thermodynamic stability of parametrically designed helical bundles. Science **346**, 481–485

2. Koga, N., Tatsumi-Koga, R., Liu, G., Xiao, R., Acton, T. B., Montelione, G. T., and Baker, D. (2012) Principles for designing ideal protein structures. Nature **491**, 222–227

3. Joh, N. H., Wang, T., Bhate, M. P., Acharya, R., Wu, Y., Grabe, M., Hong, M., Grigoryan, G., and DeGrado, W. F. (2014) De novo design of a transmembrane zn-transporting four-helix bundle. Science **346**, 1520–1524

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

5. Hughes, A. L. (2005) Gene duplication and the origin of novel proteins. Proc. Natl. Acad. Sci. U. S. A. **102**, 8791–8792

6. Blake, C. C. F. (1978). Do genes-in-pieces imply proteins-in-pieces?

7. Bashton, M. and Chothia, C. (2007) The generation of new protein functions by the combination of domains. Structure **15**, 85–99

8. Koide, S. (2009) Generation of new protein functions by nonhomologous combinations and rearrangements of domains and modules. Curr. Opin. Biotechnol. **20**, 398–404

9. Eisenbeis, S., Proffitt, W., Coles, M., Truffault, V., Shanmugaratnam, S., Meiler, J., and Höcker, B. (2012) Potential of fragment recombination for rational design of proteins. J. Am. Chem. Soc. **134**, 4019–4022

10. Vogel, C., Bashton, M., Kerrison, N. D., Chothia, C., and Teichmann, S. A. (2004). Structure, function and evolution of multidomain proteins

11. Grishin, N. V. (2001) Fold change in evolution of protein structures. J. Struct. Biol. **134**, 167–185

12. Fernandez-Fuentes, N., Dybas, J. M., and Fiser, A. (2010) Structural characteristics of novel protein folds. PLoS Comput. Biol. **6**, e1000750

13. Aronson, H. E., Royer, W. E., and Hendrickson, W. A. (1994) Quantification of tertiary structural conservation despite primary sequence drift in the globin fold. Protein Sci. **3**, 1706–1711

14. Nussinov, R. and Wolfson, H. J. (1991) Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. Proc. Natl. Acad. Sci. U. S. A. **88**, 10495–10499

15. Leaver-Fay, A., Tyka, M., Lewis, S. M. S. M., Lange, O. F., Thompson, J., Jacak, R., Kaufman, K. W., Renfrew, P. D., Smith, C. A., Sheffler, W., Davis, I. W., Cooper, S., Treuille, A., Mandell, D. J., Richter, F., Ban, Y.-E. A., Fleishman, S. J., Corn, J. E., Kim, D. E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J. J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J. J., Kuhlman, B., Baker, D., and Bradley, P. (2011) Rosetta3: An Object-Oriented software suite for the simulation and design of macromolecules. Methods Enzymol. **Volume 487**, 545–574

16. Richardson, J. S. and Richardson, D. C. (2013). Invited review: Studying and polishing the PDB's macromolecules

17. Wang, G. and Dunbrack, R. L. (2003) PISCES: A protein sequence culling server. Bioinformatics **19**, 1589–1591

18. Sheffler, W. and Baker, D. (2010) RosettaHoles2: a volumetric packing measure for protein structure refinement and validation. Protein Sci. **19**, 1991–1995

19. Kumar, M. D. S., Bava, K. A., Gromiha, M. M., Prabakaran, P., Kitajima, K., Uedaira, H., and Sarai, A. (2006) ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions. Nucleic Acids Res. **34**, D204–D206

20. Holm, L. and Rosenström, P. (2010) Dali server: conservation mapping in 3D. Nucleic Acids Res. **38**, W545–9

21. Tyka, M. D., Jung, K., and Baker, D. (2012) Efficient sampling of protein conformational space using fast loop building and batch minimization on highly parallel computers. J. Comput. Chem. **33**, 2483–2491

22. Kabsch, W. and Sander, C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers **22**, 2577–2637

23. Fleishman, S. J., Leaver-Fay, A., Corn, J. E., Strauch, E.-M., Khare, S. D., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., Meiler, J., and Baker, D. (2011) RosettaScripts: a scripting language interface to the rosetta macromolecular modeling suite. PLoS One **6**, e20161

24. Kuhlman, B. and Raleigh, D. P. (1998) Global analysis of the thermal and chemical denaturation of the n-terminal domain of the ribosomal protein L9 in H2O and D2O. determination of the thermodynamic parameters, deltah(o), deltas(o), and deltac(o)p and evaluation of solvent isotope effects. Protein Sci. **7**, 2405–2412

25. Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H., and Adams, P. D. (2012) Towards automated crystallographic structure refinement with phenix.refine. Acta Crystallogr. D Biol. Crystallogr. **68**, 352–367

26. McCoy, A. J., Grosse-Kunstleve, R. W., Adams, P. D., Winn, M. D., Storoni, L. C., and Read, R. J. (2007) Phaser crystallographic software. J. Appl. Crystallogr. **40**, 658–674

27. Emsley, P., Lohkamp, B., Scott, W. G., and Cowtan, K. (2010) Features and development of coot. Acta Crystallogr. D Biol. Crystallogr. **66**, 486–501

28. Neri, D., Szyperski, T., Otting, G., Senn, H., and Wüthrich, K. (1989) Stereospecific nuclear magnetic resonance assignments of the methyl groups of valine and leucine in the DNA-binding domain of the 434 repressor by biosynthetically directed fractional 13C labeling. Biochemistry **28**, 7510–7516

29. Kim, S. and Szyperski, T. (2003) GFT NMR, a new approach to rapidly obtain precise high-dimensional NMR spectral information. J. Am. Chem. Soc. **125**, 1385–1393

30. Atreya, H. S. and Szyperski, T. (2004) G-matrix fourier transform NMR spectroscopy for complete protein resonance assignment. Proc. Natl. Acad. Sci. U. S. A. **101**, 9642–9647

31. Cavanagh, J., Fairbrother, W. J., Arthur G. Palmer, I., Skelton, N. J., and Rance, M. (2010) Protein NMR Spectroscopy: Principles and Practice. Academic Press

32. Shen, Y., Atreya, H. S., Liu, G., and Szyperski, T. (2005) G-matrix fourier transform NOESY-based protocol for high-quality protein structure determination. J. Am. Chem.

Soc. **127**, 9085–9099

33. Pelton, J. G., Torchia, D. A., Meadow, N. D., and Roseman, S. (1993) Tautomeric states of the active-site histidines of phosphorylated and unphosphorylated IIIGlc, a signal-transducing protein from escherichia coli, using two-dimensional heteronuclear NMR techniques. Protein Sci. **2**, 543–558

34. Penhoat, C. H. d., Li, Z., Atreya, H. S., Kim, S., Yee, A., Xiao, R., Murray, D., Arrowsmith, C. H., and Szyperski, T. (2005) NMR solution structure of thermotoga maritima protein TM1509 reveals a zn-metalloprotease-like tertiary structure. J. Struct. Funct. Genomics **6**, 51–62

35. Güntert, P., Dötsch, V., Wider, G., and Wüthrich, K. (1992) Processing of multi-dimensional NMR data with the new software PROSA. J. Biomol. NMR **2**, 619–629

36. Keller, R. Cantina-Verlag; goldau, switzerland: 2004. The Computer Aided Resonance Assignment Tutorial

37. Zimmerman, D. E., Kulikowski, C. A., Huang, Y., Feng, W., Tashiro, M., Shimotakahara, S., Chien, C., Powers, R., and Montelione, G. T. (1997) Automated analysis of protein NMR assignments using methods from artificial intelligence. J. Mol. Biol. **269**, 592–610

38. Moseley, H. N., Monleon, D., and Montelione, G. T. (2001) Automatic determination of protein backbone resonance assignments from triple resonance nuclear magnetic resonance data. Methods Enzymol. **339**, 91–108

39. Shen, Y. and Bax, A. (2013) Protein backbone and sidechain torsion angles predicted from NMR chemical shifts using artificial neural networks. J. Biomol. NMR **56**, 227–241

40. Buchner, L. and Güntert, P. (2015) Systematic evaluation of combined automated NOE assignment and structure calculation with CYANA. J. Biomol. NMR **62**, 81–95

41. Herrmann, T., Güntert, P., and Wüthrich, K. (2002) Protein NMR structure determination with automated NOE assignment using the new software CANDID and the torsion angle dynamics algorithm DYANA. J. Mol. Biol. **319**, 209–227

42. Güntert, P., Mumenthaler, C., and Wüthrich, K. (1997) Torsion angle dynamics for NMR structure calculation with the new program DYANA. J. Mol. Biol. **273**, 283–298

43. Linge, J. P., Williams, M. A., Spronk, C. A. E. M., Bonvin, A. M. J. J., and Nilges, M. (2003) Refinement of protein structures in explicit solvent. Proteins **50**, 496–506

44. Brünger, A. T., Adams, P. D., Clore, G. M., DeLano, W. L., Gros, P., Grosse-Kunstleve, R. W., Jiang, J. S., Kuszewski, J., Nilges, M., Pannu, N. S., Read, R. J., Rice, L. M., Simonson, T., and Warren, G. L. (1998) Crystallography & NMR system: A new software suite for macromolecular structure determination. Acta Crystallogr. D Biol. Crystallogr. **54**, 905–921

45. Bhattacharya, A., Tejero, R., and Montelione, G. T. (2007) Evaluating protein structures determined by structural genomics consortia. Proteins **66**, 778–795

Chapter 3

FUNCTIONAL INCORPORATION OF BINDING MOTIFS INTO DE NOVO DESIGNED PROTEINS

3.1 Introduction

Protein-protein interactions mediate a wide variety of cellular processes. Designed and engineered protein-protein interactions have the potential to probe and modify these processes to great effect. Antibodies, for instance, have long-been experimentally engineered to bind a multitude of targets for applications ranging from laboratory reagents to therapeutics [1, 2]. Many methods exist for the generation and optimization of protein-protein interactions. Directed evolution is a powerful experimental method used to screen large libraries of dna and protein sequences for the generation or optimization of a selectable attribute, such as protein-protein binding. However, the library sizes available to directed evolution methods are limited by experimental constraints, and are rarely able to sample the entirety of a desired sequence-space. Additionally, directed evolution methods are often unable to select for properties such as specificity of binding partners and binding locations, and thermostability; all of which are critical properties for most downstream applications. Computational design is a rapidly improving strategy for the creation of novel protein-protein interfaces, and benefits from atomistic control over the properties of the engineered protein. Recent computational design efforts efforts have succeeded in the generation of de novo homo-dimers, metal-mediated interactions, protein nano-cages, and viral antagonists[3, 4, 5, 6]. In each of these cases, the protein binder has been designed by modifying the surface residues of an existing protein structure that has been selected for structural properties conducive to the design task, such as shape complementarity to the target molecule. An alternative starting point for the computational design of protein interfaces is the creation of new, or de novo, protein backbones. However,

many de novo protein design strategies target a single fold, and are therefore poorly suited for sampling a large number of diverse backbones for compatibility with the desired binding partner[7, 8, 9, 10]. Recently, we described a novel method of de novo protein design, called SEWING, that utilizes evolutionary principles to create large numbers of highly diverse and designable protein structures. Here, we extend SEWING to the creation of protein backbones as templates for interface design. Our results indicate that SEWING can be used for the generation of well-folded de novo proteins that retain binding to the intended target.

3.2 Materials and Methods

3.2.1 Selection of Target Interfaces

As an initial test of our method, we sought to design de novo protein backbones that incorporate binding peptides from 4 distinct protein-protein interfaces: Activated GαQ bound to its effector, phospholipase Cβ3 (PLC-β3) (PDB: 3OHM); Scallop Troponin C bound to a peptide fragment from Troponin I (PDB: 3TZ1); Vinculin head domain bound to Actinin peptide (PDB: 1YDI); and anti-apoptotic protein Mcl-1 in complex with a BH3 peptide derived from proapoptotic protein Bim (PDB: 3KJ0). These interfaces were selected due to a combination of therapeutic relevance, size and structure of the binding peptide, and availability of binding affinities.

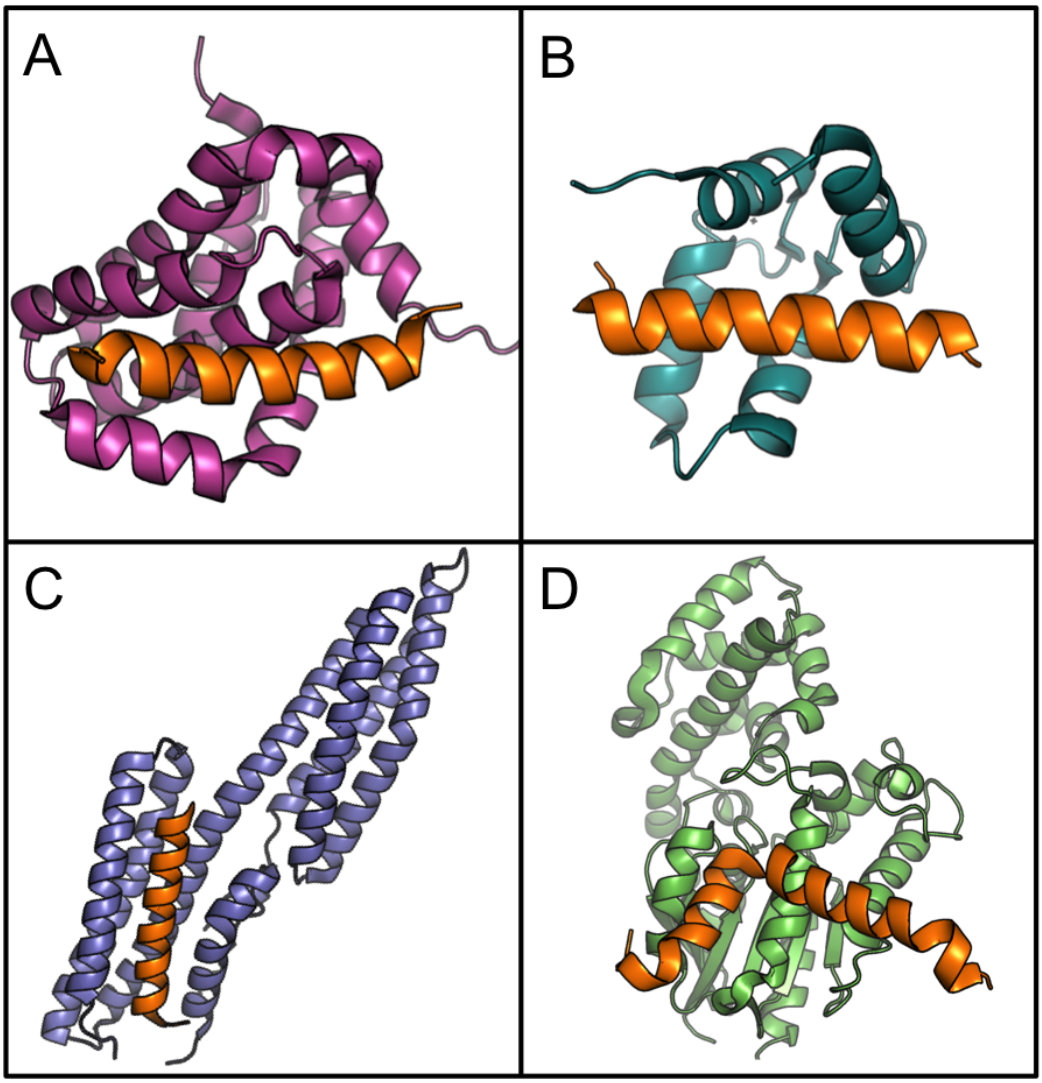

Figure 3.1: Starting scaffolds for interface design. In all cases, the binding peptide used as the starting structure is highlighed in orange. (A) Interface between Mcl-1 (pink) and BH3 peptide from Bim (PDB: 3KJ0). (B) Interface between scallop Troponin C (teal) and peptide derived from scallop Troponin I (PDB: 3TZ1). (C) Interface between Vinculin head domain (blue) and Actinin peptide (PDB: 1YDI). (D) Interface between GαQ and a peptide derived from PLC-β3 (PDB: 3OHM).

In the case of vinculin, troponin, and Mcl-1 interfaces, the binding peptide adopts a single helix conformation with affinities ranging from 1nM to 1μM[11, 12, 13]. A helix binding to a

groove on a target protein is commonly utilized interface motif in naturally occurring protein-protein interactions, and therefore these interfaces serve as an excellent proof-of-concept for the generality of this interface design method(Figure 3.1A, 3.1B, 3.1C)[14]. In the case of the GαQ interface, a 27-residue binding peptide adopts a kinked helix-turn-helix conformation(Figure 3.1D) that binds PLC-β3 with a dissociation constant of 4μM[15]. The two helices in the PLC-β3 peptide are nearly perpendicular to one another in the binding pose, a conformation not seen in previously designed proteins, and thus a highly challenging test our methods ability to design an accommodating structure.

From a clinical perspective, the Mcl-1/BH3 and GαQ/Cβ3 interfaces are of particular interest. Mcl-1, a Bcl-2 antiapoptotic protein, protects cells from programmed cell death. The sequestration of BH3 peptide by antiapoptotic proteins allows cells to evade death, and can lead to cancer development[16, 17]. There are several members of the anti-apoptotic Bcl-2 family with varying specificity for different BH3 peptides. The multi-specific nature of many BH3 peptides can convolute experimental studies into the effects of certain pathways and researchers have engineered binders with high specificity towards various Bcl-2 family members[18]. The design strategy employed here can be used to similar effect by extending the interface surface area and allowing the introduction of specificity-inducing interactions not achievable with a peptide alone.

The signalling through the GαQ/PLC-β3 interface is an important regulator of cell proliferation[15]. Recent studies have implicated excess GαQ signaling in the development of ocular cancers [19, 20]. Therefore, the inhibition of GαQ signalling via a specific antagonist to the GαQ/PLC-β3 interface represents a potential method of treatment.

3.2.2 Computational Design Protocol

The previous chapter describes a novel method of protein design in which substructures from known protein structures are recombined using structural similarity. We showed that this method gives rise to diverse sets of highly designable de novo protein backbones. In those works, substructures were defined by secondary structure boundaries, without consideration of potential function. The all-against-all structurally similarity between substructure pairs was

precomputed, and used to construct a graph of potential recombination events. In this chapter, additional nodes are defined by their functional properties (in this case, binding to the target molecule) and appended through on-the-fly structural similarity to the other nodes in the graph. When constructing novel backbones, only paths that include the desired function node are considered. In this way, de novo protein backbones can be constructed that incorporate substructures intended to impart function on the designed protein (Figure 3.2).

In this study we aimed to design novel proteins that incorporate binding epitopes from 4 target interfaces. In all cases, the contiguous SEWING framework designed in Chapter 2 was used, which recombines substructures based on overlap of a single secondary structure elements(Figure 3.2). For the 3 interface peptides consisting of a single helix, additional substructures can be appended to either the amine or carboxy terminus of the starting peptide. However, in these cases the addition of substructures from both termini was not allowed for fear that multiple structural alignments over the same region could allow the introduction of structural deviations in the peptide backbone that could reduce modeling accruacy or hinder binding. In the case of the PLC-β3 peptide, which consists of two α-helices, additions to both termini were allowed. Additionally, in order to preserve residues deemed to be critical for binding, any structural recombinations that resulted in the shortening of the peptide such that any of these critical residues was removed were not considered. Finally, in order to prevent the addition of substructures that would cause steric occlusion of the intended binding, all design models were built in the presence of the structure for the intended binding partner. Together, these additional requirements create a much more restrained system than the designs outlined in the previous chapter. In order to account for this additional complexity, a low resolution score-function was implemented to guide sampling towards backbones that not only avoid clashes, but also favor the potential to design favorable inter- and intra- chain contacts. Importantly, the implemented score-function terms do not take into consideration the amino acid identities of the protein being designed, but rather quantify the ability to design a favorable contact during later sequence optimization steps. The full designability score-function used for these simulations is described in Equation 3.1

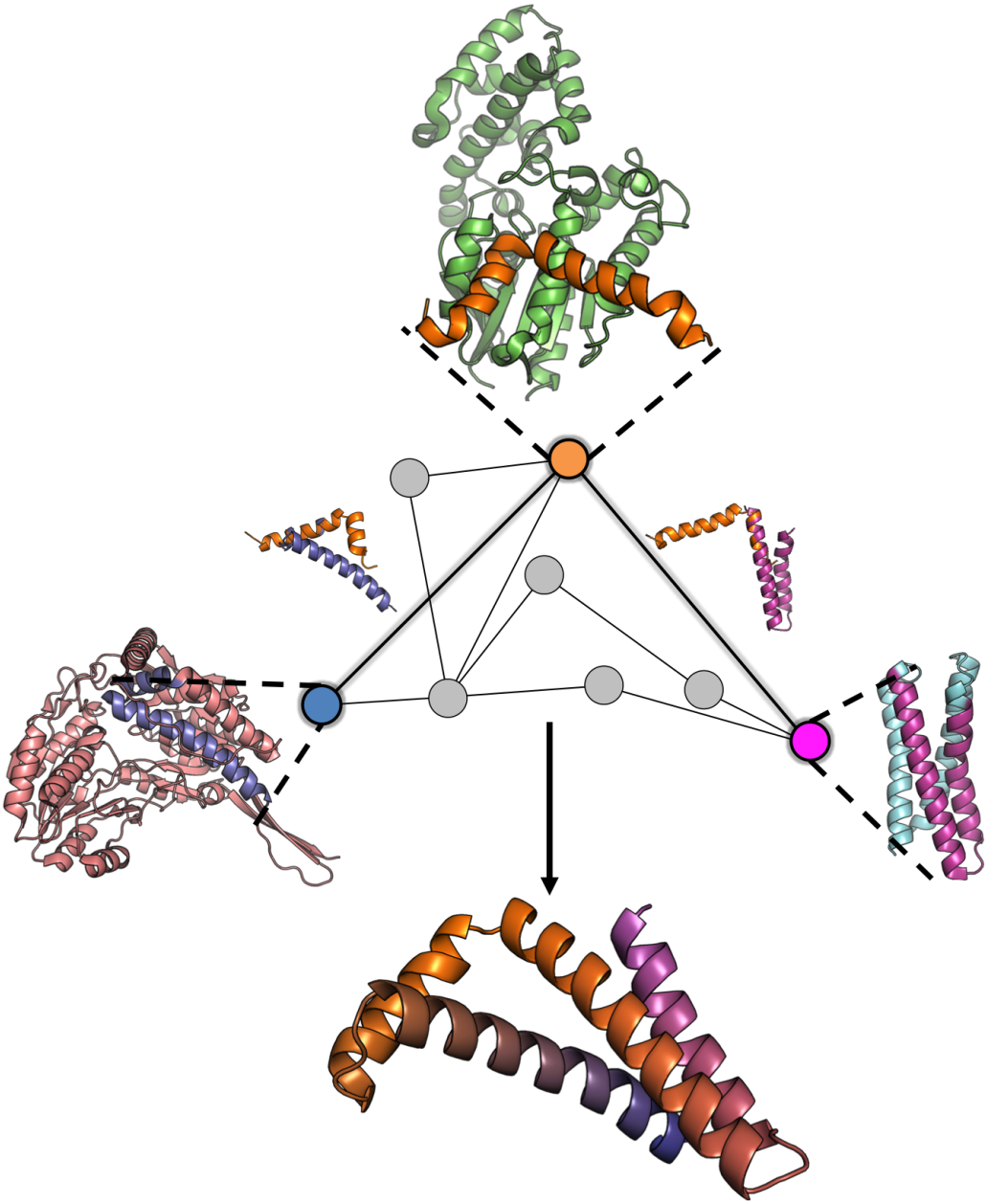

Figure 3.2: Schematic of the SEWING append method. The PLC-β3 peptide (orange) is used as the initial substructure node. Additional contiguous substructures derived from natural proteins are added by structural similarity. The final design model (rainbow) illustrates the design is a chimera from the various substructures.

$$E_{BB} = \sum_{1 \to n} \Theta_{clash} + \Theta_{intra-design} + \Theta_{inter-design} \tag{3.1}$$

In this equation, the $\Theta_{intra-design}$ favors designability of intra-chain contacts, the $\Theta_{inter-design}$ favors design of inter-chain contacts, and the Θ_{clash} term prevents both intra- and inter-chain backbone clashes. All score terms were normalized to account for the variable numbers of residues between different designs. Weights for each of the three terms were varied empirically to favor design models that demonstrate a clash-free globular topology that maximizes interface area. When designing structures, additional edges from the substructure graph can be added or removed in a monte-carlo based process that attempts to optimize the E_{BB}.

For each of the 4 binding peptides, 1,000 design simulations, each consisting of 10,000 monte-carlo cycles were run. Simulations were allowed to follow either 2 or 3 edges, resulting in final design models of either 3, 4, or 5 helices, dependent on the number of helices in the starting substructure. Following backbone creation, sequence refinement was carried out as previously described in Chapter 2, with the additional modification that mutation of critical interface residues was not permitted. Fully refined design models were filtered on metrics to favor low predicted energy, tight packing of the protein core, and shape complementarity between the designed protein and desired binding partner[21]. In total, 10 design models were selected for experimental characterization (Figure 3.3).

3.2.3 Protein Expression and Purification

Linear gene fragments (gBlocks) encoding each protein were synthesized by Integrated DNA Technologies (Coralville, Iowa). Additional 19 base-pair overhangs were added to each fragment to allow gibson cloning into in-house pCDB24 vectors previously linearized with XhoI restriction enzyme(New England Biolabs)[22]. Constructs were transformed into chemically competent BL21 Star E. coli cells and grown overnight at 37 °C, and 225 r.p.m. in Luria-Bertani (LB) medium supplemented with 50ug/mL Ampicillin. The following day, 10mL of the overnight culture were used to inoculate 1.5L cultures of LB broth. Cells were grown at 37 °C until an optical density (O.D.) of approximately 0.8 was reached. Protein expression was induced using 0.66mM of isopropyl thio-β-D-galactoside (IPTG), and cells continued to grow at 25 °C. Cells were harvested after approximately 15 hours of protein expression and collected by centrifugation for 15 minutes at 4500 r.p.m. pCDB24 constructs are expressed

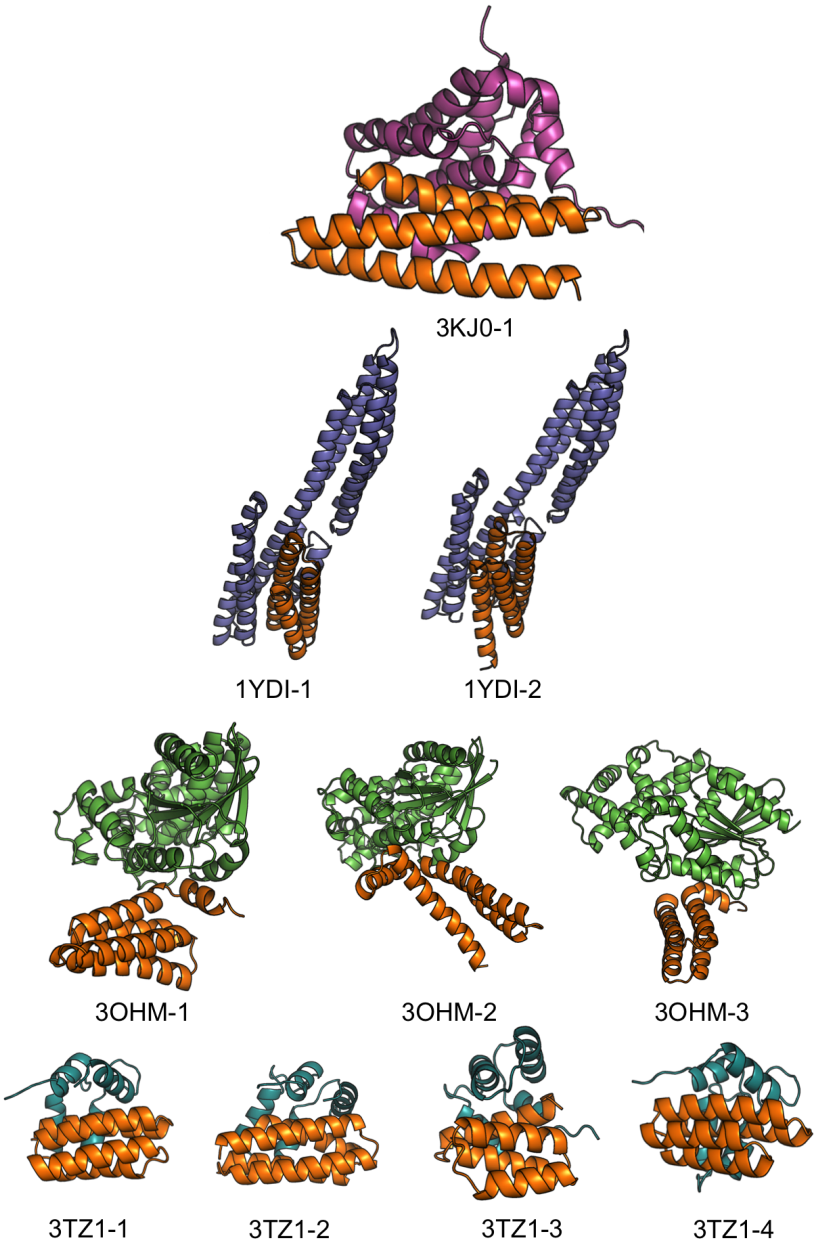

Figure 3.3: Experimentally characterized binding interfaces. Design models for each of the 10 experimentally characterized designs (orange) shown in complex with the target protein. A summary of experimental results can be found in Table 3.1

as fusions to the Saccharomyces cerevisiae protein Smt3, which aids in protein expression and solubility. A deca-histidine tag is included upstream of the Smt3 gene. Proteins were purified from cell pellets as described in the previous chapter.

3.2.4 Yeast Surface Display

In addition to pCDB24, all genes encoding proteins designed to bind GαQ were cloned into the pETCON2 yeast-display vector[6]. Briefly, 50 base-pair overhangs containing homology to the pETCON vector were used for cloning via homologous recombination. EBY100 yeast were transformed with completed constructs. Upon induction with galactose-containing media, encoded proteins are displayed on the surface of the yeast cell as fusions to the aga1p and aga2p proteins[23]. HA and C-myc tags are expressed on the amine and carboxy termini, respectively.

Biotinylated GαQ was labeled with streptavidin fused to Alexa Fluor 633 (ThermoFisher). Protein-displaying cells were labeled with chicken anti-C-myc antibodies, followed by Goat anti-chicken fused to alexa fluor 488. Simultaneous detection of protein expression and binding to GαQ was analyzed through detection of the respective fluorophores on a FACSAria flow cytometer(BD).

3.2.5 Isothermal Titration Calorimetry

The binding affinity of designs 3TZ1-2 and 3TZ1-3 to Troponin C was measured via isothermal titration calorimetry on a MicroCal Auto-iTC200. Purified designs at a concentration between 20μM and 30 μM were loaded in the sample cell, and purified Troponin C was at a concentration of 400μM was titrated in through 20 injections of 2μl over 1 hour. The results were fit to a single-state binding event, and yielded affinities of 7.1μM(N=0.95) and 3.8μM(N=1.36) for designs 3TZ1-2 and 3TZ1-3, respectively. All experiments were conducted in a buffer composed of 25mM $NaPO_4$, 50mM NaCl.

3.2.6 Surface Plasmon Resonance

The affinities of 3OHM-1, 3OHM-2, and 3OHM-3 against GαQ were measured on a ProteOn XPR36 biosensor. Biotinylated GαQ at a concentration of 100nM was loaded onto a neutravidin sensor chip (ProteOn NLC). Affinity for design models was calculated by flowing purified designs at 5 concentrations, ranging from 0.37μM to 30μM. Resultant data was fit to an equilibrium binding model for single-state binding. All experiments were conducted in a

buffer composed of 20mM HEPES pH 7.0, 10mM NaF, 30μM $AlCl_3$, 10mM MgCl2, 150mM NaCl, 2mM DTT, 0.005% Tween-20.

3.2.7 Crystallography

Sample of 3OHM-2 used for crystal growth were further purified via size exclusion chromatography on a Superdex S75 16/600 equilibrated with 100mM Ammonium Acetate. Pooled fractions were concentrated to 225μM and grown in 24-well crystallography trays (Hampton Research). Crystals formed in approximately 3 days in a buffer composed of 0.1M diammonium hydrogen citrate, and 15% Polyethylene Glycol (PEG). Crystals were flash frozen in liquid nitrogen after cryoprotection in a solution composed of the mother liquor mixed with 30% ethylene glycol. The structure was solved by molecular replacement with pairs of polyalanine helices derived from the design model, followed by further refinement with SHELXE and arp/warp.[24, 25, 26].

3.3 Results

The 10 models selected for experimental characterization are highly diverse, consistent with results from Chapter 2. The functional peptides vary in their positioning in the completed designs, and designs encompass both 3 and 4 helix-bundles (Figure 3.3). These results further demonstrate the advantage of SEWING in producing large numbers of diverse protein backbones.

design	expression	helical	well-folded	binding
3KJ0-1	no	ND	ND	ND
1YDI-1	no	ND	ND	ND
1YDI-2	no	ND	ND	ND
3OHM-1	yes	yes	ND	yes
3OHM-2	yes	yes	yes	yes
3OHM-2	yes	yes	ND	yes
3TZ1-1	no	ND	ND	ND
3TZ1-2	yes	yes	ND	yes
3TZ1-3	yes	yes	ND	yes
3TZ1-4	no	ND	ND	ND

Table 3.1: Experimental summary of interface designs. 5 of the 10 designs were expressed well in bacteria and were helical via CD. 3 of the 5 designs were well-folded via CD, and all 3 bound their desired target.

5 of the 10 designs were readily purified from bacterial culture. All of the purified proteins contained significant helical content via Circular Dichroism (CD), but display variety in melting temperature and cooperativity of unfolding (Table 3.1, Figures 3.4 and 3.5).

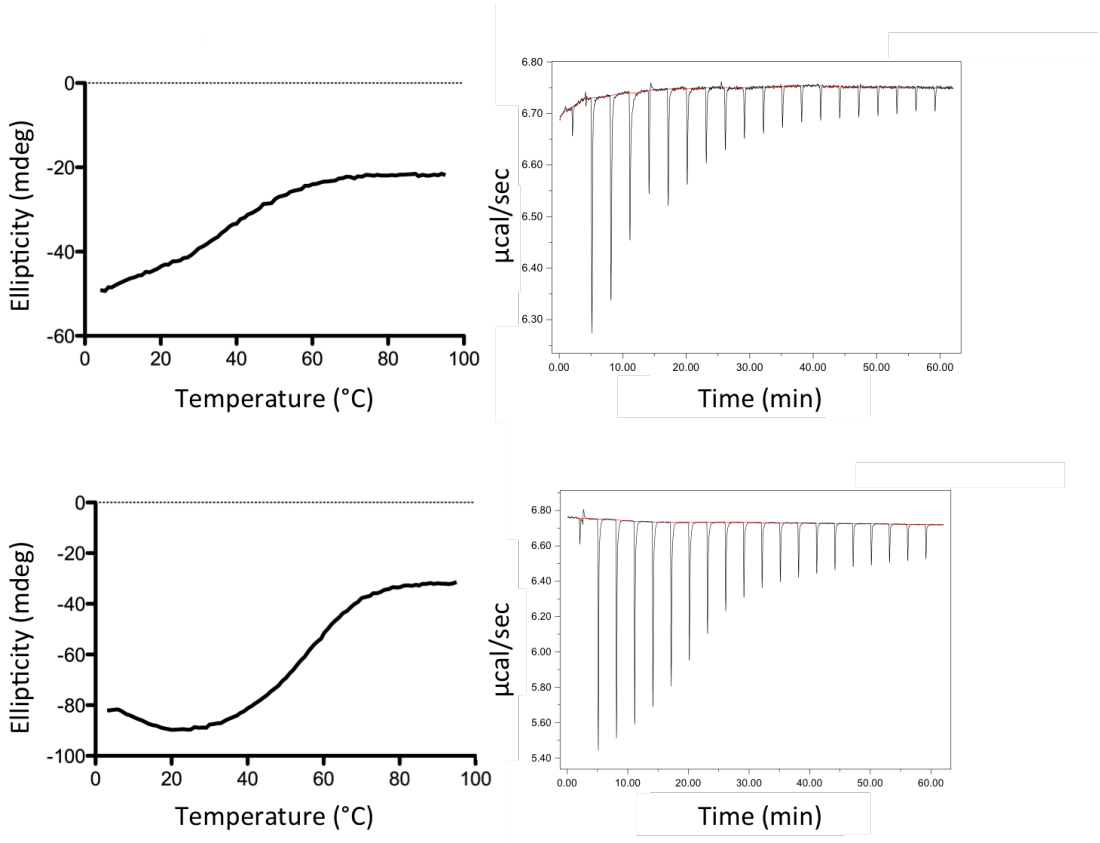

Figure 3.4: Thermal denaturation and isothermal titration calorimetry for 3TZ1-2 (top) and 3TZ1-3 (bottom). In both cases, a cooperative unfolding event is observed indicating a well folded structure. Binding to Troponin C is observed in both cases.

Binding was assayed through 3 separate methods: yeast surface display, isothermal titration calorimetry (ITC), and surface plasmon resonance (SPR). In the case of binders to the scallop Troponin C, affinity was determined via ITC. Titrations of purified Troponin C into design models showed endothermic binding for both 3TZ1-2 and 3TZ1-3 designs. In both cases, affinity for Troponin C molecule was determined to be in the low micromolar range, consistent with the reported affinities of binding the peptide alone [11, 27]. The interaction between Troponin C and Troponin I peptide has been shown to demonstrate increased affinity in the presence of calcium. In our experiments, excess calcium was not added, and yet the

affinities aligned well with previously reported data for binding in the presence of calcium, indicating that our measured affinities may represent a lower bound of affinity. Unfortunately, additional experiments conducted in the presence of calcium could not be readily fit (data not shown).

For the designs incorporating the PLC-β3, binding was assessed both via yeast display and SPR. For both assays, strong binding to GαQ was observed (Figure 3.5). Equilibrium analysis of SPR data shows that all three tested designs bind to GαQ in the high nanomolar, to low micromolar range, comparable to the 4μM binding measured for the PLC-β3 peptide alone (data not shown). For one design, 3OHM-2, a crystal structure was solved for the unbound state. The crystal structure contains two molecules in the asymmetric unit, and agreement between the design model and crystal structure is poor (Figure 3.6). Curiously, the crystallized conformation of design appears to be inconsistent with the designed binding mode(Figure 3.6). Given that it is unlikely such high affinity binding was achieved serendipitously, this suggests that either the crystallized conformation differs from the conformation in solution, or that the designed protein undergoes a conformational change upon binding.

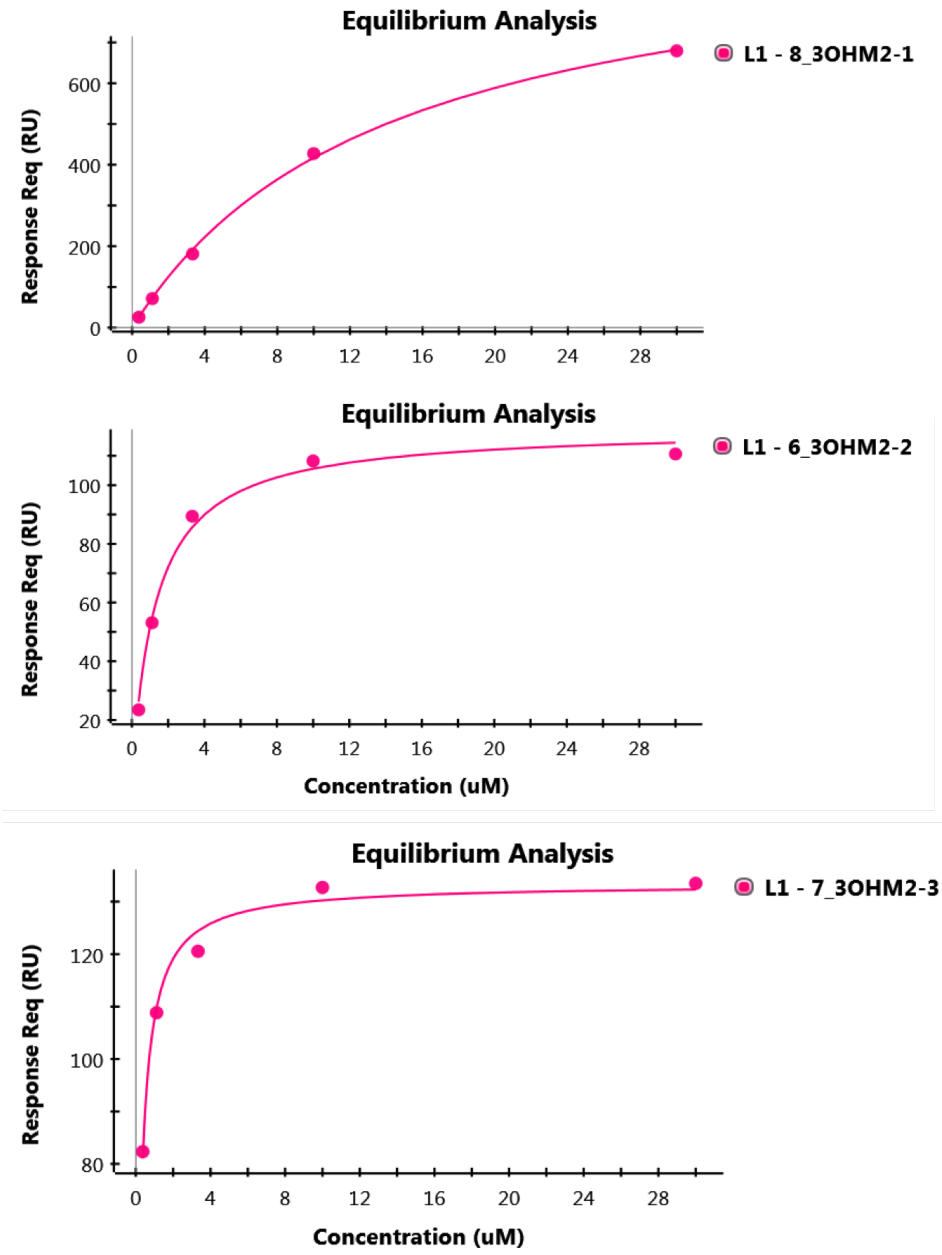

Figure 3.5: Equilibrium analysis of SPR data for the 3OHM-1, 3OHM-2, and 3OHM-3 show that all designs bind GαQ. Binding affinities were fit to 14μM, 1.3μM and 239nM, respectively

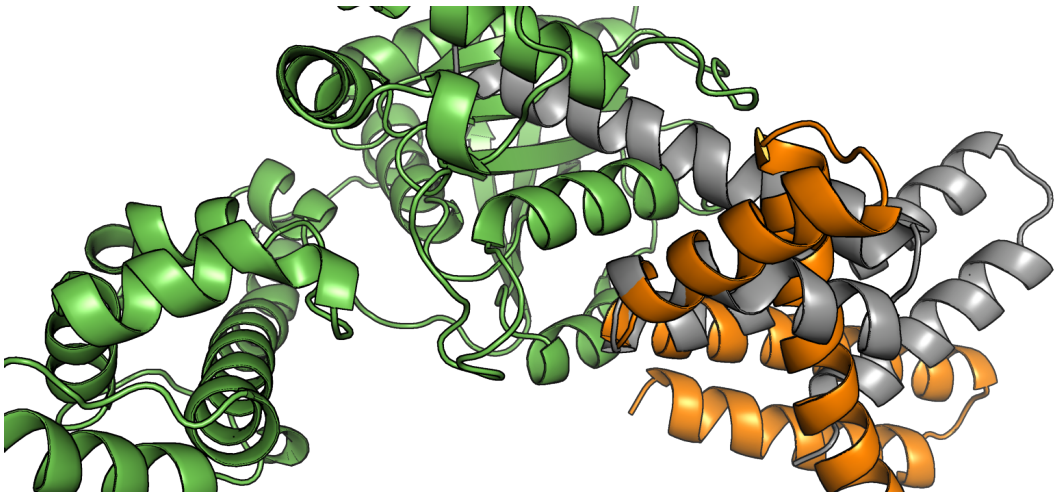

Figure 3.6: Comparison of 3OHM-2 design model and crystal structure shows that the crystallographic form (gray) is incompatible with the designed binding orientation (orange). However, binding is observed in the low μM range (Figure 3.5, Table 3.1).

3.4 Discussion and Future Directions

The results reported here provide strong preliminary data for the use of de novo design for the improvement of peptide binding affinity. Each design that was able to be purified showed strong helical character, and maintained binding affinity to the target protein that was at least as strong as the peptide alone. However, stability and cooperativity of unfolding for designs reported here is much less than than previously reported de novo designs, highlighting the difficulty in restricting de novo design by forcing the incorporation of desired structural features. One hypothesis for this difficulty is that in many of the tested cases the sequences for the binding peptide show relatively poor preference for the bound structure, suggesting the possibility of a conformational change upon binding (Figure 3.7). Consequently, de novo designs made to incorporate this structure must similarly pay the energetic cost of forcing the binding region into the disfavored conformation. This difficulty became apparent when attempting to forward-fold the molecule, which has been shown in previous studies to be an excellent predictor of design quality(i.e. predict the designed structure using only sequence data)[7]. To mitigate this difficulty, interfaces in which the binding peptide shows strong structural preference for the bound state can be utilized in future studies.

High-resolution structural validation of binding proteins was not achieved in this study and in one case a crystal structure of a design in the unbound state suggests the molecule is capable of adopting a tertiary structure that differs significantly from the design model. Additional experiments to elucidate the structural features of this design must be conducted in order to gauge the accuracy of the designed molecule. One strategy for continued development of this designed binder is to use multi-state design to introduce mutations that favor the designed conformation and disfavor the conformation observed in the experimental structure. Specific mutations that eliminate sidechain-backbone hydrogen bonds in the undesired state have been identified and further experimental analysis is ongoing. Another encouraging result from these studies was the agreement between in vitro purified proteins and yeast-displayed designs, suggesting that future workflows can involve directed evolution to further improve binding affinities or increase stability of the designed protein. In this study, all designed proteins

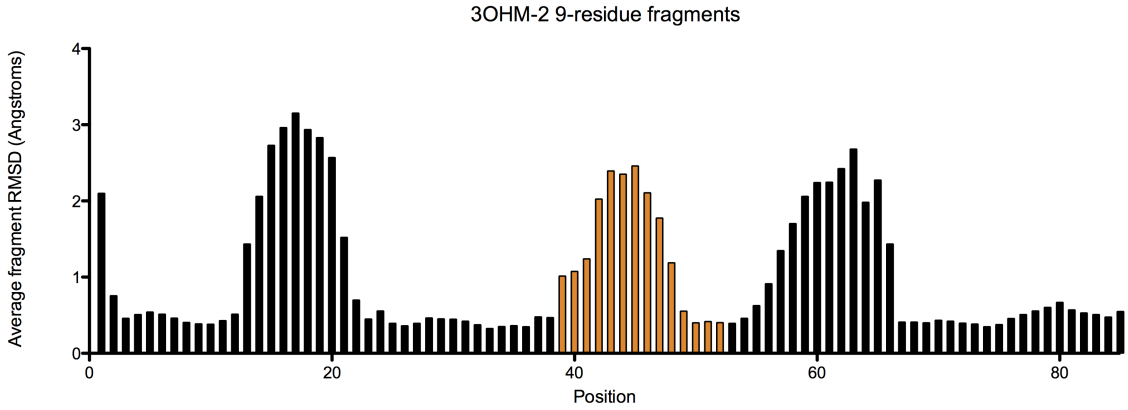

Figure 3.7: Fragment analysis of PLC-β3 binding region. The sequence of 3OHM-2 was used to select 200 9-residue fragments for each 9-residue window in the structure. The average RMSD between the selected fragments and design model is shown for each window. The region of the structure taken from PLC-β3, shown in orange, is a region in which there is especially high RMSD, indicating a poor sequence/structure agreement. Other areas of high RMSD correspond to loop regions, for which there is often poor agreement (Chapter 2)

were entirely α-helical, similar to previously reported designs from the SEWING method. Further improvements to the computational framework should allow extension of this method to alternative forms of secondary structure, and thus expand the generality of this strategy for interface design.

3.5 Command Lines and Input Files

3.5.1 Inputs for the Construction of Protein Backbones from a Starting Interface Peptide

RosettaScript

```
<ROSETTASCRIPTS>
<SCOREFXNS>
</SCOREFXNS>
<FILTERS>
</FILTERS>
    <MOVERS>
        <AppendAssemblyMover
            name=aam
cycles=10000
            add_probability=0.05
            delete_probability=0.005
            switch_probability=0.89
            max_temperature=0.10
            min_temperature=0.05
            base_native_bonus=0.5
            neighbor_cutoff=1
            min_segments=5
            max_segments
        >
            <GlobalRequirements>
                <GlobalLengthRequirement
                    dssp="H"
                    min_length=8
                    max_length=30
                 />
                <GlobalLengthRequirement
                    dssp="L"
                    min_length=1
                    max_length=5
                />
            </GlobalRequirements>
        </AppendAssemblyMover>
    </MOVERS>
    <PROTOCOLS>
        <Add mover_name=aam/>
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

RosettaScript

```
#The PDB for the starting peptide to be used
#for the SEWING append protocol
-s 3OHM_startnode.pdb

#The desired binding partner, aligned with the starting
#peptide in the designed binding orientation
-sewing:partner_pdb 3OHM_partner.pdb

#Sequence positions that must be kept during the
#structural chimerization process
-sewing:keep_model_residues 6 10 13 14 17 32
```

REFERENCES

1. Liu, J. K. H. (2014) The history of monoclonal antibody development – progress, remaining challenges and future innovations. Ann. West. Med. Surg. **3**, 113–116

2. Nissim, A. and Chernajovsky, Y. (2008) Historical development of monoclonal antibody therapeutics. In Therapeutic Antibodies, Handbook of Experimental Pharmacology, 3–18. Springer Berlin Heidelberg

3. Stranges, P. B., Machius, M., Miley, M. J., Tripathy, A., and Kuhlman, B. (2011) Computational design of a symmetric homodimer using β-strand assembly. Proc. Natl. Acad. Sci. U. S. A. **108**, 1–6

4. Der, B. S., Machius, M., Miley, M. J., Mills, J. L., Szyperski, T., and Kuhlman, B. (2012) Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer. J. Am. Chem. Soc. **134**, 375–385

5. King, N. P., Sheffler, W., Sawaya, M. R., Vollmar, B. S., Sumida, J. P., André, I., Gonen, T., Yeates, T. O., and Baker, D. (2012) Computational design of self-assembling protein nanomaterials with atomic level accuracy. Science **336**, 1171–1174

6. Fleishman, S. J., Whitehead, T. A., and Ekiert, D. C. (2011) Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. Science **332**, 816–821

7. Koga, N., Tatsumi-Koga, R., Liu, G., Xiao, R., Acton, T. B., Montelione, G. T., and Baker, D. (2012) Principles for designing ideal protein structures. Nature **491**, 222–227

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

9. Huang, P.-S., Oberdorfer, G., Xu, C., Pei, X. Y., Nannenga, B. L., Rogers, J. M., DiMaio, F., Gonen, T., Luisi, B., and Baker, D. (2014) High thermodynamic stability of parametrically designed helical bundles. Science **346**, 481–485

10. Harbury, P. B. (1998) High-Resolution protein design with backbone freedom. Science **282**, 1462–1467

11. Kato, Y. S., Yumoto, F., Tanaka, H., Miyakawa, T., Miyauchi, Y., Takeshita, D., Sawano, Y., Ojima, T., Ohtsuki, I., and Tanokura, M. (2013) Structure of the ca(2+)-saturated c-terminal domain of scallop troponin C in complex with a troponin I fragment. Biol. Chem. **394**, 55–68

12. Fire, E., Gullá, S. V., Grant, R. a., and Keating, A. E. (2010) Mcl-1-Bim complexes accommodate surprising point mutations via minor structural changes. Protein Sci. **19**, 507–519

13. Bois, P. R. J., Borgon, R. a., Vonrhein, C., and Izard, T. (2005) Structural dynamics of -actinin – vinculin interactions . Mol. Cell. Biol. **25**, 6112–6122

14. Jacobs, T. M. and Kuhlman, B. (2013) Using anchoring motifs for the computational design of protein-protein interactions. Biochem. Soc. Trans. **41**, 1141–1145

15. Waldo, G. L., Ricks, T. K., Hicks, S. N., Cheever, M. L., Kawano, T., Tsuboi, K., Wang, X., Montell, C., Kozasa, T., Sondek, J., and Harden, T. K. (2010) Kinetic scaffolding mediated by a phospholipase c-beta and gq signaling complex. Science **330**, 974–980

16. Lomonosova, E. and Chinnadurai, G. (2008) BH3-only proteins in apoptosis and beyond: an overview. Oncogene **27 Suppl 1**, S2–19

17. Letai, A., Bassik, M. C., Walensky, L. D., Sorcinelli, M. D., Weiler, S., and Korsmeyer, S. J. (2002) Distinct BH3 domains either sensitize or activate mitochondrial apoptosis, serving as prototype cancer therapeutics. Cancer Cell **2**, 183–192

18. Chen, T. S., Palacios, H., and Keating, A. E. (2013) Structure-based redesign of the binding specificity of anti-apoptotic Bcl-x(L). J. Mol. Biol. **425**, 171–185

19. Lyon, A. M., Dutta, S., Boguth, C. A., Skiniotis, G., and Tesmer, J. J. G. (2013) Full-length gαq–phospholipase c-β3 structure reveals interfaces of the c-terminal coiled-coil domain. Nat. Struct. Mol. Biol. **20**, 355–362

20. Van Raamsdonk, C. D., Bezrookove, V., Green, G., Bauer, J., Gaugler, L., O'Brien, J. M., Simpson, E. M., Barsh, G. S., and Bastian, B. C. (2009) Frequent somatic mutations of GNAQ in uveal melanoma and blue naevi. Nature **457**, 599–602

21. Sheffler, W. and Baker, D. (2010) RosettaHoles2: a volumetric packing measure for protein structure refinement and validation. Protein Sci. **19**, 1991–1995

22. Gibson, D. G., Young, L., Chuang, R.-Y., Venter, J. C., Hutchison, C. a., and Smith, H. O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. Nat. Methods **6**, 343–345

23. Boder, E. T. and Wittrup, K. D. (1997) Yeast surface display for screening combinatorial polypeptide libraries. Nat. Biotechnol. **15**, 553–557

24. Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H., and Adams, P. D. (2012) Towards automated crystallographic structure refinement with phenix.refine. Acta Crystallogr. D Biol. Crystallogr. **68**, 352–367

25. McCoy, A. J., Grosse-Kunstleve, R. W., Adams, P. D., Winn, M. D., Storoni, L. C., and Read, R. J. (2007) Phaser crystallographic software. J. Appl. Crystallogr. **40**, 658–674

26. Sheldrick, G. M. (2008) A short history of SHELX. Acta Crystallogr. A **64**, 112–122

27. Yumoto, F., Tanaka, H., Nagata, K., Miyauchi, Y., Miyakawa, T., Ojima, T., and Tanokura, M. (2008) Spectroscopic and ITC study of the conformational change upon ca2+-binding in TnC c-lobe and TnI peptide complex from akazara scallop striated muscle. Biochem. Biophys. Res. Commun. **369**, 109–114

Chapter 4

SWIFTLIB: RAPID DEGENERATE-CODON-LIBRARY OPTIMIZATION THROUGH DYNAMIC PROGRAMMING[1]

4.1 Introduction

In vitro evolution couples genetic diversity generation with either a screen or a selection to identify proteins with some desired phenotype. Although creating highly diverse DNA libraries is trivial, the efficient isolation of the sought-after phenotype presents a bottleneck, limiting the number of DNA sequences that can be tested. Techniques for selecting crossover loci in gene shuffling [1, 2, 3], for open-reading-frame selection [4, 5], for screening neutral drift libraries [6], and for screening restricted-alphabet libraries [7, 8, 9] all aim to distill sequence space to a manageable size while maximizing the likelihood that the remaining sequences will yield the desired phenotype.

Degenerate codon (DC) libraries are attractive in that they focus diversity to regions the library designer thinks will be most productive, they can be molded to include as much or as little diversity at particular positions as is necessary, and they are relatively inexpensive to make. Unfortunately, canonical "NNN" diversification (N = A, C, G, or T) can be used at only a small number of positions before exceeding the experimental size limits. Take for example the 10^7 diversity limit imposed by yeast surface display, a rather middle-of-the road limit; larger than the 10^3 limit for 96-well format screening, and smaller than the 10^{13} limit for mRNA display. NNN diversification would exceed a 10^7 diversity limit if used at a mere four positions. NNN is particularly inefficient since it commits 64 DNA sequences to produce

[1]This chapter previously appeared as an article in the Journal of Nucleic Acids Research. The original citation is as follows:
Jacobs, T. M., Yumerefendi, H., Kuhlman, B., & Leaver-Fay, A. (2015). SwiftLib: rapid degenerate-codon-library optimization through dynamic programming. *Nucleic Acids Research, 43*(5), e34.

only 20 distinct amino acid (AA) sequences. "NNK" (K = G or T) is better, since it uses only 32 DNA sequences for the same 20 AAs. There are more tricks to increase the AA:DNA ratio: the "22c trick" [10] uses three separate DCs to get all 20 AAs from only 22 DNA sequences, and the "small intelligent libraries" technique [11] uses four DCs to get all 20 AAs for exactly 20 DNA sequences. Still, using 20 DNA sequences allows full randomization at only five positions before bumping into a 10^7 diversity limit. To randomize more positions requires limiting the AA diversity. The NDT degenerate codon (D=A, G, or T), which codes for 12 AAs using 12 DNA sequences has been suggested as one way to reduce the AA diversity [9]. Multiple-sequence alignments [12] or computational protein design [13] have also been used to suggest which AAs might be worth considering at each position. However, having a set of candidate AAs at each position leaves the library designer with deciding which DCs to use so that diversity is spread out most productively while adhering to the diversity limit. This paper presents an efficient algorithm for making this decision.

Early work in automating DC optimization mostly focused on matching some target AA distribution by creating "spiked" DCs, where the nucleotide ratios are not uniform [14, 15, 16, 17, 18, 19, 20]. These efforts focused on single positions at a time and made no attempt at trading off between positions. Firth, Patrick, and Blackburn created and still host a web server for selecting DCs for a given set of amino acids [21, 22] which is a significant boon to anyone looking to partially automate the process of designing a DC library.

Mena and Daugherty introduced *LibDesign*, the first algorithm we are aware of for whole library optimization [23]. It takes as input a set of sequences for the positions to be randomized, taken either from protein design trajectories or from multiple sequence alignments, and tallies the AA frequencies at each position. Starting with the $(2^4 - 1)^3 = 3375$ possible DCs (2^4 representing the number of bit strings of length 4 to give all combinations of A, C, G, and T; the -1 to throw out the bit string where all four nucleotides are absent), LibDesign chooses small subsets (6 or 7 DCs) for each position by choosing first the smallest DC that gives the

most commonly observed AA (really, just a single codon), then the smallest DC that gives the two most commonly observed AAs, and so on, until it has a DC that covers all the observed AAs. It then uses brute-force enumeration of all codons in the subsets, looking for combinations with high scores (more on this score later) and that stay below a specified diversity limit.

Treynor *et al.* [24] introduced a technique for building DC libraries while considering energies computed by protein design software, treating the library optimization problem ("what DC should be assigned to each position?") as a variant of the rotamer optimization problem ("what rotamer should be assigned to each position?") and using previously developed dead-end elimination theorems to solve it. (Briefly, rotamers, or rotational isomers, represent discrete side-chain conformations that differ from each other only in their dihedral angles [25].) Instead of looking at rotamer-pair energies, their optimization algorithm looks at the average AA pair energies for a given pair of DCs; each AA pair energy is taken simply as the rotamer interaction energy between the two rotamers that interact most favorably with the template structure (and sequence). One obvious drawback of this technique is that it does not consider how the rotamers of two AAs might relax in each other's presence. It also lacks a way to limit the size of the resulting library, and must be run repeatedly (requiring the user to exclude AAs that might be responsible for producing too much diversity) until a library of the right size is produced.

Allen *et al.* [26] followed with another library design technique, CLEARSS. It begins by computing for every position and for every AA-subset size the optimal AA subset as defined by an input set of scored sequences (*e.g.* the 1000 lowest-energy sequences that emerged from a set of protein design trajectories). It then enumerates all assignments of per-position sizes so that the product of those sizes is within a user-specified size range and picks the best assignment. The algorithm's reliance on brute-force enumeration means that it is quite slow. It can be made to run quickly when the number of AAs at each position is either very low or very high so that there is little to enumerate, but it is much slower when an intermediate number of AAs is used. Curiously, the authors designed their algorithm to enumerate based on the number of AA sequences instead of the number of DNA sequences, which represents the actual experimental constraint. They also incorrectly assert that the optimal set of AAs

for a particular position will be no larger than the one that contains all of the AAs that appear in the input designs; this would not be the case should a DC pull in AAs in addition to the desired ones, as DCs frequently do. Based on this assertion, CLEARSS restricts the number of AA subsets it examines and as a result may miss the optimal DC choice in spite of the fact that it uses brute force enumeration.

Parker *et al.* [27] introduced both a dynamic programming (DP) solution and an integer-linear programming (ILP) solution to solve a DC library optimization problem where the library is guided by sequence alignments only and the designer has not even chosen which positions to randomize. They make the very relevant observation that trying to optimize the library so that pairwise information is included (*e.g.* including PHE at residue 10 only if ARG is included at position 20 because PHE10 is only ever seen if ARG20 is also present) makes the library optimization problem NP-Complete; thus the need for an ILP solution to optimize their pairwise quality and novelty metrics. The dynamic programming algorithm they give is quite different from the one presented here; notably, it is unable to enforce a limit on the size of the resulting library, merely a limit on the number of positions which are randomized. Their ILP solution, in contrast, is able to enforce a library size limit. Chen *et al.* [28] extended this result in designing a DC library with ILP where multiple DCs were allowed at a single position, with the restriction that only one position among those that would be covered by the same primer – those that lie in a single *stretch* of DNA – be allowed to use multiple DCs.

The dynamic programming algorithm presented here also allows multiple DCs per position, but removes the restriction that only one position per stretch use multiple DCs. Experimentally, the construction of such a library would require purchasing multiple primers to cover the stretch, where the number of primers needed is the product of the number of degenerate codons used at each position within the stretch. The algorithm allows the user to limit the total number of primers that could be used and, with that limit, determines the optimal way to distribute the use of those primers. Surprisingly, the use of multiple DCs allows the algorithm to come up with very efficient libraries where the degeneracy of the genetic code can all but be avoided so that, at least for the test cases examined here, the AA:DNA ratio approaches 1. The implementation that we have deployed online, which we call *SwiftLib*, offers an accessible alternative

pos	268	269	270	271	272	276	330	331	332
A	3	8	1	81	4	105	10	2	1
C	0	0	0	0	0	0	0	0	0
D	8	5	0	0	0	0	29	7	9
E	23	7	0	0	0	0	21	23	17
F	0	0	0	0	0	0	0	0	0
G	0	0	0	1	0	59	4	1	5
H	1	0	0	0	0	0	0	1	9
I	10	2	98	0	0	0	0	0	1
K	22	0	0	0	0	0	5	4	9
L	17	8	5	0	0	0	0	14	0
M	13	9	5	0	2	0	3	7	4
N	6	4	0	0	0	0	8	19	15
P	0	0	0	0	0	0	0	0	0
Q	42	2	0	0	1	0	6	29	18
R	35	4	0	0	0	0	27	12	21
S	7	44	8	113	43	36	45	34	29
T	6	58	23	4	30	0	42	46	57
V	7	49	60	1	120	0	0	0	5
W	0	0	0	0	0	0	0	1	0
Y	0	0	0	0	0	0	0	0	0

Table 4.1: Rosetta [29, 30, 31] was used to generate two hundred designs for a set of surface residues in a protein-interface-design application using the 1XBI PDB. These are the AA counts for each of the nine designed positions. The designable positions are contained in two stretches, divided by the vertical bar.

to the previously published ILP approaches [27, 28] because it requires no back end. SwiftLib is implemented in only ∼2K lines of JavaScript and runs inside web browsers; the code executes on the user's computer. It is, to our knowledge, the first web server for optimizing degenerate codon libraries. SwiftLib is accessible at `http://rosettadesign.med.unc.edu/SwiftLib`.

4.2 Materials and Methods

4.2.1 DP For One Degenerate Codon

We have formulated the optimization of degenerate codon libraries with a linearly-additive error function that we wish to minimize, subject to the constraint that the library size not exceed a given limit. This error function treats each residue separately, omitting any consideration of residue-pair information that might be present. Because it is linearly additive and each of its errors are integers, its optimization admits a rapid dynamic programming solution.

The input for the problem is a $20 \times n$ table of amino acid counts for n designable residues and a diversity limit, L. The count $C_i(a)$ is the number of times that amino acid $a \in A$ appeared in the set of input sequences at position i, where A is the set of amino acids. We define the error for choosing a particular degenerate codon $d \in \mathbb{D}$ at position i as

$$E_i = \sum_{a \in A} \delta(a \notin p(d)) C_i(a) \tag{4.1}$$

where $p(d)$ is the set of AAs produced by d, and $\delta(x \notin y)$ is the delta function which yeilds a value of one if x is absent from the set y and zero otherwise, and $\mathbb{D}$ is the set of all 3375 degenerate codons. Effectively, this objective function penalizes the exclusion of AAs that were observed in the input sequences, with greater penalties applied to AAs that appeared the most. The error generated by an assignment $D = \{d_1 \dots d_n\}$ to all n designable positions is simply the sum of the individual errors:

$$E(D) = \sum_{i}^{n} E_i(d_i). \tag{4.2}$$

Let $|d|$ represent the number of DNA sequences defined by a particular degenerate codon d, and $|D|$ represent the product $\prod_i^n |d_i|$. Then the optimization problem can be formulated as:

$$\min_{D \in \mathbb{D}^n} \; E(D)$$

subject to the constraint that $|D| \leq L$.

This problem can be inverted so that instead we solve for the smallest library that achieves any particular error level e. If we have a list of these library sizes for all possible errors, then we can simply pick out the smallest error that can be generated with a library beneath the given limit. This optimization strategy mirrors the one that Bellman originally described when inventing dynamic programming [32]. Let s_i^e denote the smallest size $min|d|$ of all degenerate codons that at position i generate a particular error level $E_i(d) = e$, and infinity if there are no degenerate codons that produce that error level. Let S_{i-1}^e denote the size of the smallest sub-library $D_{i-1} = \{d_1 \dots d_{i-1}\}$ defined over the range between the first designable position

up to and including position $i-1$ that generates error $E(D_{i-1}) = e$. S_i^e can be expressed recursively as:

$$S_i^e = \min_{0<e'\leq e} S_{i-1}^{e-e'} \times s_i^{e'} \tag{4.3}$$

with the recursion bottoming out at $S_1^e = s_1^e$. Since the error is integral, we can turn this recursion on its head and build up a table of partial solutions. If the maximum error that can be produced at any position is m, then two $n \times nm$ tables are needed to hold the solution. The dynamic programming algorithm is as follows: For all $1 < i \leq n$ and for all $0 \leq e \leq m \times i$, compute

$$S[i,e] = \min_{0\leq e'\leq e^*} \quad S[i-1, e-e'] \times s_i[e'] \tag{4.4}$$

and

$$T[i,e] = \arg\min_{0\leq e'\leq e^*} \quad S[i-1, e-e'] \times s_i[e']. \tag{4.5}$$

where e' represents the choice of error contributed by position i, e^* is the smaller of e and m, and T represents a traceback table that can be used to reconstruct the optimal degenerate-codon assignment after the smallest error that satisfies the library size limit has been identified. This algorithm populates the two tables in $O(n^2m^2)$ time and the traceback to reconstruct the optimal solution takes $O(n)$ time.

This solution can be trivially extended to place penalties on AAs, including the STOP codon, that the user would prefer to exclude (but not forbid) from the library, as long as those penalties are also integral. It is also possible to forbid or to require AAs by restricting the set of degenerate codons from which to chose; this is more of a preprocessing addition than it is a modification to the DP algorithm.

Table 4.2: The manual solution to the first library design problem, along with the best solution produced by LibDesign [23], and three solutions produced by dynamic programming (DP). The table reports the chosen DCs at each position, the amino acids produced by those DCs, the number of nucleic-acid sequences the DCs prescribe (#NA), the number of unique AAs and AA sequences the DCs produce (#AA), the percentage of codons that produce desired AAs and the percentage of library members that contain only desired AAs (%Des), the error as calculated from equations 4.1 and 4.2, and the LibDesign score[a]. DP solution 1 used one DC per position, solution 2 considered two DCs per position with a limit of 4 primers total, and solution 3 considered three DCs per position with a limit of 15 primers total. The automated solutions were restricted to a diversity limit of 3.2×10^8, the size of the manually designed library. The maximum achievable LibDesign score for this problem is 200 (higher is better). The manual solution took several hours. LibDesign took 18 minutes 51 seconds. The three dynamic-programming solutions took 0.15, 0.28, and 13.57 seconds.

Solution	pos	268	269	270	271	272	276	330	331	332	Totals
Manual	DCs	VDR	RBT	RYY	KCT	RBT	RVT	RVW	VNW	VVW	105[a]
	AAs	EGIKLM QRV	AGIS TV	AITV	AS	AGIS TV	ADGN ST	ADEGKN RST	ADEGHIKL NPQRSTV	ADEGHK NPQRST	
	#NAs	18	6	8	2	6	6	12	24	18	3.2×10^8
	#AAs	9	6	4	2	6	6	9	15	12	2.5×10^7
	%Des	89	83	100	100	67	50	100	75	89	16.5
	Error	29	39	18	6	3	0	9	8	10	122
LibDesign	DCs	VNK	DYG	RYA	KCA	DYG	KSA	VVK	VNK	VNK	75[a]
	AAs	ADEGHIKLM NPQRSTV	ALM STV	AITV	AS	ALM STV	AGS *STOP*	ADEGHKN PQRST	ADEGHIKL MNPQRSTV	ADEGHIKL MNPQRSTV	
	#NAs	24	6	4	2	6	4	18	24	24	2.9×10^8
	#AAs	16	6	4	2	6	3	12	16	16	5.7×10^7
	%Des	83	100	100	100	83	75	83	79	83	28.6
	Error	0	24	18	6	1	0	3	1	0	53
DP Sol. 1	DCs	VNS	DYG	DYA	KCA	DYG	RSC	RVM	VNS	VNS	157[a]
	AAs	ADEGHIKL MNPQRSTV	ALM STV	AIL STV	AS	ALM STV	AG ST	ADEG KNRST	ADEGHIKL MNPQRSTV	ADEGHIKL MNPQRSTV	
	#NAs	24	6	6	2	6	4	12	24	24	2.9×10^8
	#AAs	16	6	4	2	6	4	12	16	16	6.4×10^7
	%Des	83	100	100	100	83	75	100	79	83	34.4
	Error	0	24	5	6	1	0	9	1	0	46
DP Sol. 2	DCs	VNS	DBG, RAM	DYA	KCA	DYG	RSC	RVM	VAM, WBG	VNS	173[a]
	AAs	ADEGHIKL MNPQRSTV	ADEGKL MNRSTVW	AIL STV	AS	ALM STV	AG ST	ADEG KNRST	DEHKLM NQRSTW	ADEGHIKL MNPQRSTV	
	#NAs	24	13	6	2	6	4	12	12	24	2.9×10^8
	#AAs	16	13	6	2	6	4	9	12	16	1.0×10^8
	%Des	83	77	100	100	83	75	100	100	83	33.4
	Error	0	4	5	6	1	0	9	3	0	28
DP Sol. 3	DCs	MKC, RYG, VAM	AKS, RMC SWA	DYA	DCA	DYG	RSC	ADG, RVC, SAA	DBG, VAM	VNS	192[a]
	AAs	ADEHIKLM NQRSTV	ADEILM NQRSTV	AIL STV	AST	ALM STV	AGST	ADEGKM NQRST	ADEGHKLM NQRSTVW	ADEGHIKLM NPQRSTV	
	#NAs	14	12	6	3	6	4	11	15	24	2.9×10^8
	#AAs	14	12	6	3	6	4	11	15	16	1.9×10^8
	%Des	100	100	100	100	83	75	100	93	83	48.6
	Error	0	0	5	2	1	0	0	0	0	8

4.2.2 DP For Multiple Degenerate Codons

The dynamic programming algorithm can also be extended to allow multiple degenerate codons at a single position while constraining the total number of primers that must be purchased to accommodate the extras. Using multiple degenerate codons at a single position allows the exploration of a wider set of AAs while keeping the size of the library down; it is cheaper (in terms of library size) to get the AA set {C,W,Y} using the two degenerate codons TRT (R=A or G) and TGG, with a DNA size of 3 than it is to use the single degenerate codon TRK (K= G or T) with a DNA size of 4, and this economy is especially important

since TRK also pulls in a STOP codon. Indeed, if the library-designer wishes to forbid STOP codons entirely, then the AA set {C,W,Y} can only be designed if multiple degenerate codons are considered. However, it is more expensive (in terms of money) to use multiple degenerate codons because more primers have to be purchased.

We assume that the primer boundaries are defined ahead of time; this allows us to talk about a *stretch* of DNA that contains a set of designable positions. (We do not consider here the more challenging problem of trying to simultaneously optimize the DCs and the stretch boundaries, which would require knowledge of the G/C content of the DNA between the designable positions and the annealing temperature for the PCR reaction used for gene assembly.) For a single stretch, the number of primers that must be purchased is the product of the number of degenerate codons chosen at each of its randomized positions; the cartesian product of the selected degenerate codons must be purchased to cover all combinations.

The user may wish to constrain this problem by defining limits on the number of degenerate codons per position, L_p, on the number of primers to order per stretch, L_s, and on the number of primers to purchase total, L_T. In the analysis that follows, we consider the sensical assignment of values $1 \leq L_p \leq L_s \leq L_T$ only, and also assume that L_T is at least as large as the number of stretches.

The DP solution to this problem is similar to the one given above; it again solves for the smallest library that produces a given error, but also pays attention to the primer counts: at iteration i, DP solves for the smallest library containing all positions up to and including position i given that it produces an error level e, using j primers total, and using k primers to cover i's stretch. Now, the number of extra primers required if j' degenerate codons are used at position i depends on how many codons have already been used at all the other positions on the same stretch. If k' represents the product of the number of degenerate codons at all positions less than i that are on the same stretch as i, then using j' degenerate codons at position i means using $j' \times k'$ primers for that stretch; if j'' represents the number of primers used total for all positions less than i, then using j' degenerate codons at position i will have one of two possible effects on the total number of primers demanded, depending on whether or not position i is the first position in a stretch: if i is the first position in a stretch, it will

mean using $j'' + j'$ primers total, and if i is not the first position in a stretch, it will mean using $j'' + \frac{j'-1}{j'}k'$ primers total.

The DP solution for this problem is given by the following two equations:

$$S[i,j,j',e] = \min_{1 \le k' \le L_s} \min_{0 \le e' \le e^*} S[i-1, j-j', k', e-e'] \times s_i[j', e'] \tag{4.6}$$

if position i is the first randomized position for a stretch, and

$$S[i,j,k,e] = \min_{1 \le j' \le j^* \mid k/j' \in \mathbb{I}} \min_{0 \le e' \le e^*} S[i-1, j-\frac{j'-1}{j'}k, \frac{k}{j'}, e-e'] \times s_i[j', e'] \tag{4.7}$$

if position i is not the first randomized position for a stretch. where j^* is the smaller of j and L_p. The table s_i holds the smallest-library sizes for position i for each error value between 0 and m, and for each number of degenerate codons between 1 and L_p. The equations for the traceback table are similarly constructed.

This dynamic programming algorithm runs in $O(n^2m^2L_s^2L_T)$ time plus an initial expense of computing the s_i tables, which requires $O(n3375^{L_p})$ time to consider all combinations degenerate codons. This initial expense can be shaved by first computing the subset of degenerate codons that produces the smallest error for each distinct coverage of AAs that contribute to the error (either by their absence or their presence) and then enumerating combinations of degenerate codons from this subset. The speedup offered by this technique decreases as more and more AAs contribute to the error. In our test cases, the sizes of these subsets are in the range of 100 to 400, making the initial expense of populating the s_i tables negligible. This algorithm requires $O(n^2mL_sL_T)$ memory to store the S and T tables.

In both dynamic programming solutions, a very simple speedup can be obtained if one is only interested in knowing about the minimum-error library or about the N lowest-error libraries with sizes less than the given diversity limit: if the total library error is iterated over

in the outermost loop (instead of the position), one may stop as soon as the first library of size less than L is found, or as soon as the first N libraries of size less than L are found. This output-sensitive algorithm runs in $O(n(k+1)^2 L_s^2 L_T)$ time, where k is the total error for the worst library sought (*i.e.* the smallest total error if only one library is sought). For this reason, SwiftLib runs fastest when it is able to find a low-error solution. Furthermore, the use of a sparse array can reduce the memory overhead significantly; since SwifLib is implemented in JavaScript, the JavaScript interpreter has this option.

4.3 Results

4.3.1 Library Designs

We chose two library design test cases where manual solutions had been created before our DP algorithm was implemented. For each test case, we used DP to design three degenerate codon libraries with three different library design goals: 1) allowing only one DC per position, 2) allowing at most two DCs per position and allowing twice as many primers as there are stretches, and 3) allowing at most three DCs per position and allowing a larger number of primers. The libraries were assessed only on how well they capture the sought-after sequence diversity; the DP libraries have not been synthesized or screened.

We compared the results from DP to the previously published LibDesign algorithm [23]. This algorithm is a natural comparison for ours as its inputs and goals are nearly identical. LibDesign takes a set of input sequences, selects a small set of degenerate codons for each position based on the AA counts from the input sequences, and then enumerates all combinations of degenerate codons. The LibDesign score for a library is the number of sequences in the input set that are generated by the library: every amino acid in a particular sequence has to be encoded in the library in order for that sequence to contribute to the score. LibDesign attempts to maximize its score (in contrast to SwiftLib which minimizes its error metric). This score is at the furthest end of the spectrum from our error metric in complexity: our error term treats each position independently, Parker *et al.*'s ILP alogirthm is more complex

Table 4.3: Rosetta was used to generate 1000 sequences on residues near the Jα helix of the *A. sativa* LOV2 domain (PDBid 2V0U). The table below gives the AA counts for each position. The library was constructed using five separate DNA stretches, divided by vertical bars, which contained 12 randomized positions. The counts for the native AAs are shown in bold.

	413	475	477	479	493	495	514	520	528	529	531	532
A	990	3	1	21	323	1	0	0	934	115	0	992
C	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0
E	0	**0**	0	0	0	0	0	0	0	0	0	0
F	1	0	460	0	0	964	3	0	0	0	0	0
G	9	0	390	349	351	23	45	0	**17**	5	0	0
H	0	0	34	0	257	**12**	2	0	0	498	0	0
I	0	0	1	630	2	0	86	0	0	0	0	**1**
K	**0**	559	0	0	0	0	0	0	0	0	1	0
L	0	0	0	0	**11**	0	**10**	349	0	110	**14**	0
M	0	0	5	0	10	0	16	8	49	197	4	4
N	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	**0**	0	0	0	0	0	0	0	0
R	0	438	0	0	0	0	0	0	0	74	91	0
S	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	**0**	0	0	0	0	0	0	0	0	0
V	0	0	0	0	46	0	0	**643**	0	**0**	0	3
W	0	0	109	0	0	0	640	0	0	1	890	0
Y	0	0	0	0	0	0	198	0	0	0	0	0

since it considers residue-pair information, but LibDesign captures higher-order inter-residue dependencies still.

The AA counts for Problem 1 are given in Table 4.1. These counts came from 200 protein design trajectories at nine positions in a particular protein. The designed libraries are given in Table 4.2. The manual solution was produced by a novice library designer who focused only on the most-commonly observed AAs at each of the designable positions and sought out degenerate codons that were capable of covering these AAs. For this reason, the library contains much higher per-position error than the dynamic programming libraries. For this problem, the third dynamic programming solution was constrained to use 15 or fewer primers.

The dynamic programming solutions to Problem 1 have much lower errors than the manual solution. DP solution 1, which like the manual solution uses only a single DC per position, achieves a considerably lower error. It also produces a library with twice as much AA diversity (the number of unique protein sequences), while actually decreasing the DNA diversity. DP solution 2, using at most four primers and two degenerate codons, decreases the error still further from DP solution 1, choosing to use two degenerate codons at positions 269 and 331, where both the manual solution and DP solution 1 had the highest per-position error and a low AA:DNA ratio. It also delivers four times the AA diversity as the manual solution. DP solution 3 chooses to divide its 15 primers into one group of 9, using three DCs for both

Table 4.4: Solutions. The manual library was created by an expert library designer using degenerate codons to cover the AAs selected by Rosetta, along with the ISOR technique [33] for ensuring that the native AAs (shown in bold) were present; the library's size and error are computed as if an extra codon for the native AA (given in parentheses) were present at each position, though the ISOR technique does not guarantee that adjacent randomized positions will achieve full diversity. For the automated solutions, the library size was restricted to 10^9 and the wild type AA was required to be covered by the degenerate codons themselves. The maximum achievable LibDesign score for this problem is 1000 (higher is better). The three dynamic programming solutions represent allowing one degenerate codon per position, allowing two degenerate codons per position restricted to 10 primers total, and allowing three degenerate codons per position restricted to 25 primers total (though only 24 were chosen). See Table 4.2 for a description of row labels. The manual solution took hours to generate, LibDesign took 52 minutes 37 seconds, and the three dynamic programming solutions took 1.40, 0.23, and 0.89 seconds.

Solution	Pos	413	475	477	479	493	495	514	520	528	529	531	532	Totals
Manual	DCs	GCC, (AAA)	ARA, (GAG)	KGG, YWC, (ACG)	RBA, (CAG)	SNC, (TTG)	KKC, (CAT)	DDC, WKG, (CTG)	CTG, (GTG)	RYG, (GGG)	CDT, RYG, (GTG)	WGG, (CTG)	GCC, (ATT)	955[a]
	AAs	A**K**	**E**KR	FGHL **T**WY	AGI**Q** RTV	ADGH **L**PRV	CFG**H**V	CDFGILM NRSVWY	L**V**	A**G**M TV	AHLMR **T**V	**L**RW	A**I**	
	#NAs	2	3	7	7	9	5	14	2	5	8	3	2	8.9×10^7
	#AAs	2	3	7	7	9	5	13	2	5	7	3	2	6.4×10^7
	%Des	100	100	71	57	67	60	57	100	60	86	100	100	4.9
	Error	10	3	7	0	12	1	2	8	0	6	5	7	61
LibDesign	DCs	RMA	RRA	NNK	VNA	SNC	YWC	TDK	KTA	GSA	VNK	TKG	RYA	81[a]
	AAs	AEKT	EGKR	ACDEFGHIKLM NPQRSTVWY *STOP*	AEGIKL PQRTV	ADGHL PRV	FHLY	CFLWY *STOP*	LV	AG	ADEGHIK LMNPQR STV	LW	AITV	
	#NAs	4	4	32	12	8	4	6	2	2	24	2	4	9.1×10^8
	#AAs	4	4	21	11	8	4	6	2	2	16	2	4	3.6×10^8
	%Des	50	75	34	33	63	50	67	100	100	54	100	75	0.4
	Error	10	3	0	0	12	24	149	8	49	1	96	4	356
DP Sol. 1	DCs	RMA	RRA	DBS	VDA	SNC	YWC	WDS	STA	GSA	VNS	TKG	RYA	795[a]
	AAs	AEKT	EGKR	ACFGIL MRSTVW	EGIK LQRV	ADGH LPRV	FHLY	CFIKLM NRSWY *STOP*	LV	AG	ADEGHIK LMNPQR STV	LW	AITV	
	#NAs	4	4	18	9	8	4	12	2	2	24	2	4	7.6×10^8
	#AAs	4	4	12	8	8	4	12	2	2	16	2	4	3.0×10^8
	%Des	25	50	28	25	63	50	67	100	100	46	100	75	0.4
	Error	10	3	34	21	12	24	47	8	49	1	96	4	309
DP Sol. 2	DCs	AAA, GSA	RRA	DSG YWC	VNA	SNC	CAC KKC	DKG, WWC	STA	RBG	CWC, RBG	WKG	RYA	966[a]
	AAs	AGK	EGKR	AFGHL RSTWY	AEGIKL PQRTV	ADGHL PRV	CFG HV	FGILMN RVWY	LV	AGM RTV	AGHLM RTV	LM RW	AITV	
	#NAs	3	4	10	12	8	5	10	2	6	8	4	4	8.9×10^8
	#AAs	3	4	10	11	8	5	10	2	6	8	4	4	8.1×10^8
	%Des	100	75	60	33	63	60	70	100	50	87	100	75	1.3
	Error	1	3	6	0	12	1	2	8	0	1	1	4	39
DP Sol. 3	DCs	AAA, GSA, TTC	RVA	DBG, HWC	ATA, CAA, GSA	ATR, SNC	GSA YWC	ATR, KGG, YWC	VTG	ATG, GSA	DKG, SMC	WKG	ATR, GYA	999[a]
	AAs	AFGK	AEG KRT	AFGHILMN RSTVWY	AGIQ	ADGHI LMPRV	AFG HLY	FGHIL MWY	LMV	AGM	ADGHLM PRVW	LM RW	AI MV	
	#NAs	4	6	15	4	10	6	8	3	3	10	4	4	1.0×10^9
	#AAs	4	6	14	4	10	6	8	3	3	10	4	4	9.3×10^8
	%Des	100	67	53	100	70	67	100	100	100	80	100	100	13.3
	Error	0	0	0	0	0	0	0	0	0	0	1	0	1

positions 268 and 269, and one group of 6, using three DCs for position 330 and two DCs for position 331. As a result, its error drops to 8 while its AA:DNA ratio increases to 2:3; at only a single position does it use more than one codon for an amino acid. It delivers nearly ten times the AA diversity as the manual solution. The fraction of amino acid sequences also increases as the number of degenerate codons increases so that nearly half of all sequences in DP solution 3 contain only desired amino acids, in contrast to one sixth in the manually constructed library. The dynamic programming algorithm was much faster than the manual

process; the manual solution took hours to generate but DP solutions 1, 2, and 3 took 0.15, 0.28, and 13.57 seconds.

The AA counts for Problem 2 are given in Table 4.3. These counts came from 1000 protein design trajectories at twelve positions in a particular protein. The designed libraries are given in Table 4.4. The manual solution came from an expert library designer and took advantage of two techniques: the ISOR technique for ensuring that the native AA is present in the library [33], and the use of multiple degenerate codons. Briefly, ISOR ensures that the WT amino acids are present in the library by creating extra "primers" by partial digestion of the WT gene. The designer of this library made two mistakes. First, the library's intended size was 10^9, but it came out more than ten times smaller at just under 10^8. Second, the library designer meant to use a codon for leucine ("CTG") at position 520, but mistakenly chose (and subsequently ordered) the codon "CAG", which codes for glutamine, instead. The error of 61 reported in Table 4.3 for the manual library was calculated as if "CTG" had been ordered.

In the automated solutions for this design problem, the libraries were forced to include the wild-type AA at each position, mimicking the use of the ISOR technique in the manually-constructed library. DP solutions 1 and 2 were similarly constrained as in Problem 1. DP solution 3 was constrained to use at most 25 primers. The automated solutions were limited to a 10^9 library size, since that was the intended size for this library, even though the manual solution was an order of magnitude smaller. DP solutions 4-6, constructed with a diversity of 10^8, are given in Table S1.

Compared against the manual solution, the dynamic programming solutions to Problem 2 were mixed. The manual solution itself included the use of multiple degenerate codons at some positions and took advantage of the ISOR technique, which likely explains why it achieved a lower error than DP solution 1, which used only a single degenerate codon per position. DP solution 2, however, does produce a lower error than the manual solution, even though it does not use the ISOR technique. It chooses to place two degenerate codons at the same positions that the manual solution does, and two more in the other two stretches that the manual-library's designer chose not to explore. As a result, it achieves a slightly larger AA diversity, though, at the cost of almost 10x higher DNA diversity. Now, we have calculated the AA

Table 4.5: (in seconds) for the three variations on the two library design problems. The running times for the ILP solutions include only the amount of time for the *glpsol* solver to run, and do not include the preprocessing step of examining degenerate codon combinations; the running times for the DP solutions include both preprocessing and optimization steps. Running times were measured on a 2013 MacBookPro with a 2.3 GHz i7 processor and 4 GB of RAM. DP was run within Chrome version 34.0.1847.13.

	Prob 1 1 DC	Prob 1 2 DCs	Prob 1 3 DCs	Prob 2 1 DC	Prob 2 2 DCs	Prob 2 3 DCs
DP	0.15	0.28	13.57	1.40	0.23	0.89
ILP	0.46	661.08	1607.67	0.09	46.86	4.09

diversity for the manual library by treating the wild type AA – which the ISOR technique adds – as if it were encoded by a second (or third) degenerate codon at each position. This overestimates the manual library's diversity, since if one position receives its wild-type AA, then its adjacent positions will also receive their wild-type AAs. In spite of this overestimation, DP solution 2 exceeds the manual library's AA diversity.

DP solution 3 reduces the error still further. It achieves an error of 1, missing only the lysine at position 531 that appeared once in the Rosetta simulations. It also achieves a 14:15 AA:DNA ratio, coding for a single AA at position 377 with two codons and all others at all positions with only a single codon. DP solution 3 included several AAs that were not observed in the Rosetta designs; we cannot conclude, then, that the use of multiple DCs is perfectly molding the DNA to the desired set of AAs. The error function offers no penalty for including unobserved AAs (simply no bonus for doing so), so the DP algorithm's inclusion of extra unobserved AAs in attempting to cover all the observed AAs is expected. However, the high AA:DNA ratio is not expected since this ratio is not directly optimized. The unobserved AAs can be squeezed out if the diversity limit is dropped, at the expense of increasing the resulting error. A seventh DP solution (Table S1) which was constrained in the same ways that DP solution 3 was, except that its diversity limit was 10^7, defines a library with an error of 69 – similar to that produced by the manual solution – and which contains only six AAs that were unobserved in the original designs. DP solution 7 achieves a perfect 1:1 AA:DNA ratio.

For problem 2, the DP solutions were completed much more quickly than the manual one. The manual solution took many hours to construct, whereas DP solutions 1, 2, and 3 took 1.40, 0.23, and 0.89 seconds.

In both problems, the DP solutions achieved lower error levels than the LibDesign solutions, which is not unexpected given that LibDesign was not optimizing our error metric. What was unexpected, however, was that the dynamic programming solutions as well as the manual solutions achieved a better (higher) LibDesign score than LibDesign did. Since LibDesign relies on exhaustive enumeration of the codons it chose at the beginning, this result suggests that the original choices for which codons to consider were sub-optimal. The ability of the dynamic-programming libraries to achieve such high LibDesign scores (the highest possible score for Problem 1 was 200, the highest possible score for Problem 2 was 1000; the third DP solutions achieved scores of 192 and 999) is the result of having achieved such low errors; almost all of the AAs that came out of the computational designs were covered, meaning almost all of the input sequences were fully covered. It is likely that DP would have produced much worse LibDesign scores if its best solution had a much higher error. LibDesign's reliance on brute force enumeration meant that it took much longer to produce its libraries than the DP algorithm; at Problem 1, it took 18 minutes and 51 seconds, at Problem 2, it took 52 minutes 37 seconds.

The Supplemental Material presents the results from an additional 798 library design problems from 38 proteins where we selected subsets of residues of varying sizes and redesigned those residues with Rosetta. We report the error level (Table S3) and running time (Table S4) as we varied a) the diversity limit, b) the number of degenerate codons per position, and c) the primer limit. These results show that if one started from a degenerate codon library using only a single degenerate codon per position and had the choice between either making the library 10 times larger or allowing two degenerate codons per position and 10 primers total, that the error reduction is much greater for the latter rather than the former (23% reduction vs. 75% reduction; P-value < 0.00001, 1-tailed T test). The median running time for these 798 jobs was 0.61 seconds, with the longest job taking 104 seconds.

4.3.2 Comparison with ILP

This section concludes with a comparison between our DP algorithm and the previously-published integer-linear-programming formulations of the problem. Integer linear programming (ILP), sometimes called mixed integer programming, is an NP-Complete problem [34] where many problem instances admit a rapid solution. Indeed, the frequency with which ILP problems present themselves in process optimization has led to a wide availability of commercial ILP solvers, including the free GLPK solver, used in Chen *et.al* [28] and which we have used here. When an ILP solver produces a solution, which is not guaranteed, it produces the exact solution, and so it is a natural comparison point for our DP algorithm which also produces an exact solution.

In the Supplemental Materials, we give a reformulation of the previously published ILP solutions for a single degenerate codon per position that optimizes our error metric, and a novel ILP formulation to allow multiple degenerate codons per position. The multiple DC formulation requires computing the sum of a product of variables, which is challenging to do with ILP, and so it runs slowly.

There are four principle advantages that DP has over ILP: DP runs in polynomial time whereas ILP is NP-complete, DP is faster, a single DP execution can provide more than a single solution, and DP is more easily implemented than ILP. To show that DP is faster, we compared ILPs running time against DP's running time for the two problems presented above and in the Supplemental Materials for 268 additional jobs. The running times for problems 1 and 2 are given in Table 4.5. In five of the six cases, DP completed in less time than ILP. Library design jobs where multiple degenerate codons were allowed degraded the ILP running time substantially. For 192 of the 220 jobs using multiple degenerate codons from the Supplemental Materials, DP completed in less time than ILP. In 107 of those cases, no ILP solution was obtained in 1000 minutes at which point we killed the ILP solver. Assuming these jobs had finished in exactly 1000 minutes, the median speedup for DP over ILP when using multiple degenerate codons was 5752.

A single execution of DP produces more than just a single library. The DP algorithm computes the size of the smallest library capable of achieving every error level starting at 0

and building upwards until it finds a library that is beneath the given size limit. As a result, it can report each of those library sizes as a function of error level to give the user insight into how the error changes as a function of library size. Without having to re-run the algorithm, they can see what would happen to the error as they increased the size of their library. Furthermore, DP can provide the codons that comprise the larger libraries if the user is interested in them in $O(n)$ time. SwiftLib presents users with a scatter plot of the Pareto-optimal libraries – scored by library size and error level – with the log-library size given on the x-axis, and the error level on the y-axis. They can then click on any of the dots in the plot to display the DCs that make up that library.

The fourth advantage, that DP has a simpler implementation than ILP, means that we are able to create a website that has no back end. This site simply serves up the code to run DP (which can be downloaded from the website or from github: `https://github.com/aleaverfay/swiftlib_javascript` and the code is executed within the user's web browser; the heavy computation is performed on the user's own computer. In contrast, a website that relied on an ILP solution would have to queue ILP jobs to be performed on a back end server and then delivered results as they completed. Even if ILP were as fast as DP, such a server might accumulate a heavy backlog of work. SwiftLib, needing only to deliver the DP source code, could accommodate many more users before it started to slow down.

The largest drawback of DP in comparison to ILP is in its restricted objective function. With ILP, one is able to formulate more complicated functions to optimize, such as the pairwise quality and novelty metrics presented by Parker *et al.* [27]. It it our opinion, however, that the advantage of being able to consider multiple degenerate codons per position (within a reasonable amount of time) outweighs the weakness in not being able to incorporate pairwise data.

4.4 Discussion

Directed evolution is rapidly becoming a standard complement to computational protein design [13, 24, 35, 36, 37, 26, 38, 39, 40, 41, 42, 28, 43, 44, 45, 46]. It allows protein designers to test vastly more designs than could individually be expressed, purified, and assayed. As

a result, shortcomings in the current generation of energy functions and sampling protocols can be overcome, and useful proteins that vary only slightly in their sequences from a starting design can be found. Since there are so many ways for a design to fail, some of the best insight can come from finding a successful design and contrasting it against the sequences that the design score function most favors. Directed evolution offers a means to find such successful designs.

Degenerate codon libraries are a natural complement to computational protein design as they allow a designer to focus diversity to the active site or interface positions in ways that error-prone PCR, for example, could not. However, degenerate codon libraries are quite difficult to optimize by hand; there are thousands of possible DCs and so finding the best one that also offers a reasonable compromise with the other positions being optimized while the whole library stays beneath a given diversity limit is a daunting task at best. The task becomes decidedly harder once the possibility of choosing multiple degenerate codons at a single position is introduced. Moreover, manually designing DC libraries is highly error prone, as observed in this study. It is no surprise that automated construction of DC libraries has been the focus of several prior studies [23, 24, 26, 27, 28]. SwiftLib offers a rapid solution to the design of degenerate codon libraries. Because it expects as inputs a set of AA counts for each position to be randomized, which are readily derived from the outputs of protein design simulations, it should fit naturally into the computational-protein designer's workflow.

SwiftLib was able to find libraries that covered nearly every AA present in the input designs when it was allowed to consider multiple DCs at each position. SwiftLib could make it much easier to screen libraries that cover the potentially-useful AAs as suggested by computational design or multiple-sequence alignments by reducing the number of sequences that have to be tested to achieve full coverage.

Table S4.1: See Tables 2 and 4 for a description of the rows. Solution 4 took 0.17 s. Solution 5 took 0.33 s. Solution 6 took 0.99 s. Solution 7 took 1.7s.

Solution	Pos	413	475	477	479	493	495	514	520	528	529	531	532	Totals
Manual	DCs	GCC, (AAA)	ARA, (GAG)	KGG, YWC, (ACC)	RBA, (CAG)	SNC, (TTG)	KKC, (CAT)	DDC, WKG, (CTG)	CTG, (GTG)	RYG, (GGG)	CDT, RYG, (GTG)	WGG, (CTG)	GCC, (ATT)	955[a]
	AAs	A**K**	**E**KR	FGHL **T**WY	AGI**Q** RTV	ADGH **L**PRV	CFG**H**V	CDFGI**L**M NRSVWY	**L**V	A**G**M TV	AHLMR **T**V	**L**RW	A**I**	
	#NAs	2	3	7	7	9	5	14	2	5	8	3	2	8.9×10^7
	#AAs	2	3	7	7	9	5	13	2	5	7	3	2	6.4×10^7
	%Des	100	100	71	57	67	60	57	100	60	86	100	100	4.9
	Error	10	3	7	0	12	1	2	8	0	6	5	7	61
DP Sol. 4	DCs	RMA	RRA	DBS	VDA	SNC	YWC	TDS	STA	GSA	SHC	TKG	RYA	581[a]
	AAs	AEKT	EGKR	ACFGIL MRSTVW	EGIK LQRV	ADGH LPRV	FHLY	CFLWY *STOP*	LV	AG	ADHL PV	LW	AITV	
	#NAs	4	4	18	9	8	4	6	2	2	6	2	4	9.6×10^7
	#AAs	4	4	12	8	8	4	6	2	2	6	2	4	5.6×10^7
	%Des	50	75	56	33	63	50	67	100	100	67	100	75	0.7
	Error	10	3	34	21	12	24	149	8	49	277	96	4	687
DP Sol. 5	DCs	AAA, GCA	RRA	DSG YWC	VNA	SNC	CAC TTC	DKG, WWC	STA	GSA	CWC, RBG	WKG	RYA	893[a]
	AAs	AK	EGKR	AFGHL RSTWY	AEGIKL PQRTV	ADGHL PRV	FH	FGHILM NRVWY	LV	AG	AGHLM RTV	LM RW	AITV	
	#NAs	2	4	10	12	8	2	12	2	2	8	4	4	9.4×10^7
	#AAs	2	4	10	11	8	2	11	2	2	8	4	4	7.9×10^7
	%Des	100	75	60	33	63	100	75	100	100	88	100	75	4.6
	Error	10	3	6	0	12	24	0	8	49	1	1	4	118
DP Sol. 6	DCs	AAA, GSA	RRA	AYG, KGG, YWC	GSA, MWA	CWC, RBG	CAC, GGA, TTC	ATR, KGG, YWC	VTG	ATG, GSA	CWC, RBG	WKG	ATR, GCA	987[a]
	AAs	AGK	EG KR	FGHL MTWY	AGIKLQ	AGHL MRTV	FGH	FGHIL MWY	LMV	AGM	AGHLM RTV	LM RW	AI M	
	#NAs	3	4	8	6	8	3	8	3	3	8	4	3	9.6×10^7
	#AAs	3	4	8	6	8	3	8	3	3	8	4	3	9.6×10^7
	%Des	100	75	75	67	75	100	100	100	100	88	100	100	24.6
	Error	1	3	2	0	2	1	0	0	0	1	1	3	14
DP Sol. 7	DCs	AAA, GCA	RRA	DSG, YWC	ATA, CAA, GSA	CAC, GBA, MTG	CAC, TTC	KGG, TAC, WTS	STA	ATG, GSA	CDC, RYG	WKG	ATA, GCA	944[a]
	AAs	AK	EKR	AFGHL RSTWY	AGIQ	AGHLMV	FH	FGHLMWY	LV	AGM	AHLMRTV	LMRW	AI	
	#NAs	2	4	10	4	6	2	7	2	3	7	4	2	9.0×10^6
	#AAs	2	4	10	4	6	2	7	2	3	7	4	2	9.0×10^6
	%Des	100	75	60	100	100	100	100	100	100	86	100	100	38.6
	Error	10	3	6	0	2	24	2	8	0	6	1	7	69

4.5 Supplemental Materials

4.5.1 Additional libraries for problem 2

In addition to the set of libraries presented for Problem 2 in the main text, Table S4.1 presents four more dynamic programming solutions. These solutions were constructed with smaller library-size limits. Solutions 4-6, which mirror solutions 1-3 in the number of DCs considered per position and in the primer limits, were limited to a size of 10^8 – the size of the manual solution. Solution 7 was limited to a size of 10^7. These solutions all result in higher error levels than the solutions generated with a size limit of 10^9; however, they allocate more of their diversity toward the desired amino acids, with 39% of the library 7 sequences containing only desired amino acids – 10-fold higher than for the manual solution. Solution 6 has a much lower error than the manual solution, and Solutions 6 and 7 both achieve a perfect 1:1 AA:DNA ratio.

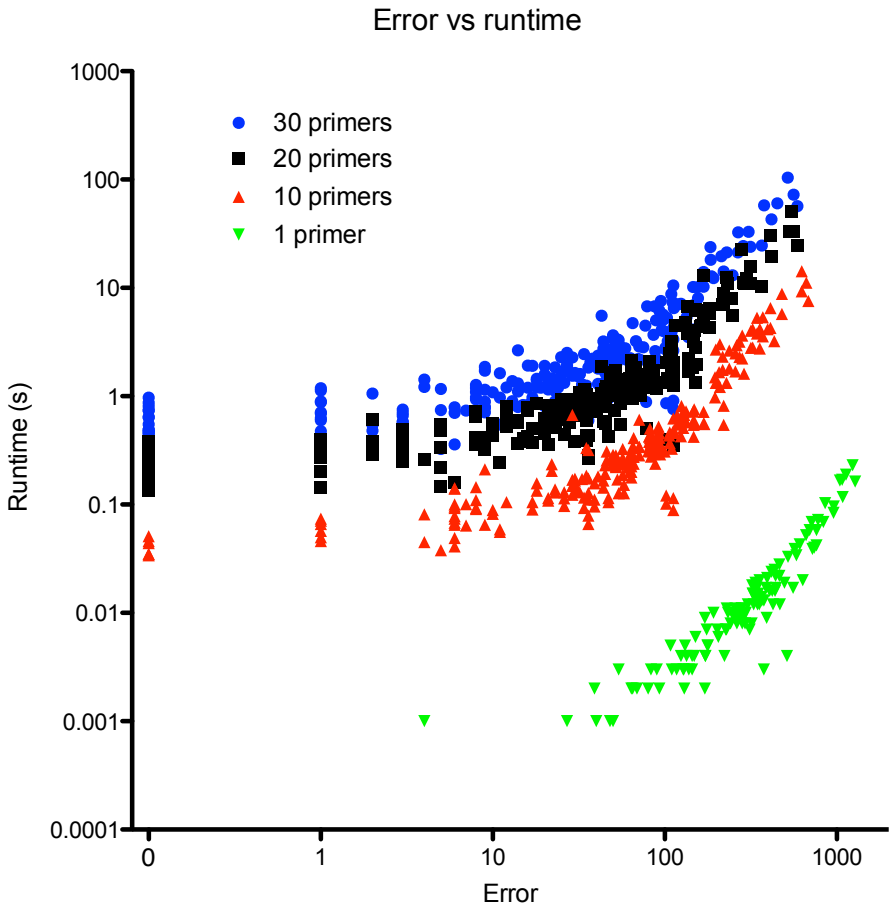

Figure S4.1: DP's running time as a function of the amount of error in 798 library design problems. The output-sensitive DP algorithm has an $O(n(k+1)^2 L_s^2 L_T)$ running time complexity, which is evidenced in the quadratic shape of the running time as a function of the error the problems produced. In these problems L_s was set to L_T; no other limit besides L_T was placed on L_s. The running time curves shift upwards as the limit on the number of primers increase. (Error values of 0 are plotted at 0.1 and that x-axis value relabeled.)

4.5.2 ILP with One Degenerate Codon

ILP is a problem of optimizing a linear objective function of v variables subject to a set of constraints formulated as linear inequalities. An example ILP program would be to maximize $x_1 + 2x_2 - 1.5x_3$ with the constraints $x_3 > x_2$, $x_1 + x_2 + x_3 < 12$, $x1 \geq 0$, $x_2 \geq 0$, and $x_3 \geq 0$ and where $x_1, x_2, x_3 \in \mathbb{I}$. ILP is a more restricted version of linear programming, which allows the variables to be real-valued, and which is often solved rapidly. One of the more common approaches to ILP is to solve the linear program, and then to "round down" to the nearest

integers; the rounding, however, may produce a solution that violates one of the constraints, at which point either the 2^v nearby integer assignments to the variables must be considered or some other complex heuristic must be employed. Our DP solution represents a theoretical improvement over the ILP solution; polynomial time solutions are preferable to NP-complete solutions.

We reformulated the ILP solutions put forward by Parker *et al.* and Chen *et al.* to optimize the error metric given in Equation 2: one formulation considers only a single DC per position, the second (novel) formulation allows multiple DCs per position and multiple positions per stretch to use multiple DCs. Our error metric is in fact very similar to the one used by Chen *et al.*, where they sought a particular set of AAs for each position and minimized the number of AAs that were excluded; this would be equivalent to putting a count of 1 on each desired AA and using our error metric.

To optimize a single DC per position, first introduce for every position i and every achievable error level e at that position a binary-valued variable d_e^i to represent whether the smallest DC which gives error e has been chosen. For each position i, add a constraint

$$1 = \sum_e d_e^i \tag{4.8}$$

to make sure that one and only one DC is chosen per position. Then add a constraint on the library size

$$\log(L) \geq \sum_i \sum_e \log(|d_e^i|) \times d_e^i \tag{4.9}$$

where $|d_e^i|$ represents the size of the degenerate codon that the variable d_e^i corresponds to. The ILP optimization problem becomes

$$\text{Minimize} \quad \sum_i \sum_e e \times d_e^i \tag{4.10}$$

subject to the above linear constraints.

4.5.3 ILP with Multiple Degenerate Codons

The ILP formulation becomes much more difficult to express when allowing multiple DCs at multiple positions per stretch. The challenge stems from needing to constrain the sum of a product. Given that all constraints must be formulated as linear inequalities, it is trivial to constrain a pure sum of variables in ILP, and it is also easy to constrain a pure product of variables by computing a sum of logs; however, constraining a mixture of sums and products is not straight forward. The total number of primers used is the sum of the number used for each stretch, and the number used for each stretch is the product of the number used at each position in that stretch; thus constraining the total number of primers used is challenging.

Let n be the number of designable positions, m be the total number of error levels, and o be the number of degenerate codons to consider at each position. The user will have provided two constraints L on the size of the library, and L_T on the number of primers total that can be purchased.

To begin, add the binary-valued variable (henceforth boolean variable) $d^i_{e,j}$ for each position i, for each number of degenerate codons that could be assigned j, and for each achievable error level e, to represent whether or not the smallest set of exactly j DCs that achieves an error level of e has been chosen at position i. Then for each position add a one-and-only-one constraint

$$1 = \sum_e \sum_j d^i_{e,j} \tag{4.11}$$

to ensure that only a single set of degenerate codons is chosen at each position.

Boolean AND and OR operations, which are needed to solve this problem, can been represented by the addition of new variables and new constraints. To represent the boolean OR of k boolean variables $\{x_1, x_2, ...x_k\}$ requires adding one new boolean variable $orvar$, adding

one constraint of the form

$$orvar \leq x_1 + x_2 + ... + x_k \tag{4.12}$$

and adding k constraints of the form

$$orvar \geq x_i \tag{4.13}$$

for each $i \leq k$. To represent the boolean AND of k boolean variables requires adding one new boolean variable *andvar*, adding one constraint of the form

$$andvar + k - 1 \geq x_1 + x_2 + ... + x_k \tag{4.14}$$

and adding k constraints of the form

$$andvar \leq x_i \tag{4.15}$$

for each $i \leq k$.

To count the number of DCs used at each position, add $n \times o$ binary variables q_j^i representing whether a set of DCs containing exactly j elements has been chosen for position i and a set of constraints representing the boolean OR of the set of $d_{e,j}^i$ variables.

To count the number of primers used for each stretch, add for each stretch with p positions o^p boolean variables representing all possible combinations of numbers of DCs assigned to each position. For example, consider a single stretch made up of three positions, a, b, and c that could each take on two DCs. Let the six boolean variables, q_1^a, q_2^a, q_1^b, q_2^b, q_1^c, q_2^c represent whether the positions were assigned either 1 or 2 degenerate codons. Then it is possible to express the number of primers needed for this stretch using a set of $2^3 = 8$ additional boolean variables $s_{x,y,z}^{abc}$ each representing the boolean AND of q_x^a, q_y^b, and q_z^c for $x, y, z \in \{1, 2\}$. With the $s_{x,y,z}^{abc}$ variables, the following linear inequality can be used to constrain the total number

Table S4.2: Proteins are labeled by their PDB id; lower-case letters after the PDB id represent which chain from the PDB was used. The amino acid diversity (AA diversity) is calculated as the product over all positions of the number of amino acids that appeared at those positions in the Rosetta designs. Proteins are sorted by the number of designable positions, but it is worth highlighting that the amino acid diversity varies quite widely – some proteins accommodate more amino acids than others and so have higher amino acid diversities as a result.

PDB	Designable positions																	AA diversity
1BKF	52	53	54															2.9e+02
2BOPa	366	368	369															1.4e+03
1AMM	87	88	89	106	128													3.6e+02
3LZM	81	82	83	85	86													4.9e+04
1PLC	47	48	50	51	76													7.1e+04
1JBC	117	118	119	122	185	187												7.2e+05
1RRO	48	51	52	53	54	55												1.3e+06
1CYO	43	44	45	46	47	48												1.6e+06
1CTJ	37	38	41	42	45	46	49											3.5e+05
1KNB	479	482	486	487	488	489	490											6.3e+06
1NIF	171	172	173	174	239	240	241											1.2e+07
1RA9	77	78	79	80	81	82	83											1.8e+07
4FGF	76	77	79	81	82	83	126											3.4e+07
1MLA	151	152	154	155	156	172	174	175										9.5e+06
1AKY	6	105	106	109	110	111	112	113										4.1e+08
1TTAa	3	5	59	60	61	62	63	64	65									2.9e+08
1IFC	45	63	64	65	66	67	81	82	83									1.8e+09
2CPL	31	35	77	79	82	83	84	106	108	109								5.4e+08
1IGD	25	26	27	28	29	30	31	32	33	34								1.5e+09
2END	65	66	67	68	69	70	71	72	73	103								5.8e+09
1LAM	237	238	239	240	241	242	243	244	294	295								7.2e+09
1PHP	196	197	198	199	200	201	220	221	237	239								5.4e+10
1HFC	182	183	184	185	194	195	196	219	220	222	223							1.2e+09
2RN2	73	74	75	76	77	78	79	80	81	82	104							5.7e+11
1XIC	182	183	184	185	190	191	192	193	194	195	198	224						1.2e+11
1SNC	9	10	11	12	71	72	73	74	75	92	93	96						2.7e+11
5P21	11	13	14	81	83	84	85	86	89	90	116	117	125					4.2e+08
1RCF	50	51	52	82	83	84	85	86	116	117	141	163	166					1.0e+11
1FNC	164	165	166	167	168	199	201	258	261	267	268	269	270					2.2e+11
3CHY	11	56	57	58	59	60	63	64	66	67	68	69	85					4.1e+12
1CKAa	134	135	136	137	159	160	161	162	163	164	170	171	172					6.7e+12
2MHR	55	56	57	58	59	60	61	62	63	70	73	74	77					7.5e+14
2PHY	31	33	38	39	60	61	62	63	64	65	66	67	70	120				4.4e+14
1DAD	1	2	3	4	33	34	36	102	105	106	109	111	112	113	114			3.1e+12
1PHB	207	208	209	210	211	212	213	214	215	216	218	220	221	224	225			1.9e+15
1WHI	37	38	39	59	60	61	62	63	84	85	87	91	106	109	111	114		2.1e+14
1CEM	211	212	213	214	215	216	240	243	244	247	257	258	259	260	261	279	282	3.1e+14
1XYZa	598	626	630	640	642	671	672	673	674	675	676	677	678	679	682	684	714	9.3e+16

of primers:

$$
\begin{aligned}
L_T \geq\ & \ldots + 8s^{abc}_{2,2,2} + \\
& 4s^{abc}_{2,2,1} + 4s^{abc}_{2,1,2} + 4s^{abc}_{1,2,2} + \\
& 2s^{abc}_{2,1,1} + 2s^{abc}_{1,2,1} + 2s^{abc}_{1,1,2} + \\
& 1s^{abc}_{1,1,1} + \ldots
\end{aligned} \tag{4.16}
$$

where the ellipses represent the counts of the number of primers needed by the other stretches besides the one composed of positions a, b, and c; each stretch must be represented by a similar set of variables in this constraint.

Finally add a constraint on the library size

$$\log(L) \geq \sum_i \sum_j \sum_e \log(|d_e^i|) \times d_{e,j}^i \tag{4.17}$$

where $|d_{e,j}^i|$ represents the size of the set of degenerate codons that the variable $d_{e,j}^i$ corresponds to. The ILP optimization problem becomes

$$\text{Minimize} \quad \sum_i \sum_j \sum_e e \times d_{e,j}^i \tag{4.18}$$

subject to the above linear constraints.

A set of python scripts to prepare the input to the ILP solver and to interpret the output generated by the GLPK solver can be downloaded from the SwiftLib website.

4.5.4 Additional Library Design Problems

To expand our set of library design problems, we selected a subset of residues from 38 PDBs [47], ran fixed-backbone redesign 1000 times using Rosetta, and counted the number of times each amino acid appeared at each of the design positions. For each design problem, we calculated the amino-acid diversity as the product of the number of amino-acids with non-zero counts, and then tested each design problem while varying the limit on the library size (at 10%, 100% and 1000% of the amino acid diversity), the limit on the number of primers (10, 20, and 30), and the number of degenerate codons per position (1, 2, or 3). When using multiple degenerate codons per position, we artificially declared primer boundaries at every 20 residues. We forbade the stop codon in these library design jobs.

Table S4.2 gives the residues involved in the 38 design problems. The errors and running times are given in Tables S4.3 and S4.4; the running times represent only the dynamic programming portion of the running time, the running time for the initial enumeration of degenerate codon combinations is not included. Running time was measured on a dual-CPU, 10-core per CPU, 2.5GHz Xeon E5-2670 machine with 54 GB of RAM running Ubuntu 14.04.

Table S4.3: Error for SwiftLib Solutions for 38 additional Library Design Problems.

PDB	div. limit	1 DC	2 DCs			3 DCs			PDB	div. limit	1 DC	2 DCs			3 DCs		
#primers		1	10	20	30	10	20	30	#primers		1	10	20	30	10	20	30
	2.80e+01	514	112	112	112	112	112	112		1.43e+02	377	102	102	102	102	78	78
1BKF	2.88e+02	50	5	5	5	4	1	1	2BOPa	1.43e+03	130	6	6	6	6	1	1
	2.88e+03	4	0	0	0	0	0	0		1.43e+04	48	0	0	0	0	0	0
	3.60e+01	171	36	36	36	36	36	36		4.91e+03	172	36	36	36	36	26	23
1AMM	3.60e+02	40	1	1	1	1	0	0	3LZM	4.91e+04	80	6	3	3	6	3	2
	3.60e+03	27	0	0	0	0	0	0		4.91e+05	69	1	0	0	1	0	0
	7.10e+03	138	47	47	47	47	35	30		7.19e+04	172	58	55	55	26	21	20
1PLC	7.10e+04	65	11	11	11	11	5	2	1JBC	7.19e+05	129	39	35	35	9	4	3
	7.10e+05	39	1	0	0	1	0	0		7.19e+06	110	34	30	30	7	1	0
	1.35e+05	311	147	112	112	147	112	98		1.64e+05	293	100	89	89	94	80	70
1RRO	1.35e+06	128	26	9	9	26	9	6	1CYO	1.64e+06	134	30	16	16	30	16	9
	1.35e+07	64	10	1	1	10	1	0		1.64e+07	93	6	3	3	6	2	1
	3.49e+04	222	64	45	45	64	44	41		6.34e+05	557	168	137	137	168	137	109
1CTJ	3.49e+05	144	21	15	15	21	15	13	1KNB	6.34e+06	426	57	33	33	57	33	24
	3.49e+06	117	8	5	5	8	5	3		6.34e+07	320	6	1	1	6	1	1
	1.22e+06	633	220	151	151	220	144	101		1.83e+06	282	146	112	112	146	112	98
1NIF	1.22e+07	466	54	17	14	44	14	7	1RA9	1.83e+07	177	45	23	23	45	23	19
	1.22e+08	391	6	3	3	6	0	0		1.83e+08	144	29	9	9	29	9	8
	3.35e+06	364	124	104	104	124	103	96		9.53e+05	229	61	51	50	61	48	42
4FGF	3.35e+07	282	63	35	35	63	34	26	1MLA	9.53e+06	192	32	20	17	30	17	11
	3.35e+08	228	17	9	9	17	8	6		9.53e+07	171	7	0	0	4	0	0
	4.08e+07	366	133	112	112	133	107	95		2.94e+07	272	115	72	72	106	68	54
1AKY	4.08e+08	296	71	45	45	71	39	29	1TTAa	2.94e+08	178	39	23	23	33	21	16
	4.08e+09	241	36	21	21	36	20	14		2.94e+09	148	12	8	8	6	2	1
	1.75e+08	421	128	61	61	128	61	51		5.37e+07	263	91	65	63	80	57	47
1IFC	1.75e+09	354	93	35	31	93	35	30	2CPL	5.37e+08	226	44	20	18	41	20	10
	1.75e+10	329	83	25	23	83	25	22		5.37e+09	205	29	3	1	29	3	0
	1.47e+08	202	92	71	71	92	71	65		5.78e+08	131	52	41	41	52	40	37
1IGD	1.47e+09	175	66	45	45	66	45	41	2END	5.78e+09	108	30	22	22	30	21	18
	1.47e+10	151	46	25	25	46	25	23		5.78e+10	90	18	12	12	18	12	9
	7.17e+08	335	130	109	109	130	109	103		5.41e+09	582	196	152	134	196	149	124
1LAM	7.17e+09	280	84	57	57	84	57	52	1PHP	5.41e+10	464	112	68	49	112	62	46
	7.17e+10	254	53	32	32	53	32	27		5.41e+11	389	57	27	21	57	27	20
	1.23e+08	438	170	154	145	82	62	50		5.66e+10	1280	683	591	591	665	529	451
1HFC	1.23e+09	377	110	103	101	29	20	14	2RN2	5.66e+11	1082	433	366	366	396	289	229
	1.23e+10	346	94	90	88	9	5	2		5.66e+12	958	289	248	248	262	163	120
	1.21e+10	394	118	86	86	118	82	77		2.65e+10	762	107	65	60	107	44	38
1XIC	1.21e+11	344	81	56	56	81	55	54	1SNC	2.65e+11	738	83	45	42	83	24	18
	1.21e+12	325	71	47	47	71	46	45		2.65e+12	722	66	32	30	66	8	5
	4.16e+07	124	57	40	32	57	35	26		1.01e+10	304	88	55	53	87	49	39
5P21	4.16e+08	83	23	14	8	23	14	8	1RCF	1.01e+11	268	58	28	23	53	20	13
	4.16e+09	54	8	3	1	8	3	1		1.01e+12	247	39	10	8	34	2	0
	2.18e+10	497	149	86	75	148	73	53		4.09e+11	441	228	147	147	207	135	111
1FNC	2.18e+11	443	93	43	36	93	28	17	3CHY	4.09e+12	324	125	68	68	125	68	51
	2.18e+12	407	64	23	22	64	10	4		4.09e+13	257	82	43	43	82	43	34
	6.73e+11	960	357	183	156	357	183	146		7.55e+13	1043	626	561	561	626	547	519
1CKAa	6.73e+12	835	257	109	98	257	109	93	2MHR	7.55e+14	856	479	417	417	479	411	378
	6.73e+13	762	209	64	54	209	64	50		7.55e+15	715	344	286	286	344	278	267
	4.41e+13	695	324	245	192	324	233	185		3.14e+11	349	92	66	59	92	64	53
2PHY	4.41e+14	568	237	161	113	237	157	109	1DAD	3.14e+12	280	50	32	27	50	28	20
	4.41e+15	520	198	135	89	198	132	87		3.14e+13	233	22	12	9	22	8	4
	1.93e+14	778	372	315	315	372	315	307		2.07e+13	463	81	55	48	81	52	44
1PHB	1.93e+15	663	272	220	220	272	220	213	1WHI	2.07e+14	434	52	32	27	52	32	25
	1.93e+16	608	217	168	168	217	168	168		2.07e+15	417	35	16	11	35	16	9
	3.05e+13	356	83	52	44	83	51	43		9.34e+15	1235	412	298	266	412	230	184
1CEM	3.05e+14	337	65	35	29	65	34	27	1XYZa	9.34e+16	1132	317	218	187	317	152	112
	3.05e+15	322	50	21	16	50	18	14		9.34e+17	1073	282	183	155	282	116	79

The median running time was 0.61 seconds for the entire set of 798 design problems; the longest running time was 104 seconds. Figure S4.1 shows that the observed running time scales quadratically in an output-sensitive fashion with the error and that running time increases with the limit on the number of primers.

Increasing the number of degenerate codons from two to three had a relatively minor change in the error; the mean reduction in error was 22% (median = 6%, $\sigma = 31\%$). Increasing the primer limit from 10 to 30 had a larger reduction in error of 49% (median = 49%, $\sigma = 27\%$). The greatest improvement came from going from a single degenerate codon per position to multiple degenerate codons per position, with a 75% (median = 75%, $\sigma = 14\%$) reduction in error when switching to 2 degenerate codons per position and a limit of 10 primers total, and an 84% (median = 86%, $\sigma = 12\%$) reduction in error when switching to 2 degenerate codons

Table S4.4: Running times in seconds for SwiftLib Solutions for 38 Library Design Problems.

PDB	div. limit	1 DC	2 DCs			3 DCs			PDB	div. limit	1 DC	2 DCs			3 DCs		
#primers		1	10	20	30	10	20	30	#primers		1	10	20	30	10	20	30
	2.80e+01	0.00	0.09	0.35	0.76	0.12	0.37	0.91		1.43e+02	0.00	0.10	0.40	0.86	0.12	0.51	0.89
1BKF	2.88e+02	0.00	0.04	0.15	0.33	0.04	0.14	0.31	2BOPa	1.43e+03	0.00	0.05	0.16	0.36	0.04	0.14	0.31
	2.88e+03	0.00	0.04	0.14	0.30	0.03	0.26	0.31		1.43e+04	0.00	0.05	0.14	0.30	0.04	0.15	0.30
	3.60e+01	0.00	0.08	0.26	0.60	0.07	0.28	0.67		4.91e+03	0.00	0.11	0.44	0.98	0.13	0.44	0.89
1AMM	3.60e+02	0.00	0.07	0.20	0.48	0.06	0.21	0.47	3LZM	4.91e+04	0.00	0.07	0.25	0.58	0.07	0.25	0.54
	3.60e+03	0.00	0.05	0.22	0.46	0.04	0.20	0.44		4.91e+05	0.00	0.07	0.22	0.49	0.07	0.23	0.49
	7.10e+03	0.00	0.10	0.42	0.95	0.12	0.44	0.84		7.19e+04	0.00	0.13	0.55	1.25	0.10	0.36	0.81
1PLC	7.10e+04	0.00	0.06	0.24	0.55	0.06	0.22	0.48	1JBC	7.19e+05	0.00	0.10	0.39	0.91	0.07	0.26	0.58
	7.10e+05	0.00	0.05	0.26	0.46	0.05	0.20	0.65		7.19e+06	0.00	0.09	0.38	0.83	0.06	0.40	0.55
	1.35e+05	0.01	0.42	1.26	2.63	0.54	1.82	5.65		1.64e+05	0.01	0.31	1.17	2.20	0.35	1.21	2.21
1RRO	1.35e+06	0.00	0.11	0.32	0.71	0.13	0.34	0.70	1CYO	1.64e+06	0.00	0.13	0.40	0.83	0.15	0.42	0.80
	1.35e+07	0.00	0.08	0.27	0.60	0.09	0.34	0.81		1.64e+07	0.00	0.08	0.30	0.66	0.09	0.29	0.62
	3.49e+04	0.00	0.18	0.56	1.36	0.20	0.58	1.30		6.34e+05	0.02	0.56	1.68	3.51	0.72	2.20	3.82
1CTJ	3.49e+05	0.00	0.11	0.38	0.83	0.12	0.41	0.90	1KNB	6.34e+06	0.01	0.19	0.69	1.28	0.23	0.66	1.21
	3.49e+06	0.00	0.09	0.34	0.75	0.09	0.55	0.75		6.34e+07	0.01	0.10	0.32	0.70	0.14	0.32	0.71
	1.22e+06	0.02	0.54	1.34	2.99	0.82	1.86	2.71		1.83e+06	0.01	0.58	1.59	3.35	0.72	2.06	3.99
1NIF	1.22e+07	0.01	0.13	0.37	0.80	0.14	0.37	0.74	1RA9	1.83e+07	0.01	0.19	0.52	1.12	0.23	0.57	1.20
	1.22e+08	0.01	0.10	0.29	0.70	0.07	0.38	0.64		1.83e+08	0.00	0.15	0.40	0.89	0.67	0.42	0.93
	3.35e+06	0.01	0.43	1.47	3.23	0.56	2.05	4.10		9.53e+05	0.01	0.22	0.75	1.68	0.25	1.51	1.84
4FGF	3.35e+07	0.01	0.21	0.60	1.25	0.26	0.68	1.33	1MLA	9.53e+06	0.01	0.13	0.46	0.97	0.15	0.51	0.97
	3.35e+08	0.01	0.09	0.34	0.78	0.10	0.35	0.79		9.53e+07	0.01	0.10	0.32	0.75	0.08	0.31	0.74
	4.08e+07	0.01	0.58	1.86	4.26	0.73	2.49	7.58		2.94e+07	0.01	0.46	1.02	2.17	0.43	1.23	2.23
1AKY	4.08e+08	0.01	0.28	0.80	1.65	0.34	0.84	1.55	1TTAa	2.94e+08	0.01	0.16	0.62	1.19	0.17	0.57	1.20
	4.08e+09	0.01	0.17	0.53	1.16	0.32	0.57	1.10		2.94e+09	0.00	0.10	0.42	0.98	0.14	0.38	0.88
	1.75e+08	0.02	0.51	0.97	2.22	0.71	1.32	2.49		5.37e+07	0.01	0.29	0.96	2.25	0.31	1.08	2.17
1IFC	1.75e+09	0.01	0.36	0.69	1.53	0.45	0.78	1.64	2CPL	5.37e+08	0.01	0.17	0.57	1.20	0.18	0.56	1.09
	1.75e+10	0.01	0.31	0.61	1.28	0.41	0.93	1.41		5.37e+09	0.01	0.12	0.39	0.89	0.15	0.43	0.87
	1.47e+08	0.01	0.50	1.33	2.88	0.52	1.59	4.74		5.78e+08	0.01	0.22	0.87	1.79	0.26	0.95	2.06
1IGD	1.47e+09	0.01	0.33	1.01	2.06	0.41	1.16	2.36	2END	5.78e+09	0.01	0.17	0.62	1.43	0.17	0.64	1.40
	1.47e+10	0.01	0.26	0.72	1.58	0.31	0.79	1.84		5.78e+10	0.00	0.13	0.52	1.20	0.16	0.80	1.15
	7.17e+08	0.01	0.62	1.88	4.01	0.67	2.49	5.30		5.41e+09	0.03	0.97	2.84	5.34	1.50	4.03	7.12
1LAM	7.17e+09	0.01	0.34	1.08	2.34	0.42	1.25	3.15	1PHP	5.41e+10	0.02	0.48	1.24	2.24	0.62	1.45	2.57
	7.17e+10	0.01	0.24	0.74	1.60	0.26	0.83	1.78		5.41e+11	0.02	0.27	0.70	1.42	0.33	0.83	1.59
	1.23e+08	0.02	0.55	1.99	4.13	0.32	1.16	3.66		5.66e+10	0.16	7.58	24.45	56.79	11.16	33.03	60.45
1HFC	1.23e+09	0.01	0.33	1.34	2.80	0.17	0.62	1.31	2RN2	5.66e+11	0.12	3.23	10.32	24.58	4.16	10.93	21.28
	1.23e+10	0.01	0.29	1.19	2.61	0.21	0.48	1.06		5.66e+12	0.08	1.61	5.53	13.04	2.96	4.52	7.05
	1.21e+10	0.02	0.55	1.73	3.68	0.66	2.08	4.49		2.65e+10	0.04	0.37	1.06	2.17	0.46	0.99	2.21
1XIC	1.21e+11	0.02	0.38	1.21	2.81	0.43	1.42	3.29	1SNC	2.65e+11	0.04	0.28	0.83	1.85	0.34	0.68	1.45
	1.21e+12	0.01	0.34	1.05	2.41	0.61	1.24	2.74		2.65e+12	0.04	0.24	0.72	1.60	0.28	0.73	1.17
	4.16e+07	0.00	0.26	0.73	1.60	0.25	0.93	1.68		1.01e+10	0.01	0.24	0.85	2.08	0.29	1.02	2.10
5P21	4.16e+08	0.00	0.13	0.57	1.19	0.14	0.59	1.27	1RCF	1.01e+11	0.01	0.15	0.57	1.33	0.17	0.55	1.38
	4.16e+09	0.00	0.11	0.49	1.13	0.14	0.49	1.19		1.01e+12	0.01	0.11	0.44	1.10	0.12	0.61	0.97
	2.18e+10	0.02	0.57	1.44	2.83	0.76	1.59	2.86		4.09e+11	0.02	1.33	3.23	6.94	1.67	6.68	7.36
1FNC	2.18e+11	0.02	0.32	0.86	1.83	0.41	0.75	1.91	3CHY	4.09e+12	0.01	0.63	1.48	3.22	0.81	1.82	3.27
	2.18e+12	0.02	0.23	0.65	1.47	0.31	0.56	1.22		4.09e+13	0.01	0.41	1.07	2.41	0.50	1.87	2.44
	6.73e+11	0.10	2.76	4.30	8.01	4.17	6.41	10.21		7.55e+13	0.17	9.32	32.85	72.58	14.25	50.86	104.34
1CKAa	6.73e+12	0.07	1.69	2.46	5.06	2.49	3.18	6.15	2MHR	7.55e+14	0.10	5.73	19.47	43.00	8.79	30.32	57.90
	6.73e+13	0.06	1.22	1.54	2.98	3.02	2.15	3.54		7.55e+15	0.07	3.79	11.28	24.33	5.29	22.53	32.66
	4.41e+13	0.06	2.82	7.95	12.28	4.16	10.85	18.18		3.14e+11	0.01	0.38	1.47	2.77	0.46	1.75	3.34
2PHY	4.41e+14	0.04	1.80	4.40	6.49	2.64	6.19	8.76	1DAD	3.14e+12	0.01	0.24	0.86	1.90	0.29	0.92	1.91
	4.41e+15	0.03	1.33	3.78	5.02	2.70	4.81	6.75		3.14e+13	0.01	0.20	0.66	1.87	0.24	0.64	1.42
	1.93e+14	0.07	3.56	10.98	23.92	5.36	15.84	32.95		2.07e+13	0.03	0.34	1.29	2.71	0.42	1.71	3.20
1PHB	1.93e+15	0.05	2.31	7.02	14.15	3.18	8.93	19.65	1WHI	2.07e+14	0.03	0.23	0.94	1.94	0.28	1.08	2.29
	1.93e+16	0.04	1.75	4.87	10.25	2.31	13.11	14.01		2.07e+15	0.02	0.18	0.74	1.63	0.33	0.78	1.73
	3.05e+13	0.02	0.37	1.24	2.74	0.44	1.65	5.54		9.34e+15	0.23	4.26	12.23	21.31	6.51	12.36	23.80
1CEM	3.05e+14	0.02	0.28	1.05	2.28	0.34	1.15	2.51	1XYZa	9.34e+16	0.19	2.80	7.02	12.77	4.02	6.38	10.55
	3.05e+15	0.02	0.23	0.83	1.92	0.27	0.86	2.66		9.34e+17	0.17	2.24	5.78	10.13	3.64	4.54	6.74

per position and a limit of 30 primers total. Surprisingly, the use of multiple degenerate codons per position makes increasing the library size more effective: a 10 fold increase in the library size when using 2 or 3 degenerate codons per position reduces the error by 55% (median = 50%, $\sigma = 26\%$) and 60% (median = 61%, $\sigma = 26\%$); however, the same increase in size when using a single degenerate codon per position results in only a 23% (median = 16%, $\sigma = 20\%$) reduction in error.

The library design problems where the diversity limits were set as 100% of the amino-acid diversities were given to the GLPK ILP solver. The running times for these jobs are reported in Table S4.5; specifically, only the running time for the solver is reported, the running time of the python script used to enumerate the degenerate codon combinations and to create the input file for the solver is not reported. Several of the ILP jobs did not run correctly. None of the 1PHB library design jobs with 3 degenerate codons per position could be run. This protein had a single stretch with 12 designable positions, requiring $3^{12} = 531,441$ binary variables to

Table S4.5: Running times in seconds for ILP Solutions for 38 Library Design Problems. Jobs that took more than 60000 seconds were killed before they completed, and so the running times reported for these jobs represent a lower bound on how long those jobs would have taken to complete. The 1PHB running times for the jobs that could not be run are reported with a dash. Jobs that exited with an assertion failure are indicated with an asterix; the shortest job that failed ran for 8 hours first.

PDB	div. limit	1 DC	2 DCs			3 DCs			PDB	div. limit	1 DC	2 DCs			3 DCs		
#primers		1	10	20	30	10	20	30	#primers		1	10	20	30	10	20	30
1BKF	2.88e+02	0	0	0	0	0	0	0	2BOPa	1.43e+03	0	0	0	0	2	0	0
1AMM	3.60e+02	0	0	0	0	0	0	0	3LZM	4.91e+04	0	0	0	0	33	2	1
1PLC	7.10e+04	0	0	0	0	22	2	0	1JBC	7.19e+05	0	1	0	0	12	3	0
1RRO	1.35e+06	0	28	2	2	19233	626	122	1CYO	1.64e+06	0	120	7	6	48024	1586	676
1CTJ	3.49e+05	0	6	2	2	54880	5628	2667	1KNB	6.34e+06	0	32	6	6	60000	60000	60000
1NIF	1.22e+07	0	12	2	0	13759	41	3	1RA9	1.83e+07	0	2311	65	52	60000	60000	*30440
4FGF	3.35e+07	0	98	16	24	60000	60000	60000	1MLA	9.53e+06	0	265	5	0	60000	1868	112
1AKY	4.08e+08	0	60000	5354	5504	60000	60000	60000	1TTAa	2.94e+08	0	267	5	6	60000	60000	32343
1IFC	1.75e+09	0	129	5	1	60000	60000	8853	2CPL	5.37e+08	0	406	10	2	60000	739	72
1IGD	1.47e+09	0	60000	60000	60000	60000	60000	60000	2END	5.78e+09	0	1196	820	678	60000	60000	60000
1LAM	7.17e+09	0	60000	60000	60000	60000	60000	60000	1PHP	5.41e+10	0	60000	7261	502	60000	60000	60000
1HFC	1.23e+09	0	99	25	9	*55109	60000	60000	2RN2	5.66e+11	0	60000	60000	60000	60000	60000	60000
1XIC	1.21e+11	0	60000	60000	60000	60000	60000	60000	1SNC	2.65e+11	0	13681	34	16	60000	60000	21730
5P21	4.16e+08	0	97	7	0	*36467	60000	1376	1RCF	1.01e+11	0	2190	44	4	60000	60000	60000
1FNC	2.18e+11	0	60000	256	22	60000	60000	27801	3CHY	4.09e+12	0	60000	60000	60000	60000	60000	60000
1CKAa	6.73e+12	0	60000	60000	*46226	60000	60000	60000	2MHR	7.55e+14	0	60000	60000	60000	60000	60000	60000
2PHY	4.41e+14	0	60000	60000	60000	60000	60000	60000	1DAD	3.14e+12	0	60000	495	98	60000	60000	60000
1PHB	1.93e+15	0	60000	60000	60000	-	-	-	1WHI	2.07e+14	0	60000	974	512	60000	60000	60000
1CEM	3.05e+14	0	60000	60000	15474	60000	60000	60000	1XYZa	9.34e+16	0	60000	60000	*46180	60000	60000	60000

count the number of primers used for that stretch. The python script to generate the ILP input file for these jobs created a 72GB input file in free MPS format (using 140 GB of memory to do so). The GLPK solver, however, exited with the error message "too many rows" when it tried to read in this file. Additionally, several library design jobs failed with the assertion error ("piv1 != 0.0"); these jobs, identified by their PDBid, the number of DCs per position, and the primer limit, were (5P21,3,10), (1XYZa,2,30), (1HFC,3,10), (1CKAa,2,30), and (1RA9,3,30).

The ILP jobs with a single degenerate codon per position all finished in under half a second. None the less, DP was faster than ILP for 29 of the 38 test cases with a median speedup of 2.0x. The use of multiple degenerate codons per position caused the ILP running times to increase dramatically. If it had not already finished, we stopped the solver after 1000 minutes. Of the 220 jobs using multiple degenerate codons that could be run, 107 (49%) did not complete in 1000 minutes. We have reported the running time for these jobs in Table S4.5 as 1000 minutes (60,000 seconds), though this represents only a lower bound on their running time. SwiftLib outperformed ILP in 194 of the 220 jobs in which multiple degenerate codons were considered and which could be run by the GLPK solver. For these 220 jobs, the median speedup for DP over ILP was 5752.

REFERENCES

1. Voigt, C. A., Martinez, C., Wang, Z.-G., Mayo, S. L., and Arnold, F. H. (2002) Protein building blocks preserved by recombination. Nature Structural & Molecular Biology **9**, 553–558

2. Saraf, M. C., Horswill, A. R., Benkovic, S. J., and Maranas, C. D. (2004) Famclash: a method for ranking the activity of engineered enzymes. Proceedings of the National Academy of Sciences **101**, 4142–4147

3. Zheng, W., Ye, X., Friedman, A. M., and Bailey-Kellogg, C. (2007) Algorithms for selecting breakpoint locations to optimize diversity in protein engineering by site-directed protein recombination. In Proc. CSB, volume 6, 31–40

4. Waldo, G. S. (2003) Improving protein folding efficiency by directed evolution using the gfp folding reporter. In Directed Enzyme Evolution, 343–359. Springer

5. Gerth, M. L., Patrick, W. M., and Lutz, S. (2004) A second-generation system for unbiased reading frame selection. Protein Engineering Design and Selection **17**, 595–602

6. Gupta, R. D. and Tawfik, D. S. (2008) Directed enzyme evolution via small and effective neutral drift libraries. Nature methods **5**, 939–942

7. Fellouse, F. A., Wiesmann, C., and Sidhu, S. S. (2004) Synthetic antibodies from a four-amino-acid code: a dominant role for tyrosine in antigen recognition. Proceedings of the National Academy of Sciences of the United States of America **101**, 12467–12472

8. Koide, A., Gilbreth, R. N., Esaki, K., Tereshko, V., and Koide, S. (2007) High-affinity single-domain binding proteins with a binary-code interface. Proceedings of the National Academy of Sciences **104**, 6632–6637

9. Reetz, M. T., Kahakeaw, D., and Lohmer, R. (2008) Addressing the numbers problem in directed evolution. ChemBioChem **9**, 1797–1804

10. Kille, S., Acevedo-Rocha, C. G., Parra, L. P., Zhang, Z.-G., Opperman, D. J., Reetz, M. T., and Acevedo, J. P. (2012) Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. ACS synthetic biology **2**, 83–92

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

12. Reetz, M. T. and Wu, S. (2008) Greatly reduced amino acid alphabets in directed evolution: making the right choice for saturation mutagenesis at homologous enzyme positions. Chemical Communications 5499–5501

13. Hayes, R. J., Bentzien, J., Ary, M. L., Hwang, M. Y., Jacinto, J. M., Vielmetter, J., Kundu, A., and Dahiyat, B. I. (2002) Combining computational and experimental screening for rapid optimization of protein properties. Proceedings of the National Academy of Sciences **99**, 15926–15931

14. Arkin, A. P. and Youvan, D. C. (1992) Optimizing nucleotide mixtures to encode specific subsets of amino acids for semi-random mutagenesis. Nature Biotechnology **10**, 297–300

15. Labean, T. H. and Kauffman, S. A. (1993) Design of synthetic gene libraries encoding random sequence proteins with desired ensemble characteristics. Protein Science **2**, 1249–1254

16. Jensen, L. J., Andersen, K. V., Svendsen, A., and Kretzschmar, T. (1998) Scoring functions for computational algorithms applicable to the design of spiked oligonucleotides. Nucleic acids research **26**, 697–702

17. Wolf, E. and Kim, P. S. (1999) Combinatorial codons: a computer program to approximate amino acid probabilities with biased nucleotide usage. Protein science **8**, 680–688

18. Wang, W. and Saven, J. G. (2002) Designing gene libraries from protein profiles for combinatorial protein experiments. Nucleic acids research **30**, e120–e120

19. Craig, R. A., Lu, J., Luo, J., Shi, L., and Liao, L. (2010) Optimizing nucleotide sequence ensembles for combinatorial protein libraries using a genetic algorithm. Nucleic acids research **38**, e10–e10

20. Nov, Y. and Segev, D. (2013) Optimal codon randomization via mathematical programming. Journal of Theoretical Biology

21. Patrick, W. M., Firth, A. E., and Blackburn, J. M. (2003) User-friendly algorithms for estimating completeness and diversity in randomized protein-encoding libraries. Protein engineering **16**, 451–457

22. Firth, A. E. and Patrick, W. M. (2008) GLUE-IT and PEDEL-AA: new programmes for analyzing protein diversity in randomized libraries. Nucleic acids research **36**, W281–W285

23. Mena, M. A. and Daugherty, P. S. (2005) Automated design of degenerate codon libraries. Protein Engineering Design and Selection **18**, 559–561

24. Treynor, T. P., Vizcarra, C. L., Nedelcu, D., and Mayo, S. L. (2007) Computationally designed libraries of fluorescent proteins evaluated by preservation and diversity of function. Proceedings of the National Academy of Sciences **104**, 48–53

25. Ponder, J. W. and Richards, F. (1987) Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequences for different structural classes. Journal of Molecular Biology **193**, 775–791

26. Allen, B. D., Nisthal, A., and Mayo, S. L. (2010) Experimental library screening demonstrates the successful application of computational protein design to large structural ensembles. Proceedings of the National Academy of Sciences **107**, 19838–19843

27. Parker, A. S., Griswold, K. E., and Bailey-Kellogg, C. (2011) Optimization of combinatorial mutagenesis. Journal of Computational Biology **18**, 1743–1756

28. Chen, T. S., Palacios, H., and Keating, A. E. (2012) Structure based re-design of the binding specificity of anti-apoptotic Bcl-xL. Journal of molecular biology

29. Kuhlman, B. and Baker, D. (2000) Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences, USA **97**, 10383–8

30. Das, R. and Baker, D. (2008) Macromolecular modeling with rosetta. Annual Review of Biochemistry **77**, 363–382

31. Leaver-Fay, A., Tyka, M., Lewis, S. M. S. M., Lange, O. F., Thompson, J., Jacak, R., Kaufman, K. W., Renfrew, P. D., Smith, C. A., Sheffler, W., Davis, I. W., Cooper, S., Treuille, A., Mandell, D. J., Richter, F., Ban, Y.-E. A., Fleishman, S. J., Corn, J. E., Kim, D. E., Lyskov, S., Berrondo, M., Mentzer, S., Popović, Z., Havranek, J. J., Karanicolas, J., Das, R., Meiler, J., Kortemme, T., Gray, J. J., Kuhlman, B., Baker, D., and Bradley, P. (2011) Rosetta3: An Object-Oriented software suite for the simulation and design of macromolecules. Methods Enzymol. **Volume 487**, 545–574

32. Bellman, R. (1957) Dynamic Programming. Princeton University Press, Princeton, NJ

33. Herman, A. and Tawfik, D. S. (2007) Incorporating synthetic oligonucleotides via gene reassembly (ISOR): a versatile tool for generating targeted libraries. Protein Engineering Design and Selection **20**, 219–226

34. Karp, R. (1972) Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, Complexity of Computer Computations, 85–103. Plenum Press

35. Mena, M. A., Treynor, T. P., Mayo, S. L., and Daugherty, P. S. (2006) Blue fluorescent proteins with enhanced brightness and photostability from a structurally targeted library. Nature biotechnology **24**, 1569–1571

36. Guntas, G., Purbeck, C., and Kuhlman, B. (2010) Engineering a protein–protein interface using a computationally designed library. Proceedings of the National Academy of Sciences **107**, 19296–19301

37. Chica, R. A., Moore, M. M., Allen, B. D., and Mayo, S. L. (2010) Generation of longer emission wavelength red fluorescent proteins using computationally designed libraries. Proceedings of the National Academy of Sciences **107**, 20257–20262

38. Lippow, S. M., Moon, T. S., Basu, S., Yoon, S.-H., Li, X., Chapman, B. A., Robison, K., Lipovšek, D., and Prather, K. L. (2010) Engineering enzyme specificity using computational design of a defined-sequence library. Chemistry & biology **17**, 1306–1315

39. Khersonsky, O., Röthlisberger, D., Dym, O., Albeck, S., Jackson, C. J., Baker, D., and Tawfik, D. S. (2010) Evolutionary optimization of computationally designed enzymes: Kemp eliminases of the KE07 series. Journal of molecular biology **396**, 1025–1042

40. Fleishman, S. J., Whitehead, T. A., Ekiert, D. C., Dreyfus, C., Corn, J. E., Strauch, E.-M., Wilson, I. A., and Baker, D. (2011) Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. Science **332**, 816–821

41. Khersonsky, O., Röthlisberger, D., Wollacott, A. M., Murphy, P., Dym, O., Albeck, S., Kiss, G., Houk, K., Baker, D., and Tawfik, D. S. (2011) Optimization of the in-silico-designed kemp eliminase KE70 by computational design and directed evolution. Journal of molecular biology **407**, 391–412

42. Azoitei, M. L., Correia, B. E., Ban, Y.-E. A., Carrico, C., Kalyuzhniy, O., Chen, L., Schroeter, A., Huang, P.-S., McLellan, J. S., Kwong, P. D., et al. (2011) Computation-guided backbone grafting of a discontinuous motif onto a protein scaffold. Science **334**, 373–376

43. Khersonsky, O., Kiss, G., Röthlisberger, D., Dym, O., Albeck, S., Houk, K. N., Baker, D., and Tawfik, D. S. (2012) Bridging the gaps in design methodologies by evolutionary optimization of the stability and proficiency of designed kemp eliminase ke59. Proceedings of the National Academy of Sciences **109**, 10358–10363

44. Whitehead, T. A., Chevalier, A., Song, Y., Dreyfus, C., Fleishman, S. J., De Mattos, C., Myers, C. A., Kamisetty, H., Blair, P., Wilson, I. A., et al. (2012) Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing. Nature biotechnology **30**, 543–548

45. Dutta, S., Chen, T. S., and Keating, A. E. (2013) Peptide ligands for pro-survival protein bfl-1 from computationally guided library screening. ACS chemical biology **8**, 778–788

46. Blomberg, R., Kries, H., Pinkas, D. M., Mittl, P. R., Grütter, M. G., Privett, H. K., Mayo, S. L., and Hilvert, D. (2013) Precision is essential for efficient catalysis in an evolved kemp eliminase. Nature

47. Ding, F. and Dokholyan, N. V. (2006) Emergence of protein fold families through rational design. PLoS Comput Biol **2**, e85

Chapter 5

CONCLUSIONS

5.1 Introduction

These chapters describe the development of new computational tools to aid in the engineering of protein structures and functions. The unifying theme of this work is the combination of evolutionary principles and computational structural modeling. The application of these methods has demonstrated the great potential of this combination strategy, and the resultant tools will enable continued improvement to protein engineering methods. The purpose of this chapter is to explore the limitations of these methods, and conjecture their continued improvement.

5.2 Lessons and Future Directions of Protein Engineering with SEWING

Chapters 2 and 3 of this dissertation describe the development and application of a novel strategy for designing proteins and protein interfaces that leverages evolutionary concepts to create large numbers of highly diverse and designable protein backbones. We describe two strategies, contiguous and discontiguous SEWING, which attempt to capture and reuse different structural attributes from native protein structure (Chapter 2). Accurate designs from both strategies were achieved, but by and large, contiguous SEWING was more successful than discontiguous. A critical difference between these two methods is the necessity to design loops between elements of secondary structure. Our results indicate that our attempts at creation of these de novo loops were largely unsuccessful, and could be responsible for the lower success rate of discontiguous SEWING. To further test this hypothesis, we have attempted to redesign the fifth helix of discontiguous design DA05, which was shown to be disordered in solution,

using the contiguous SEWING method. To accomplish this, the final loop and helix from the DA05 model was removed and the remainder of the structure was added to the SEWING graph, similar to the addition of binding motifs described in chapter 3. Three designs from this method were experimentally characterized. All three designs expressed well in bacteria and displayed cooperative unfolding transitions (Figure 5.1). Two of the designs, DA05R1 and DA05R2 were significantly more thermostable than the original DA05 design, with DA05R1 displaying an increase in melting temperature of over 20 °C. Given that the first four helices of the redesigned structure remained largely identical to the original DA05, this result provides strong evidence that the new helix is responsible for the improved thermostability of this design, and is likely well-folded.

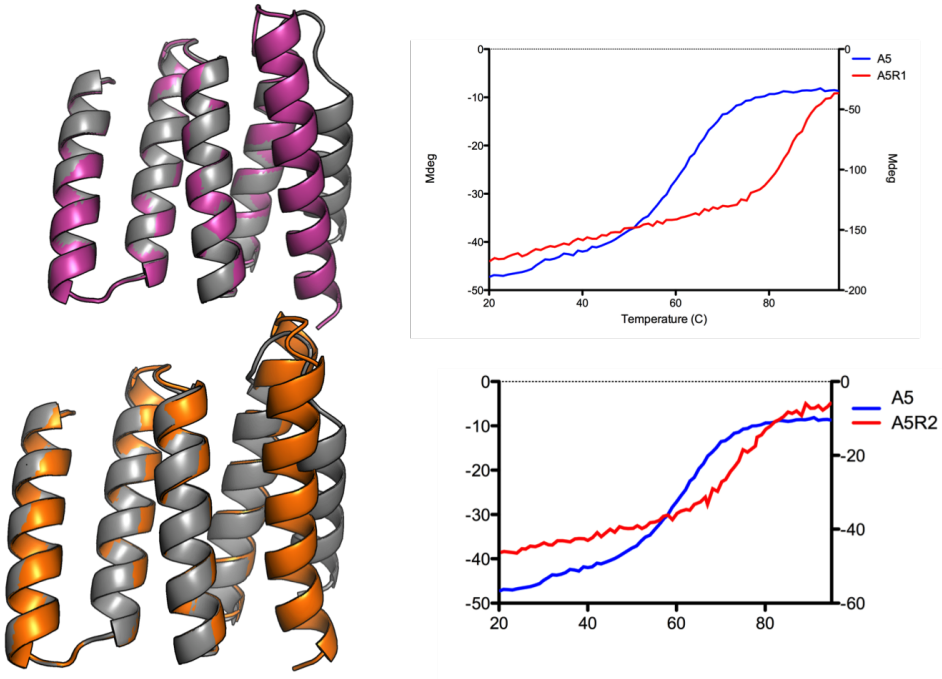

Figure 5.1: Thermal denaturation of DA05 redesigns. Design models of DA05R1 (pink) and DA05R2 (orange) superimposed on the original DA05 design model (gray). Next to each design model is a thermal denaturation of the redesigned protein (red line) compared with the original DA05 thermal melt (blue line). In both cases, the redesigned protein displays higher thermostability than the initial design.

This result provides further support for the contiguous SEWING method, and illustrates the advantage of utilizing native loops in the creation of de novo designs. Since the initial

work with discontiguous SEWING, alternative methods for designing loops de novo have been developed[1]. The application of these methods to discontiguous SEWING may yield improved success rates, allowing de novo designs to incorporate many of the complex tertiary interactions frequently seen in areas of protein function, such as enzymatic active sites and nucleic acid binding motifs. Additionally, the redesign of DA05 illustrates that contiguous and discontiguous SEWING are not mutually exclusive, and that their combination can allow for both increased diversity, and improved accuracy.

The SEWING method differs starkly from alternative strategies of de novo backbone generation in that a target topology is not necessarily provided. While this attribute is a benefit in many cases, it can also be a limitation. A targeted fold directs sampling towards a much more confined region of conformation space and therefore increased sampling of that space can be achieved. This fine-grained sampling can be highly useful in the engineering of protein structures that rely critically on exact orientation, such as with protein nanomaterials and capsid-like cages[2, 3]. The limited sampling of a specified conformational space in SEWING has proven to be a challenge in the design of β-sheet containing proteins, which often rely on the specific orientation of non-adjacent β-strands to accommodate the backbone-backbone hydrogen bonding pattern. Currently, SEWING relies on a single structural superimposition between any two substructures. One way to increase the local sampling necessary to generate these types of structural features is to generate multiple alignments for each pair of substructures, allowing two substructures to be combined in many ways that generate similar but distinct chimeras. Similarly, parametric approaches could be applied to sample the local space around a given structural supimposition. Alternatively, larger substructures that already contain these types of interactions can be used in the SEWING graph. To test this strategy an alternative SEWING graph containing $\beta\alpha\beta$ motifs that exhibit hydrogen bonding between the first and last β-strand is currently being tested. Initial computational results suggest that this strategy may help design these difficult interactions (Figure 5.2).

The Sewing strategy was also employed towards the generation of de novo proteins that bind a target molecule. While binding was observed, high resolution structural validation of the designed structures was not achieved. This result could indicate that designed structures

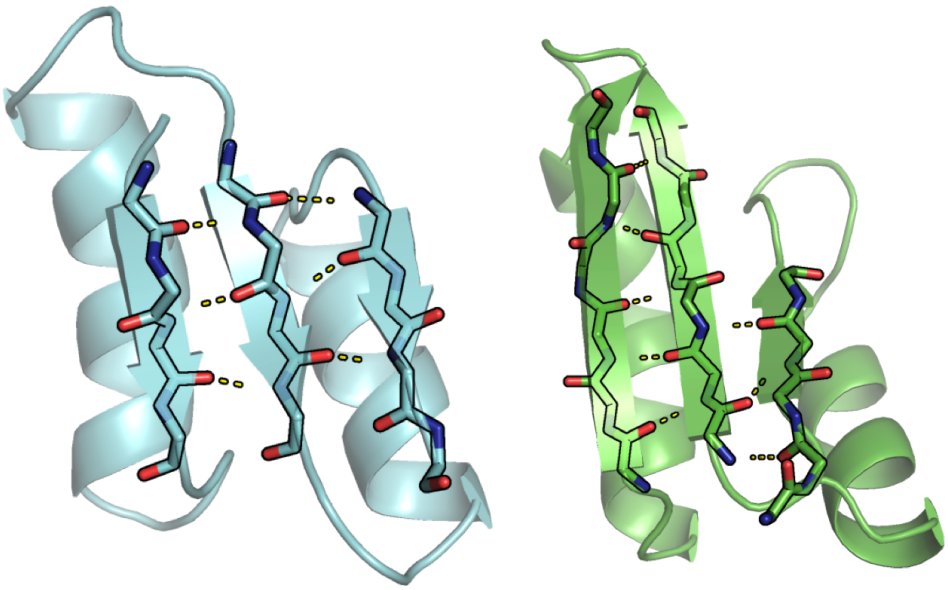

Figure 5.2: Computational design models of two SEWING designs constructed from $\beta\alpha\beta$ substructures. β-strand residues are shown in sticks to highlight the formation of hydrogen bonds between non-adjacent β-strands

contain elements that adopt multiple conformations. The use of multi-state design, and careful binding motif selection was proposed in chapter 3 as a potential solution to this issue. The interfaces designed in chapter 3 relied on existing structures of a known binding interface, and therefore this method is not readily adopted to design of binders to targets with no known binders. One potential future direction is to incorporate successful anchoring motifs into SEWING designs[4, 5, 6, 7]. The ability to incorporate these types of functional motifs into SEWING designs is ongoing, with preliminary computational results in the generation of calcium binding proteins.

5.3 Lessons and Limitations in Refinement of Directed Evolution Libraries

Chapter 4 describes the development of a web tool called SwiftLib for the rapid generation of degenerate codon libraries. The worldwide availability through a website, and near

instantaneous runtime allow researchers to very rapidly test large numbers of variables, such as number of primers, in order to best accommodate the desired sequence diversity in a degenerate codon library. The tool does not, however, attempt to define the best strategy for selecting how to identify sequences likely to give the desired function. This topic is a subject of much work, with solutions ranging from computational design[8, 9], evolutionary sequence profiles[10], and over-enrichment of specific amino-acids[11, 12]. These methods have illustrated that frequently better results can be achieved through small enriched libraries rather than large random ones[13]. Few comparisons between these various methods have been tested, which could in part be attributed to the lack of a deterministic and rapid way of generating a library from a given collection of sequences predicted to be informative. SwiftLib offers a unique solution to this problem and could enable quantitative comparison between various library refinement methods. This information would be highly valuable to protein engineers, and could allow for the development of improved methods. One limitation of SwiftLib is the inability to incorporate information regarding coupled sequence positions, a common phenomenon in evolutionary data. Additional studies into the importance of this data are necessary to fully determine the extent of this limitation.

5.4 Future of Protein Engineering Techniques

Protein engineering techniques have developed rapidly over the past several decades and have been applied to the generation of proteins used extensively in clinical and industrial applications. The chapters here develop several new strategies and tools to aid in protein engineering, with the goal of combining effective elements of natural protein evolution, and structure guided computational design. These results provide valuable preliminary data that this strategy can be used to engineer proteins and protein-interactions that are difficult to design with alternative techniques. Our analysis has highlighted several areas of improvement, and continued development will elucidate the extent to which these techniques can be applied to a wider variety of engineering challenges.

REFERENCES

1. Das, R. (2013) Atomic-Accuracy prediction of protein loop structures through an RNA-Inspired ansatz. PLoS One **8**, e74830

2. King, N. P., Sheffler, W., Sawaya, M. R., Vollmar, B. S., Sumida, J. P., André, I., Gonen, T., Yeates, T. O., and Baker, D. (2012) Computational design of self-assembling protein nanomaterials with atomic level accuracy. Science **336**, 1171–1174

3. Huang, P.-S., Oberdorfer, G., Xu, C., Pei, X. Y., Nannenga, B. L., Rogers, J. M., DiMaio, F., Gonen, T., Luisi, B., and Baker, D. (2014) High thermodynamic stability of parametrically designed helical bundles. Science **346**, 481–485

4. Jacobs, T. M. and Kuhlman, B. (2013) Using anchoring motifs for the computational design of protein-protein interactions. Biochem. Soc. Trans. **41**, 1141–1145

5. Der, B. S., Machius, M., Miley, M. J., Mills, J. L., Szyperski, T., and Kuhlman, B. (2012) Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer. J. Am. Chem. Soc. **134**, 375–385

6. Stranges, P. B., Machius, M., Miley, M. J., Tripathy, A., and Kuhlman, B. (2011) Computational design of a symmetric homodimer using β-strand assembly. Proc. Natl. Acad. Sci. U. S. A. **108**, 1–6

7. Fleishman, S. J., Corn, J. E., Strauch, E.-M., Whitehead, T. a., Karanicolas, J., and Baker, D. (2011) Hotspot-centric de novo design of protein binders. J. Mol. Biol. **413**, 1047–1062

8. Guntas, G., Hallett, R. A., Zimmerman, S. P., Williams, T., Yumerefendi, H., Bear, J. E., and Kuhlman, B. (2015) Engineering an improved light-induced dimer (iLID) for controlling the localization and activity of signaling proteins. Proc. Natl. Acad. Sci. U. S. A. **112**, 112–117

9. Guntas, G., Purbeck, C., and Kuhlman, B. (2010) Engineering a protein-protein interface using a computationally designed library. Proc. Natl. Acad. Sci. U. S. A. **107**, 19296–19301

10. Verma, R., Schwaneberg, U., and Roccatano, D. (2012) Computer-Aided protein directed evolution: a review of web servers, databases and other computational tools for

protein engineering. Comput. Struct. Biotechnol. J. **2**, e201209008

11. Schilling, J., Schöppe, J., and Plückthun, A. (2014) From DARPins to LoopDARPins: novel LoopDARPin design allows the selection of low picomolar binders in a single round of ribosome display. J. Mol. Biol. **426**, 691–721

12. Koide, S. and Sidhu, S. S. (2009) The importance of being tyrosine: Lessons in molecular recognition from minimalist synthetic binding proteins. ACS Chem. Biol. **4**, 325–334

13. Lutz, S. and Patrick, W. M. (2004) Novel methods for directed evolution of enzymes: quality, not quantity. Curr. Opin. Biotechnol. **15**, 291–297