Explicit Consideration of Solubility and Interaction Specificity in Computational Protein Design

Ronald Jerzy Jacak

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 Biochemistry and Biophysics.

Chapel Hill
2011

Approved by:

Advisor: Brian Kuhlman

Reader: Sharon Campbell

Reader: Charles Carter

Reader: Channing Der

Reader: Tim Elston

Abstract

Ronald Jerzy Jacak: Explicit Consideration of Solubility and Interaction Specificity in Computational Protein Design
(Under the direction of Brian Kuhlman)

Most successes to date in computational protein design have relied on optimizing sequences to fit well for a single structure. Multistate design represents a new approach to designing proteins in which sequences are optimized for multiple contexts usually given by multiple structures (states). In multistate design simulations, sequences that either stabilize the target state or destabilize alternate competing states are selected. This dissertation describes the application of multistate design to two problems in protein design: designing sequences for solubility and increasing binding specificity in protein-protein interface design.

Previous studies with the modeling program Rosetta have shown that many designed proteins have patches of hydrophobic surface area that are considerably larger than what is seen in native proteins. These patches can lead to nonspecific association and aggregation. We use a multistate design approach to address protein solubility by disfavoring the aggregated state through the addition of a new solubility term to the Rosetta energy function. The score term explicitly detects and penalizes the formation of hydrophobic patches during design. Designing with this new score term results in proteins with naturally occurring frequencies of hydrophobic amino acids on the surface without large hydrophobic patches.

Designing protein-protein interfaces with high affinity and specificity is still a challenge for protein design algorithms. Multistate design is well-suited for addressing the problem of specificity because it can explicitly disfavor off-target interactions. Using a new implementation in Rosetta, multistate design is applied to the orthogonal interface design problem: redesign a protein with many partners to interact with only one of the partners. We use the RalA signaling network as the model system and make our design goal a redesigned RalA that only interacts with the effector RalBP1. Multistate design is able to recover several of the known mutants important for effector binding and predicts many new mutations that alter binding specificity. From *in silico* predictions, single-state design for Ral/RalBP1 by itself is not sufficient to destabilize RalA's interactions with its other effectors. Only multistate design is able to destabilize both of the negative states and give the desired interaction specificity.

To my parents, George and Wanda Jacak

Acknowledgements

First and foremost I would like to acknowledge and thank my advisor, Brian Kuhlman, for being a wonderful mentor and friend, supporting me as graduate student, and being patient with my perfectionist nature.

I must also greatly thank Andrew Leaver-Fay. Much of this thesis would not be possible without his help. The hpatch score is based off of his implementation of another non-pairwise decomposable score term in Rosetta. Andrew also implemented the weight optimization and multistate design protocols in Rosetta.

I would like to thank all of the members of the Kuhlman lab for providing a fun and intellectually stimulating work environment, and especially thank Doug Renfrew, Steven Lewis, and Ben Stranges for the many helpful discussions about Rosetta and my work.

I owe a great deal of my success to my teachers and mentors. I would like to thank Rod Fortney, for being like a second father to me. Without his encouragement, I may have never come to graduate school. I want to say thanks to Phyllis Boudreaux, for making physics class fun, and to Kathleen Finch, for teaching me how to write. I would like to thank Chris Shaw and Robert Latek, for allowing me to work on interesting projects and helping me get into graduate school. Finally, a big thanks goes to Barry Lentz, for getting me to really learn biophysics and for all his advice during my first year at UNC.

I want to thank my committee members, Charlie Carter, Sharon Campbell, Tim Elston, and Channing Der, for the insight they've provided during my yearly meetings. I want to especially thank Charlie, for getting me excited about protein structure with his class, and Sharon, for many helpful discussions regarding working with GTPases.

Many individuals have assisted me in collecting data during my time at UNC. I would like to recognize two individuals in particular: Ashutosh Tripathy, for assistance with collecting ITC data at the Macromolecular Interactions Facility, and Edward Dale, for assistance with running energy function weight optimization runs on the BASS compute cluster.

My time at UNC has been very enjoyable, due in large part to all my friends. I would like to extend special thanks to Jenn Cable, Doug Renfrew, Anh Nguyen, Ben Stranges, Charles Davis, Erin Heenan, Kyle Wilcox, Michael Gaddis, Amber Wells, JD Daw, Ashton Verdery, Amanda Byrd, and to Liz Johns, for helping me get through graduate school sanely.

Finally, I would like to thank my parents, George and Wanda Jacak, for their love and support. They came to the United States so that my life might be better, and for that I am grateful.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CD	Circular dichroism
DEE	Dead-End Elimination
FP	Fluorescence polarization
MC	Monte-Carlo
NMR	Nuclear Magnetic Resonance
PDB	Protein data bank
PSO	Particle Swarm Optimization
REU	Rosetta energy units
RMSD	Root mean square deviation
RPE	Rotamer pair energies
SASA	Solvent accessible surface area

Chapter 1

Introduction

1.1 Computational protein design

Simply stated, the goal of protein design is to find a sequence that will perform some desired function. There are a myriad of functions for which protein design can be used. For example, one may want to design a protein that binds to another protein, or to design an enzyme that catalyzes a specific reaction. The reason protein design is difficult is that, even for a small protein, the number of possible amino acid sequences is vast. Searching through this space of all possible sequences to find the ones with the desired function is the problem of protein design. In computational protein design, computer algorithms are used to search through this space and to find sequences that hopefully have the desired function. Thus, the challenge for computational protein design programs is to reliably find these sequences and to do so in a reasonable amount of time.

Although the field of computational protein design is relatively young, it has already had a large number of successes. In one of the first examples of computational protein design, Desjarlais et al. (1) made redesigns of the core of phage 434 cro protein, with two of them having native-like stabilities. Dahiyat and Mayo (2; 3) used their design algorithm to design a zinc finger $\beta\beta\alpha$motif. Their design model displayed very good agreement with an NMR solution structure. Nauli et al. (4) were able to stabilize and change the folding pathway of protein G. In 2003, Kuhlman et al. (5) used computational protein design to make a completely novel protein fold. Lippow et al. (6) were able to enhance the affinity of several antibodies for their antigens beyond the *in vivo* maturation affinity. More recently, two studies reported success in the design of novel enzymes: Rothlisberger et al. (7) were able to computationally design an enzyme that catalyzes the Kemp elimination reaction and Jiang et al. (8) designed retro-aldolase activity into a number of distinct proteins. A geometric hashing algorithm was used to identify scaffolds suitable for the reaction, followed by design of the residues around the substrate binding site to optimize the reaction.

A number of exciting results have also been obtained in computational protein interface design, a subset of protein design, where the goal is to create novel or change existing protein-protein interfaces. Reynolds et al. redesigned the β-lactamase inhibitor protein (BLIP) to have higher affinity for SHV-1 β-lactamase over the wild type's preferred target TEM-1 BETA-lactamase(9). Sammond et al. used a variety of simulation schemes and experimental testing to develop a protocol for increasing affinity at protein-protein interfaces(10). Liu et al. were able to make a new erythropoietin receptor (EPOR) binding protein by grafting the key interaction residues from erythropoietin onto rat PLCδ1-PH (pleckstrin homology domain of phospholipase Cδ1) (11). This approach is only effective if a structure of a complex between the target protein and another protein is known. Huang and Mayo were the first to design a de novo protein interface(12). They redesigned the β1 domain of protein G to form a heterodimer with an affinity of 300 μM. Recent work by Jha et al. improved upon this result by designing a new p21-activated kinase 1 (PAK1) binding protein(13). Using a computational protocol that combines rigid-body docking with design and minimization, they were able to redesign human hyperplastic discs protein to bind to PAK1 with an affinity of 100 μM. Unfortunately, high-resolution structures of both interfaces confirming the design model are not available. Furthermore, in the realm of natural protein-protein interactions, both of these designed proteins have relatively weak affinities. These results show that we still have a long way to go when it comes to designing protein protein interfaces.

Another very common approach to designing proteins is through directed evolution. Techniques such as phage display and yeast display have been used to design high-affinity interactions for many target proteins and small molecules. Koide et al. obtained high-affinity interactions between a fibronectin type III domain and maltose-binding protein using only the amino acids tyrosine and serine for interface positions(14). Using yeast cell surface display, Boder et al. obtained single chain antibody fragments with femtomolar affinity for fluorescein, with dissociation kinetics even slower than that of biotin-streptavidin(15). Cochran et al. engineered a 30-fold increase in affinity between human epidermal growth factor (EGF) and the EGF receptor using yeast display(16). The success of directed evolution is limited by the size of the library of variants that can be screened with a given technique. If all amino acids are allowed at all positions, there are approximately 10^{13} possible sequences for a set of 10 residues. The maximum sized library that can be screened by phage display is 10^{12}(17) with 10^{10} being more common. This limit means that at best 10% and in most cases only 0.1% of all possible sequences are tested in the experiment. Some cell-free directed evolution methods such as ribosome display can screen $10^{13} - 10^{14}$ sequences(18), but even these methods would come up short for larger design problems. The advantage computational protein design has over directed evolution lies in being able to sample a much larger sequence space than is available to even the best screening methods. Routine protein design simulations consider 10^{14} sequences with many orders of magnitude more possible(19). Another disadvantage to directed evolution techniques is that none of them can tell you in what orientation the protein is binding the target. Knowing where on the target and how the protein is binding may be important for certain applications.

Despite the successes described above, many challenges remain for the field of protein design. Although Kuhlman et al. was successful in designing a novel α/βprotein, the *de novo* design of a helix bundle protein or a β-sheet protein has met with limited success. Summa et al. reported the design of a heterotetrameric helical structure using a computational approach(20). Their design is composed of four peptides which associate to form a tetramer, not one chain which folds into a helix bundle. Kraemer-Pecore et al. designed sequences to fold into a WW domain, a small β-sheet motif(21). A 1D NMR of their design had good overlap with a wild-type WW domain. In neither case were high-resolution structures obtained for these designs, leaving the design models unverified. The *de novo* design of protein interfaces has only been slightly more successful. As described above, the affinities of computationally designed interfaces are a long way off native protein-protein interaction affinities. Thus far, the best *de* *novo* interfaces have come about with the use of directed evolution(22). Computational protein design, however, is very complementary to directed evolution. Combining protein design with directed evolution in a hybrid approach has the potential to speed up the creation of new protein-protein interfaces(23) and maybe will shed light on where protein design programs fail. Another challenge for protein interface design lies in designing specific interactions. It is not uncommon for designed proteins to interact with their target protein, but also to interact with other proteins as strongly as or even better than with the target. Along the same lines, designed proteins may bind their target, but not at the interface for which they were designed. Multistate design approaches, described below, take a step toward addressing some of these specificity challenges and hopefully will improve the success-rate of de novo protein and protein-protein interface design.

1.2 Programs for CPD

The protein design program used in this work, Rosetta, was originally developed for protein structure prediction(24; 25). Since that time, it has expanded and has demonstrated accomplishments in protein design(26; 27), protein-protein docking(28), ligand docking(29; 30; 31), protein stability estimation(32), and enzyme design(7; 8). More recently, newer functions for Rosetta have been emerging including RNA structure prediction(33; 34), crystal structure refinement (35; 36; 37), and NMR structure determination (38; 39). All computational protein design programs have two components: a search algorithm which traverses sequence space generating candidate sequences and a scoring, or energy, function which determines how well a candidate sequence fits for the desired structure. Rosetta uses a Monte Carlo search algorithm and a combination of physically based and knowledge based score terms as part of its energy function. A complete description of the energy function and search algorithm, and how a design simulation is performed with Rosetta is the subject of Chapter 2.

A number of other computational protein design programs have been developed. Of these other protein design programs, ORBIT and EGAD are the most popular. ORBIT(40; 2; 3), developed by Mayo and coworkers, uses a similar energy function but different search algorithm than Rosetta. Originally, ORBIT used the dead-end elimination (DEE) optimization method for exploring sequence space. The advantage of DEE is that if it converges, the solution it finds is the global minimum energy conformation(41). The algorithm works by eliminating rotamers and rotamer pairs that have higher energy than other rotamers and thus would not be part of the minimum energy conformation. Because it is an elimination algorithm and not a deterministic one, it does not always converge. The DEE algorithm is also computationally intensive. For these reasons, an alternative search algorithm called FASTER is more often used in ORBIT(42). The FASTER algorithm always converges and is much faster than DEE(43). The protein design program EGAD(44), developed by Handel and coworkers, has a more physically-based energy function compared to Rosetta and uses either Monte Carlo or a genetic algorithm to search sequence space. EGAD uses the OPLS-AA molecular mechanics force field(45) for bonded interactions, generalized Born for electrostatics, and a solvent accessible surface area-dependent solvation term(46). Details of EGAD can be found in reference (47).

1.3 Multistate protein design

Multistate design optimizes a sequence for multiple criteria by considering multiple states simultaneously during the simulation. In single-state design, also called positive design, the optimal sequence is the one that has the lowest energy for the desired structure. Some design goals, however, are not amenable to single-state design. For example, designing a protein to interact with the activate form of a target protein but not the inactive form would be difficult with single-state design. In multistate design, the active and inactive conformations of the target protein are modeled at the same time. During the simulation, substitutions that either stabilize the interaction with the active form or destabilize the interaction with the inactive form of the target protein are kept. This purposeful destabilization of undesired interactions in multistate design is frequently called negative design. The use of multistate design to improve the solubility of designed proteins and to improve the specificity of designed protein-protein interactions is described below.

1.3.1 Addressing solubility during protein design

It is widely accepted that protein cores are predominantly made up of hydrophobic residues and that hydrophilic residues cover the surface. Large amounts of exposed hydrophobic surface area tend to cause proteins to oligomerize as the free energy of desolvating those residues is favorable (48). Exposed hydrophobic surface area can lead proteins to aggregate into non-functional precipitate. Indeed, one study has shown a correlation between amount of exposed hydrophobic surface area and rate of aggregation (49). However, small clusters of hydrophobic residues are commonly found on the surface of proteins (50) and surface hydrophobics can contribute to protein stability(51; 52). Surface hydrophobic residues are also commonly found at protein-protein interfaces(53). Clearly, proteins surfaces play an important role in how proteins will behave in solution. The difficulty in protein and protein interface design is in designing structures containing enough exposed hydrophobic surface area that the proteins are stable but not so much that the proteins are insoluble.

Protein design programs have developed various methods for designing protein surfaces. ORBIT restricts all surface positions to hydrophilic residues(40; 54). The design program DESIGNER upweights hydrogen bond and electrostatics interaction energies at the surface to favor the design of polar residues(55). Pokala and Handel experimented with using a hydrophobic SASA metric to filter designs made by their design program, EGAD, but found that a simple reference state for surface residues was more effective for designing soluble sequences(44). Rosetta implicitly designs sequences for solubility through the use of the reference energy term (56). Each amino acid type has an energy associated with it, the reference state energy, which affects how often that residue type is used in a design run. The values for each amino acid type were derived by iteratively performing design and adjusting the values until redesigns had high sequence identity to the native proteins. The assumption is that, since the native sequences are soluble, if Rosetta is trained to produce native-like sequences the designs it makes should be soluble, too.

Despite the use of amino acid reference energies, Rosetta designed proteins have a tendency to aggregate when expressed. It is true that the reference energies ensure that the surfaces of designs have similar hydrophobic content as native proteins. However, in Rosetta redesigns, the surface hydrophobics tend to cluster together leading to large hydrophobic patches on the surface. In a comparison of four proteins redesigned using EGAD and Rosetta, all of the redesigns made with Rosetta had larger surface-exposed hydrophobic patches(44). Additionally, several Rosetta designed proteins that have been expressed by members of the lab have failed due to aggregation. In fact, one of the most common ways Rosetta fails is by identifying sequences that, when tested experimentally, are insoluble.

We decided to apply a multistate design approach to address protein solubility during design. If the folded and functional form of a protein is considered the target state, then the aggregated form can be thought of as an alternative state. Negative design against this alternate state should lead to more designed sequences adopting the target state. Instead of trying to model the aggregated state as a separate structure for which energies would be made unfavorable, we imposed negative design against this state by adding a solubility term to the energy function. The development and application of this new, non-pairwise decomposable score term against the aggregated state is described in chapter 3. The score term, named hpatch, penalizes the exposure of large contiguous amounts of hydrophobic surface area. As the desolvation of nonpolar surface area is very favorable energetically, we expect that proteins designed with this new score will be less likely to oligomerize and/or aggregate when expressed.

1.3.2 Protein-protein interface specificity

Instead of creating novel protein-protein interfaces, many interface design studies have concentrated on changing interaction specificities. The Ca^{2+} binding protein calmodulin has been used for a number of these studies. Calmodulin is a good test system because it interacts with many proteins and structures of several calmodulin-peptide complexes have been solved(57). In two separate studies, Shifman and coworkers (58; 59) redesigned calmodulin to bind preferentially to a peptide construct of smooth muscle myosin light-chain kinase (smMLCK) over several other calmodulin partners. Their best design showed a 155-fold increase in binding specificity for smMLCK against a peptide designed to bind calmodulin(60). Taking into consideration all of the calmodulin partners, however, they only achieved a 5-fold increase in binding specificity. Only positive design for smMLCK was performed in their simulations.

Yosef et al. (61) followed up on this by further redesigning calmodulin, in this case to have an 880-fold increase in specificity for calmodulin-dependent protein kinase (CaMKII) over another calmodulin partner, calcineurin (CaN). The calmodulin/CaN interface is destabilized by three orders of magnitude in the design, providing the large change in specificity. The authors did not report the affinity of this design against the other calmodulin targets used by Shifman et al. (59). Green and coworkers (62) used a less-automated approach that filters designs based on the difference in energy between the cognate and non-cognate complexes. They were able to design a calmodulin/M13 kinase pair that had affinity comparable to wild-type and altered, but not "fully orthogonal" specificity. The above studies demonstrate that recovering wild-type binding affinity in redesigned complexes is feasible, but that new approaches for obtaining specificity are needed.

To obtain higher amounts of specificity in redesigned interfaces, a computational 'second-site suppressor' strategy was developed by Kortemme et al. (63). Their protocol begins with finding mutations on one partner that will eliminate binding, and then making mutations on the other partner that will compensate and restore binding. Affinity is recovered by optimizing the Rosetta predicted binding energy and specificity comes as a result of the disrupting mutation. The strategy was used to redesign the interface between bacterial DNase colicin E7 and its inhibitor protein Im7. Joachimiak et al. (64) extended this protocol by including rigid-body translations of one of the bound proteins while evaluating mutations. Rigid-body translations and rotations of Im7 helped identify a different set of mutants than those found previously. In both studies, the final designed cognate pairs had affinity comparable to wild-type, but specificity in only one direction.

Multistate design is well suited for the problem of designing specificity in protein-protein interfaces because it expressly considers multiple structures simultaneously. Harbury and coworkers (65) originally used the protocol to design sequences that would form coiled-coil heterodimers, the target complex, and not homodimers, the alternative states. Bolon and coworkers (66) were able to successfully design a specific, albeit less stable, heterodimer from the wild-type homodimer SspB when using positive and negative design. Multistate design was used by Ambroggio and Kuhlman (67) to design a sequence compatible with two different folds.

The sequence changes conformations between a zinc finger-like fold and a trimeric coiled-coil fold depending on whether Zn^{2+} is present in the solution. In their protocol sequences were optimized for scoring well on both structures. Most recently, Grigoryan et al. (68) used a multistate design approach to design specific peptide partners for 46 different human basic-region leucine zipper (bZIP) transcription factors. A considerable amount of knowledge of how bZIP coiled coils interact was included in their computational protocol. Nevertheless, their results are a great example of the promise of negative design for altering protein-protein interaction specificity.

We decided to use multistate design to redesign interactions in the small GTPase Ral signaling network. To date, most of the work done with multistate design has been on coiled-coils. The interfaces between Ral and its effectors are structurally more diverse than coiled coils and make for a good next test for the Rosetta energy function and multistate design. Chapter 4 describes a new implementation of multistate design in Rosetta and its application to the design of a specific Ral-effector complex. In this case, multistate design is used in the more intuitive sense. The off-target state structures are modeled explicitly and their energies are purposefully made worse while trying to make the target state energy better. By using a multistate design approach and modeling the off-target interactions as negative states, we expect to recover the known and find new specificity-changing mutations in the Ral signaling network.

References

1. Desjarlais, J. and Handel, T. (1995) De novo design of the hydrophobic cores of proteins. Protein Science **4**, 2006–2018. ISSN 1469-896X

2. Dahiyat, B., Sarisky, C., and Mayo, S. (1997) De novo protein design: towards fully automated sequence selection. Journal of molecular biology **273**, 789–796. ISSN 0022-2836

3. Dahiyat, B. and Mayo, S. (1997) De novo protein design: Fully automated sequence selection. Science **278**, 82. ISSN 0036-8075

4. Nauli, S., Kuhlman, B., Le Trong, I., Stenkamp, R., Teller, D., and Baker, D. (2002) Crystal structures and increased stabilization of the protein G variants with switched folding pathways NuG1 and NuG2. Protein science **11**, 2924–2931. ISSN 1469-896X

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

6. Lippow, S., Wittrup, K., and Tidor, B. (2007) Computational design of antibody-affinity improvement beyond in vivo maturation. Nature biotechnology **25**, 1171–1176. ISSN 1087-0156

7. Rothlisberger, D., Khersonsky, O., Wollacott, A., Jiang, L., DeChancie, J., Betker, J., Gallaher, J., Althoff, E., Zanghellini, A., Dym, O., et al. (2008) Kemp elimination catalysts by computational enzyme design. Nature **453**, 190–195. ISSN 0028-0836

8. Jiang, L., Althoff, E., Clemente, F., Doyle, L., Rothlisberger, D., Zanghellini, A., Gallaher, J., Betker, J., Tanaka, F., Barbas, C., et al. (2008) De novo computational design of retro-aldol enzymes. Science **319**, 1387. ISSN 0036-8075

9. Reynolds, K., Hanes, M., Thomson, J., Antczak, A., Berger, J., Bonomo, R., Kirsch, J., and Handel, T. (2008) Computational Redesign of the SHV-1 [beta]-Lactamase/[beta]-Lactamase Inhibitor Protein Interface. Journal of molecular biology **382**, 1265–1275. ISSN 0022-2836

10. Sammond, D., Eletr, Z., Purbeck, C., Kimple, R., Siderovski, D., and Kuhlman, B. (2007) Structure-based protocol for identifying mutations that enhance protein-

protein binding affinities. Journal of molecular biology **371**, 1392–1404. ISSN 0022-2836

11. Liu, S., Liu, S., Zhu, X., Liang, H., Cao, A., Chang, Z., and Lai, L. (2007) Nonnatural protein–protein interaction-pair design by key residues grafting. Proceedings of the National Academy of Sciences **104**, 5330

12. Huang, P., Love, J., and Mayo, S. (2007) A de novo designed protein–protein interface. Protein Science **16**, 2770–2774. ISSN 1469-896X

13. Jha, R., Leaver-Fay, A., Yin, S., Wu, Y., Butterfoss, G., Szyperski, T., Dokholyan, N., and Kuhlman, B. (2010) Computational design of a PAK1 binding protein. Journal of molecular biology **400**, 257–270. ISSN 0022-2836

14. Koide, A., Gilbreth, R., 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

15. Boder, E., Midelfort, K., and Wittrup, K. (2000) Directed evolution of antibody fragments with monovalent femtomolar antigen-binding affinity. Proceedings of the National Academy of Sciences of the United States of America **97**, 10701

16. Cochran, J., Kim, Y., Lippow, S., Rao, B., and Wittrup, K. (2006) Improved mutants from directed evolution are biased to orthologous substitutions. Protein Engineering Design and Selection **19**, 245. ISSN 1741-0126

17. Sidhu, S., Lowman, H., Cunningham, B., and Wells, J. (2000) Phage display for selection of novel binding peptides. Methods in enzymology **328**, 333. ISSN 0076-6879

18. Lipovsek, D. and Pluckthun, A. (2004) In-vitro protein evolution by ribosome display and mRNA display. Journal of immunological methods **290**, 51–67. ISSN 0022-1759

19. Gordon, D., Hom, G., Mayo, S., and Pierce, N. (2003) Exact rotamer optimization for protein design. Journal of computational chemistry **24**, 232–243. ISSN 0192-8651

20. Summa, C., Rosenblatt, M., Hong, J., Lear, J., and DeGrado, W. (2002) Computational de novo design, and characterization of an A2B2 diiron protein. Journal of molecular biology **321**, 923–938. ISSN 0022-2836

21. Kraemer-Pecore, C., Lecomte, J., and Desjarlais, J. (2003) A de novo redesign of the

WW domain. Protein science **12**, 2194–2205. ISSN 1469-896X

22. Sidhu, S. and Koide, S. (2007) Phage display for engineering and analyzing protein interaction interfaces. Current opinion in structural biology **17**, 481–487. ISSN 0959-440X

23. Hayes, R., Bentzien, J., Ary, M., Hwang, M., Jacinto, J., Vielmetter, J., Kundu, A., and Dahiyat, B. (2002) Combining computational and experimental screening for rapid optimization of protein properties. Proceedings of the National Academy of Sciences of the United States of America **99**, 15926

24. Rohl, C., Strauss, C., Misura, K., and Baker, D. (2004) Protein structure prediction using Rosetta. Methods in enzymology **383**, 66–93. ISSN 0076-6879

25. Leaver-Fay, A., Tyka, M., Lewis, S., Lange, O., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P., Smith, C., Sheffler, W., et al. (2011) ROSETTA3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. Methods in enzymology **487**, 545. ISSN 1557-7988

26. 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. Journal of molecular biology **332**, 449–460. ISSN 0022-2836

27. Hu, X., Wang, H., Ke, H., and Kuhlman, B. (2007) High-resolution design of a protein loop. Proceedings of the National Academy of Sciences **104**, 17668

28. Gray, J., Moughon, S., Wang, C., Schueler-Furman, O., Kuhlman, B., Rohl, C., and Baker, D. (2003) Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. Journal of Molecular Biology **331**, 281–299. ISSN 0022-2836

29. Meiler, J. and Baker, D. (2006) ROSETTALIGAND: Protein–small molecule docking with full side-chain flexibility. PROTEINS: Structure, Function, and Bioinformatics **65**, 538–548. ISSN 1097-0134

30. Davis, I. and Baker, D. (2009) RosettaLigand docking with full ligand and receptor flexibility. Journal of molecular biology **385**, 381–392. ISSN 0022-2836

31. Davis, I., Raha, K., Head, M., and Baker, D. (2009) Blind docking of pharmaceutically

relevant compounds using RosettaLigand. Protein Science **18**, 1998–2002. ISSN 1469-896X

32. Kellogg, E., Leaver-Fay, A., and Baker, D. (2011) Role of conformational sampling in computing mutation-induced changes in protein structure and stability. Proteins: Structure, Function, and Bioinformatics ISSN 1097-0134

33. Das, R. and Baker, D. (2007) Automated de novo prediction of native-like RNA tertiary structures. Proceedings of the National Academy of Sciences **104**, 14664

34. Das, R., Kudaravalli, M., Jonikas, M., Laederach, A., Fong, R., Schwans, J., Baker, D., Piccirilli, J., Altman, R., and Herschlag, D. (2008) Structural inference of native and partially folded RNA by high-throughput contact mapping. Proceedings of the National Academy of Sciences **105**, 4144

35. Sheffler, W. and Baker, D. (2009) RosettaHoles: Rapid assessment of protein core packing for structure prediction, refinement, design, and validation. Protein Science **18**, 229–239. ISSN 1469-896X

36. Sheffler, W. and Baker, D. (2010) RosettaHoles2: A volumetric packing measure for protein structure refinement and validation. Protein Science **19**, 1991–1995. ISSN 1469-896X

37. DiMaio, F., Tyka, M., Baker, M., Chiu, W., and Baker, D. (2009) Refinement of protein structures into low-resolution density maps using rosetta. Journal of molecular biology **392**, 181–190. ISSN 0022-2836

38. Shen, Y., Lange, O., Delaglio, F., Rossi, P., Aramini, J., Liu, G., Eletsky, A., Wu, Y., Singarapu, K., Lemak, A., et al. (2008) Consistent blind protein structure generation from NMR chemical shift data. Proceedings of the National Academy of Sciences **105**, 4685

39. Raman, S., Lange, O., Rossi, P., Tyka, M., Wang, X., Aramini, J., Liu, G., Ramelot, T., Eletsky, A., Szyperski, T., et al. (2010) NMR structure determination for larger proteins using backbone-only data. Science **327**, 1014. ISSN 0036-8075

40. Dahiyat, B. and Mayo, S. (1996) Protein design automation. Protein Science **5**, 895–903. ISSN 1469-896X

41. Desmet, J., De Maeyer, M., and Lasters, I. (1997) Theoretical and algorithmical op-

timization of the dead-end elimination theorem. In Pac Symp Biocomput, 122–33. Citeseer

42. Shah, P., Hom, G., Ross, S., Lassila, J., Crowhurst, K., and Mayo, S. (2007) Full-sequence computational design and solution structure of a thermostable protein variant. Journal of molecular biology **372**, 1–6. ISSN 0022-2836

43. Desmet, J., Spriet, J., and Lasters, I. (2002) Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins: Structure, Function, and Bioinformatics **48**, 31–43. ISSN 1097-0134

44. Pokala, N. and Handel, T. (2005) Energy functions for protein design: adjustment with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. Journal of molecular biology **347**, 203–227. ISSN 0022-2836

45. Jorgensen, W., Maxwell, D., and Tirado-Rives, J. (1996) Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. Journal of the American Chemical Society **118**, 11225–11236. ISSN 0002-7863

46. Eisenberg, D. and McLachlan, A. (1986) Solvation energy in protein folding and binding. Nature

47. Chowdry, A., Reynolds, K., Hanes, M., Voorhies, M., Pokala, N., and Handel, T. (2007) An object-oriented library for computational protein design. Journal of Computational Chemistry **28**, 2378–2388. ISSN 1096-987X

48. Janin, J., Miller, S., and Chothia, C. (1988) Surface, subunit interfaces and interior of oligomeric proteins. Journal of molecular biology **204**, 155–164. ISSN 0022-2836

49. Chiti, F., Stefani, M., Taddei, N., Ramponi, G., and Dobson, C. (2003) Rationalization of the effects of mutations on peptide andprotein aggregation rates. Nature **424**, 805–808. ISSN 0028-0836

50. Lijnzaad, P. and Argos, P. (1997) Hydrophobic patches on protein subunit interfaces: characteristics and prediction. Proteins: Structure, Function, and Bioinformatics **28**, 333–343. ISSN 1097-0134

51. Schindler, T., Perl, D., Graumann, P., Sieber, V., Marahiel, M., and Schmid, F. (1998) Surface-exposed phenylalanines in the RNP1/RNP2 motif stabilize the cold-shock

protein CspB from Bacillus subtilis. Proteins: Structure, Function, and Bioinformatics **30**, 401–406. ISSN 1097-0134

52. Poso, D., Sessions, R., Lorch, M., and Clarke, A. (2000) Progressive stabilization of intermediate and transition states in protein folding reactions by introducing surface hydrophobic residues. Journal of Biological Chemistry **275**, 35723. ISSN 0021-9258

53. Prasad Bahadur, R., Chakrabarti, P., Rodier, F., and Janin, J. (2004) A dissection of specific and non-specific protein-protein interfaces. Journal of molecular biology **336**, 943–955. ISSN 0022-2836

54. Dahiyat, B., Benjamin Gordon, D., and Mayo, S. (1997) Automated design of the surface positions of protein helices. Protein Science **6**, 1333–1337. ISSN 1469-896X

55. Wernisch, L., Hery, S., and Wodak, S. (2000) Automatic protein design with all atom force-fields by exact and heuristic optimization1. Journal of Molecular Biology **301**, 713–736. ISSN 0022-2836

56. Kuhlman, B. and Baker, D. (2000) Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America **97**, 10383

57. Crivici, A. and Ikura, M. (1995) Molecular and structural basis of target recognition by calmodulin. Annual review of biophysics and biomolecular structure **24**, 85–116. ISSN 1056-8700

58. Shifman, J. and Mayo, S. (2002) Modulating calmodulin binding specificity through computational protein design. Journal of molecular biology **323**, 417–423. ISSN 0022-2836

59. Shifman, J. and Mayo, S. (2003) Exploring the origins of binding specificity through the computational redesign of calmodulin. Proceedings of the National Academy of Sciences of the United States of America **100**, 13274

60. O'Neil, K., Wolfe, H., Erickson-Viitanen, S., and DeGrado, W. (1987) Fluorescence properties of calmodulin-binding peptides reflect alpha-helical periodicity. Science **236**, 1454. ISSN 0036-8075

61. Yosef, E., Politi, R., Choi, M., and Shifman, J. (2009) Computational design of calmodulin mutants with up to 900-fold increase in binding specificity. Journal of molecular

biology **385**, 1470–1480. ISSN 0022-2836

62. Green, D., Dennis, A., Fam, P., Tidor, B., and Jasanoff, A. (2006) Rational Design of New Binding Specificity by Simultaneous Mutagenesis of Calmodulin and a Target Peptide— . Biochemistry **45**, 12547–12559

63. Kortemme, T. and Baker, D. (2004) Computational design of protein-protein interactions. Current opinion in chemical biology **8**, 91–97. ISSN 1367-5931

64. Joachimiak, L., Kortemme, T., Stoddard, B., and Baker, D. (2006) Computational design of a new hydrogen bond network and at least a 300-fold specificity switch at a protein-protein interface. Journal of molecular biology **361**, 195–208. ISSN 0022-2836

65. Havranek, J. and Harbury, P. (2002) Automated design of specificity in molecular recognition. Nature Structural & Molecular Biology **10**, 45–52

66. Bolon, D., Grant, R., Baker, T., and Sauer, R. (2005) Specificity versus stability in computational protein design. Proceedings of the National Academy of Sciences of the United States of America **102**, 12724

67. Ambroggio, X. 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

68. Grigoryan, G., Reinke, A., and Keating, A. (2009) Design of protein-interaction specificity gives selective bZIP-binding peptides. Nature **458**, 859–864. ISSN 0028-0836

Chapter 2

Protein design with Rosetta

All protein design programs have two main components: a search algorithm which traverses the space of all sequences and generates candidate sequences, and a scoring, or energy, function, which determines how well a candidate sequence fits for the desired structure. The details of both components, including how the energy function is trained, are provided below. The rest of the chapter describes how a typical design simulation works and looks at some characteristics of proteins designed with Rosetta.

2.1 Rosetta energy function

The scoring function in Rosetta contains a combination of terms that consider van der Waals interactions, solvation, hydrogen bonding, electrostatics, and sterics (Figure 2.1). Some of the terms are physically-based, such as the van der Waals energy, while the remaining terms are knowledge-based, derived from statistics gathered from the Protein Data Bank (PDB) (1). The expanded form of each energy term can be found in the Appendix. In a design run, all of the energy terms are evaluated for every candidate sequence and their sum becomes the final score for that sequence.

$$\begin{aligned} E_{protein} = \quad & W_{lj\,atr}E_{lj\,atr} + W_{lj\,rep}E_{lj\,rep} + W_{hbond}E_{hbond} + W_{solvation}E_{solvation} + W_{aa}E_{aa} + \\ & W_{pair}E_{pair} + W_{rama}E_{rama} + W_{rotamer}E_{rotamer} - W_{reference}E_{reference} \end{aligned}$$

Figure 2.1: Rosetta energy function. The current form of the Rosetta energy function is shown.

2.1.1 van der Waals energy

Rosetta uses the standard 12-6 Lennard-Jones potential to model van der Waals interactions. The Lennard-Jones energy favors having atoms close to each other, but not so close that they clash. This term is important for ensuring well-packed, hydrophobic cores in designed proteins. In some applications, Rosetta uses a dampened Lennard-Jones potential, in which the exponential component of the potential is linearized. This modification helps alleviate large clashes that can results when designing with a fixed-backbone scaffold and discrete side-chain conformations(2; 3). Well depths for the potential are taken from the CHARMM19 parameter set(4; 5) and atom radii are taken from (6).

2.1.2 solvation energy

The solvation energy of a protein is the change in free energy observed when transferring the protein from a vacuum to solvent water. Generally speaking, an accurate solvation energy term favors the burial of hydrophobic surface area ensuring a mostly hydrophobic protein core. While burial of polar surface area is penalized, favorable electrostatics interactions can overcome the unfavorable energy of desolvating a polar group. The most precise way of calculating solvation free energy is through molecular dynamics (MD) simulations using explicit water. However, MD simulations take thousands of CPU hours for just one structure, thereby making it impossible to do design with. Instead, all protein design program use implicit solvent models, also called continuum models because they treat water as a continuous medium. The first implicit solvent models estimated the solvation energy by combining the solvent-accessible surface area (SASA) of every atom with a solvation parameter for that atom(7). The atomic solvation parameters represent the amount of energy per unit area a given atom type contributes to the free energy of solvation. More recently, continuum electrostatics models such as Poisson-Boltzmann (PB) and Generalized Born (GB), an approximation to the PB equation, have been incorporated into protein design programs(8; 9; 10). These methods only model the electrostatic contribution to solvation free energy, and therefore sometimes are augmented with a surface area term to model the nonpolar (or the solvent entropy) contribution to solvation free energy(11). Both of these methods are still very demanding computationally.

Furthermore, solving the PB equation analytically for proteins is not possible, so numerical methods must be used to obtain solutions.

Instead of using one of the above implicit solvation models, Rosetta uses the Lazaridis-Karplus, solvent-exclusion solvation model, also called EEF1(12). EEF1 estimates the solvation free energy by taking the solvation free energy of a group i in a fully solvent-exposed reference state and subtracting some energy to account for neighboring desolvating groups. The total solvation free energy for the protein is then obtained by summing over all groups in the protein. How much energy is subtracted from the reference state energy is determined by looking at how much volume is excluded by each neighbor j around group i. The model is parameterized so that the solvation energy of deeply buried groups is zero. The Lazaridis-Karplus method for calculating solvation energy is very fast because it does not require solving the Poisson-Boltzmann equation or calculation of the SASA.

2.1.3 electrostatics energy

Electrostatics are modeled in Rosetta using two energy terms: an orientation-dependent hydrogen-bond potential(13) and a residue-pair potential(14). Hydrogen bonds are important for the stabilization of secondary and tertiary structure in proteins and provide specificity to protein-protein interactions. The pair energy term captures electrostatic interactions not scored by the EEF1 solvation energy term. Both of these terms are knowledge-based terms derived from statistics of structures deposited in the PDB. The hbond potential is a linear combination of a distance-dependent energy term for the hydrogen-acceptor atom distance, and three angular-dependent energy terms. The angular-dependent energy terms capture preferences for the angle at the hydrogen bond, the angle at the acceptor atom, and the acceptor/acceptor-base dihedral angle (for sp2 hybridized acceptors). The pair energy term is based on the probability of seeing two amino acids close together after adjusting for the probability of seeing those two amino acids in the given environment. The pair residue potential is only evaluated for polar residues.

2.1.4 torsional energy

Bonded atom interactions in Rosetta are evaluated by the ramachandran torsional energy term and a rotamer self-energy term. The ramachandran energy is the inverse log of the probability of seeing specific $\phi$and ψbackbone angles given a particular amino acid and secondary structure (helix, strand, or loop). The rotamer self-energy term measures the internal energy of a side chain. More specifically, it measures the probability of an amino acid type in a specific rotamer given specific $\phi$and ψbackbone torsion angles, adjusted for amino acid frequencies in the PDB. Rosetta uses the rotamer probabilities derived by Dunbrack and Cohen(15).

2.1.5 reference energy

The reference energy term in the energy function serves as a pseudo unfolded state energy and as a way to bias surface amino acid composition. The reference energies were originally parameterized to reproduce native protein sequences(16). The reference energies, which favor the placement of polar residues, offset the solvation energy term which, in general, favors the design of hydrophobic residues.

2.2 Search algorithm

All protein design programs use a search function of some sort. A search function is necessary, as exhaustively testing all possible sequences for a given fold is computationally prohibitive. Search functions fall into two categories: stochastic and deterministic. Stochastic search functions include Monte Carlo (MC) and genetic algorithms, while deterministic search algorithms include dead-end elimination (DEE) and self-consistent mean field algorithms. There is a tradeoff between speed and finding the optimal solution that has to be considered when selecting a search algorithm(17). MC is significantly faster than the DEE method, but does not guarantee finding the global minimum. If it converges, DEE is guaranteed to find the global minimum(18). Other search algorithms including branch-and-bound(19), integer linear programming(20), and tree decomposition methods (21) have been described.

Rosetta uses a Metropolis Monte Carlo(22) simulated annealing(23) search algorithm to explore sequence space. During a simulation, a random position is selected for substitution. The change in energy for that substitution is determined using the energy function described above. Changes are accepted or rejected based on the Metropolis criterion. If the change in energy is favorable, the substitution is accepted. If the change in energy is unfavorable, it is accepted with probability exp(ΔE/kT) where E is the energy, T is the temperature, and k is Boltzmann's constant. The Metropolis condition ensures that more time is spent evaluating low-energy states. To sample a larger amount of conformation space, a simulated annealing approach is used in conjunction with MC. In simulated annealing, the temperature used for the Metropolis criterion is started at a high value, thereby increasing the probability with which substitutions that increase the energy are accepted. During the optimization, the temperature is gradually decreased, decreasing the probability of accepting unfavorable substitutions. At the end of simulated annealing, only substitutions which lower the energy are accepted. By starting off with high temperatures, simulated annealing algorithms are less likely to get trapped in local minima.

2.3 Fixed-backbone design

Protein design simulations take as input a set of backbone coordinates and output a structure with a sequence that has low energy for that structure. A typical design simulation proceeds as follows. A PDB file for an existing protein structure is given to Rosetta with a specification of which residues should be allowed to vary. Rosetta removes all of the side chains from the input structure and builds conformations, or rotamers, of all possible side chains for every designable position. By default, all positions in the input structure are allowed to change. Disallowing certain positions from being changed (e.g. active-site residues in an enzyme) may be desirable for some applications. The Dunbrack backbone-dependent rotamer library(24), and ideal bond lengths and angles are used when making rotamers. Using a rotamer library to create side chain conformations significantly reduces the size of conformation space that needs to be searched. The simulation starts by first calculating energies between all pairs of rotamers using the energy function described above. Most of the time in a design simulation is spent in calculating these rotamer pair energies (RPE), which are stored in a large table in memory. After calculation of the RPEs, side chain optimization, or packing, can start. Using the table of RPEs and the search algorithm described above, Rosetta simultaneously optimizes the identity and conformation of all positions of the backbone. Millions of substitutions are considered and evaluated with the Metropolis criterion. When the simulated annealing finishes, the side chain assignment that had the lowest energy is recovered and that structure is output.

Native sequence recovery represents one way to train and/or test protein design energy functions. In these tests, Rosetta is used to redesign an existing protein and how much of the native sequence is recovered is calculated. Table 2.1 shows the native sequence recovery broken down by amino acid type for proteins redesigned with the current Rosetta energy function. Rosetta redesigned proteins have 49% core and 33% overall identity to the native sequences. Core sequence recovery is higher than surface recovery because neighboring residues in the core only allow for certain amino acid types to fit. The Rosetta energy function has been trained to reproduce native protein sequences. Thus, the sequence recoveries obtained by Rosetta are high compared to other protein design programs. Although the recoveries are good, the amino acid composition of the designed proteins differs somewhat from the composition seen in native proteins. For example, Rosetta designs twice as many tryptophanes and less than half as many prolines as the number present in native proteins. A measure of this difference between the two distributions will be given later.

2.4 Energy function weight optimization

The weights on the terms in the Rosetta energy function can be optimized to create energy functions that perform better in certain applications. For example, there is considerable interest in obtaining an energy function that can reliably discriminate the native structure of a protein from thousands of low energy decoys. Such an energy function would be extremely useful in ab initio structure prediction. Alternatively, one may want an energy function that can correctly predict the experimental change in free energy for mutations to a protein. The weights currently in Rosetta were parameterized so that redesigned proteins would reproduce

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
LEU	66	122	92	0.54	180	378	454	0.48	18	54	174	0.33
GLY	56	69	66	0.81	302	400	368	0.75	164	210	205	0.78
ASP	10	20	29	0.5	80	268	354	0.3	49	167	221	0.29
SER	20	37	106	0.54	79	258	348	0.31	25	119	128	0.21
GLU	2	7	16	0.29	54	289	308	0.19	31	180	158	0.17
PHE	37	60	69	0.62	98	193	302	0.51	6	23	68	0.26
LYS	2	11	9	0.18	62	325	285	0.19	30	193	148	0.16
ARG	4	14	20	0.29	37	185	262	0.2	10	85	76	0.12
ALA	57	103	94	0.55	118	385	259	0.31	9	144	30	0.06
ILE	41	83	70	0.49	105	242	231	0.43	7	33	39	0.21
THR	19	41	62	0.46	61	291	225	0.21	30	145	110	0.21
TYR	9	40	27	0.22	39	158	224	0.25	6	23	98	0.26
VAL	47	98	67	0.48	107	308	194	0.35	8	60	37	0.13
HIS	10	18	41	0.56	30	118	176	0.25	5	37	32	0.14
GLN	3	17	6	0.18	18	186	172	0.1	10	94	110	0.11
ASN	3	14	11	0.21	33	206	158	0.16	20	110	120	0.18
TRP	10	23	21	0.43	32	76	145	0.42	4	12	56	0.33
PRO	6	18	7	0.33	61	229	77	0.27	29	144	40	0.2
MET	10	31	25	0.32	20	90	70	0.22	1	20	5	0.05
CYS	0	12	0	0	0	27	0	0	0	2	0	0
Total	412	838		0.492	1516	4612		0.329	462	1855		0.249

Table 2.1: Native sequence recovery of Rosetta redesigns. Native sequence recoveries by amino acid type are reported for a set of 32 proteins redesigned with Rosetta.

native protein sequences. This result was obtained by maximizing the Boltzmann probability of the native amino acid over all positions in a set of 30 proteins. Sharabi et al. (25) recently described an optimized energy function for protein-protein interface design.

A new, flexible, weight-fitting protocol was recently implemented in Rosetta (Andrew Leaver-Fay, in preparation). The protocol works by searching the space of all weights for a combination that maximizes the fitness of the selected objective function(s). For example, the protocol can be used to train weights for highest native sequence recovery. In this case, the protocol starts by calculating the unweighted energies of every possible rotamer at every position in a set of proteins. The positions surrounding the one being considered are held fixed at their native amino acid. The dot product of a candidate set of weights with the unweighted energies is used to obtain a fitness. In the case of native sequence recovery, the fitness is calculated according to the equation below:

$$Fitness = \sum_{proteins} \sum_{positions} -\ln \left[\frac{e^{-E(aanat)}}{\sum_{aa,i} e^{-E(aai)}} \right]$$

where E(aanat) is the energy of the native amino acid at a position and the denominator is the partition function for all 20 amino acids at that position. Instead of multiplying many small probabilities, the sum of the inverse log of the probabilities is minimized. If the protocol is instead used to optimize for predicting changes in free energy, the fitness is the sum of squared differences between the predicted and experimental $\Delta\Delta$G for all mutants. Candidate weight sets are obtained by using particle swarm optimization (PSO) to search weight space. The best weight set found by PSO in each round is minimized using conjugate gradient-based minimization. If optimizing for native sequence recovery, the minimized weight set is then used to do full protein redesigns. This complete redesign step ensures that weights optimized in a fixed environment are still good for whole protein redesigns. More details of the protocol are provided in the Methods section of chapter 3.

Added later to the weight fitting protocol was the ability to optimize the weights so that redesigns had native-like amino acid (AA) composition. Unlike native amino acid probability or $\Delta\Delta$G prediction, energy function weights cannot be optimized directly for AA composition. Instead, AA composition is optimized for after the complete redesign step of the protocol, by adjusting the reference energies up or down depending on whether that residue type is designed too much or too little. The cross entropy, a measure of the difference between two distributions, between the designed and native amino acid distributions was used to determine if AA composition was becoming more native-like. Only weight sets that increased the overall sequence recovery and decreased the cross entropy were accepted.

The ideal energy function for protein design would produce native-like designed proteins and be accurate in predicting changes in stability for point mutants. Using the weight-fitting protocol described above, a great amount of effort was spent in optimizing weights and different energy term combinations during the development of the score term described in chapter 3. Energy function optimization is a very hard problem because of the vast size of weight space, the number of options present in Rosetta, and the variety of metrics which must be examined for each energy function. The first generations of weight optimization runs trained for overall native sequence recovery. The only metrics considered were core and overall native sequence recovery. Shortly later, the goal was changed to optimizing weights to do well at sequence recovery and $\Delta\Delta$G prediction. This added the $\Delta\Delta$G correlation coefficient to the list of metrics that had to be considered for a set of weights. Many different options in Rosetta were tested to see what effect they had on the metrics: extra rotamers at surface positions, extra rotamers throughout, multiple packing runs, inclusion of crystal structure rotamers, and modifications to the pair energy term and solvation term. Over time, the list of metrics expanded to include total hydrophobic surface area, percent of residues on the surface that were hydrophobic, and AA composition. In the end, because of difficulties in accurately predicting $\Delta\Delta$G, energy function weights were only optimized for native sequence recovery and AA composition.

Training weights to accurately predict $\Delta\Delta$G proved considerably more difficult than expected. We found that if weights were trained to reproduce changes in stability, designing proteins with this energy function resulted in proteins that were composed almost entirely of hydrophobic residues. This result makes sense because the folded state is almost always desolvated to some extent compared to the unfolded state. Because solvation energy represents one of the biggest contributions to total energy, hydrophobic residues are favored on the surface because of the favorable energy of desolvation. We also found many mutations were predicted to be significantly more destabilizing than they were in reality because of high Lennard Jones repulsive energies. This result made us realize that predicting $\Delta\Delta$G depends greatly on how the mutant structures are created. In fact, a study describing different methods of creating mutant structures and what effect that had on $\Delta\Delta$G prediction(26) was published around the same time we were experimenting with $\Delta\Delta$G prediction. The authors found that allowing more conformational freedom to relax away clashes during mutant structure creation greatly improves the correlations that are obtained. Instead of trying to optimize weights for $\Delta\Delta$G prediction on a set of poorly made mutant structures, we elected to optimize energy functions only for native sequence recovery and AA composition and then test $\Delta\Delta$G prediction accuracy using the protocol described by Kellogg et al.(26).

References

1. Berman, H., Henrick, K., Nakamura, H., and Markley, J. (2006) The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data. Nucleic acids research ISSN 0305-1048

2. Rohl, C., Strauss, C., Misura, K., and Baker, D. (2004) Protein structure prediction using Rosetta. Methods in enzymology **383**, 66–93. ISSN 0076-6879

3. Grigoryan, G., Ochoa, A., and Keating, A. (2007) Computing van der Waals energies in the context of the rotamer approximation. Proteins: Structure, Function, and Bioinformatics **68**, 863–878. ISSN 1097-0134

4. Brooks, B., Bruccoleri, R., Olafson, B., et al. (1983) CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. Journal of computational chemistry **4**, 187–217. ISSN 1096-987X

5. MacKerell Jr, A., Bashford, D., Bellott, M., Dunbrack Jr, R., Evanseck, J., Field, M., Fischer, S., Gao, J., Guo, H., Ha, S., et al. (1998) All-atom empirical potential for molecular modeling and dynamics studies of proteins. The Journal of Physical Chemistry B **102**, 3586–3616. ISSN 1520-6106

6. Neria, E., Fischer, S., and Karplus, M. (1996) Simulation of activation free energies in molecular systems. The Journal of chemical physics **105**, 1902

7. Eisenberg, D. and McLachlan, A. (1986) Solvation energy in protein folding and binding. Nature

8. Marshall, S., Vizcarra, C., and Mayo, S. (2005) One-and two-body decomposable Poisson-Boltzmann methods for protein design calculations. Protein science **14**, 1293–1304. ISSN 1469-896X

9. Vizcarra, C., Zhang, N., Marshall, S., Wingreen, N., Zeng, C., and Mayo, S. (2008) An improved pairwise decomposable finite-difference Poisson–Boltzmann method for computational protein design. Journal of Computational Chemistry **29**, 1153–1162. ISSN 1096-987X

10. Pokala, N. and Handel, T. (2004) Energy functions for protein design I: Efficient and

accurate continuum electrostatics and solvation. Protein science **13**, 925–936. ISSN 1469-896X

11. Dzubiella, J., Swanson, J., and McCammon, J. (2006) Coupling nonpolar and polar solvation free energies in implicit solvent models. The Journal of chemical physics **124**, 084905

12. Lazaridis, T. and Karplus, M. (1999) Effective energy function for proteins in solution. Proteins: Structure, Function, and Bioinformatics **35**, 133–152. ISSN 1097-0134

13. Kortemme, T., Morozov, A., and Baker, D. (2003) An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein-protein complexes. Journal of molecular biology **326**, 1239–1259. ISSN 0022-2836

14. Simons, K., Ruczinski, I., Kooperberg, C., Fox, B., Bystroff, C., and Baker, D. (1999) Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. Proteins: Structure, Function, and Bioinformatics **34**, 82–95. ISSN 1097-0134

15. Dunbrack Jr, R. and Cohen, F. (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Science **6**, 1661–1681. ISSN 1469-896X

16. Kuhlman, B. and Baker, D. (2000) Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America **97**, 10383

17. Voigt, C., Gordon, D., and Mayo, S. (2000) Trading accuracy for speed: a quantitative comparison of search algorithms in protein sequence design1. Journal of Molecular Biology **299**, 789–803. ISSN 0022-2836

18. Desmet, J., Maeyer, M., Hazes, B., and Lasters, I. (1992) The dead-end elimination theorem and its use in protein side-chain positioning. Nature **356**, 539–542. ISSN 0028-0836

19. Gordon, D. and Mayo, S. (1999) Branch-and-terminate: A combinatorial optimization algorithm for protein design. Structure **7**, 1089–1098. ISSN 0969-2126

20. Eriksson, O., Zhou, Y., and Elofsson, A. (2001) Side chain-positioning as an integer programming problem. In Proceedings of the First International Workshop on Algorithms

in Bioinformatics, WABI '01, 128–141. Springer-Verlag. ISBN 3-540-42516-0

21. Leaver-Fay, A., Kuhlman, B., and Snoeyink, J. (2005) An adaptive dynamic programming algorithm for the side chain placement problem. In Pacific Symposium on Biocomputing, volume 10, 16–27

22. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., et al. (1953) Equation of state calculations by fast computing machines. The journal of chemical physics **21**, 1087. ISSN 0021-9606

23. Kirkpatrick, S., Jr., D., and Vecchi, M. (1983) Optimization by simmulated annealing. science **220**, 671–680. ISSN 1095-9203

24. Dunbrack, R. and Karplus, M. (1994) Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. Nature Structural & Molecular Biology **1**, 334–340

25. Sharabi, O., Yanover, C., Dekel, A., and Shifman, J. (2011) Optimizing energy functions for protein–protein interface design. Journal of Computational Chemistry ISSN 1096-987X

26. Kellogg, E., Leaver-Fay, A., and Baker, D. (2011) Role of conformational sampling in computing mutation-induced changes in protein structure and stability. Proteins: Structure, Function, and Bioinformatics ISSN 1097-0134

Chapter 3

Computational Protein Design with Explicit Consideration of Surface Hydrophobic Patches

3.1 Abstract

De novo protein design requires the identification of amino acid sequences that favor the target folded conformation and are soluble in water. One strategy for promoting solubility is to disallow hydrophobic residues on the protein surface during design. However, naturally occurring proteins often have hydrophobic amino acids on their surface that contribute to protein stability via the partial burial of hydrophobic surface area or play a key role in the formation of protein-protein interactions. A less restrictive approach for surface design that is used by the modeling program Rosetta is to parameterize the energy function so that the number of hydrophobic amino acids designed on the protein surface is similar to what is observed in naturally occurring monomeric proteins. Previous studies with Rosetta have shown that this limits surface hydrophobics to the naturally occurring frequency (~28%) but that it does not prevent the formation of hydrophobic patches that are considerably larger than those observed in naturally occurring proteins. Here, we describe a new score term that explicitly detects and penalizes the formation of hydrophobic patches during computational protein design. With the new term we are able to design protein surfaces that include hydrophobic amino acids at naturally occurring frequencies, but do not have large hydrophobic patches. By adjusting the strength of the new score term the emphasis of surface redesigns can be switched

between maintaining solubility and maximizing folding free energy.

3.2 Introduction

In addition to adopting a stable folded conformation, many proteins must be soluble in water in order to perform their biological function. This requirement constrains protein evolution, as sequences that are optimized only for folding free energy may not be optimized for solubility, and vice a versa (1). Folding free energy is equal to the difference in free energy of the folded and unfolded states. In the unfolded state proteins adopt an ensemble of conformations that are less compact and more solvated than folded protein. In the folded state proteins adopt a unique set of structures with desolvated cores. The desolvation of hydrophobic amino acids is a primary driving force for protein folding, and increasing the difference in buried hydrophobic surface area between the folded and unfolded state will often stabilize proteins (2; 3). Even partially buried hydrophobic amino acids on the surface of a protein can dramatically boost protein stability. For example, introducing a cluster of four hydrophobic amino acids on to the surface of procarboxypeptidase A2 stabilizes the protein by more than 5 kcal/mol (4).

Protein solubility is determined by many factors, including net electrostatic charge(5), folding free energy, and the amount of exposed hydrophobic surface area in the folded state. A comparison of the surfaces of proteins that are monomeric and water-soluble with the surfaces of proteins that form obligate oligomers provides an indication of what surface features prevent association. The most striking difference between the two sets of proteins is the amount of exposed hydrophobic surface area. Jones and Thornton(6) found that the interfaces of oligomeric proteins are more hydrophobic than the interfaces of other protein-protein complexes and of non-interface surfaces. In the set of oligomeric proteins examined by Janin et al.(7), the average amount of non-polar surface area at oligomer interfaces is 8% greater than the amount seen in monomeric protein surfaces. In agreement with these findings, Chiti et al.(8) found that the rate of aggregation of proteins and peptides increases as the amount of exposed hydrophobic surface area increases. Because exposed hydrophobic surface area can be so detrimental to protein fitness, computer-based methods for protein design must take this in to account when designing sequences for the surfaces of proteins.

Protein design programs contain two key components, a score function for evaluating the fitness of an amino acid sequence for a given target structure and an optimization procedure for identifying low scoring sequences. Several studies have shown that if the score function is constructed to model only folding free energy then the surfaces of the designed proteins do not resemble the surfaces of naturally occurring proteins (9; 10). In these cases, structural models of the unfolded and folded state are used to calculate the free energy difference between the folded and unfolded state. Pokala and Handel observed protein surfaces dominated by hydrophobic amino acids because their model emphasizes the importance of the hydrophobic effect in driving protein folding and surface residues were predicted to bury more hydrophobic surface area in the folded state than in the unfolded state. This problem can be alleviated by explicitly disallowing hydrophobic amino acids at all surface positions(11; 12), but this solution is not ideal because partially exposed hydrophobics can contribute significantly to protein stability and surface hydrophobic amino acids are often important for protein function. A more permissive approach is to allow surface hydrophobics, but modify the score function so that it disfavors surfaces that are likely to promote aggregation.

A variety of scoring schemes have previously been used to control the placement of surface hydrophobic amino acids in design simulations(10; 12; 13; 14; 15). In many cases, the end result is that the score function represents more than folding free energy. This outcome can be achieved by including separate scoring terms for aggregation propensity and folding energy, or by creating a single score that implicitly reflects both criteria. Explicit scoring terms that have been used include penalties for exposed hydrophobic surface area(13; 14) and negative design against sequences with favorable energy in a low dielectric environment(16). More implicit strategies include constraining amino acid composition to match naturally occurring proteins(17; 18; 19) and up weighting the strength of hydrogen bonds and electrostatic interactions on protein surfaces(15).

The computer program we use for protein design studies, Rosetta, produces a single score that depends on both protein stability and aggregation propensity. Instead of parameterizing the Rosetta score function to predict folding free energies, Rosetta was trained to recapitulate naturally occurring sequences when performing design simulations with naturally occurring protein backbones. Critical to this recapitulation is the inclusion of reference energies for each amino acid type. These energies are subtracted from the total energy of the protein, and their main function is to control amino acid composition during design simulations. By setting the reference values to favor the correct ratio of polar and hydrophobic amino acids in protein sequences, the design of polar protein surfaces is implicitly favored. However, we have observed that in many cases the Rosetta scoring function fails to prevent large hydrophobic clusters on the surface of proteins, even though the overall amino acid composition of the protein surface is not significantly different from other soluble proteins. This result reflects the favorable energetics of placing similar types of amino acids near each other. Pokala and Handel(10) used an alternative strategy for setting amino acid reference values. They used their force field to calculate the average energy of each amino acid type on the surface for a large set of proteins and used these values as reference values. This strategy reduced the formation of hydrophobic clusters, but it also resulted in an underrepresentation of leucine, isoleucine, valine, phenylalanine and tyrosine on protein surfaces.

Here we describe the implementation of a new, non-pairwise-decomposable scoring term (called hpatch) that penalizes the formation of hydrophobic patches on the surfaces of designed proteins. Unlike previously described scoring terms that disfavor aggregation, the new term explicitly disfavors large patches, rather than the total amount of exposed hydrophobic surface area. We find that redesigning proteins with the hpatch score term reduces the size of hydrophobic patches to levels seen in native proteins, but preserves a natural ratio of hydrophobic and polar amino acids on the surface. We parameterize a new Rosetta scoring function using the hpatch score and assess its performance on native sequence recovery and ability to predict changes in free energy for characterized protein mutants. We find that we are able to create a single score function that performs well in sequence recovery tests when the hpatch score is included, and additionally performs well in predicting changes in protein stability if the weight on the hpatch score is set to zero.

3.3 Methods

3.3.1 Rosetta energy function

Rosetta uses a Monte Carlo search algorithm with simulated annealing to find low scoring sequences for a target backbone(19; 20; 21; 22). The energy function uses the 12-6 Lennard-Jones potential, the Lazardis-Karplus implicit solvation model(23), a statistics-based electrostatics term(24), an explicit hydrogen-bond potential(25), a side-chain rotamer preference term, a knowledge-based backbone torsional term, and reference energies that are assigned to each amino acid type. Side chain conformations are restricted to those found in the Dunbrack backbone-dependent rotamer library(26).

3.3.2 Monomer protein set

A monomer protein set was assembled from structures available in the PDB using metadata from the EBI macromolecule database PISA(27) and the PDB header. First, all structures listed as monomers in PISA with the keyword Protein were downloaded. This query resulted in 2489 structures. All PDB files which contained the words dimer or any higher-order oligomer were removed, leaving ~2300 structures. ~570 of these were definite monomers with a line indicating the biological unit to be monomeric in the PDB header. The remaining structures had nothing to indicate oligomerization, and were assumed to be monomeric. Additional monomeric structures were downloaded from the RCSB(28) using the Advanced Search page. The PDB files for all single-chain, protein-containing structures determined using X-ray crystallography, having <1.8Åresolution and <50% sequence identity were downloaded and those containing monomer as the biological unit were saved. This query resulted in 285 structures, approximately 50 of which were also contained in the previous set. The two sets of ~2500 structures were then clustered with CD-HIT(29), using a sequence identity threshold of 40%. CD-HIT performs sequence-based clustering using a greedy incremental algorithm, making it much faster than doing all-by-all comparisons using BLAST. The algorithm generated 1300 clusters, of which the representative PDB from each cluster was used for statistics.

A subset of the monomer protein set was used for energy function weight optimization.

The 64 smallest structures in the set of structures returned by the RCSB search page were randomly assigned into training and testing sets of equal size. The PDB codes for these structures are listed in the supplementary material (Table S3.1).

3.3.3 Development of a score term that disfavors hydrophobic patches

Our goal was to create a score term that favors protein surfaces with distributions of hydrophobic amino acids similar to the distributions observed in naturally occurring soluble proteins. Two different implementations of the hpatch score, hpatch-fast and hpatch-SASA, were developed and tested. Both versions are knowledge-based and are derived from the typical amounts of hydrophobic surface area exposed on protein surfaces. Statistics on hydrophobic accessible surface area were calculated from the set of monomeric structures described above.

The hpatch-fast score assigns all surface residues a score that depends on the amount of exposed hydrophobic surface area (hSASA) in their vicinity. Precalculated average hSASA values that depend on amino acid type and degree of burial (as measured by number of neighbors) are used to rapidly estimate the hSASA for each residue. To derive these average values, the exact amount of hSASA exposed by every residue with 24 or fewer neighbors (Cβ within 10 Å) in the monomer protein set was calculated using Rosetta. The areas were grouped by residue type and number of neighbors and averaged (Table S3.2). Five different levels of burial were considered: residues with 10 or fewer neighbors, 11 to 13 neighbors, 14 to 16 neighbors, 17 to 20 neighbors, and 21 to 24 neighbors. Using these precalculated values avoids the slow calculation of exact SASA, making optimization of the score during a design run fast.

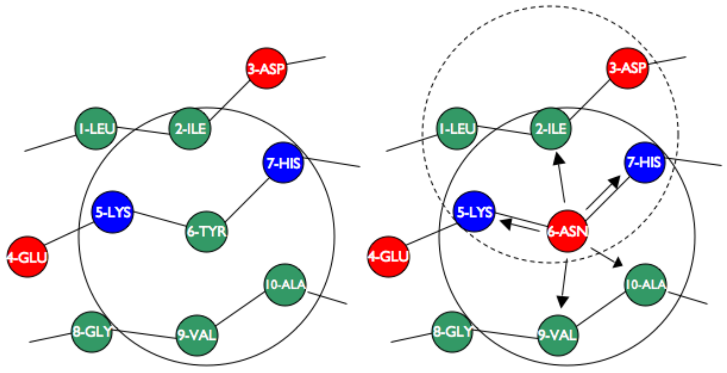
The hpatch-fast score for a given position depends on two things: the total amount of hSASA surrounding that position and the number of neighbors it has (Cβ distance), both within 10 Å. To parameterize the score, the hSASA around every surface residue in the set of naturally occurring monomeric structures was calculated using Rosetta. Residues were considered surface residues if they had 20 neighbors or less. Using the precalculated average values for the hydrophobic area exposed by each residue type, the sum of the amount of hydrophobic area exposed by a given surface residue and all of its neighbors within 10 Å, along with that residues number of neighbors, was saved. Neighboring residues with greater than 24 neighbors were assumed to have zero exposed hydrophobic surface area. All of the hSASA values were then grouped by number of neighbors. The distribution of areas was binned into increments of 25 Å^2 and the inverse log of the probabilities was taken to create a score that favors native-like amounts of hydrophobic area surrounding residues on protein surfaces (Figure S3.1). The score values are reported in the supplementary material (Table S3.3). The score is defined out to a maximum area size of 1100 Å^2. Areas of size greater than 1100 Å^2 are given a score of 25.

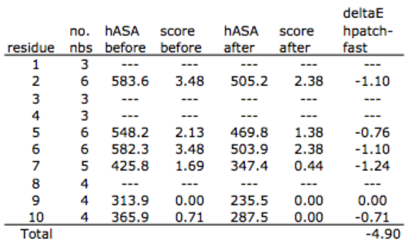
The hpatch-SASA score uses the exact SASA for each atom in the protein and, instead of assigning a score to each surface residue, explicit patches that can span many residues are detected and given a score. During rotamer optimization, the SASA of the protein is kept up-to-date in the same manner as in Leaver-Fay et al.(30). Briefly, a set of dots is distributed evenly on a sphere centered on an atom, where the radius of the sphere is the radius of the atom plus the probe radius. Each dot keeps track of the number of other residues that "cover" it, determined by using distance and angle calculations and precalculated masks that specify which dots are covered given two spheres(31). When a rotamer substitution is considered, only the dot coverage counts for atoms which have overlapping SASA radii with either the previous rotamer or the new rotamer are updated. The SASA of each atom is determined by counting the number of dots not covered by any other atoms.

The hpatch-SASA score also uses a more rigorous method for finding hydrophobic patches, similar to that of the program QUILT(32). After the SASA computation has completed, all hydrophobic atoms with nonzero SASA are assigned as nodes in a graph. An edge is placed between two nodes in this graph if their corresponding atoms have exposed overlap. The requirements for being considered exposed overlap are given in the following section. The union-find algorithm(33) is run on the graph to find all of its connected components. Each connected component represents a hydrophobic patch on the surface of the protein.

Statistics on the patches found in native proteins were calculated to derive the function used for the hpatch-SASA score. A distribution of patch size for all patches with four or more atoms in the set of monomeric structures is shown in Figure S3.2. Using the inverse log of the probabilities does not provide a score bonus for splitting a large patch into two smaller sized patches as the score is mostly linear with a slope close to 0.5. Therefore, various exponential curves were plotted with the inverse log probabilities and $score = 0.4 * ((patch\_area/50) - 1)^2$ was selected for the score because it greatly penalizes large patches without overpenalizing smaller, native-sized patches. The score values were adjusted so that patches with an area of 50 Å^2 or less receive a score of 0.0 (Table S3.4). During scoring, patch areas are binned to the nearest 50 Å^2. Patches of size 900 Å^2 or greater are assigned a score of 100.

Hydrogen atoms are excluded from both SASA calculations and patch identification. The van der Waals radii used here were taken from Chothia et al.(34) (Table S3.5). For comparison, hydrophobic patch areas were also calculated using QUILT(32). All QUILT runs used the maximum number of dots per atom, 252, the recover option -R, and a polar expansion radius of 1.4.





residue	no. nbs	hASA before	score before	hASA after	score after	deltaE hpatch-fast
1	3	---	---	---	---	---
2	6	583.6	3.48	505.2	2.38	-1.10
3	3	---	---	---	---	---
4	3	---	---	---	---	---
5	6	548.2	2.13	469.8	1.38	-0.76
6	6	582.3	3.48	503.9	2.38	-1.10
7	5	425.8	1.69	347.4	0.44	-1.24
8	4	---	---	---	---	---
9	4	313.9	0.00	235.5	0.00	0.00
10	4	365.9	0.71	287.5	0.00	-0.71
Total						-4.90

Figure 3.1: Overview of how the hpatch-fast score updates in response to a sequence substitution. Consider a substitution from tyrosine to asparagine at position (node) 6. Each of the neighbors within 10 Å(solid circle) of node 6 - nodes 2, 5, 7, 9 and 10 (indicated by arrows) - updates its record of the total amount of hydrophobic accessible surface area (hASA) within 10 Åassuming the substitution is accepted. The sum of the change in the hpatch-fast score at node 6 and all of the neighboring nodes becomes the hpatch-fast score change for the substitution. The table shows how the hASA and hpatch-fast score change at all of the nodes. Dashed circle, neighbors within 10 Åof node 2.

3.3.4 Implementation of the hpatch scores as non-pairwise decomposable terms in Rosetta

As patch identification is not pairwise decomposable, the hpatch scores were implemented differently than the other score terms in the Rosetta energy function. Their implementation closely follows that of the SASApack score described in Leaver-Fay et al.(30). During a design simulation using the hpatch-fast score, each surface-exposed residue keeps track of the sum of exposed hydrophobic area within 10 Å. An amino acid substitution at an exposed residue causes that residue and all its neighboring residues to update their hSASA sums (Fig. 3.1). The updated hSASA is used to get the new hpatch-fast score for that residue. The hpatch-fast score of the protein is the sum of the hpatch-fast score of all residues. For the hpatch-SASA score, two sets of calculations are performed after every amino acid or rotamer substitution (Fig. 3.2). First, the SASA values of the residues near the changing residue are updated. Then, all of the hydrophobic patches for the current rotamer assignment are found using the union-find algorithm. The sum of the scores of all patches with four or more atoms becomes the hpatch-SASA energy for that state assignment.

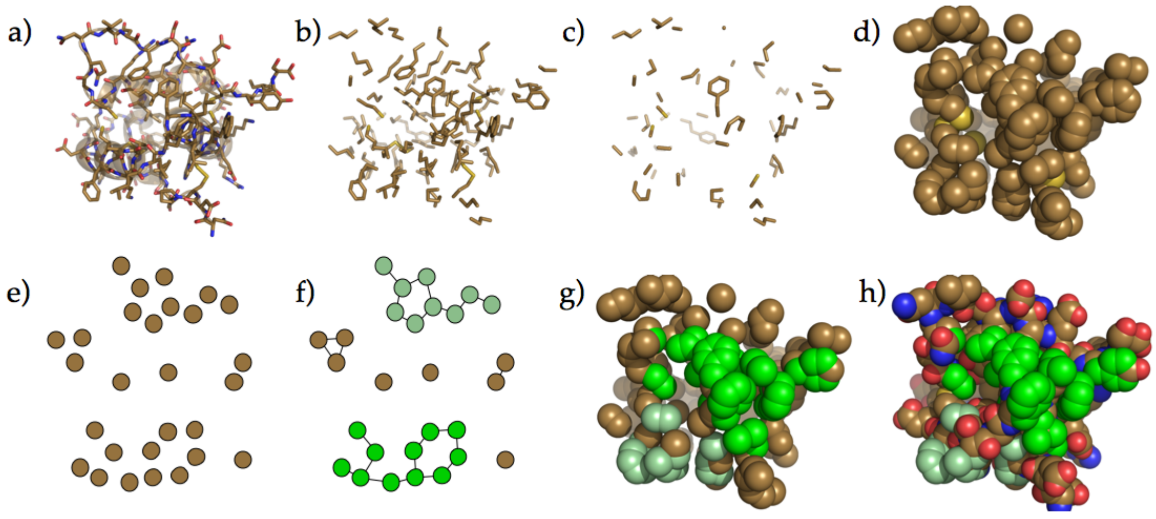


Figure 3.2: Overview of how the hpatch-SASA score finds and scores hydrophobic patches. Consider a protein being designed (A). During simulated annealing, after each substitution all nonpolar atoms with nonzero SASA (B) are assigned to nodes in a graph (C). The union-find algorithm is run on this graph, which places edges between nodes whose atoms have exposed overlap. The output of the union-find algorithm is the set of all connected components in the input graph (D), which represents all of the hydrophobic patches on the protein. The largest hydrophobic patch on the input protein is shown in green in (E), with the atom radii expanded to their SASA radii in (F).

Two additional considerations are necessary with the hpatch-SASA score to prevent assigning all of the hydrophobic surface area to one large patch. One way an overly large patch

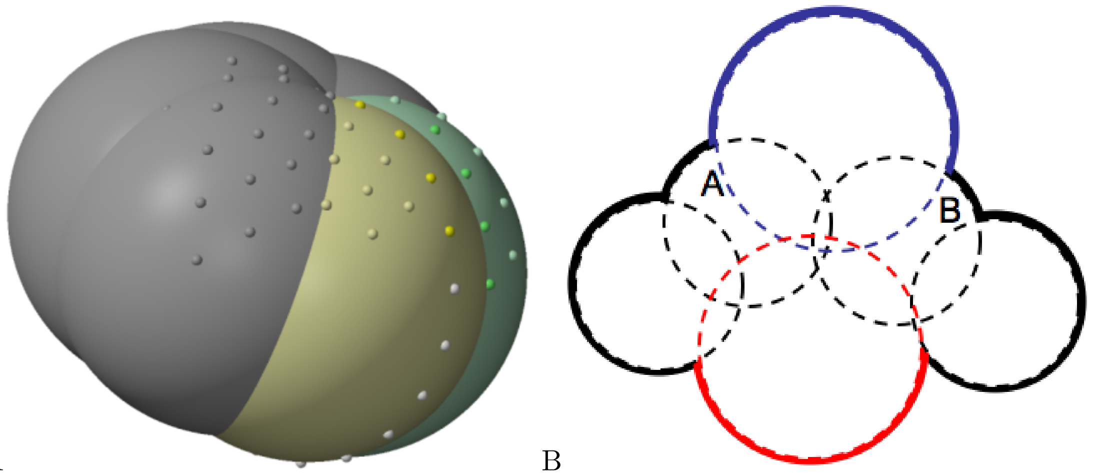


Figure 3.3: Checking for exposed overlap. A) Dots on the surface of PHE4 from ubiquitin (PDB id: 1UBQ). The white dots are buried, all other dots are exposed. The neighbors of this PHE, which bury most of its surface, are not shown. Atoms CZ (yellow) and CE1 (green) have exposed overlap according to the criteria used in this paper. The white dots and dark-yellow dots represent the set of dots on CZ adjacent to the plane of intersection with CE1. The dark green dots on the surface of CE1 are both exposed and adjacent to the intersection with CZ. Because both the dark-yellow set and the dark-green set are non-empty, atoms CZ and CE1 are said to have exposed overlap. B) If the adjacency condition for considering two atoms as part of the same patch were merely sphere overlap, instead of exposed overlap, then the two dimensional atoms A and B would be considered part of the same patch. Intuitively, this is mistaken, since the nitrogen (blue) and oxygen (red) atoms pictured here disrupt the patches from joining. A and B overlap, but the region where they overlap is not exposed to solvent.

can arise is if narrow strips of hydrophobic surface area connect large regions of hydrophobic surface area. As was done by Lijnzaad et al.(32) to avoid this situation, the polar atom SASA radii are expanded by 1.4 Å. Expanding the polar atom radii reduces the number of thin strips of hydrophobic area, delimiting the surface into separate hydrophobic patches. The other way in which an overly large patch can arise is if atom-pair adjacency is considered by sphere-overlap alone. Instead, the overlap region must be exposed for two atoms to be considered to contribute to the same patch. Two overlapping atoms, a and b, are defined to have exposed overlap if there exists an exposed dot on a adjacent to the plane of intersection with atom b, and if there exists an exposed dot on the surface of b adjacent to the plane of intersection with atom a (Figure 3.3A). Computing whether any dot on atom a is adjacent to the plane of intersection with atom b is logically a boolean AND of the bit-vector representing a's exposed dots and the pre-computed overlap mask(31) for b's overlap on a taken at distance max(0, r

- τ), where r is the actual distance between a and b, and τ is the distance threshold limiting a dot's distance from the plane of intersection to be considered adjacent to the intersection. In this work, we use a cutoff distance τ= 0.8Å. Not checking for exposed overlap between two atoms can result in overlapping atoms with noncontiguous surface area being assigned to the same connected component (Figure 3.3B). With this approach, it is possible that two atoms are placed into the same patch even though their accessible hydrophobic surface area is not contiguous. This result would occur when the exposed dots in each ring are on opposite sides of the plane of intersection between the two atoms. We assume this case happens rarely and do not check for it during simulations.

As with the SASApack score, to speed up design simulations, hpatch score evaluations are procrastinated if the change in energy of the pairwise-decomposable terms for a rotamer substitution exceeds some threshold value. If the substitution is later accepted by the Metropolis criterion, the hpatch calculations are performed. This optimization is particularly helpful at the end of simulated annealing when most rotamer substitutions are rejected.

3.3.5 Explicit unfolded state energy term

An explicit unfolded state energy was used in place of the reference energies for some of the simulations. The unfolded state energy was calculated using a peptide-based model, which uses the energy of amino acids in fragments of structure to approximate the unfolded state energy. The average energy of each amino acid in the unfolded state was obtained by excising fragments from a set of PDB files and calculating the energy of the central residue. The set of PDB files used was the Dunbrack non-redundant subset of crystal structures with resolution $\leq$2.0 Å and R-factor $\leq$0.25 assembled in June 2005(35). Fragments of size 13 were randomly selected from each structure and repacked. Additional rotamers were created by expanding all χ angles $\pm$1 standard deviation around their preferred values. The number of residues in the protein multiplied by 0.1 determined the number of fragments taken from each structure. The unweighted energies of the central residue in every fragment were stored and then grouped by residue type. Unweighted, as opposed to weighted, energies were kept so that the same weights found during weight optimization and applied to the folded state could be used for the unfolded state. The mean values of the total unfolded energy for each residue type are shown in the supplementary material (Table S3.6).

3.3.6 Rosetta energy function and weight optimization

The various energy functions tested in this work were each optimized for native sequence recovery using a weight-fitting protocol implemented in Rosetta (Andrew Leaver-Fay, in preparation). The current Rosetta energy function was optimized using an approach similar to the one used here(20). The protocol works by adjusting the weights of the energy terms and the reference energies so that the Boltzmann probability of the native amino acid is maximal over all positions in a set of proteins. More formally, the fitness is defined as

$$\sum_{proteins} \sum_{positions} -\ln \left[\frac{e^{-E(aanat)}}{\sum_{aa,i} e^{-E(aai)}} \right]$$

where E(aanat) is the energy of the native amino acid at a position and the denominator is the partition function for all 20 amino acids at that position. To reduce floating-point errors from multiplying probabilities, the sum of inverse log of the probability was minimized. At each position in a representative set of proteins, the unweighted energy for all rotamers for every amino acid were obtained at that position, holding the other positions in the protein fixed at their native rotamers. Extra $\chi 1$ and $\chi 2$ torsion angles were used for all residue types at all positions. The best scoring rotamer for each residue type was used for evaluation of the fitness function. Candidate weight sets were created using particle swarm optimization followed by conjugate-gradient-based minimization of the best set of weights found using the swarm. The best, minimized weight set is then used to fully redesign all proteins in the set. Weight sets that improve both the overall sequence recovery and the designed amino acid composition are accepted, and the weight optimization-redesign cycle is repeated until the weights converge. If the overall sequence recovery or amino acid composition worsens, the reference energies are adjusted and redesign of the training set is repeated iteratively until both improve or until a predefined limit of 6 iterations is reached. If the limit is reached, the weight set is rejected and the next cycle of weight optimization begins from the previously accepted weight set. Natural amino acid composition is obtained by minimizing the cross entropy between the distribution of designed amino acids and native amino acids. The cross entropy is minimized by raising the reference energy of amino acids overrepresented in the redesigned proteins and lowering those that are underrepresented. Typically, 6-8 rounds of reference energy adjustment are needed to obtain native-like amino acid compositions. Only a subset of the terms in the standard Rosetta energy function were optimized. The omega, long-range and short-range backbone-backbone hydrogen bond weights were held fixed at 0.5, 1.17, and 0.585, respectively. In fixed-backbone design, these terms do not help in improving sequence recovery as all backbone coordinates are fixed. The fa_atr term, representing the attractive portion of the Lennard-Jones potential, was also held fixed at 0.8 so that the optimized weights could be compared to the current Rosetta weights. Because only one residue is being considered at a time during fitness function evaluation, the weight optimization procedure is also not appropriate for fitting the weights for the hpatch scores. Therefore, multiple weight optimization simulations were performed with varying fixed weights on the hpatch scores. The weight on the hpatch-fast score was left at 1.0. A weight of 0.3 on the hpatch-SASA score gave the best results without changing the sequence recoveries and amino acid composition. In addition to the standard Rosetta energy function, additional energy functions using the hpatch score and/or the unfolded state energy term described above were optimized (Table S3.7). The other energy functions optimized are as follows: the standard energy function with the hpatch-fast term (standard + hpatch-fast) and with the hpatch-SASA term (”standard + hpatch-SASA”); a standard one which replaces the reference energies with the unfolded state energy term (standard, no refE + unfoldedE), and the same with the hpatch-SASA score added (”standard, no refE + unfoldedE, hpatch-SASA”).

3.3.7 Predicting changes in free energy for protein mutants

Some of the optimized energy functions were also used to predict the change in free energy for a set of experimentally characterized mutants. Wild type and mutant structures were relaxed using the protocol described in Row 16 of Table 1 in Kellogg et al.(36). Briefly, all of the side chains are first repacked using a soft repulsive energy function. Then the structures side-chain and backbone torsions are minimized using a hard-repulsive energy function. During minimization, harmonic restraints are placed on all pairs of C-alpha atoms within 9 Å keeping the backbone from moving too far from the crystal structure. Three rounds of minimization are performed, where the weight on the repulsive term is increased, starting at 1/10th of its full weight, then at 1/3rd of its full weight, and ending at the full weight. This protocol is applied 50 times to both the wild type and mutant species, and the average of the three-lowest energies for each species is taken as its energy. The predicted ΔΔG is the difference between the energies of the mutant and wild type species. A set of 1210 mutants assembled by Yin et al.(37) and Guerois et al.(38) were used for testing prediction accuracy. When weight sets that included the hpatch term were used in ΔΔG prediction, the weight on the hpatch term was set to 0. Prediction accuracy was measured by calculating the Pearson correlation coefficient.

3.4 Results

We first examined the performance of the current full atom energy function from Rosetta (version 3.1), which was originally parameterized to best reproduce native amino acid sequences when performing whole protein redesigns of high-resolution crystal structures(4; 19). Sequence redesigns were performed on a test set of 32 monomeric proteins. The results were similar to what we have observed previously(19). In the core of the proteins, 49% of the wild type amino acids and 33% of all residues were recovered (Table 3.1). The surfaces of the redesigns have 1270 Å^2 of hSASA, on average, similar to the wild type proteins which have 1100 Å^2 on average. However, in the redesigns the surface hydrophobics are more clustered than in the wild type proteins. The average size of the largest hydrophobic patch on the wild type proteins is 476 Å^2, while for the redesigns it is 813 Å^2. For three designs there were extremely large hydrophobic patches, with areas greater than 1200 Å^2. Surface residue design is heavily influenced by the amino acid reference energies. To see if the patches are a result of the current reference energies, we used a weight optimization protocol to refit the reference energies holding the weights on the other energy terms fixed. Designing with this energy function results in redesigns with sequence recoveries of 52% in the core and 35% overall. The amount of total hydrophobic surface area in the redesigns, 1206 Å^2, is again similar to what is seen in natives, 1100 Å^2. As with the current Rosetta energy function, though, large hydrophobic patches are found on the surfaces of the redesigns. The average size of the largest hydrophobic patch in the redesigns is 694 Å^2, 1.5 fold larger than the patches seen on wild type proteins. As before, there are several proteins with very large patches. This set includes two all-β proteins, on which the largest hydrophobic patch in each protein spans the surface of a β-sheet. Refitting the reference energies is not sufficient for producing native-like surfaces.

	sequence recovery						
energy function	core	overall	surface	avg QUILT -ep 1.4 (Å^2)	avg total hASA (Å^2)	% hp on surface	hpatch score weight
natives (all/train/test)	—	—	—	462 448 476	1090 1080 1100	27.4 27.0 27.8	—
current Rosetta weights	49.2	32.9	24.9	813	1270	29.5	—
current Rosetta, fit refEs only, training	56.5	38.0	24.9	775	1213	31.2	—
current + hpatch-SASA, fit refEs only, training	55.8	38.2	26.3	385	984	26.1	0.3
current Rosetta, fit refEs only, test	51.7	35.1	27.9	694	1206	31.8	—
current + hpatch-SASA, fit refEs only, test	52.5	35.5	27.3	446	1046	27.1	0.2
standard, training	56.7	38.1	25.9	697	1134	27.7	—
standard + hpatch-fast, training	56.9	37.9	24.1	590	1078	24.9	1.0
standard + hpatch-SASA, training	55.4	37.4	25.3	374	970	24.6	0.5
standard, test	51.6	35.2	27.3	735	1225	28.9	—
standard + hpatch-fast, test	51.6	35.2	26.6	723	1105	23.8	1.0
standard + hpatch-SASA, test	52.3	36.5	28.9	433	1089	27.7	0.3

Table 3.1: Recoveries and energies of weight optimized standard and standard + hpatch energy functions. This table reports the native sequence recoveries, largest hydrophobic patch area and total hydrophobic surface area averages for various energy functions. The energy functions tested include the current Rosetta energy function, the current energy function with the hpatch-SASA score (current + hpatch-SASA), a reweighted standard energy function, the standard energy function with the hpatch-fast term (standard + hpatch-fast) and with the hpatch-SASA term ("standard + hpatch-SASA"). Each of the reweighted energy functions were optimized for native sequence recovery and native amino acid composition. Extra $\chi1$ and $\chi2$ rotamers were used for all residues (-ex1 -ex2 -extrachi_cutoff 0) except for training of the standard + hpatch-SASA energy function which used extra rotamers around $\chi1$ only.

The energy function currently used by Rosetta has been modified since the weights on the score terms were last parameterized. New smoothing functions have been applied to the Lennard-Jones and solvation potentials and hydrogen bond energies are scaled so that buried interactions score more favorably. To create a more appropriate point of reference, we also used the weight optimization protocol to refit the weights on the Rosetta score terms in addition to the amino acid reference energies. With the reweighted energy function 52% of amino acids are recovered in the core of proteins and 35% are recovered for all residues. The surfaces of these redesigns have 1225 Å^2 of hydrophobic surface area, on average. Large hydrophobic patches are still present in these redesigns, with the average largest hydrophobic patch being 735 Å^2. Refitting the weights on the energy terms and/or the reference energies improves native sequence recovery, but does not change how hydrophobic residues are clustered on the surface of designed proteins.

3.4.1 Redesigning proteins with the hpatch score

To counter the tendency of Rosetta to place hydrophobic residues near other hydrophobic residues on the surface, we developed and tested two score terms that explicitly penalize surface hydrophobic patches. Our first implementation was that of the hpatch-fast score, which gives all surface residues a score that depends on the amount of exposed hydrophobic surface area within 10 Å and the number of neighbors that residue has. Each surface residue calculates the sum of the amount of hydrophobic area exposed by all of its neighbors within 10 Å. Average precalculated SASA values based on residue type and number of neighbors are used instead of explicit SASA calculations to approximate how much exposed hydrophobic area a residue adds to the total. Residues with more exposed hydrophobic area surrounding them than what is seen in native proteins get a high score. From tests on individual structures, we found that the hpatch-fast score slightly reduced the size of hydrophobic patches in redesigned proteins. As part of a larger test, and to see what effect the score has on native sequence recovery, we optimized an energy function that included the hpatch-fast score. The weight on the hpatch-fast score was held fixed at 1.0. Sequence recoveries for the proteins created by this optimized energy function are given in Table 3.1. Core and overall recovery are 52% and 35%, respectively, the same as recoveries obtained from the reweighted standard energy function. No significant change is seen in the sizes of the hydrophobic patches in the redesigns, however. The average largest patch size goes from 735 Å^2 in the standard redesigns to 723 Å^2 in the hpatch-fast redesigns. These designs have more native-like amounts of total hydrophobic surface area, 1105 Å^2 on average, but this improvement does not extend to the hydrophobic patches. Increasing the weight on the hpatch-fast score does result in smaller patches, but only because the surfaces become less hydrophobic overall compared to natives.

Since the hpatch-fast redesigns still had large hydrophobic patches, we implemented another version of the score that more rigorously identifies and penalizes patches. Our hypothesis was that the hpatch-fast score was not effective for two reasons. First, we noticed that using precomputed average values for the amount of hydrophobic area each residue adds to a patch introduces a considerable amount of error into the patch areas. Second, hydrophobic patches can easily extend beyond the 10 Å threshold the score considers. For these reasons, the hpatch-SASA score uses the exact SASA for patch areas and the union-find graph algorithm for patch detection (see Methods). In tests on individual structures, adding the hpatch-SASA score with a weight of 1.0 to the standard energy function caused a dramatic decrease in the size and number of hydrophobic patches in the designs.

Confident that the score was penalizing hydrophobic patch formation, we again used the weight fitting protocol to optimize the weights of the other energy terms around the hpatch-SASA score. As was done for the standard Rosetta energy function, the protocol was used to refit the values of the reference energies alone and for all energy terms and reference energies together. Weight fitting was done for both the standard Rosetta energy function and the standard energy function with the hpatch-SASA score. The recoveries and surface metrics for proteins redesigned with the reweighted energy functions are given in Table 3.1. For the energy function where only the reference energies were optimized, core and overall recovery stand at 53% and 36%, respectively. These recoveries are very close to the recoveries obtained with the reweighted standard Rosetta energy function. The average largest hydrophobic patch size in these redesigns is 446 Å^2, much smaller than what is seen in the current and reweighted standard Rosetta redesigns and smaller also than what is seen in the native proteins. The total amount of hydrophobic surface area in these redesigns, 1046 Å^2, is close to what is seen in natives, 1100 Å^2. When allowing all weights and reference energies to be optimized, the hpatch-SASA energy function gets recoveries of 52% in the core and 37% overall. The average largest hydrophobic patch size in these redesigns is 433 Å^2 and the average total amount of

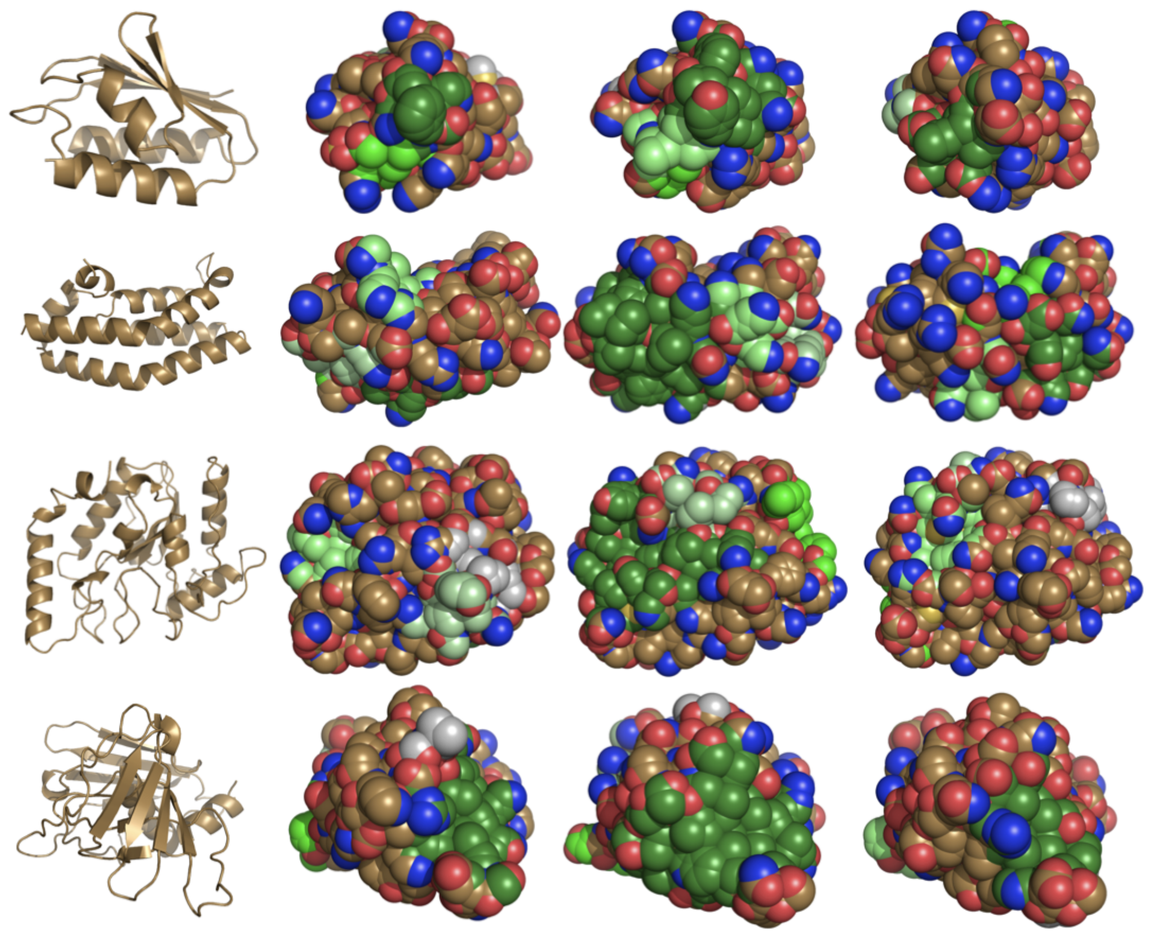


Figure 3.4: Surfaces of native and redesigned proteins. From left to right, the native protein in cartoon representation, and spherical representations of the native protein, a current Rosetta redesign, and a reweighted Rosetta + hpatch-SASA redesign. Hydrophobic patches are colored according to size (largest to smallest: dark green, green, lime, pale green, gray). Oxygen and nitrogen atoms colored red and blue, respectively, and all other atoms are colored gold. Proteins shown are histidine-containing protein (1OPD), histidine-containing protein phosphotransfer domain (2A0B), uracil DNA glycosylase (3EUG), and toxic shock syndrome toxin-1 (3TSS). Figures created with PyMOL(39).

hydrophobic surface area in these redesigns is 1089 Å^2. The reduction in the average largest hydrophobic patch size is achieved without a change to the total amount of hydrophobic surface area. Examples of the difference in largest hydrophobic patch size between a native protein, a reweighted standard Rosetta redesign, and a redesign with the hpatch-SASA score are shown in Figure 3.4.

Designing with the hpatch-SASA score results in distributions of hydrophobic patch sizes

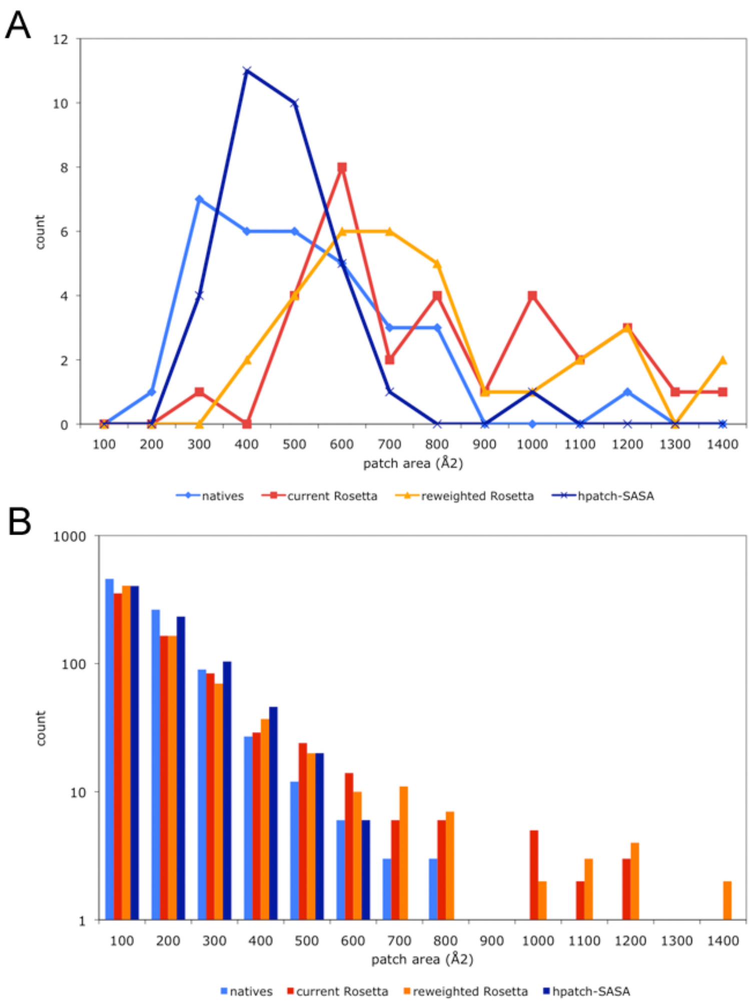


Figure 3.5: Hydrophobic patches in native and redesigned proteins. The bar graphs show hydrophobic patch area distributions for the largest (A) and all (B) patches in native (blue), current Rosetta redesigns (red), reweighted Rosetta redesigns (orange) and reweighted Rosetta + hpatch-SASA redesigns (dark blue).

more like that of native proteins. A histogram of largest hydrophobic patch size for the native and redesigned proteins is shown in Figure 3.5A. There is a noticeable shift toward larger patches in the proteins redesigned with the current and reweighted Rosetta energy functions that is shifted back to near-native levels with the addition of the hpatch-SASA score. The score also corrects the size distribution of all patches, not just the largest patch. Figure 3.5B shows the distribution of patch sizes for all patches in the native and redesigned proteins. The redesigns created with the hpatch-SASA score have patch sizes that track the sizes seen in native proteins better than the current and reweighted standard energy function redesigns. Using the hpatch-SASA score, it is possible to design surfaces with native-like amino acid composition and smaller than native sized patches of hydrophobic area.

protein	no. residues	rotamers	score function	hpatch-SASA score	area largest QUILT patch (Å^2)	total time (s)	sim annealing time (s)
1HZ5A	72	—	native	6.88	258	—	—
		96520	standard redesign	26.3	548	344	327
		96551	standard + hpatch-SASA redesign	5.1	399	6691	6667
1LMBA	87	—	native	14.2	563	—	—
		114005	standard redesign	15.7	493	478	457
		114057	standard + hpatch-SASA redesign	2.2	463	8184	8152
1QYS	92	—	native	12.2	481	—	—
		127087	standard redesign	21.8	661	546	522
		127132	standard + hpatch-SASA redesign	5.9	588	12419	12390
1FKB	107	—	native	7.7	585	—	—
		134025	standard redesign	16.2	674	578	551
		134096	standard + hpatch-SASA redesign	7.4	525	11330	11297
1IFC	131	—	native	11.0	554	—	—
		183274	standard redesign	15.8	767	879	840
		183342	standard + hpatch-SASA redesign	8.0	558	21755	21715
1GBS	185	—	native	6.2	411	—	—
		195483	standard redesign	27.7	426	1161	1115
		195563	standard + hpatch-SASA redesign	2.6	309	22965	22905

Table 3.2: Energies, hydrophobic patch areas and run times of proteins redesigned with the hpatch score. Each protein was redesigned with the current Rosetta energy function and the optimized standard + hpatch-SASA energy function. All residue types were allowed at all positions, and extra $\chi1$ and $\chi2$ torsion angles were used for all residues. Hydrophobic patch areas were calculated using QUILT, with a polar expansion radius of 1.4Å. Sim. annealing time represents the time spent in the sequence optimization part of the simulation.

Design simulations using the hpatch-SASA score take longer to complete because patch energies cannot be precalculated and stored in memory, as can rotamer pair energies. To see what effect the hpatch-SASA score has on the final energies and run times of design simulations, we performed complete redesigns of seven proteins using the standard Rosetta energy function and the hpatch-SASA optimized energy function. Protein names, final energies, and running times are reported in Table 3.2. The total time of simulations with the hpatch-SASA score increases by a factor of 21, on average. This increase appears to be independent of protein size as the fold increase in runtime for the 72 residue protein 1HZ5A is roughly the same as the increase for the 185 residue protein 1GBS.

3.4.2 Design using an energy function with an explicit unfolded state energy

Instead of explicitly modeling the unfolded state Rosetta uses amino acid-specific reference energies that are parameterized to favor a native-like distribution of amino acids in designed sequences. This implicitly favors proteins with hydrophobic cores and polar surfaces as the solvation model in Rosetta strongly penalizes the burial of polar chemical groups. Because the hpatch score provides an alternative mechanism for controlling amino composition on the protein surface we were curious if we could replace Rosettas reference energies with an explicit unfolded state energy term in combination with the hpatch score. The attractiveness of this approach is that it removes 19 adjustable parameters from the weight fitting process and creates a score function with an explicit protein stability term (energy of the folded state minus the unfolded state) and an explicit measure of protein solubility (hpatch score). A variety of approaches can be used to model the unfolded state. Creamer and Rose(40) found that using fragments excised from the structures of folded proteins serve as better models of the unfolded state than do tripeptides. Based on this conclusion, Pokala and Handel(10) used the average energy of amino acids in short fragments as a per-residue unfolded state energy in their design algorithm. They found that the fragment-based unfolded state model outperforms the tripeptide model in predicting changes in stability for a large number of protein mutants. Here, we use 13-residue fragments excised from folded proteins to estimate the average energy of each amino acid type in the unfolded state (see Methods).

Several energy functions using the unfolded state energy in place of the reference energies with and without the hpatch-SASA score were optimized for native sequence recovery and amino acid composition (Table 3.3). Using the unfolded state energy term in place of the Rosetta reference energies lowers native sequence recovery and leads to very hydrophobic surfaces with an excessive amount of tryptophans on the surface (Table S3.8). Core and overall recovery with this energy function are 48% and 31%, respectively, and the average amount of hydrophobic SASA is 1844 Å^2, nearly twice as large as the wild type proteins. The average largest hydrophobic patch size jumps to 1479 Å^2 compared to 735 Å^2 for the reweighted standard Rosetta energy function.

We next added the hpatch-SASA score term to the energy function with the unfolded state

energy function	sequence recovery core	sequence recovery overall	sequence recovery surface	avg QUILT -ep 1.4 (Å^2)	% hp on surface	avg total hASA (Å^2)	hpatch score weight
natives	—	—	—	476	27.8	1100	—
current, no refE + unfoldedE, test	37.2	24.4	17.8	1173	51.2	1935	—
current, no refE + unfoldedE, hpatch-SASA, test	37.1	25.2	18.7	445	35.8	1206	0.30*
standard, no refE + unfoldedE, training	55.9	32.8	17.9	1403	48.1	1837	—
standard, no refE + unfoldedE, hpatch-SASA, training	50.6	31.8	18.9	424	29.3	1024	0.50*
standard, no refE + unfoldedE, test	47.7	30.5	21.0	1479	46.4	1844	—
standard, no refE + unfoldedE, hpatch-SASA, test	48.4	29.8	19.9	464	30.3	1064	0.50*

Table 3.3: Recoveries and energies of weight optimized standard and standard + unfolded state energy and/or hpatch energy functions. This table reports the native sequence recoveries, largest hydrophobic patch area and total hydrophobic surface area averages for various weight optimized energy functions. The energy functions tested include the current energy function which replaces the reference energies with the unfolded state energy term (current, no refE + unfoldedE), the current energy function with the unfolded state energy and the hpatch-SASA score ("standard, no refE + unfoldedE, hpatch-SASA"), a reweighted standard energy function, and a reweighted standard energy function with the hpatch-SASA score (standard, no refE + unfoldedE, hpatch-SASA). Both of the reweighted energy functions were optimized for native sequence recovery alone. Extra $\chi1$ and $\chi2$ rotamers used for all residues (-ex1 -ex2 -extrachi_cutoff 0).

term. The core and overall recoveries with this energy function, 48% and 30% respectively, are similar to the recoveries of the energy function without the hpatch-SASA score. However, the surfaces of these redesigns have considerably smaller hydrophobic patches than when the hpatch-SASA score is not present. The average largest hydrophobic patch size goes from 1479 Å^2 to 464 Å^2 upon addition of the hpatch-SASA score and the total average hSASA, 1064 Å^2, is similar to the wild type proteins. Despite having a more native-like distribution of hydrophobic surface area on the surface, the amino acid composition on the surface is still significantly different than the native sequences (Table S3.8). While the native sequences have 37 histidines and 12 tryptophans on their surfaces in total, the designs have 308 histidines and 168 tryptophans. Conversely, alanine and lysine are grossly underrepresented on the surfaces of the designs. These results can be interpreted in a variety of ways. The peptide model of the unfolded state may be missing important features that determine the favorability of each amino acid in the unfolded state. The over abundance of tryptophan in the design models indicate that tryptophan makes more favorable interactions on a protein surface (using the Rosetta energy function) than in the peptide fragments used here. This could indicate that true unfolded states allow for more contacts and burial than is present in the fragments. Alternatively, the additional tryptophans on the surface may be favorable for folding free energy, but may have other negative consequences, such as favoring misfolded conformations or non-specific interactions with other proteins. Amino acid composition may also be partially determined by metabolic constraints that influence the overall fitness of an organism.

energy function	$\Delta\Delta$G R
current Rosetta weights	0.69
current Rosetta, fit refEs only	0.68
current + hpatch-SASA	0.68
current, no refE + unfoldedE	0.66
current, no refE + unfoldedE, hpatch-SASA	0.66
standard	0.61
standard + hpatch-SASA	0.63
standard, no refE + unfoldedE	0.58
standard, no refE + unfoldedE, hpatch-SASA	0.44

Table 3.4: Correlation coefficients for predicting changes in stability. Correlation coefficients between experimental and predicted $\Delta\Delta$G for a set of protein mutants using the various optimized energy functions.

3.4.3 Predicting changes in stability for mutations

Rosetta can also be used to predict the change in free energy for protein mutants ($\Delta\Delta$G). Given that we optimized the energy functions described above only for native sequence recovery, we wanted to see how they would perform at predicting $\Delta\Delta$G. We follow the protocol described in Kellogg et al.(36), which uses repacking and side-chain and backbone torsion minimization to create mutant structures. First, all residues in the mutant structure are repacked using the Rosetta energy function with a dampened Lennard Jones energy. Then the structures are cycled through side-chain and backbone torsion minimization using either the standard Rosetta energy function or one of the weight-optimized energy functions described above. The results of using the various energy functions to predict $\Delta\Delta$G of stability can be found in Table 3.4. With the current Rosetta energy function there is a correlation coefficient of 0.69 between the experimental and predicted $\Delta\Delta$G values(36). When only the reference energies are reweighted, both the standard Rosetta energy function and the standard energy function with the hpatch-

SASA score have correlation coefficients of 0.68. If all of the score terms and the reference energies are reweighted, the standard Rosetta energy function and the hpatch-SASA energy function get correlation coefficients of 0.61 and 0.64 respectively. Using the unfolded term in place of the reference energies and reweighting all of the score terms for native sequence recovery gives a correlation coefficient of 0.58.

3.5 Discussion

In previous de novo protein design projects with Rosetta it has been necessary to restrict the amino acid alphabet available at specific surface residue positions in order to avoid the design of large hydrophobic patches on the protein surface(20; 41; 42). As seen in the tests performed here, the primary problem is not the overall amino acid composition of the protein surface, but rather the clumping of similarly typed amino acids. Hydrophobic and polar amino acids probably segregate on the surfaces of the designs for the same reason that oil and water do not mix, the hydrophobic amino acids can not satisfy the hydrogen bonding potential of the polar amino acids. Most protein design algorithms, including Rosetta, use energy functions that are pairwise additive at the residue level. In this case, there is not a straightforward mechanism for explicitly disfavoring hydrophobic patches while maintaining a native-like distribution of amino acids on the surface. Here, we have shown that a non-pairwise additive score function that explicitly detects patches of exposed hydrophobic surface area can be combined with the standard Rosetta energy function to design surfaces that more closely resemble naturally occurring monomeric proteins.

Our use of the hpatch score to disfavor hydrophobic patches is an example of negative design. The purpose of the score term is not to increase the thermodynamic favorability of the folded state relative to the unfolded state, but rather to disfavor aggregation. This suggests that by changing the weight on the hpatch score it will be possible to shift the emphasis of surface redesigns between maintaining solubility and maximizing folding free energy. For example, in a previous study, Rosetta was used to redesign the sequence of the activation domain of human procarboxypeptidase A2(43). The redesigned protein was 10 kcal/mol more stable than the wild type protein. In these simulations, all amino acids were allowed at each sequence position in the protein and a large hydrophobic patch, with an area of 730 Å^2, was created on the surface of the proteins β-sheet. Subsequent NMR analysis indicated that at concentrations >100 μM the redesigned protein self-associates and buries the hydrophobic residues on the surface of the β-sheet(4). A similar interaction was seen in the crystal structure of the protein. If we redesign procarboxypeptidase (1VJQ) using the hpatch score, a large hydrophobic patch is no longer placed on the surface of the sheet. Instead, the largest patch in this redesign is created by one of the loops of A2 and has an area of 257 Å^2.

Because patch identification is not pairwise-decomposable, simulations with either implementation of the hpatch score take longer to complete than standard Rosetta design runs. For comparison, designing with another non-pairwise-decomposable score term in Rosetta, the SASApack score, increased the runtimes of simulation by 26-fold. Design simulations with the hpatch-SASA score increase the runtime by a factor of 21. In addition to making the surface area calculations, time is also spent finding patches using the union-find algorithm with the hpatch-SASA score. As with the SASApack score, procrastination of hpatch score calculations helps to speed up the simulations. If a substitution causes an increase in the energy of the other energy terms over some threshold amount, the hpatch score is not calculated unless the substitution is later accepted. This optimization applies to both forms of the hpatch score. As simulations with the hpatch-SASA score take longer than current Rosetta, we recommend that the hpatch-SASA score only be used during the final design runs of a protocol. Alternatively, as the score is fast to compute for a single structure, it could be used as a filter at the end of a protocol. Chennamsetty et al.(44) recently used a spatial aggregation propensity score that gives positions with exposed hydrophobic neighbors a high score to increase the solubility of two antibodies.

As discussed above, one function of the 20 amino acid reference values used in Rosetta is to implicitly disfavor the placement of large numbers of hydrophobic amino acids on the surfaces of redesigns. For this reason, we tested if the reference values could be replaced with the hpatch score and explicitly calculated unfolded state energies. The hpatch score was successful at preventing large hydrophobic patches on the surface; however, the amino acid composition of the redesigned surfaces was considerably different than naturally occurring proteins. The amino acid composition of proteins is probably determined by several factors including: protein stability, protein solubility, metabolic constraints and negative design against alternative structures and complexes. Without an empirical energy term that can be varied to titrate amino acid compositions, it is difficult to match the naturally occurring frequencies on protein surfaces. Pokala and Handel had some success using amino acid reference energies that were derived from calculating the average energies of the various amino acids on a protein surface, but hydrophobic amino acids were underrepresented with this approach. In future surface redesigns with Rosetta, we plan to continue using the empirically determined reference values in combination with the hpatch score.

In this study we have focused on the surfaces of monomeric proteins. The hpatch-SASA score may also prove useful when designing transient protein-protein interactions. In this scenario, proteins must be soluble in the unbound state, and therefore, cannot rely on large hydrophobic surfaces to mediate the interaction with the target protein. When performing standard single-state computational protein design on a protein-protein complex there is no energetic penalty for designing an interface mediated by hydrophobics as the solubility of the unbound state is not being considered during the simulation. The solution to this problem is to perform a multi-state design simulation in which the sequence is simultaneously optimized for binding as well as solubility in the unbound state(16). The hpatch-SASA score could be used to provide a measure of solubility in the unbound state for candidate sequences.

Code availability

Source code for the hpatch scores is available for free as part of the Rosetta molecular modeling program, version 3.3.

3.6 Supplementary Material

Computational Protein Design with Explicit Consideration of Surface Hydrophobic Patches

Ron Jacak, Andrew Leaver-Fay, and Brian Kuhlman

training set	size	fold	test set	size	fold
2igd	61	a/b	1a8o	70	all a
1orc	64	a/b	1hyp	75	a
1hoe	74	b	1bdo	80	all b
1aba	87	a/b	1opd	85	a/b
2acy	98	a/b	1opc	99	a/b
1bm8	99	a/b	1by2	110	a/b
1bxv	99	b	2mcm	112	b
1co6	107	a/b	1bea	116	a
1tmy	118	a/b	2a0b	118	all a
1mai	119	a/b	2mhr	118	a
3pyp	125	a/b	1b9o	123	a
1c52	131	a	1bqk	124	a/b
1vsr	134	a/b	1bfg	126	all b
1akr	147	a/b	1ifc	131	b
1bd8	156	all a	1bk9	134	a
1bgc	165	all a	1pne	139	a/b
1kao	167	a/b	2sns	141	a/b
1koe	172	a/b	1amx	150	b
1b2v	173	a/b	1bj7	150	a/b
2sga	181	a/b	1a6m	151	all a
1gbs	185	a/b	1ra9	159	a/b
1qf9	194	a/b	1qst	160	a/b
1nkr	195	all b	1cjw	166	a/b
1cex	197	a/b	1mh1	181	a/b
1rgp	199	all a	3tss	190	a/b
1ppn	212	a/b	2pth	193	a/b
1a7s	225	a/b	2eng	210	a/b
1azo	226	a/b	1lbu	213	a/b
1uch	226	a/b	1g3p	217	a/b
1bio	228	b	3eug	225	a/b
1amf	231	a/b	1atg	231	a/b
3seb	238	a/b	1lst	240	a/b

Table S3.1: PDB codes of proteins used in design and weight optimization runs

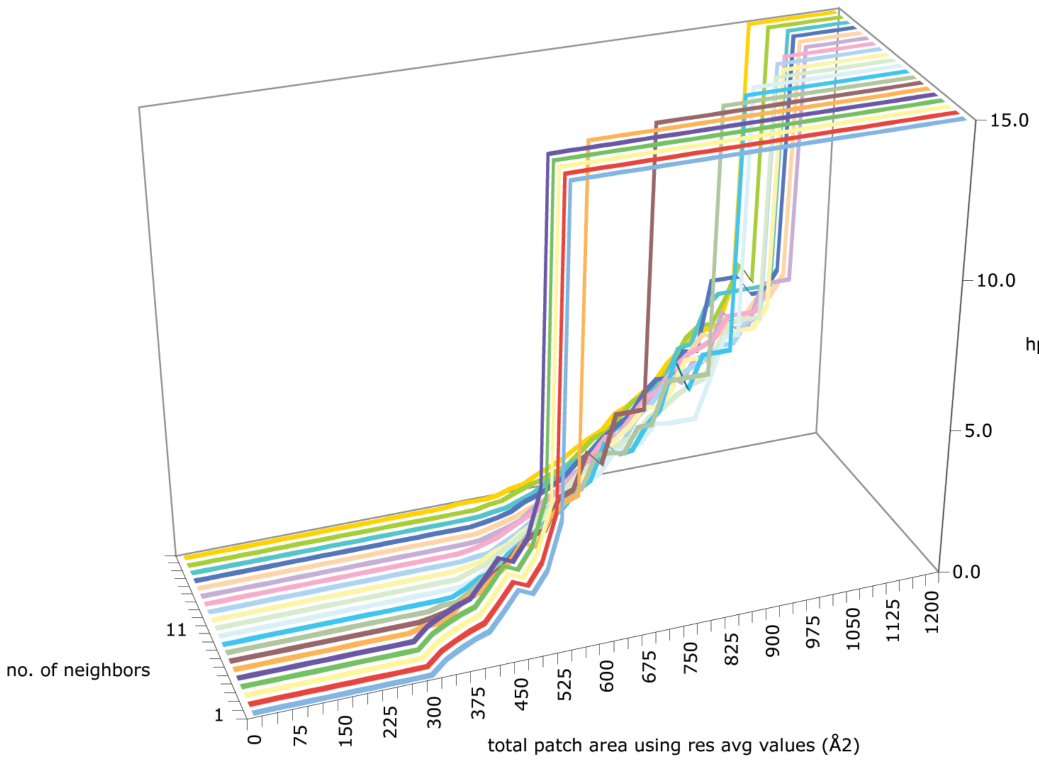


Figure S3.1: Plot of hpatch-fast score by hydrophobic patch area and number of neighbors. Each line represents the score for a given number of neighbors. Residues are only given hpatch-fast scores for having too much surrounding hydrophobic surface area. No score is given to residues with a patch area smaller than the most common patch area for that number of neighbors observed in native proteins. Patches of size greater than 1200 Å^2 are given a score of 25.

amino acid	nbs1-10	nbs11-13	nbs14-16	nbs17-20	nbs21-24
PHE	140.33	97.01	58.46	30.86	11.48
TRP	139.78	95.72	58.85	34.59	16.70
MET	129.12	97.25	61.79	33.03	11.05
LEU	119.66	85.25	53.91	26.87	7.93
TYR	114.72	83.56	55.49	30.61	13.05
ILE	114.64	80.00	52.77	25.21	7.48
LYS	101.83	81.16	63.23	42.99	23.74
VAL	97.18	70.04	47.46	23.49	6.79
PRO	93.59	72.40	52.45	31.07	12.69
HIS	90.26	68.55	48.19	30.81	15.02
CYS	80.59	53.46	32.72	16.52	5.64
ARG	71.21	56.79	42.58	26.33	11.48
THR	68.93	53.70	38.04	21.79	8.33
ALA	63.72	50.20	35.41	17.71	5.48
GLU	59.10	44.45	32.77	19.46	8.16
GLN	51.72	39.30	29.56	17.62	7.27
SER	47.41	37.45	27.29	14.82	5.37
ASP	42.50	31.62	22.33	12.67	5.78
GLY	38.24	31.12	23.67	13.42	5.24
ASN	36.32	27.02	19.65	10.92	4.51

Table S3.2: Average amount of hydrophobic accessible surface area exposed by each residue type at 5 levels of burial: 10 of fewer neighbors, 11 to 13 neighbors, 14 to 16 neighbors, 17-20 neighbors, and 21-24 neighbors. The aromatic residues expose the most hydrophobic surface area, while asparagine and glycine expose the least.

patch area	nb1	nb2	nb3	nb4	nb5	nb6	nb7	nb8	nb9	nb10	nb11	nb12	nb13	nb14	nb15	nb16	nb17	nb18	nb19	nb20
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
25	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
125	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
150	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
175	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
225	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
250	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
275	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
300	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
325	0.443	0.443	0.443	0.443	0.443	0.127	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
350	0.706	0.706	0.706	0.706	0.706	0.138	0.076	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
375	0.929	0.929	0.929	0.929	0.929	0.348	0.182	0.113	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
400	1.099	1.099	1.099	1.099	1.099	0.849	0.379	0.124	0.023	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
425	1.686	1.686	1.686	1.686	1.686	0.937	0.579	0.418	0.243	0.017	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
450	2.315	2.315	2.315	2.315	2.315	1.376	1.084	0.789	0.534	0.169	0.174	0.080	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000
475	2.092	2.092	2.092	2.092	2.092	1.713	1.372	1.079	0.628	0.472	0.320	0.145	0.066	0.063	0.000	0.000	0.000	0.000	0.000	0.000
500	2.785	2.785	2.785	2.785	2.785	2.378	1.883	1.340	1.160	0.732	0.617	0.415	0.118	0.184	0.039	0.000	0.000	0.008	0.000	0.053
525	4.394	4.394	4.394	4.394	4.394	2.133	2.261	1.960	1.470	1.055	0.910	0.571	0.362	0.330	0.165	0.100	0.071	0.018	0.005	0.049
550	25.000	25.000	25.000	25.000	25.000	3.071	2.346	2.072	1.769	1.359	1.211	0.897	0.595	0.580	0.295	0.216	0.147	0.098	0.101	0.035
575	25.000	25.000	25.000	25.000	25.000	3.476	3.387	2.531	2.103	1.741	1.660	1.154	0.911	0.815	0.466	0.354	0.279	0.152	0.149	0.087
600	25.000	25.000	25.000	25.000	25.000	3.659	3.569	3.070	2.430	2.302	1.912	1.493	1.171	1.032	0.836	0.539	0.532	0.328	0.344	0.250
625	25.000	25.000	25.000	25.000	25.000	25.000	4.773	3.267	2.920	2.552	2.503	2.038	1.572	1.443	1.039	0.725	0.694	0.490	0.515	0.322
650	25.000	25.000	25.000	25.000	25.000	25.000	4.262	4.323	3.201	2.918	2.963	2.204	1.828	1.803	1.370	0.987	0.902	0.796	0.646	0.579
675	25.000	25.000	25.000	25.000	25.000	25.000	5.872	4.323	4.414	3.059	3.196	2.747	2.297	2.126	1.789	1.340	1.161	0.961	0.831	0.766
700	25.000	25.000	25.000	25.000	25.000	25.000	5.872	4.205	3.922	3.697	3.433	3.004	2.591	2.640	2.046	1.574	1.610	1.375	1.004	0.985
725	25.000	25.000	25.000	25.000	25.000	25.000	5.872	5.016	3.979	3.851	3.743	3.633	2.947	3.209	2.482	1.927	1.903	1.469	1.407	1.284
750	25.000	25.000	25.000	25.000	25.000	25.000	25.000	5.016	4.733	3.894	3.907	3.989	3.242	3.163	2.670	2.271	2.343	1.949	1.825	1.550
775	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.402	5.203	4.464	4.467	4.549	4.057	3.812	3.309	2.766	2.687	2.175	1.953	1.780
800	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.402	6.812	4.390	4.675	4.488	4.357	4.323	3.670	3.039	2.978	2.532	2.426	2.287
825	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.402	5.714	4.390	5.042	4.924	4.955	4.534	4.156	3.504	3.686	3.092	2.914	2.498
850	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.402	6.812	4.390	5.294	5.935	5.156	5.170	4.695	3.921	4.106	3.425	3.068	2.849
875	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.812	5.643	5.448	5.935	5.407	5.576	4.338	4.343	4.063	3.720	3.401	3.171
900	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	6.812	7.029	6.546	5.530	5.561	5.758	5.137	5.091	4.938	4.788	4.027	3.787
925	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.029	7.239	6.628	5.967	5.981	5.137	5.678	4.843	4.883	4.860	4.193
950	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.239	6.628	5.967	6.674	6.641	5.496	7.241	6.087	5.217	4.685
975	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.239	6.628	6.660	6.674	5.948	5.902	7.241	6.493	5.217	4.781
1000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.322	6.660	6.674	5.948	5.902	7.241	6.493	6.064	4.886
1025	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.368	7.334	6.595	6.548	6.493	7.163	6.390
1050	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.334	6.595	6.548	6.493	6.470	25.000
1075	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	7.334	7.288	7.241	6.493	25.000	25.000
1100	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000
1125	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000
1150	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000
1175	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000
1200	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000	25.000

Table S3.3: hpatch-fast score by hydrophobic patch area and number of neighbors. The number of occurrences of each patch area and neighbor count for every residue with 20 or fewer neighbors in a set of monomeric proteins was determined using Rosetta. The inverse log of the probabilities was taken to obtain scores. The score for patch areas smaller than the most common patch area was set to the minimum score at that number of neighbors. The entire score was then shifted down so that the minimum hpatch-fast score at each number of neighbors is 0. Patches of size greater than 1100 Å^2 are given a score of 25.

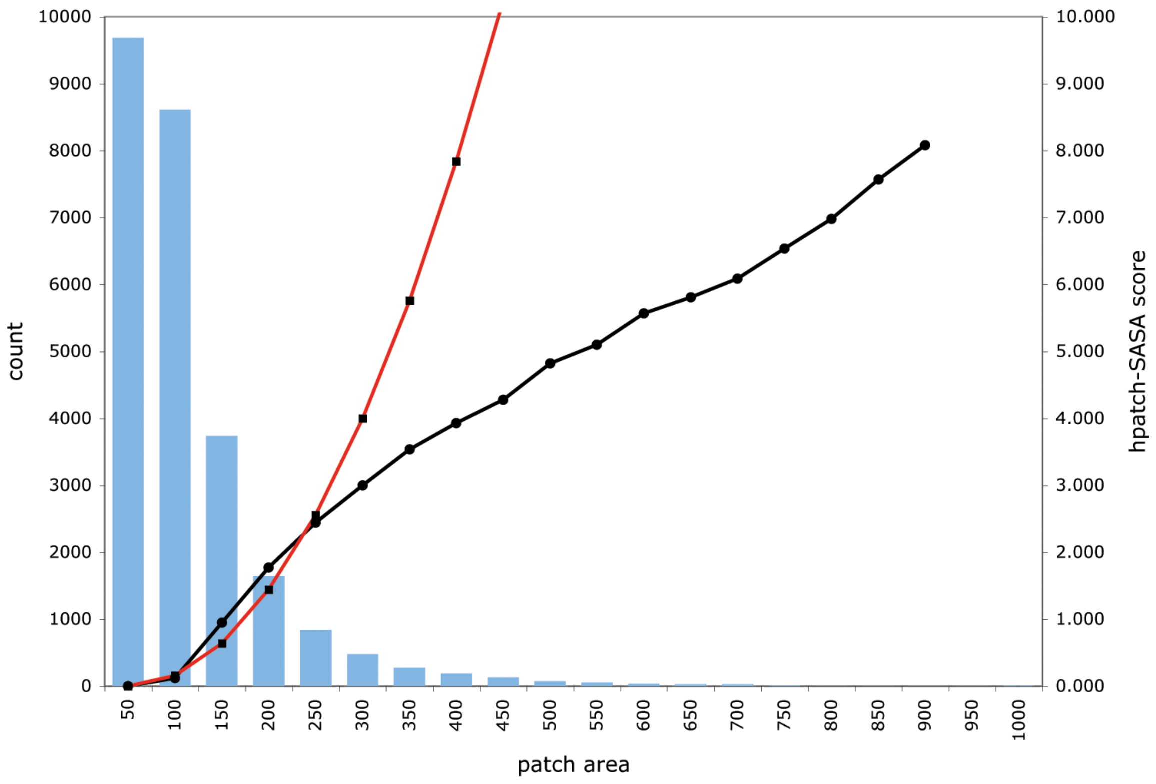


Figure S3.2: Plot of hpatch-SASA score by hydrophobic patch area. Blue bars represent the counts of the indicated patch size found in the set of monomeric structures. The red line is the hpatch-SASA score. The black line shows the score if the inverse log of the probabilities is used. Patches of size greater than 900 Å^2 are given a score of 100.

patch area	count	hpatch-SASA score
0	9692	0.00
50	8613	0.16
100	3740	0.64
150	1645	1.44
200	844	2.56
250	484	4.00
300	281	5.76
350	190	7.84
400	134	10.24
450	78	12.96
500	59	16.00
550	37	19.36
600	29	23.04
650	22	27.04
700	14	31.36
750	9	36.00
800	5	40.96
850	3	46.24
900		100.00
950		100.00

Table S3.4: hpatch-SASA score by hydrophobic patch area. The number of occurrences of each patch area in the set of monomeric proteins was determined using Rosetta. An exponential curve that more strongly penalizes large patches was used instead of the inverse log of the probabilities to obtain the score. The entire score was shifted down so that the minimum hpatch-SASA score is 0. Patches of size 900 Å^2 or greater are given a score of 100.

atom type	vdW radius
CNH2	1.76
COO	1.76
CH1	1.87
CH2	1.87
CH3	1.87
aroC	1.76
Ntrp	1.65
Nhis	1.65
NH2O	1.65
Nlys	1.5
Narg	1.65
Npro	1.65
OH	1.4
ONH2	1.4
OOC	1.4
Oaro	1.4
S	1.85
Nbb	1.65
CAbb	1.87
CObb	1.76
OCbb	1.4

Table S3.5: van der Waals radii used for SASA calculations

AA	fa_atr	fa_rep	fa_sol	fa_intra_rep	pro_close	fa_pair	hbond sr_bb	hbond lr_bb	hbond bb_sc	hbond sc	rama	omega	fa_dun	p_aa_pp
ALA	2.56	-0.57	-1.57	-0.23	0.00	0	0.53	0.027	0.006	0	0.21	-0.15	0	0.16
ARG	3.18	-0.62	-2.22	-2.10	-0.01	0.09	0.49	0.017	0.021	0.013	0.26	-0.17	-2.33	0.10
ASN	2.87	-0.63	-2.41	-1.66	-0.01	0.08	0.39	0.017	0.031	0.006	-0.03	-0.18	-1.17	0.25
ASP	3.00	-0.68	-2.46	-1.50	-0.01	0.10	0.40	0.017	0.042	0.013	0.01	-0.16	-0.87	0.19
CYS	2.54	-0.62	-1.70	-0.40	-0.01	0	0.39	0.032	0.007	0	-0.02	-0.24	-0.55	-0.13
GLN	3.15	-0.67	-2.24	-1.55	-0.01	0.05	0.50	0.011	0.018	0.004	0.23	-0.17	-1.72	0.07
GLU	3.07	-0.63	-2.17	-1.56	0.00	0.14	0.48	0.012	0.015	0.009	0.41	-0.14	-1.72	0.14
GLY	1.69	-0.49	-1.20	-0.02	0.00	0	0.31	0.014	0.009	0	-0.66	-0.15	0	1.06
HIS	3.53	-0.70	-2.35	-0.99	0.00	0.02	0.41	0.017	0.023	0.006	0.03	-0.21	-1.36	0.08
ILE	2.93	-0.61	-1.58	-4.74	-0.01	0	0.44	0.022	0.004	0	0.20	-0.17	-0.49	0.36
LEU	3.24	-0.65	-1.77	-2.05	0.00	0	0.54	0.034	0.005	0	0.13	-0.17	-0.61	0.17
LYS	2.81	-0.66	-1.81	-1.08	0.00	0.12	0.45	0.015	0.009	0.001	0.29	-0.16	-1.53	0.02
MET	3.27	-0.68	-1.87	-1.12	-0.01	0	0.53	0.027	0.005	0	0.10	-0.17	-1.92	0.07
PHE	3.68	-0.73	-1.73	-6.68	-0.01	0	0.43	0.022	0.006	0	0.12	-0.22	-1.03	0.10
PRO	2.23	-1.04	-1.14	-0.36	-0.39	0	0.18	0.002	0.002	0	0.15	-0.18	-0.63	1.13
SER	2.45	-0.62	-1.93	-0.37	-0.01	0.09	0.37	0.018	0.036	0.010	-0.02	-0.19	-0.62	0.08
THR	2.62	-0.61	-2.00	-1.82	-0.01	0.04	0.37	0.011	0.033	0.009	0.02	-0.18	-0.27	0.14
TRP	4.51	-0.87	-2.23	-6.39	-0.01	0	0.43	0.018	0.009	0.001	0.09	-0.21	-1.95	0.04
TYR	3.75	-0.74	-1.92	-6.55	0.00	0	0.41	0.020	0.008	0.003	0.11	-0.23	-0.97	0.10
VAL	2.63	-0.58	-1.46	-3.26	-0.01	0	0.37	0.019	0.004	0	0.22	-0.18	-0.24	0.34

Table S3.6: Unweighted unfolded state energies by energy term for all amino acids. The dot product of the unweighted energies for an amino acid with a set of weights gives the unfolded state energy of that amino acid.

energy term	current Rosetta	current Rosetta, fit refEs only	hpatch EF, reweight refEs only	unfolded EF, score12 unfold-edEs (normal-ized)	unfolded + hpatch EF, score12 unfold-edEs (normal-ized)	standard REF, reweight all terms	hpatch EF, reweight all terms	unfolded EF, reweight all terms (normal-ized)	unfolded + hpatch EF, reweight all terms (normal-ized)
fa_atr	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
fa_rep	0.44	0.44	0.44	0.44	0.44	0.560924	0.437555	0.593519	0.497965
fa_sol	0.65	0.65	0.65	0.65	0.65	0.997569	0.938886	1.33295	1.43525
fa_intra_rep	0.004	0.004	0.004	0.004	0.004	0.1822	0.2196	0.0556	0.5576
pro_close	1	1	1	1	1	0.010	0.008	0.001	0.095
fa_pair	0.49	0.49	0.49	0.49	0.49	0.665853	0.642108	0.457219	0.428062
hbond_sr_bb	0.585	0.585	0.585	0.585	0.585	0.585	0.585	0.585	0.585
hbond_lr_bb	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
hbond_bb_sc	1.17	1.17	1.17	1.17	1.17	3.39777	3.50261	3.21955	3.6468
hbond_sc	1.1	1.1	1.1	1.1	1.1	1.3483	1.23576	1.30316	1.83379
dslf_ss_dst	1	1	1	0.5	1	1	1	1	1
dslf_cs_ang	1	1	1	2	1	1	1	1	1
dslf_ss_dih	1	1	1	5	1	1	1	1	1
dslf_ca_dih	1	1	1	5	1	1	1	1	1
rama	0.2	0.2	0.2	0.2	0.2	0.0045	0.0018	0.0452	0.0381
omega	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
fa_dun	0.56	0.56	0.56	0.56	0.56	0.507731	0.332697	0.282501	0.27986
p_aa_pp	0.32	0.32	0.32	0.32	0.32	0.981461	1.04745	0.949626	1.02642
ref/unfolded	1	1	1	1	1	1	1	1	1
hpatch	—	—	0.2	—	0.3	—	0.3	—	0.5
ALA	0.16	0.12	0.20	0.56	0.56	0.83	0.60	0.52	1.53
CYS	1.7	-0.04	0.07	-0.14	-0.14	0.44	0.36	-0.27	0.63
ASP	-0.67	-0.48	-0.42	-0.22	-0.22	-0.66	-0.45	-0.58	-0.26
GLU	-0.81	-0.61	-0.55	-0.34	-0.34	-0.65	-0.27	-0.42	-0.12
PHE	0.63	1.17	1.01	0.54	0.54	0.26	-0.18	0.29	-1.94
GLY	-0.17	-0.40	-0.27	0.12	0.12	0.42	0.18	1.07	2.29
HIS	0.56	0.98	0.97	-0.13	-0.13	1.02	1.11	-0.36	0.21
ILE	0.24	0.38	0.31	0.53	0.53	-0.03	-0.20	0.50	-0.73
LYS	-0.65	-0.42	-0.41	-0.33	-0.33	-0.28	-0.10	-0.27	0.29
LEU	-0.1	0.20	0.09	0.57	0.57	0.27	-0.03	0.48	0.57
MET	-0.34	-0.20	-0.25	-0.25	-0.25	0.02	0.13	-0.07	0.48
ASN	-0.89	-0.76	-0.72	-0.49	-0.49	-0.87	-0.59	-0.71	-0.47
PRO	0.02	-0.81	-0.89	-0.32	-0.32	0.57	0.88	1.02	2.10
GLN	-0.97	-0.82	-0.62	-0.45	-0.45	-0.80	-0.45	-0.59	-0.29
ARG	-0.98	-0.79	-0.74	-0.69	-0.69	-0.87	-0.47	-0.64	-0.62
SER	-0.37	-0.23	-0.26	-0.24	-0.24	0.27	0.14	-0.32	0.60
THR	-0.27	-0.18	-0.20	0.04	0.04	-0.19	-0.34	-0.22	-0.04
VAL	0.29	0.22	0.25	0.47	0.47	0.15	-0.03	0.53	0.05
TRP	0.91	1.73	1.50	0.28	0.28	0.29	0.12	-0.08	-2.20
TYR	0.51	0.95	0.91	0.48	0.48	-0.18	-0.42	0.11	-2.07
ddG	0.69	0.68	0.681	0.662	0.662	0.609	0.635	0.579	0.436
core	49.2	51.7	52.5	36.3	37.1	51.6	52.3	47.7	48.4
overall	32.9	35.1	35.5	24.0	25.2	35.2	36.5	30.5	29.8
surface	24.9	27.9	27.3	17.6	18.7	27.3	28.9	21.0	19.9
%hp/surface	29.5	31.8	27.1	50.3	35.8	28.9	27.7	46.4	30.3
largest patch	813	694	446	1169	445	735	433	1479	464

Table S3.7: Final weights obtained from weight optimization runs for various energy functions including the hpatch score and/or the unfolded state energy term.

Table S3.8: Sequence recoveries by amino acid for various weight optimized energy functions. Core, overall, and surface native sequence recoveries for a test set of 32 proteins, reported in total and by amino acid type. Core positions are defined as those residues having 24 or more neighbors, and surface positions are defined as having 16 or fewer neighbors.

Recoveries are reported for each of the following energy functions:
current Rosetta - the current Rosetta energy function previously optimized for native sequence recovery only
current Rosetta, fit refEs only - the current Rosetta energy function with the reference energies optimized for native sequence recovery and amino acid composition
current Rosetta + hpatch-SASA - the current Rosetta energy function using the hpatch-SASA score, with the reference energies optimized for native sequence recovery and amino acid composition
current Rosetta, no refE + unfoldedE - the current Rosetta energy function with reference energies replaced by an explicit unfolded state energy term (no optimization)
current Rosetta, no refE + unfoldedE, hpatch-SASA - the current Rosetta energy function with the reference energies replaced by an explicit unfolded state energy term, and using the hpatch-SASA score (no optimization)
standard energy function, reweight all terms - a reweighted Rosetta energy function optimized for native sequence recovery and amino acid composition
standard + hpatch-SASA energy function - a reweighted Rosetta energy function using the hpatch-SASA score, optimized for native sequence recovery and amino acid composition
standard, no refE + unfoldedE - a reweighted Rosetta energy function with the reference energies replaced by an explicit unfolded state energy term, optimized for native sequence recovery and amino acid composition
standard, no refE + unfoldedE, hpatch-SASA - a reweighted Rosetta energy function using the hpatch-SASA score and with the reference energies replaced by an explicit unfolded state energy term, optimized for native sequence recovery and amino acid composition

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
current Rosetta weights												
LEU	66	122	92	0.54	180	378	454	0.48	18	54	174	0.33
GLY	56	69	66	0.81	302	400	368	0.75	164	210	205	0.78
ASP	10	20	29	0.5	80	268	354	0.3	49	167	221	0.29
SER	20	37	106	0.54	79	258	348	0.31	25	119	128	0.21
GLU	2	7	16	0.29	54	289	308	0.19	31	180	158	0.17
PHE	37	60	69	0.62	98	193	302	0.51	6	23	68	0.26
LYS	2	11	9	0.18	62	325	285	0.19	30	193	148	0.16
ARG	4	14	20	0.29	37	185	262	0.2	10	85	76	0.12
ALA	57	103	94	0.55	118	385	259	0.31	9	144	30	0.06
ILE	41	83	70	0.49	105	242	231	0.43	7	33	39	0.21
THR	19	41	62	0.46	61	291	225	0.21	30	145	110	0.21
TYR	9	40	27	0.22	39	158	224	0.25	6	23	98	0.26

Continued on next page

Table S3.8 – continued from previous page

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
VAL	47	98	67	0.48	107	308	194	0.35	8	60	37	0.13
HIS	10	18	41	0.56	30	118	176	0.25	5	37	32	0.14
GLN	3	17	6	0.18	18	186	172	0.1	10	94	110	0.11
ASN	3	14	11	0.21	33	206	158	0.16	20	110	120	0.18
TRP	10	23	21	0.43	32	76	145	0.42	4	12	56	0.33
PRO	6	18	7	0.33	61	229	77	0.27	29	144	40	0.2
MET	10	31	25	0.32	20	90	70	0.22	1	20	5	0.05
CYS	0	12	0	0	0	27	0	0	0	2	0	0
Total	412	838		0.492	1516	4612		0.329	462	1855		0.249

Continued on next page

Table S3.8 – continued from previous page

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
current Rosetta, fit refEs only												
GLY	55	69	71	0.8	311	400	431	0.78	172	210	252	0.82
ALA	67	103	128	0.65	146	385	403	0.38	17	144	100	0.12
VAL	60	98	93	0.61	146	308	364	0.47	18	60	104	0.3
LEU	61	122	84	0.5	170	378	336	0.45	16	54	85	0.3
ASP	9	20	30	0.45	79	268	286	0.29	46	167	158	0.28
GLU	2	7	9	0.29	47	289	278	0.16	26	180	144	0.14
ARG	6	14	22	0.43	36	185	265	0.19	7	85	81	0.08
THR	17	41	51	0.41	75	291	259	0.26	43	145	146	0.3
ILE	45	83	79	0.54	110	242	254	0.45	9	33	48	0.27
LYS	1	11	7	0.09	44	325	233	0.14	19	193	97	0.1
SER	16	37	61	0.43	60	258	229	0.23	19	119	90	0.16
GLN	2	17	7	0.12	20	186	228	0.11	9	94	142	0.1
PRO	14	18	19	0.78	141	229	217	0.62	79	144	143	0.55
TYR	14	40	36	0.35	46	158	214	0.29	4	23	59	0.17
ASN	3	14	7	0.21	37	206	181	0.18	23	110	144	0.21
PHE	31	60	53	0.52	74	193	162	0.38	5	23	21	0.22
HIS	8	18	28	0.44	22	118	97	0.19	2	37	11	0.05
MET	11	31	25	0.35	24	90	90	0.27	1	20	15	0.05
TRP	9	23	14	0.39	26	76	43	0.34	2	12	3	0.17
CYS	2	12	14	0.17	4	27	42	0.15	0	2	12	0
Total	433	838		0.517	1618	4612		0.351	517	1855		0.279
current Rosetta + hpatch-SASA												
GLY	55	69	67	0.8	308	400	402	0.77	168	210	228	0.8
LEU	66	122	96	0.54	182	378	361	0.48	19	54	81	0.35
SER	19	37	88	0.51	84	258	325	0.33	28	119	131	0.24
ALA	64	103	110	0.62	127	385	314	0.33	8	144	59	0.06
GLU	3	7	15	0.43	51	289	312	0.18	27	180	178	0.15
THR	14	41	46	0.34	77	291	306	0.26	44	145	181	0.3
VAL	57	98	88	0.58	129	308	296	0.42	11	60	75	0.18
ASP	10	20	27	0.5	77	268	289	0.29	47	167	176	0.28
LYS	2	11	11	0.18	50	325	285	0.15	19	193	120	0.1
ARG	4	14	14	0.29	35	185	263	0.19	9	85	97	0.11
ILE	45	83	80	0.54	114	242	248	0.47	8	33	38	0.24
ASN	2	14	5	0.14	40	206	227	0.19	23	110	182	0.21
PRO	13	18	17	0.72	138	229	213	0.6	76	144	140	0.53
TYR	14	40	27	0.35	46	158	208	0.29	6	23	66	0.26
PHE	38	60	56	0.63	87	193	192	0.45	4	23	19	0.17
HIS	7	18	26	0.39	22	118	106	0.19	2	37	16	0.05
MET	10	31	28	0.32	21	90	88	0.23	1	20	7	0.05
GLN	3	17	3	0.18	10	186	78	0.05	3	94	44	0.03
TRP	13	23	23	0.57	34	76	70	0.45	3	12	9	0.25
CYS	1	12	11	0.08	4	27	29	0.15	1	2	8	0.5
Total	440	838		0.525	1636	4612		0.355	507	1855		0.273

Continued on next page

Table S3.8 – continued from previous page

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
current Rosetta, no refE + unfoldedE												
TRP	14	23	44	0.61	48	76	878	0.63	10	12	517	0.83
HIS	8	18	127	0.44	45	118	816	0.38	12	37	344	0.32
SER	20	37	166	0.54	75	258	445	0.29	20	119	117	0.17
GLY	50	69	60	0.72	292	400	356	0.73	162	210	197	0.77
PHE	29	60	53	0.48	82	193	304	0.42	4	23	100	0.17
TYR	10	40	21	0.25	25	158	182	0.16	3	23	88	0.13
VAL	35	98	54	0.36	78	308	167	0.25	6	60	32	0.1
ILE	28	83	45	0.34	73	242	161	0.3	5	33	28	0.15
MET	9	31	29	0.29	23	90	159	0.26	3	20	47	0.15
ASP	8	20	31	0.4	42	268	156	0.16	20	167	60	0.12
ARG	3	14	15	0.21	21	185	155	0.11	6	85	57	0.07
ALA	38	103	53	0.37	75	385	142	0.19	5	144	13	0.03
THR	13	41	46	0.32	34	291	126	0.12	11	145	33	0.08
GLU	1	7	18	0.14	16	289	104	0.06	5	180	28	0.03
LEU	22	122	27	0.18	62	378	103	0.16	2	54	20	0.04
PRO	8	18	11	0.44	73	229	103	0.32	37	144	59	0.26
LYS	2	11	4	0.18	19	325	97	0.06	8	193	50	0.04
CYS	1	12	27	0.08	3	27	96	0.11	1	2	29	0.5
ASN	2	14	4	0.14	14	206	40	0.07	6	110	24	0.05
GLN	3	17	3	0.18	5	186	22	0.03	1	94	12	0.01
Total	304	838		0.363	1105	4612		0.24	327	1855		0.176
current Rosetta, no refE + unfoldedE, hpatch-SASA												
HIS	8	18	121	0.44	57	118	1016	0.48	16	37	472	0.43
TRP	11	23	40	0.48	43	76	693	0.57	8	12	383	0.67
SER	22	37	163	0.59	84	258	463	0.33	24	119	140	0.2
GLY	52	69	63	0.75	294	400	358	0.74	163	210	199	0.78
ARG	6	14	16	0.43	28	185	218	0.15	7	85	103	0.08
TYR	8	40	24	0.2	30	158	218	0.19	5	23	95	0.22
ASP	8	20	38	0.4	49	268	192	0.18	28	167	86	0.17
PHE	28	60	49	0.47	74	193	188	0.38	5	23	36	0.22
GLU	2	7	20	0.29	32	289	176	0.11	11	180	66	0.06
ILE	29	83	49	0.35	76	242	151	0.31	4	33	15	0.12
VAL	37	98	58	0.38	80	308	150	0.26	3	60	26	0.05
ALA	36	103	49	0.35	73	385	135	0.19	4	144	12	0.03
THR	15	41	41	0.37	41	291	133	0.14	14	145	40	0.1
PRO	7	18	9	0.39	72	229	99	0.31	34	144	55	0.24
LYS	2	11	2	0.18	21	325	92	0.06	8	193	46	0.04
LEU	27	122	34	0.22	65	378	91	0.17	3	54	8	0.06
MET	8	31	26	0.26	17	90	86	0.19	2	20	16	0.1
CYS	1	12	27	0.08	4	27	82	0.15	0	2	19	0
ASN	2	14	6	0.14	16	206	40	0.08	7	110	22	0.06
GLN	2	17	3	0.12	4	186	31	0.02	1	94	16	0.01
Total	311	838		0.371	1160	4612		0.252	347	1855		0.187

Continued on next page

Table S3.8 – continued from previous page

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
standard energy function, reweight all terms												
GLY	55	69	79	0.8	309	400	420	0.77	169	210	223	0.8
ALA	73	103	139	0.71	138	385	370	0.36	10	144	47	0.07
VAL	70	98	121	0.71	147	308	358	0.48	15	60	75	0.25
LEU	67	122	96	0.55	181	378	358	0.48	14	54	82	0.26
GLU	3	7	10	0.43	64	289	296	0.22	38	180	196	0.21
PRO	12	18	21	0.67	155	229	290	0.68	87	144	178	0.6
ASP	6	20	22	0.3	74	268	285	0.28	45	167	182	0.27
LYS	1	11	11	0.09	48	325	279	0.15	17	193	110	0.09
ILE	46	83	67	0.55	112	242	262	0.46	6	33	47	0.18
THR	12	41	34	0.29	74	291	242	0.25	44	145	137	0.3
ARG	3	14	16	0.21	28	185	242	0.15	8	85	90	0.09
SER	19	37	72	0.51	68	258	234	0.26	10	119	62	0.08
ASN	3	14	7	0.21	46	206	207	0.22	24	110	163	0.22
GLN	3	17	8	0.18	14	186	198	0.08	6	94	134	0.06
PHE	31	60	55	0.52	83	193	162	0.43	5	23	14	0.22
TYR	6	40	16	0.15	26	158	158	0.16	5	23	63	0.22
MET	11	31	36	0.35	19	90	102	0.21	1	20	12	0.05
HIS	3	18	10	0.17	15	118	74	0.13	2	37	22	0.05
TRP	8	23	11	0.35	22	76	48	0.29	3	12	12	0.25
CYS	0	12	7	0	1	27	27	0.04	0	2	6	0
Total	432	838		0.516	1624	4612		0.352	509	1855		0.274
standard + hpatch-SASA energy function												
LEU	79	122	123	0.65	205	378	429	0.54	18	54	96	0.33
GLY	53	69	72	0.77	299	400	405	0.75	165	210	225	0.79
SER	21	37	79	0.57	88	258	333	0.34	25	119	127	0.21
ALA	68	103	107	0.66	133	385	303	0.35	13	144	56	0.09
THR	17	41	45	0.41	84	291	293	0.29	50	145	178	0.34
VAL	53	98	84	0.54	128	308	288	0.42	12	60	64	0.2
LYS	2	11	10	0.18	65	325	288	0.2	24	193	104	0.12
ASP	6	20	25	0.3	80	268	278	0.3	47	167	164	0.28
GLU	3	7	10	0.43	58	289	269	0.2	36	180	178	0.2
ILE	49	83	85	0.59	114	242	256	0.47	4	33	39	0.12
ARG	4	14	20	0.29	27	185	235	0.15	6	85	79	0.07
PRO	12	18	19	0.67	146	229	229	0.64	85	144	141	0.59
GLN	3	17	11	0.18	16	186	191	0.09	7	94	124	0.07
PHE	36	60	63	0.6	92	193	188	0.48	4	23	20	0.17
ASN	3	14	8	0.21	50	206	182	0.24	28	110	131	0.25
TYR	7	40	15	0.17	31	158	165	0.2	4	23	64	0.17
HIS	7	18	20	0.39	24	118	116	0.2	4	37	32	0.11
MET	9	31	27	0.29	16	90	80	0.18	0	20	5	0
TRP	6	23	11	0.26	26	76	68	0.34	4	12	22	0.33
CYS	0	12	4	0	0	27	16	0	0	2	6	0
Total	438	838		0.523	1682	4612		0.365	536	1855		0.289

Continued on next page

Table S3.8 – continued from previous page

Residue	No. correct core	No. native core	No. designed core	No. correct/ No. native core	No. correct	No. native	No. designed	No. correct/ No. native	No. correct surface	No. native surface	No. designed surface	No. correct/ No. native surface
standard, no refE + unfoldedE												
TRP	6	23	12	0.26	22	76	480	0.29	4	12	314	0.33
ALA	78	103	177	0.76	155	385	451	0.4	11	144	44	0.08
HIS	4	18	22	0.22	31	118	438	0.26	10	37	275	0.27
PHE	41	60	94	0.68	123	193	428	0.64	7	23	85	0.3
GLY	50	69	61	0.72	284	400	345	0.71	160	210	190	0.76
LEU	59	122	89	0.48	171	378	328	0.45	10	54	46	0.19
ILE	47	83	95	0.57	113	242	321	0.47	5	33	53	0.15
SER	18	37	74	0.49	68	258	282	0.26	13	119	110	0.11
MET	8	31	40	0.26	20	90	250	0.22	3	20	83	0.15
VAL	53	98	93	0.54	111	308	232	0.36	9	60	33	0.15
PRO	12	18	19	0.67	139	229	224	0.61	74	144	127	0.51
ARG	3	14	8	0.21	16	185	142	0.09	5	85	71	0.06
LYS	1	11	3	0.09	27	325	120	0.08	13	193	64	0.07
TYR	4	40	9	0.1	12	158	104	0.08	2	23	52	0.09
THR	6	41	14	0.15	34	291	102	0.12	18	145	52	0.12
ASN	1	14	1	0.07	21	206	93	0.1	14	110	77	0.13
GLN	2	17	2	0.12	13	186	88	0.07	6	94	69	0.06
CYS	1	12	18	0.08	2	27	67	0.07	0	2	24	0
GLU	2	7	2	0.29	20	289	65	0.07	12	180	53	0.07
ASP	4	20	5	0.2	23	268	52	0.09	13	167	33	0.08
Total	400	838		0.477	1405	4612		0.305	389	1855		0.21
standard, no refE + unfoldedE, hpatch-SASA												
HIS	4	18	17	0.22	32	118	495	0.27	11	37	308	0.3
ALA	76	103	181	0.74	150	385	462	0.39	7	144	42	0.05
GLY	49	69	61	0.71	282	400	348	0.7	158	210	189	0.75
SER	17	37	80	0.46	74	258	339	0.29	15	119	112	0.13
TRP	4	23	15	0.17	23	76	313	0.3	5	12	168	0.42
LEU	60	122	88	0.49	176	378	301	0.47	10	54	29	0.19
ILE	52	83	95	0.63	114	242	298	0.47	6	33	52	0.18
PHE	39	60	75	0.65	101	193	293	0.52	5	23	51	0.22
ARG	3	14	5	0.21	20	185	247	0.11	10	85	153	0.12
VAL	59	98	97	0.6	115	308	237	0.37	8	60	30	0.13
GLN	3	17	4	0.18	17	186	213	0.09	9	94	174	0.1
MET	10	31	40	0.32	19	90	161	0.21	0	20	32	0
TYR	7	40	15	0.17	19	158	158	0.12	5	23	73	0.22
PRO	10	18	16	0.56	98	229	134	0.43	50	144	69	0.35
THR	7	41	16	0.17	38	291	130	0.13	18	145	67	0.12
GLU	1	7	3	0.14	21	289	117	0.07	12	180	88	0.07
ASP	3	20	5	0.15	29	268	104	0.11	18	167	77	0.11
ASN	1	14	2	0.07	27	206	101	0.13	16	110	85	0.15
LYS	1	11	3	0.09	19	325	94	0.06	6	193	40	0.03
CYS	0	12	20	0	1	27	67	0.04	1	2	16	0.5
Total	406	838		0.484	1375	4612		0.298	370	1855		0.199

```
Command lines used in this work:
Design runs
mini/bin/fixbb.macosgccrelease
-database minirosetta_database
-s 1FKB.pdb
-ex1 -ex2 -extrachi_cutoff 0 -linmem_ig 10
-ignore_unrecognized_res -no_optH false -skip_set_reasonable_fold_tree -no_his_his_pairE
-mute core.io core.scoring core.conformation
[-score:weights ./hpatch_weights.txt]

Weight optimization runs
mpirun mini/bin/surface_optE_parallel.linuxgccrelease
-database minirosetta_database
-s nataa_recovery_pdbids.test.list
-optE:optimize_nat_aa true
-optE:fit_reference_energies_to_aa_profile_recovery true
-optE:n_design_cycles 10
-optE:mpi_weight_minimization true
-optE:optimize_starting_free_weights true
[ -optE:rescore:weights weights.txt -optE:rescore:outlog rescore.log ]
[ -optE:rescore:measure_sequence_recovery true ]
-optE:number_of_swarm_particles 75 -optE:number_of_swarm_cycles 30
-optE:repeat_swarm_optimization_until_fitness_improves true
-optE:free free_wts.txt -optE:fixed fixed_wts.txt
-ignore_unrecognized_res -no_optH false -skip_set_reasonable_fold_tree -no_his_his_pairE
-mute core.io core.pack core.scoring core.conformation
-ex1 -ex2 -extrachi_cutoff 0 -linmem_ig 10 -options:user

free.txt          fixed.txt
fa_rep            fa_atr 0.8
fa_sol            omega 0.5
fa_intra_rep      hbond_lr_bb 1.17
pro_close         hbond_sr_bb 0.585
fa_pair           dslf_ss_dst 1.0
hbond_bb_sc       dslf_cs_ang 1.0
hbond_sc          dslf_ss_dih 1.0
rama              dslf_ca_dih 1.0
fa_dun            hpatch 0.3
p_aa_pp

QUILT runs
quilt n 252 ep 1.4 R 1FKB.pdb

Sequence recovery runs
~/minibin/sequencerecovery.macosgccrelease -database ~/minidb/
-native_pdb_list seqrecovery.list -redesign_pdb_list redesigned.list
-ignore_unrecognized_res -mute core

Scoring runs
~/minibin/score.macosgccrelease -database ~/minidb/
-l redesigns.list
-ignore_unrecognized_res mute core
[-score:weights design_hpatch.wts]
```

References

1. Schreiber, G., Buckle, A., and Fersht, A. (1994) Stability and function: two constraints in the evolution of barstar and other proteins. Structure **2**, 945–951. ISSN 0969-2126

2. Poso, D., Sessions, R., Lorch, M., and Clarke, A. (2000) Progressive stabilization of intermediate and transition states in protein folding reactions by introducing surface hydrophobic residues. Journal of Biological Chemistry **275**, 35723. ISSN 0021-9258

3. Schindler, T., Perl, D., Graumann, P., Sieber, V., Marahiel, M., and Schmid, F. (1998) Surface-exposed phenylalanines in the RNP1/RNP2 motif stabilize the cold-shock protein CspB from Bacillus subtilis. Proteins: Structure, Function, and Bioinformatics **30**, 401–406. ISSN 1097-0134

4. Dantas, G., Corrent, C., Reichow, S., Havranek, J., Eletr, Z., Isern, N., Kuhlman, B., Varani, G., Merritt, E., and Baker, D. (2007) High-resolution structural and thermodynamic analysis of extreme stabilization of human procarboxypeptidase by computational protein design. Journal of molecular biology **366**, 1209–1221. ISSN 0022-2836

5. Lawrence, M., Phillips, K., and Liu, D. (2007) Supercharging proteins can impart unusual resilience. Journal of the American Chemical Society **129**, 10110–10112. ISSN 0002-7863

6. Jones, S. and Thornton, J. (1996) Principles of protein-protein interactions. Proceedings of the National Academy of Sciences of the United States of America **93**, 13

7. Janin, J., Miller, S., and Chothia, C. (1988) Surface, subunit interfaces and interior of oligomeric proteins. Journal of molecular biology **204**, 155–164. ISSN 0022-2836

8. Chiti, F., Stefani, M., Taddei, N., Ramponi, G., and Dobson, C. (2003) Rationalization of the effects of mutations on peptide andprotein aggregation rates. Nature **424**, 805–808. ISSN 0028-0836

9. Jaramillo, A., Wernisch, L., Héry, S., and Wodak, S. (2002) Folding free energy function selects native-like protein sequences in the core but not on the surface. Proceedings of the National Academy of Sciences of the United States of America **99**, 13554

10. Pokala, N. and Handel, T. (2005) Energy functions for protein design: adjustment with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. Journal of molecular biology **347**, 203–227. ISSN 0022-2836

11. Dahiyat, B., Benjamin Gordon, D., and Mayo, S. (1997) Automated design of the surface positions of protein helices. Protein science **6**, 1333–1337. ISSN 1469-896X

12. Dahiyat, B., Sarisky, C., and Mayo, S. (1997) De novo protein design: towards fully automated sequence selection. Journal of molecular biology **273**, 789–796. ISSN 0022-2836

13. Dahiyat, B. and Mayo, S. (1997) Probing the role of packing specificity in protein design. Proceedings of the National Academy of Sciences of the United States of America **94**, 10172

14. Sun, S., Brem, R., Chan, H., and Dill, K. (1995) Designing amino acid sequences to fold with good hydrophobic cores. Protein Engineering **8**, 1205. ISSN 1741-0126

15. Wernisch, L., Hery, S., and Wodak, S. (2000) Automatic protein design with all atom force-fields by exact and heuristic optimization1. Journal of Molecular Biology **301**, 713–736. ISSN 0022-2836

16. Havranek, J. and Harbury, P. (2002) Automated design of specificity in molecular recognition. Nature Structural & Molecular Biology **10**, 45–52

17. Alvizo, O. and Mayo, S. (2008) Evaluating and optimizing computational protein design force fields using fixed composition-based negative design. Proceedings of the National Academy of Sciences **105**, 12242

18. Koehl, P. and Levitt, M. (1999) De novo protein design. I. in search of stability and specificity1. Journal of molecular biology **293**, 1161–1181. ISSN 0022-2836

19. Kuhlman, B. and Baker, D. (2000) Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America **97**, 10383

20. Kuhlman, B., Dantas, G., Ireton, G., Varani, G., Stoddard, B., and Baker, D. (2003) Design of a novel globular protein fold with atomic-level accuracy. Science **302**, 1364. ISSN 0036-8075

21. Rohl, C., Strauss, C., Misura, K., and Baker, D. (2004) Protein structure prediction using Rosetta. Methods in enzymology **383**, 66–93. ISSN 0076-6879

22. Leaver-Fay, A., Tyka, M., Lewis, S., Lange, O., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P., Smith, C., Sheffler, W., et al. (2011) ROSETTA3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. Methods in enzymology **487**, 545. ISSN 1557-7988

23. Lazaridis, T. and Karplus, M. (1999) Effective energy function for proteins in solution. Proteins: Structure, Function, and Bioinformatics **35**, 133–152. ISSN 1097-0134

24. Simons, K., Ruczinski, I., Kooperberg, C., Fox, B., Bystroff, C., and Baker, D. (1999) Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. Proteins: Structure, Function, and Bioinformatics **34**, 82–95. ISSN 1097-0134

25. Kortemme, T., Morozov, A., and Baker, D. (2003) An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein-protein complexes. Journal of molecular biology **326**, 1239–1259. ISSN 0022-2836

26. Dunbrack Jr, R. and Cohen, F. (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Science **6**, 1661–1681. ISSN 1469-896X

27. Krissinel, E. and Henrick, K. (2007) Inference of macromolecular assemblies from crystalline state. Journal of molecular biology **372**, 774–797. ISSN 0022-2836

28. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., and Bourne, P. (2000) The protein data bank. Nucleic acids research **28**, 235. ISSN 0305-1048

29. Li, W. and Godzik, A. (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics **22**, 1658. ISSN 1367-4803

30. Leaver-Fay, A., Butterfoss, G., Snoeyink, J., and Kuhlman, B. (2007) Maintaining solvent accessible surface area under rotamer substitution for protein design. Journal of Computational Chemistry **28**, 1336–1341. ISSN 1096-987X

31. Le Grand, S. and Merz Jr, K. (1993) Rapid approximation to molecular surface area via the use of Boolean logic and look-up tables. Journal of computational chemistry **14**,

349–352. ISSN 1096-987X

32. Lijnzaad, P., Berendsen, H., and Argos, P. (1996) A method for detecting hydrophobic patches on protein surfaces. Proteins: Structure, Function, and Bioinformatics **26**, 192–203. ISSN 1097-0134

33. Galler, B. and Fisher, M. (1964) An improved equivalence algorithm. Communications of the ACM **7**, 301–303. ISSN 0001-0782

34. Chothia, C. (1976) The nature of the accessible and buried surfaces in proteins. Journal of molecular biology **105**, 1–12. ISSN 0022-2836

35. Wang, G. and Dunbrack, R. (2003) PISCES: a protein sequence culling server. Bioinformatics **19**, 1589. ISSN 1367-4803

36. Kellogg, E., Leaver-Fay, A., and Baker, D. (2011) Role of conformational sampling in computing mutation-induced changes in protein structure and stability. Proteins: Structure, Function, and Bioinformatics ISSN 1097-0134

37. Yin, S., Ding, F., and Dokholyan, N. (2007) Modeling backbone flexibility improves protein stability estimation. Structure **15**, 1567–1576. ISSN 0969-2126

38. Guerois, R., Nielsen, J., and Serrano, L. (2002) Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. Journal of molecular biology **320**, 369–387. ISSN 0022-2836

39. DeLano, W. (2002) The PyMOL molecular graphics system

40. Creamer, T., Srinivasan, R., and Rose, G. (1995) Modeling unfolded states of peptides and proteins. Biochemistry **34**, 16245–16250. ISSN 0006-2960

41. Correia, B., Ban, Y., Friend, D., Ellingson, K., Xu, H., Boni, E., Bradley-Hewitt, T., Bruhn-Johannsen, J., Stamatatos, L., Strong, R., et al. (2010) Computational Protein Design Using Flexible Backbone Remodeling and Resurfacing: Case Studies in Structure-Based Antigen Design. Journal of Molecular Biology ISSN 0022-2836

42. Hu, X., Wang, H., Ke, H., and Kuhlman, B. (2008) Computer-Based Redesign of a [beta] Sandwich Protein Suggests that Extensive Negative Design Is Not Required for De Novo [beta] Sheet Design. Structure **16**, 1799–1805. ISSN 0969-2126

43. 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. Journal of molecular biology **332**, 449–460. ISSN 0022-2836

44. Chennamsetty, N., Voynov, V., Kayser, V., Helk, B., and Trout, B. (2009) Design of therapeutic proteins with enhanced stability. Proceedings of the National Academy of Sciences **106**, 11937. ISSN 0027-8424

Chapter 4

A Generic Program for Multistate Protein Design

The text of this chapter is adapted from a submitted manuscript that was co-authored with Andrew Leaver-Fay, Ben Stranges, and Brian Kuhlman

4.1 Abstract

Multistate protein design optimizes a single protein sequence in multiple contexts given by multiple structures (states). After a sequence has been imposed on each of the states, and the rotamers on those states have been optimized, a fitness function must be used to capture how well that sequence meets the design goals. The fitness function guides the search through sequence space. We present here an implementation of multistate design where the user may specify their fitness function from a text file. We test the implementation *in silico* with the orthogonal interface redesign problem: redesigning a promiscuous protein to bind only a single partner. We choose the RalA signaling network as the model system and make our design goal a redesigned RalA that only interacts with the effector RalBP1 and not the other effectors, Sec5 and Exo84. Negative design is used to explicitly disfavor binding between RalA/Sec5 and RalA/Exo84. We find that multistate design is able to recapitulate experimentally characterized mutations known to disrupt Ral-effector interactions and predict many new mutations that alter binding specificity. Single state design for Ral/RalBP1 does not significantly affect the Ral/Sec5 and Ral/Exo84 interactions; only multistate design is able to destabilize both

negative states. Finally, we observe that expanding the set of negative state structures by computationally redocking the structures output from design leads to greater accuracy in predicting negative state binding energies. The paper concludes with a discussion of some of the challenges we encountered while performing negative design.

4.2 Introduction

In single-state protein design, the sequence of one or more proteins is optimized to minimize the energy of a single conformation. Computational approaches for this optimization typically search through sidechain sequence and conformation space simultaneously(1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12). In multistate protein design, the sequence of one or more proteins is optimized so that they will behave in different ways in different contexts. Computational approaches for this optimization typically divide the search through sequence space and the search through conformation space(13; 14; 15; 16; 17; 18); the sequence space search is performed in an outer loop, and rotamer optimization for the outer-loop sequence is performed in an inner loop. Formulation of the multistate design optimization problem is considerably more complicated than in single-state design.

In multistate design, a fitness function must capture the quality of a sequence mathematically from the optimal energies of each of the states being modeled. Havranek and Harbury maximized the probability of homodimer formation for a pair of coiled coils where their partition function included competing homodimer states as well as an aggregate state(13). Ambroggio and Kuhlman optimized the sum of the energies of two conformations for a single sequence so that it would form a monomer in the presence of zinc, and a trimer in its absence(14). Grigoryan, Reinke, and Keating optimized the energy of basic-region leucine zipper (bZIP) peptide heterodimerization under the constraint that the energy gap between heterodimers and homodimers exceed some threshold(19). Ashworth et al. optimized the specificity of the I-Msol1 homoendonuclease by favoring the binding energy for I-Msol1 to the target DNA sequence over alternate DNA sequences(18). A serious challenge in each of these design tasks was formulating the correct fitness function to produce the desired protein behaviors.

This chapter presents a multistate design implementation for solving arbitrary multistate design problems: the software is generic in that it allows the user to program their fitness function from a text file, encouraging the user to search through fitness-function space, and not just sequence space. We test our multistate design implementation with the orthogonal interface design problem where promiscuous protein A, which naturally binds proteins B, C and D, must be redesigned so that A continues to bind B, but no longer binds C or D. We demonstrate, *in silico*, that multistate design does a better job than single-state design in preserving the AB binding energy while decreasing the AC and AD binding energies.

To succeed in obtaining specificity for the desired interaction, explicit destabilization, or negative design, of some of the states must be used. To properly perform negative design, we present an iterative approach where, after each round of multistate sequence design, we redock the negative states (the AC and the AD species) to relieve strain introduced during design. The resulting redocked conformations are then fed back into the next round of design, expanding the pool of negative states that should be designed against. Due to the ease with which states can be added to the design problem, and their energies managed through the fitness function definition, we are able to demonstrate *in silico* what had previously been hypothesized about this approach (13): that representing many conformations for the negative states improves the accuracy of multistate design.

The Ral signaling network was selected as the model system for the interface redesign task. Ral is a small GTPase protein that is involved in a wide variety of cellular functions including endocytosis, transport and tethering of secretory vesicles to the plasma membrane, regulation of transcription, and maintenance of the cytoskeleton, among many others (22) (Figure 4.1). Ral exists in two isoforms, RalA and RalB, which are 82% identical, and has five known effectors: RalBP1, Sec5, Exo84, Filamin, and ZONAB. The Ral signaling network is an attractive system for testing the multistate design protocol for two reasons. First, structures of RalB in complex with RalBP1 (PDB: 2KWI) (23) and RalA in complex with Sec5 (PDB: 1UAD) (24) and Exo84 (PDB: 1ZC3) (25) have been solved. Second, some amino acid positions on Ral are contacted by more than one effector, making it a nontrivial design problem. We decided to make our design goal to redesign RalA so that it only binds to RalBP1 and not the

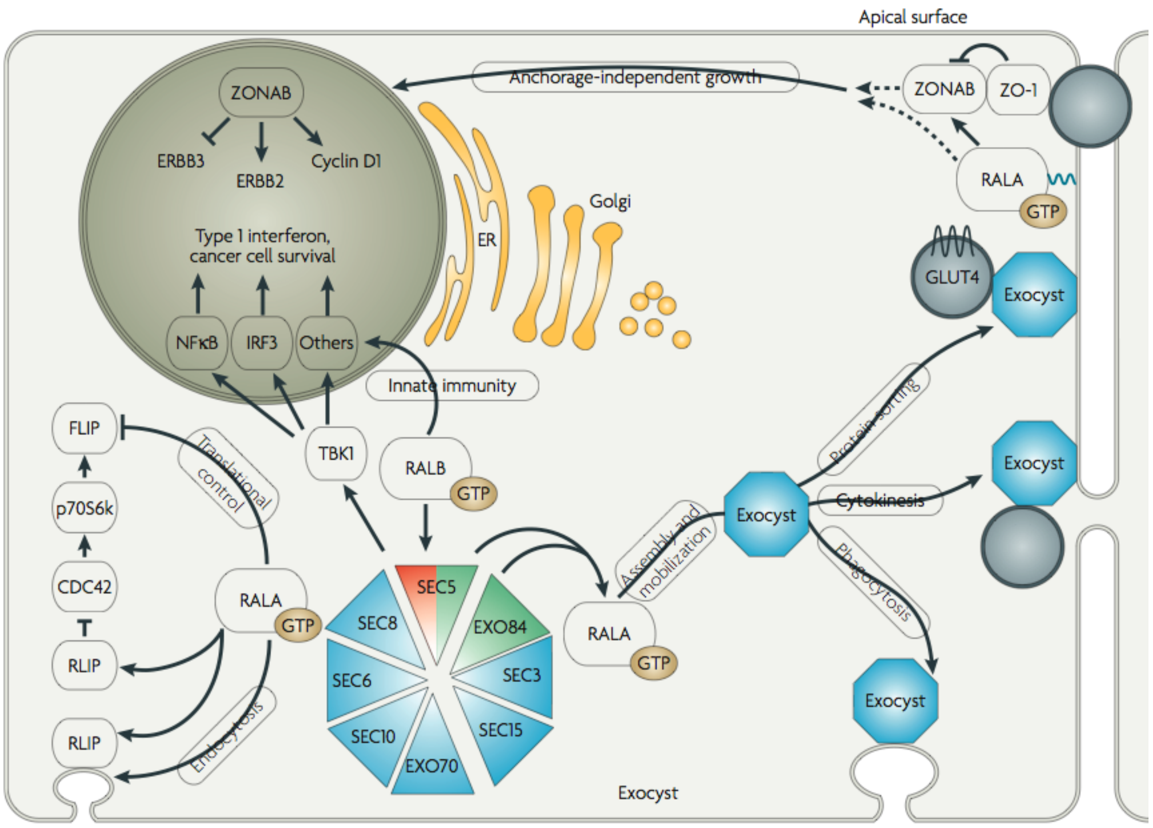


Figure 4.1: RalA signaling interactions and associated downstream functions. All effectors shown interact with RalA in a GTP-dependent manner. Sec5 and Exo84 are subunits in the heterooctameric exocyst complex, which delivers secretory vesicles to the plasma membrane. RalBP1 has been implicated in receptor-mediated endocytosis from its interaction with POB1 and Reps1 (20). ZONAB is a transcription factor which regulates the expression of genes containing inverted CCAAT boxes in their promoters(21). Not pictured is the interaction between RalA and Filamin A. Figure taken from reference (22).

other effectors. Given the flexibility of our method, the goal could be easily changed to design for one of the other effectors. We show that by using multistate design, we can recover the known effector domain mutants of Ral and propose new RalBP1-specific mutants for testing.

4.3 Methods

4.3.1 Software

Our software separates its search through sequence space and its search through conformation space. A genetic algorithm explores sequence space in an outer loop, and each state optimizes its rotamers for a given sequence in an inner loop. The energies produced in this inner loop are fed to a user-defined fitness function that guides the genetic algorithm's search through sequence space. To keep simulations fast, the implementation uses MPI to distribute the inner-loop calculations across multiple processors. The software is written as part of the ROSETTA3 molecular modeling suite (26) and will be available in the 3.3 release. We rely on Rosetta's standard "score12" score function (27) and refer to units of this score function when referring to Rosetta Energy Units (REU).

Genetic Algorithm

The genetic algorithm, described first in the context of mulitistate design by Havranek and Harbury (13) and whose implementation comes from Ashworth et al.(18), maintains a population of 100 sequences and is run for $15 * |seq|$ generations, where $|seq|$ is the length of the sequence being designed (i.e. the number of positions being mutated). Between generation i and generation $i+1$, the genetic algorithm propagates the 50 sequences with the best (lowest) fitness, and generates 50 new sequences with 98% generated as point mutants from the best 50 sequences of the previous generation, and 2% generated as crossover combinations of existing sequences. These parameters were chosen by testing the algorithm at interface sequence recovery with a fitness function described by the energy of the complex – effectively, single-state design. These parameters yielded energies and sequence recovery rates comparable to Rosetta's existing single-state design algorithm (data not shown).

State Definition

A state in our implementation refers to one of the many possible structures on which a sequence is being optimized. Each state is defined by three things: 1) a fixed backbone scaffold, 2) a mapping between some or all of the residues on this scaffold and positions in the sequence being optimized in the outer loop, and 3) a secondary packing file. The fixed backbone scaffold is given by a PDB file; the sidechains present on this backbone at repackable positions can be included in the rotamer set if the user desires. The mapping is given in a correspondence file that lists which residues on the scaffold follow which positions in the sequence optimized by the genetic algorithm. Each of the residues listed in the correspondence file are repacked in each iteration through the outer-loop. The secondary packing file defines which additional residues besides those listed in the correspondence file are allowed to repack; this is also referred to as the secondary resfile, as Rosetta's standard input file for describing how residues are allowed to change is called a resfile. Examples of state definition files are given in the Supplementary Materials.

Fixed Sequence Sidechain Optimization

At the start of execution, the program builds a fixed set of rotamers for all allowed amino acids at each residue for each state. When a particular sequence is assigned to a state, the program selects the appropriate subset of rotamers and performs rotamer optimization with this subset. It uses a slight variation on the original FASTER algorithm(10) of first assigning the backbone-minimum-energy conformation (BMEC) and then performing iterative single-residue perturbation / relaxation (sPR) until convergence. It incorporates a performance enhancement of only relaxing the ten neighbors of the perturbation residue that have the greatest-magnitude-interaction energies with the perturbed rotamer (28).

Over the course of a multistate design trajectory, rotamer-pair energies are computed as needed and stored in an interaction graph data structure for reuse (29; 11; 30) instead of all being computed up-front; this saves roughly 25% of the pair energy calculations and the memory needed to store those pair energies. Optionally, the user may set a ceiling on the amount of memory dedicated toward pair energy storage. The interaction graph storing pair energies for reuse honors that ceiling by discarding submatrices of rotamer-pair energies for particular amino-acid pair interactions; it maintains a binary heap of amino-acid-pair-submatrix-access orders and, when discarding a submatrix, chooses the submatrix whose most recent access was furthest in the past. This behavior means that the same rotamer-pair energies may be computed multiple times.

Fitness Function Definition

The genetic algorithm evaluates the fitness function once for every sequence it examines; the format for the fitness file is geared toward describing how the energies of all the states being modeled, once they have been repacked with a particular sequence, should be combined to compute the fitness for that sequence. This file has two responsibilities: state declaration and fitness-function specification. The fitness-function-definition file format provides seven commands to meet these two responsibilities. The seven commands are STATE, STATE_VECTOR, VECTOR_VARIABLE, SCALAR_EXPRESSION, VECTOR_EXPRESSION, ENTITY_FUNCTION and FITNESS.

Four of the commands rely on a common expression syntax, which allows arbitrary complicated combinations of addition, subtraction, multiplication, division and (predefined) function evaluations. The set of predefined functions available are: abs, exp, ln, vmax, vmin, pow, sqrt, eq, gt, gte, lt, lte, and, or, not, and ite. The vmin and vmax functions are the vector-minimum and vector-maximum functions, returning the smallest or largest element from a vector expression. The pseudo-boolean logic functions (e.g. gt is "greater than") return 0.0 for false and 1.0 for true. The if-then-else function (ite) takes three arguments: if the first argument evaluates to a non-zero value, then it returns the value of the second argument, else, it returns the value of the third argument. New functions can be easily added, but their addition would require recompiling.

The STATE and the STATE_VECTOR commands define states that are to be repacked in each iteration through the outer-most loop. The syntax for a STATE command is:

```
STATE <varname> <pdbfile> <correlationfile> <secondaryresfile>
```

which both declares that a particular state should be modeled and declares the scalar variable with the name `varname` which will be assigned the energy of that state after that state is repacked. This variable can then be used in subsequent expressions. The STATE_VECTOR command is:

```
STATE_VECTOR <varname> <statefile>
```

where each line of the specified state file should contain an ordered triple with the names of 1) a PDB file, 2) a correlation file, and 3) a secondary resfile. This command declares a set of states which should be modeled, and declares a vector variable with the name `varname` which will be assigned the energies of each of the states after they have been repacked. This variable can be used in subsequent expressions. A vector variable is useful for when modeling many states which are in some way interchangeable. In this paper, all states are declared inside state vectors; and always, the lowest-energy state is extracted from the state vector with the vmin (vector-min) function.

The SCALAR_EXPRESSION command is used to express succinctly some value which is useful in one or more later expressions or the fitness function itself. The SCALAR_EXPRESSION command is:

```
SCALAR_EXPRESSION <varname> = <expression>
```

which declares the variable with the name `varname` which may be used in subsequent expressions. Note that variables may only be declared once and are only assigned a single value. They cannot be reassigned values in the way variables in most programming languages can be.

The ENTITY_FUNCTION command allows the user to define arbitrary contributions to the fitness function given the sequence assigned. The command itself is:

```
ENTITY_FUNCTION <varname> <entityfunctionfile>
```

The expressions defined in the entity-function file may again be composed using the same mathematical expressions used to define the fitness function; the full file format is described in the Supplementary Material. The user may compare the amino acid assigned to any position in the given sequence against any other position, examine whether an assigned amino acid belongs to a set of amino acids, and can, with that information, compose arbitrarily complicated (e.g. non-linear and/or non-pairwise decomposable) penalties or bonuses. Such penalties could be used a) to reward the placement of net positive charge on chain A and net negative charge on chain B (assuming that local pH effects are negligible), b) to penalize making more than X mutations to the native sequence, c) to penalize the design of a homodimer while trying to design a heterodimer, or d) to reward the placement of exactly one tryptophan on each of chain A and chain B. Such penalties are notably non-pairwise decomposable and could not be expressed in any piece of software performing single-state design with a purely pairwise decomposable energy function. Of course, there is no guarantee that the penalties will produce the sequences that the user is seeking, and may require the user to strengthen or weaken individual terms to meet their needs. It should also be noted that these constraints do not have access to the conformation of the states or to their energies, so it would not be possible with these constraints to, for example, give a bonus to a sequence in which one structure forms a hydrogen bond with residue 18's backbone carbonyl.

Exactly one FITNESS command must be included in the fitness definition file. The command is:

```
FITNESS <expression>
```

The value the fitness expression evaluates to is the value that will be fed to the genetic algorithm as the sequence's fitness. Output PDB files are generated for all states whose energies contribute to the fitness expression for the best sequences encountered in each design trajectory. In the case that only one state from a state-vector contributes to the fitness for a given sequence (e.g. the lowest energy state amongst the set, selected through the vmin function), then only that state will be output.

Redocking

After each round of multistate design, we redocked the negative states to find alternate low-energy conformations, and then designed against these alternate docked conformations in subsequent rounds. We used the dock_pert rigid-body docking protocol (31) that begins with a small random rigid body perturbations of an initial docked conformation. Starting from the output structures from multistate design, we split the two chains, repacked each chain separately, and concatenated the repacked structures. This step relieved intra-chain collisions frequently present in the negative states which the shorter docking_local_refine protocol seemed willing to leave intact. These structures were then fed as input for fifty trajectories of the dock_pert protocol. The lowest energy docked conformation of these fifty was split, its chains repacked individually, and the ΔG_{bind} was calculated as the difference in energy of the bound and unbound chains.

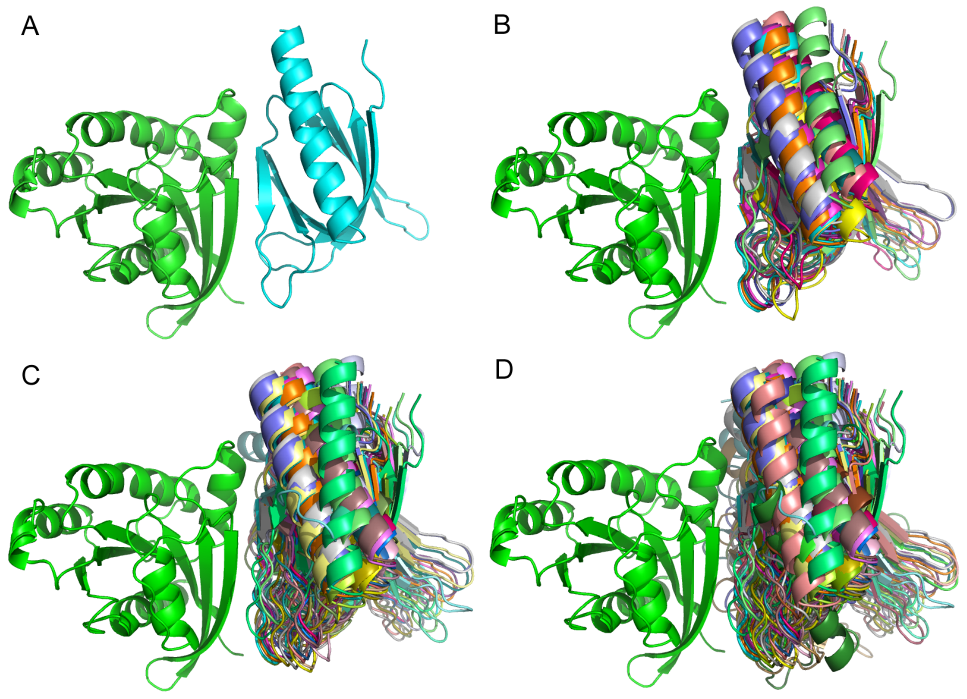


Figure 4.2: Iterative expansion of the negative state set. A) the crystal structure of RalA/Exo84 used as the negative state in the first round of multistate design, B) the thirteen negative states used in the second round, C) the thirty-one negative states used in the third round, and D) the forty-nine negative states used in the fourth round.

4.3.2 Orthogonal Interface Redesign Task Workflow

For the actual design simulation, an iterative process was used where, starting with the crystal and NMR structures as models for the positive and negative states, we ran multistate design to generate candidate sequences and then ran rigid-body docking to relax the structures of the negative states. Sequences were evaluated on the basis of the binding energy of both the desired and undesired interactions after redocking.

Input file preparation: An important step in setting up the orthogonal interface design is obtaining reliable starting structures for design. The crystal structures for the RalA/Sec5 and RalA/Exo84 interactions were repacked and minimized using Rosetta to obtain low energy models. The structure of Ral/RalBP1 is more difficult to handle in this way because its PDB entry is an ensemble of NMR models which vary in conformation considerably. (While 2KWI is actually the structure of the interaction between RalB/RalBP1, RalA and RalB have complete sequence identity at all of the interface positions considered in this study. We assume the two Ral isoforms bind RalBP1 in the same manner and use this structure to model the RalA/RalBP1 state.) All of the models in the 2KWI structure were separated into individual models, repacked, and minimized. We then chose the four lowest-energy structures, models 1, 15, 29 and 30, and redocked them with Rosetta (31). Model 30 produced the best docking funnel and binding energy, and did not substantially change the conformation of the interface (C_α RMSD < 2.0 Å). The lowest-energy docked conformation starting from model 30 was used for the Ral/RalBP1 complex.

For convenience it is useful to describe the proteins modeled by different chemical species. Each protein monomer is described as A (RalA), B (RalBP1), C (Sec5) or D (Exo84). The three dimers can be described as AB (Ral/RalBP1), AC (RalA/Sec5) or AD (RalA/Exo84). The AB species refers to the best repacked, minimized, and docked model 30 from 2KWI while AC and AD refer to the repacked and minimized 1UAD and 1ZC3 structures respectively. The A (RalA) species was further subdivided into A_b, A_c, and A_d, which use the RalA coordinates in the AB, AC, and AD dimers, respectively. The A species was divided in this way because of differences in the backbone between the three dimer structures.

Choosing what to design: We decided to make our design goal a redesigned RalA that retains its affinity for RalBP1 and has no affinity for Sec5 and Exo84. Two different setups of the redesign task were performed with multistate design. In the first setup, we selected residues on RalA that we thought could destabilize the interface between RalA/Sec5 and RalA/Exo84 without disturbing the Ral/RalBP1 interaction. The following list of residues were designed in this setup: L14, Y36, E38, K47, A48, R49, S50, R52, Q63, and E73. Residues 14, 36, 63, and 73 were chosen because they interact with either Sec5 or Exo84 and not RalBP1. The other positions were included because of mutagenesis studies indicating these residues affect Sec5 and Exo84 binding (25). Note that RalA residues A48, D49, S50 and R52 are in close proximity to RalBP1; mutations to these residues would impact both RalA's interactions with

Sec5 and Exo84 and its interactions with RalBP1. Design at these positions is non-trivial.

For the second setup of this task, we excluded some of the already-characterized specificity-determining positions and also allowed more residues at the Ral/RalBP1 interface to be designed. From the structures of the Ral-effector complexes, mutations on Ral that disrupt binding to each individual effector have already been identified. For example, the D49N mutant of RalA disrupts binding to RalBP1 but not Sec5 or Exo84, and the D49E mutant disrupts binding to Sec5 and Exo84 but not to RalBP1 (32; 33). Similarly, the mutations E38R and A48W have been shown to destroy binding with Sec5 and Exo84, respectively (25). In order to make the design task more challenging, we left these positions fixed to their native amino acids. Additionally, for this setup we allowed more residues at the interface between Ral/RalBP1 to be designed, to see if the binding energy of the positive state could be further improved. Together, these changes expanded the number of designable residues from 8 to 16. The residues allowed to change in this setup were L14, K16, Y36, K47, S50, R52, Q63, D65, L67, E73, D74, Y75, A77, I78, N81, and Y82. With this design definition, we hoped to identify new specificity-conferring mutations for Ral.

Fitness function definition: The fitness function used for design was constructed to use the binding energy of the desired Ral/RalBP1 interaction and the binding energies of the undesired RalA/Sec5 and RalA/Exo84 interactions. Using the nomenclature described above, with AB, AC, AD, A_b, A_c, A_d, B, and C, representing the energy of each of the corresponding dimer or monomer under a particular sequence assignment (A_b representing the energy of the Ral backbone taken from the Ral/RalBP1 structure, A_c representing the energy of the RalA backbone taken from the RalA/Sec5 structure, andA_d representing the energy of the RalA backbone taken from the RalA/Exo84 structure), the fitness function we minimized was as follows:

$$\begin{aligned}
fitness &= AB + w * (\Delta\Delta G_{AB,AC} + \Delta\Delta G_{AB,AD}) \\
\Delta\Delta G_{AB,AC} &= \Delta G_{AB} - \Delta G_{AC} \\
\Delta\Delta G_{AB,AD} &= \Delta G_{AB} - \Delta G_{AD} \\
\Delta G_{AB} &= AB - A_b - B \\
\Delta G_{AC} &= min(AC - A_c - C, 0) \\
\Delta G_{AD} &= min(AD - A_d - D, 0)
\end{aligned}$$

where $\Delta\Delta G_{AB,AC}$ and $\Delta\Delta G_{AB,AD}$ represent the binding-energy gaps, and the binding-energy-gap weight, w, balances the total energy of the AB complex with the binding-energy-gaps for AC and AD. The weight was varied in independent runs between 1 and 12. We computed binding energies by comparing the energies of the dimers with the energies of the monomers sharing the same backbone conformations; this meant modeling extra states (A_c and A_d), but gave more reliable results than if we had only modeled the A_b monomer. The Discussion section raises this point again. Binding energies of the negative states were capped at 0. This cap is also described in greater detail in the Discussion section.

We ran two separate single state design (SSD) jobs as controls for this method. These jobs only optimized the binding energy of AB and ignored the binding energies of AC and AD. In the first control run, we allowed design of all of the same residues included in the multistate design setup-scheme 2 (SSD1). In the second control run, we designed only residues on RalA that are at the interface with RalBP1, in an effort to mirror the way typical redesigns of only one complex are done. The set of residues in this case was as follows: K16, A48, D49, S50, R52, D65, L67, N81, Y82, R84, S85, G86. For both single state jobs, we used the same multistate protocol as above except that the weight (w) of the fitness function is set to 0 to force the design algorithm to ignore the binding energies of AC and AD.

Job management: Each batch of jobs was composed of two main features: the set of PDB files defining the states which should be optimized (the state version), and the set of residues which were allowed to redesign and repack on each of the states (the design definition). Each batch ran separate jobs for each combination of models of the positive state (AB) and weight, w, on the binding-energy-gap bonus. A Python v2.6 script created the set of files necessary for a single batch of multistate design jobs. This script, the set of input files necessary for it, and the command lines we used to execute this script are provided in the Supplementary Material. Following each round of multistate design, we redocked the AB, AC and AD dimers using the RosettaScripts executable (34), and then repacked the monomers using the fixbb (fixed-backbone design) executable. All simulations were performed with SVN revision 39931 of the Rosetta3 source code and SVN revision 39914 of the Rosetta3 database. Sequence logos were created using WebLogo v.2.8.2 (35).

Design no.	Designed sequence	Total Energy	ΔG_{AB}	ΔG_{AB-AC}	ΔG_{AB-AD}	RMSD to native		
						AB	AC	AD
wt	`lkyk srqd ledy ainy`	-466.2	-22.5	2.8	8.8	—	—	—
msd,1	`WFKF sFSG lKQH SWDy`	-460.4	-25.9	-17.0	-11.8	0.1	4.5	0.8
2	`EHKN sFEd YGRE STDF`	-465.5	-25.0	-15.8	-16.2	0.1	6.6	2.2
3	`RRTQ sLVV YKRE SSDF`	-465.6	-25.0	-12.9	-15.5	0.0	6.0	1.3
4	`DHTF sITd lKNQ SWDy`	-463.2	-24.7	-10.1	-13.8	0.1	0.6	1.0
5	`EHKT sFES lKSR SLDy`	-462.6	-24.5	-14.6	-13.7	0.1	6.2	0.4
6	`EHKT sFES lDSR SLDy`	-464.5	-24.5	-15.5	-14.9	0.1	6.2	0.4
7	`KRRF sLVV lKQH SWDy`	-462.5	-24.2	-14.4	-14.5	0.1	0.9	3.3
8	`EHKT sFES lDSR SLDy`	-464.3	-24.1	-14.8	-14.2	0.1	5.4	0.6
9	`EHKT sFES lNSR SLDy`	-464.0	-24.0	-14.5	-15.8	0.1	5.4	1.8
10	`EHKG sFEd lKQH SRDy`	-464.1	-23.8	-14.9	-12.8	0.1	6.3	4.8

Table 4.1: Selected orthogonal design sequences from setup-scheme 1. Sequences, energies (in REUs), and RMSD's of designs created multistate design (MSD) setup-scheme 1. All of the MSD designs shown have binding energy gaps between the positive and negative states greater than 10 REU.

4.4 Results

As a test of our multistate design implementation, we decided to use the protocol to redesign specificity in the Ral signaling network. Our design goal for this task was to redesign RalA so that it would only interact with RalBP1 and not with Sec5 or Exo84. The protocol starts

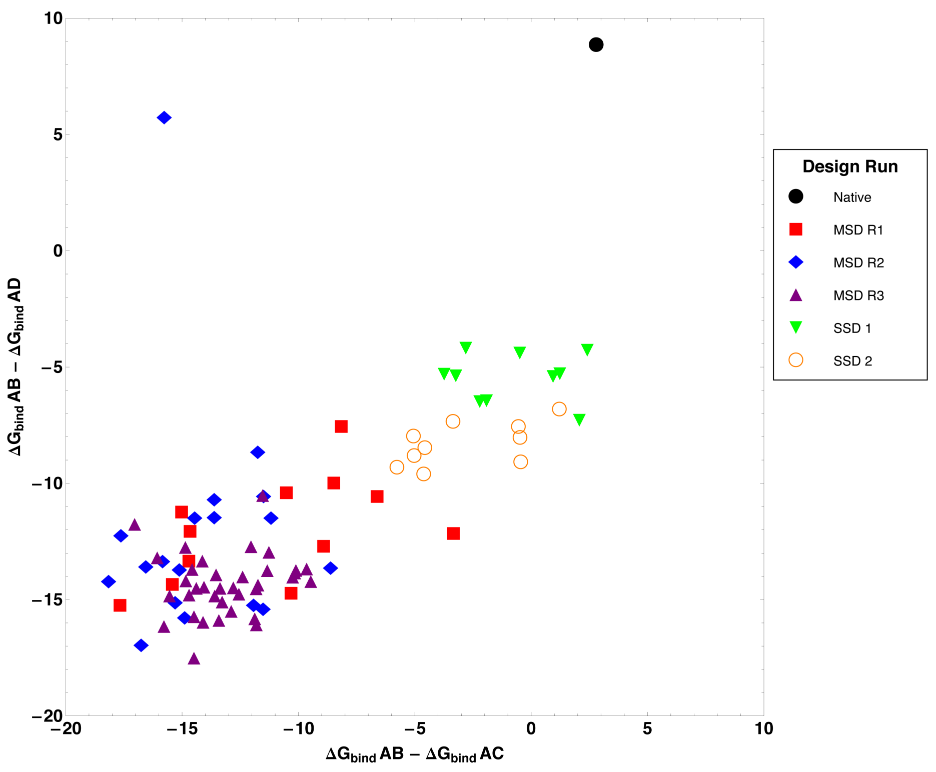


Figure 4.3: Binding energy differences using multistate and single state design. Binding energy differences between the positive state AB (Ral/RalBP1) and negative states AC (RalA/Sec5) and AD(RalA/Exo84) following multistate design (MSD) and single state design (SSD). Binding energy differences between the native AB and AC, and AB and AD states (black) are shown for reference. Consecutive rounds of MSD (red, blue, and purple) on protein A residues, listed in Methods, decrease the binding energy to C and D by a larger magnitude than SSD. Two different methods of SSD are shown: SSD 1 (green) allows design on the same residues as MSD, and SSD 2 (orange) allows design on residues that are at the AB interface. Neither of the SSD methods explicitly disfavor binding to C or D. AB binding energy maintained, in all cases, between -22.0 and -25.0 REU.

with fixed-backbone design on each of the states, identifying mutations that either stabilize the Ral/RalBP1 interface or destabilize the RalA/Sec5 and RalA/Exo84 interfaces. After the design finishes, the RalA/Sec5 and RalA/Exo84 complexes are redocked. The redocked complexes with the lowest binding energies are used as alternate conformations for the second round of design. Figure 4.2 shows the expansion of negative state conformations for RalA/Exo84 seen during three rounds of the design protocol. This process of design, redocking and feeding in the low-energy docked conformation back into the next round of design can be iterated until the the desired binding energy gaps are achieved or the sequences converge.

Two different setups of the design problem were tested: one where known specificity-changing positions on RalA were allowed to change and one where these positions were held fixed. The set of residues allowed to change in each setup is described in the Methods. For the first setup, any position on RalA that we thought could be used to improve specificity for RalBP1 was allowed to change. The results of using the multistate protocol on this design setup are shown in Table 4.1. All of the complexes output by the design protocol were redocked before calculating the predicted binding energy. After only one round of design and docking, many designs showed large destabilizations to the RalA/Sec5 and RalA/Exo84 interfaces while maintaining native-like Ral/RalBP1 binding energies.

It was reassuring to us to see that the multistate protocol recapitulated some known specificity-changing mutations. In some of the round 2 designs, glutamic acid-38 was mutated to tryptophan. This mutation decreases Sec5 binding ~600-fold while having no effect on Exo84 binding (24). Lysine-47 in wild-type RalA was mutated exclusively to glutamic acid in the round 1 designs. Fukai et al. found that the K47E mutation weakens binding to Sec5 10-fold and to Exo84 100-fold (24). Alanine-48 of RalA, part of the switch I region and at the interface of all three effectors, is mutated to arginine in all of the round-1 and most of the round-2 designs (and mutated to tryptophan in the other round-2 designs). A tryptophan mutation at this residue was previously found to prevent binding of Exo84 but had no effect on Sec5 (24). We suspect that this tryptophan's effect on Exo84 binding is due to steric repulsion and hypothesize that an arginine at this residue would work equally well. Unfortunately, not all specificity changing mutations were recovered. The multistate design protocol failed to identify the destabilization of both Sec5 and Exo84 binding induced by the glutamic acid mutation at residue 49; instead, it placed an aspartic acid at this residue in all of the designs.

For the second design setup, we again used the iterative design and redock protocol to create RalA variants optimized for RalBP1 binding. In this setup, we also compared the iterative design strategy against the simpler single-state design strategy, which used the multistate design algorithm but optimized only for the energy of the Ral/RalBP1 state (Figure 4.3). Single state design produced designs that have good binding energy for the target interface Ral/RalBP1, but that also have good binding energy for the RalA/Exo84 interaction. In contrast, multistate design is able to produce the desired destabilization of both off-target interactions, as is shown by the points in the lower left quadrant of Figure 4.3. Table 4.2 highlights a few designs where both RalA/Sec5 and RalA/Exo84 were destabilized by at least 10 REU relative to Ral/RalBP1 and the Ral/RalBP1 total energy has not been overly compromised relative to the native.

Our results also show that iterative negative design improves multistate design. In the first round of multistate design, the fixed-backbone models of RalA/Sec5 and RalA/Exo84 suggest that these complexes cannot form; however, subsequent redocking of these complexes is able to relieve collisions and find low-energy conformations. We measured the difference in predicted binding energy gaps as reported by 1) the multistate design algorithm and 2) the subsequent redocking step to yield a pair of $\Delta\Delta\Delta G$s, which we interpreted as a vector. The "magnitude" of this vector, taken as $\sqrt{\Delta\Delta\Delta G^2_{AB,AC} + \Delta\Delta\Delta G^2_{AB,AD}}$, gives a measure of the inaccuracy of the fixed-backbone assumption. The median $\Delta\Delta\Delta G$-vector magnitudes for rounds 1, 2, and 3 were 46.6, 16.3, and 10.1 REU showing that multistate design's accuracy at negative design increases as the number of negative states increases.

Many new RalA mutations that have not been previously characterized were found in the second design setup. The designed amino acids from this second design setup fell into three categories: those which appeared important for RalA stability or RalBP1 binding (often including the native amino acid), those which appeared to destabilize either Sec5 binding or Exo84 binding, and those which showed no clear preference. The sequence profile of these designs is given in Figure 4.4. In most of the designs, multistate design chose the native Ral amino acid for positions which are important for RalBP1 binding, or for Ral stability. For example, serine-50, which is consistently recovered, forms hydrogen bonds with two residues on RalBP1, threonine-437 and glutamine-433, and tyrosine-82, in the core of the interface, maintains its contact with RalBP1 histidine-413. The wild type leucine at the very buried position 67 is the most frequently selected amino acid at that position. Tyrosine is also designed frequently at this position because it can form a good intramolecular hydrogen bond with arginine-78. Similarly, multistate design preferred arginine or histidine, instead of the wild-type lysine, at position 16 because of weak intramolecular hydrogen bonds.

Multistate design readily identified positions that destabilized the negative states. Ral positions 36 and 52 are important specificity positions for Sec5. Multistate design favored lysine at position 36 because it creates a clash with Sec5 residue glycine-28. Similarly, it liked to mutate position 52, an arginine in wild-type Ral which points out into solvent, to phenylalanine, leucine, or isoleucine. These residues all create clashes with threonine-28 on Sec5. Several positions appear to be important for preventing association with Exo84. Multistate design frequently mutated residue 14 to glutamic acid which clashes with a loop in Exo84. The wild type asparagine at position 81 in Ral makes two sidechain-backbone hydrogen bonds with Exo84. Multistate design changed this position to aspartic acid exclusively, and its sidechain cannot form hydrogen bonds in the low-energy redocked Exo84 structures. This aspartic acid interacts favorably with RalBP1's lysine at residue 421 in the Ral/RalBP1 design models. Any large, bulky residue at position 78 can produce a clash with Exo84. Multistate design favored placing arginine at this position, but even leucine is enough to cause problems. Finally, multistate design almost always placed either the wild-type alanine or a serine at position 77. Serine is a good choice for this position as it forms a small clash with the Exo84 backbone and adds a favorable interaction with RalBP1 residue glutamine-417.

A number of positions, specifically 47, 73, 74 and 75, displayed no clear preference among the designed sequences. Multistate design generally favored placing polar amino acids at these positions given that they are surface-exposed. Except for position 47, none of these positions look like they could provide significant amounts of specificity to the interface. The wild type Ral tyrosine at position 75 participates in a hydrogen bond with Exo84 serine-276. Multistate design removed this favorable interaction, and placed mostly histidine and arginine at this position. Positions 63 and 65, natively glutamine and aspartic acid, respectively, are in the middle of a beta-sheet in RalA and were also mutated to a wide variety of amino acids. Multistate design displayed a small preference for glutamic and aspartic acids at these positions. These mutations make sense as in the wild-type Ral/RalBP1 structure an arginine residue on RalBP1, arginine-434, interacts with the aspartic acid at Ral position 65. This same arginine residue can interact with a glutamic acid at position 63, if an aspartic acid at position 65 is not present.

Design no.	Designed sequence	Total Energy	ΔG_{AB}	$\Delta G_{AB}-$ ΔG_{AC}	$\Delta G_{AB}-$ ΔG_{AD}	RMSD to native		
						AB	AC	AD
wt	`lkyk srqd ledy ainy`	-466.2	-22.5	2.8	8.8	—	—	—
ssd,1	`VRyE srEd YDKy SRDy`	-472.5	-24.0	-2.2	-6.5	0.1	0.3	0.2
2	`VRyF srEd FDEH STDy`	-469.1	-23.7	-1.9	-6.5	0.1	0.2	0.3
3	`VRyE srEd YDKy SRDy`	-472.2	-23.3	-0.5	-4.4	0.1	0.1	0.2
4	`VRyE srEd YDKy SRDy`	-472.1	-23.3	-3.7	-5.3	0.1	0.2	0.2
5	`VRyE srEd YDKy SRDy`	-472.0	-23.1	-3.2	-5.4	0.1	0.1	0.4
6	`VRyE srEd YDKy SRDy`	-471.9	-23.0	-2.8	-4.2	0.1	0.1	0.1
7	`VRyE srEd YDKy SRDy`	-470.8	-22.9	0.9	-5.4	0.1	0.1	0.1
8	`RRyE sHEE YDKy SRDy`	-471.1	-22.6	2.1	-7.3	0.2	0.1	0.9
9	`VRyE srEd YDKy SRDy`	-471.1	-22.4	1.2	-5.3	0.1	0.1	0.2
10	`VRyE srEd YDKy SRDy`	-471.1	-22.3	2.4	-4.3	0.1	0.1	0.1
msd,1	`WFKF sFSG lKQH SWDy`	-460.4	-25.9	-17.0	-11.8	0.1	4.5	0.8
2	`EHKN sFEd YGRE STDF`	-465.5	-25.0	-15.8	-16.2	0.1	6.6	2.2
3	`RRTQ sLVV YKRE SSDF`	-465.6	-25.0	-12.9	-15.5	0.0	6.0	1.3
4	`DHTF sITd lKNQ SWDy`	-463.2	-24.7	-10.1	-13.8	0.1	0.6	1.0
5	`EHKT sFES lKSR SLDy`	-462.6	-24.5	-14.6	-13.7	0.1	6.2	0.4
6	`EHKT sFES lDSR SLDy`	-464.5	-24.5	-15.5	-14.9	0.1	6.2	0.4
7	`KRRF sLVV lKQH SWDy`	-462.5	-24.2	-14.4	-14.5	0.1	0.9	3.3
8	`EHKT sFES lDSR SLDy`	-464.3	-24.1	-14.8	-14.2	0.1	5.4	0.6
9	`EHKT sFES lNSR SLDy`	-464.0	-24.0	-14.5	-15.8	0.1	5.4	1.8
10	`EHKG sFEd lKQH SRDy`	-464.1	-23.8	-14.9	-12.8	0.1	6.3	4.8

Table 4.2: Selected orthogonal design sequences from setup-scheme 2. Sequences, energies (in REUs), and RMSD's of designs created with single state design (SSD) and multistate design (MSD). All of the MSD designs shown have binding energy gaps between the positive and negative states greater than 10. None of the SSD designs are predicted to have this amount of specificity.

4.5 Discussion

Here we have presented a generic implementation of multistate design which allows users to rapidly customize the fitness function to be optimized, and have shown how the implementation can be used in the orthogonal interface design problem. In fact, the ease with which new states can be added and their energies managed through the fitness-function-definition file made it possible to perform multiple rounds of negative design, which to our knowledge has not previously been reported. The implementation separates its search through sequence space

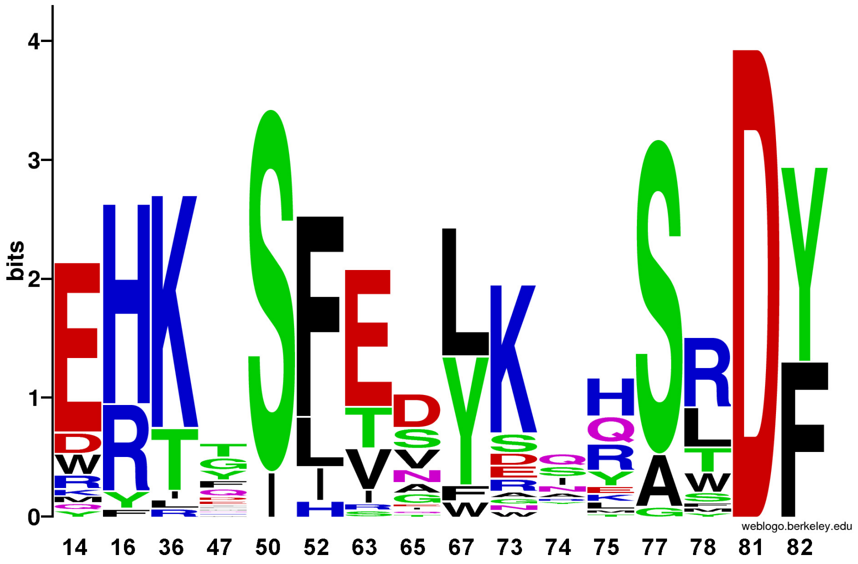


Figure 4.4: Sequence propensity of RalA residues in designs created with multistate design setup-scheme 2 for the orthogonal interface redesign task. Sequence logo of the designs produced by multistate design. Positions 50, 67, and 16 showed preferences for amino acids that stabilized the RalA monomer or that stabilized the Ral/RalBP1 complex. Positions 36 and 52 showed preferences for amino acids that destabilized the RalA/Sec5 interaction; positions 14, 77, 78, and 81 showed preferences for amino acids that destabilized the RalA/Exo84 interface. Positions 47, 73, 74 and 75 displayed no clear preferences, except for non-wildtype amino acids, as the native amino acids formed favorable contacts with either Sec5 or Exo84.

and conformation space as many prior examples of multistate design have (13; 14; 15; 18), as opposed to their simultaneous optimization in the belief-propagation algorithm presented by Fromer et al. (36), or the reduced-representation, sequence-space-only optimizations presented by Nautiyal et al.(37) and by Grigoryan et al. (38; 19). The explicit rotamer optimization we perform in our inner loop was able to find interesting through-residue interactions where one residue can pre-order a neighboring residue such that this residue's interaction with a third (or fourth) residue would be unfavorable; in contrast, Grigoryan et al.'s (38) score function, which represents amino-acid pair interactions by their average rotamer-pair interaction energies, would likely be unable to capture this pre-ordering effect. In contrast to the multi-specificity algorithms presented by Humphris and Kortemme (15) and Fromer et al.(39; 36), the implementation is suited to perform both positive and negative design. We have tuned the parameters of our genetic algorithm to behave as well as Rosetta's existing single-state design algorithm at single-state design problems, but we have not compared the genetic algorithm's performance to the intriguing FASTER-MSD algorithm presented by Allen and Mayo (16), whose implementation starting from our existing code should be straightforward.

The use of negative design in our simulations had some interesting and unexpected effects on the results of design. Most of these effects stem from the use of fixed-backbones in our simulations. There are three ways in which the fixed-backbone assumption affected our results. The first two relate to restrictions on the rigid-body degrees of freedom connecting the two chains, and the third relates to the restriction on the internal degrees of freedom in each individual chain. We discuss our findings below which will be of interest to those seeking to perform negative design.

In the first case, we found that multistate design would often introduce the largest collision it could in the negative states in order to increase the gap between the positive and negative state energies. This result is desirable if all negative states are destabilized; however, in many trajectories, multistate design would introduce collisions into one of the negative states and fail to destabilize the other. The fitness function rewarded a pair of binding energy gaps of (-1000, +3) more than it rewarded binding energy gaps of (-10, -10). Allen and Mayo observed a similar behaviors in negative design and chose to cap repulsive interactions between residue pairs at +50 (16). This problem is due to the fixed backbone assumption. Once the apparent binding energy from a particular conformation goes positive, that conformation can no longer be considered valid; the model of two proteins held rigidly docked against each other breaks down. There are two solutions to this problem: cap the binding energies for the negative states at zero (which we did) or add an alternate undocked conformation containing both chains, but where they are physically separated; this undocked conformation would presumably be chosen as the minimum energy conformation once collisions had been introduced into all the other docked conformations. The first solution is one CPU per negative species cheaper to execute.

Second, we found that rigid-body docking was often able to relax away collisions present in the negative states that came out of the early rounds of design. Multistate design can only design against states it can see, and there are a surprising number of low-energy docked conformations for the negative states. Keating et al. (40) similarly noticed that allowing their backbones to relax after introducing mutations improved their ability to predict the adopted conformations and binding energies of heterodimeric coiled-coils. Havrakek and Harbury (13) noticed that a single round of multistate design overstated the destabilization of the heterodimeric species they were designing against; they suggested that the addition of more states could overcome this problem and our *in silico* results are consistent with this hypothesis. Our simulations demonstrate that the addition of alternate conformations for the undesired interactions decreases the discrepancy between the energies that multistate design believes it produces and the energies obtained after redocking.

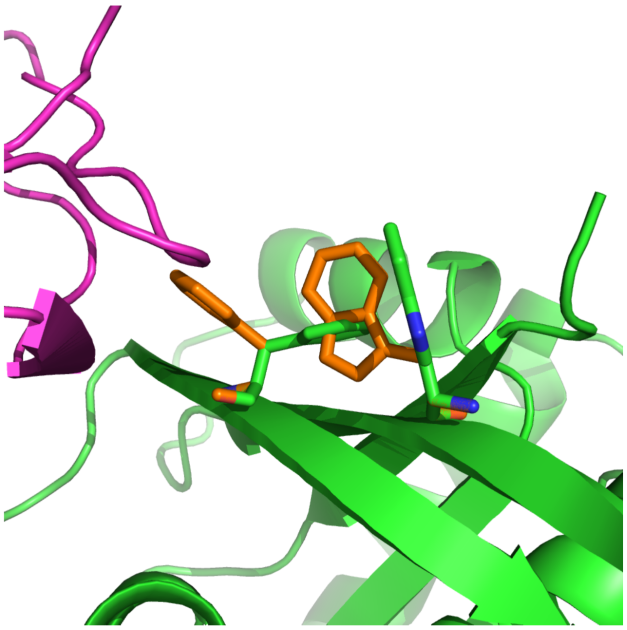


Figure 4.5: Pitfalls of designing on multiple backbone conformations. Placing both F52 and W63 on RalA (green) destabilizes its interaction with Sec5 (magenta). In the docked conformation, the F52 and W63 rotamers collide in the least-awful-rotamer placement available. In the unbound state (orange) these residues relax out of collision. W63 disrupts binding with Sec5 through F52, but neither residue disrupts binding on its own. Unfortunately, W63 is incompatible with the RalA backbones from the crystal structures, though it is compatible with the RalB backbone in the NMR structure. Here, a discrepancy between the backbone conformations of Ral in its various states leads to a questionable design.

The third way that the fixed backbone assumption impacted our results is more difficult to describe. In the setup-scheme 1 designs for the RalA task, multistate design found a pair of mutations, W63 and F52 (Figure 4.5), where the binding with Sec5 was disrupted, but at the cost of destabilizing the RalA backbone taken from the crystal structures (states A_c and A_d). In contrast, the NMR models of RalB bound to RalBP1 were able to accommodate these mutations. Since the A_c and A_d energies of the RalA monomers from the negative states were invisible to the fitness function, multistate design dutifully chose these mutations. The destabilization of the backbone conformation for RalA from the Sec5 crystal structure is worrisome in this case because the section of the RalA backbone being designed has such high agreement between the Sec5-bound and Exo84-bound crystal structures (though, the RalBP1-bound NMR models showed significant disagreement). We did not want to disrupt the crystal structure conformation. The fixed-backbone assumption was more of a requirement in this case: we designed for a backbone we were unsure about (the NMR model) without considering a backbone we were interested in preserving (the crystal backbone), but, if the same backbone had been present in all three models, we would not have encountered this issue. We tried twice to skirt this problem by docking the crystal structure of RalA against the RalBP1 models, and by docking the RalA-NMR structures against the Sec5 and Exo84 models, but neither approach resulted in good docking funnels or satisfactory binding energies. What to do?

There were two possible solutions: modify the fitness function to disfavor the destabilization of the RalA crystal structure, or redefine the set of positions which are allowed to design. Taking the first approach, one could have included the energies of the crystal forms of the unbound RalA states in the fitness function: $fitness = AB + w * (\Delta\Delta G_{AB,AC} + \Delta\Delta G_{AB,AD}) + A_c + A_d$. Such a fitness function has the unfortunate consequence of triple-counting stabilizing mutations to the RalA structure. Alternatively, one could penalize the destabilization of the crystal forms of RalA beyond some threshold: $fitness = AB + w * (\Delta\Delta G_{AB,AC} + \Delta\Delta G_{AB,AD}) + max(A_c - x, 0)^2 + max(A_d - y, 0)^2$ where x and y are some predetermined constants representing an upper bound on how destabilized the RalA monomers could become before the penalty kicks in. We went with the second option and expanded the set of designable positions. This change had the serendipitous effect of favoring sequences on the RalA backbone which were compatible with all three structures; the fitnesses for the best designs which lacked the F52/W63 pair were better than those with them.

It should be noted that the W63/F52 pair was preferred by multistate design not because it destabilized the RalA monomer, but because it destabilized the RalA/Sec5 interface. However, if we had not compared the AC energy against the A_c monomer energy, but instead compared it against the energy from a different backbone conformation, (e.g. if we had defined ΔG_{AC} as $AC - A_b - C$), then multistate design would have been able to destabilize the RalA/Sec5 interface by destabilizing the backbone conformation for RalA present in the RalA/Sec5 interaction. This would be done by introducing intra-chain collisions within the RalA chain as long as those collisions were not present in the A_b state. In some other design problem, this might be a valid design strategy. For example, in orthogonal interface redesign for a protein known to adopt different conformations to interact with different partners, destabilizing the conformation required to interact with one partner is a fine way to destabilize that interaction. The fitness function must be carefully constructed when designing on different backbone conformations to ensure that the desired result is obtained.

References

1. Desmet, J., Maeyer, M., Hazes, B., and Lasters, I. (1992) The dead-end elimination theorem and its use in protein side-chain positioning. Nature **356**, 539–542. ISSN 0028-0836

2. Koehlt, P. and Delarue, M. (1994) Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy. J. Mol. Biol **239**, 249–275

3. Desjarlais, J. and Handel, T. (1995) De novo design of the hydrophobic cores of proteins. Protein Science **4**, 2006–2018. ISSN 1469-896X

4. Dahiyat, B. and Mayo, S. (1996) Protein design automation. Protein Science **5**, 895–903. ISSN 1469-896X

5. Harbury, P., Plecs, J., Tidor, B., Alber, T., and Kim, P. (1998) High-resolution protein design with backbone freedom. Science **282**, 1462

6. Kortemme, T., Ramirez-Alvarado, M., and Serrano, L. (1998) Design of a 20-Amino Acid, Three-Stranded beta-Sheet Protein. Science **281**, 253

7. Kuhlman, B. and Baker, D. (2000) Native protein sequences are close to optimal for their structures. Proceedings of the National Academy of Sciences of the United States of America **97**, 10383

8. Looger, L. and Hellinga, H. (2001) Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics1. Journal of molecular biology **307**, 429–445. ISSN 0022-2836

9. Eriksson, O., Zhou, Y., and Elofsson, A. (2001) Side chain-positioning as an integer programming problem. Algorithms in Bioinformatics 128–141

10. Desmet, J., Spriet, J., and Lasters, I. (2002) Fast and accurate side-chain topology and energy refinement (FASTER) as a new method for protein structure optimization. Proteins: Structure, Function, and Bioinformatics **48**, 31–43. ISSN 1097-0134

11. Leaver-Fay, A., Kuhlman, B., and Snoeyink, J. (2005) An adaptive dynamic programming algorithm for the side chain placement problem. In Pacific Symposium on Biocomputing, volume 10, 16–27

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

13. Havranek, J. and Harbury, P. (2002) Automated design of specificity in molecular recognition. Nature Structural & Molecular Biology **10**, 45–52

14. Ambroggio, X. 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

15. Humphris, E. and Kortemme, T. (2007) Design of multi-specificity in protein interfaces. PLoS Comput Biol **3**, e164

16. Allen, B. and Mayo, S. (2010) An efficient algorithm for multistate protein design based on FASTER. Journal of Computational Chemistry **31**, 904–916. ISSN 1096-987X

17. Babor, M. and Kortemme, T. (2009) Multi-constraint computational design suggests that native sequences of germline antibody H3 loops are nearly optimal for conformational flexibility. Proteins: Structure, Function, and Bioinformatics **75**, 846–858. ISSN 1097-0134

18. Ashworth, J., Taylor, G., Havranek, J., Quadri, S., Stoddard, B., and Baker, D. (2010) Computational reprogramming of homing endonuclease specificity at multiple adjacent base pairs. Nucleic Acids Research **38**, 5601. ISSN 0305-1048

19. Grigoryan, G., Reinke, A., and Keating, A. (2009) Design of protein-interaction specificity gives selective bZIP-binding peptides. Nature **458**, 859–864. ISSN 0028-0836

20. Feig, L. (2003) Ral-GTPases: approaching their 15 minutes of fame. Trends in cell biology **13**, 419–425. ISSN 0962-8924

21. Frankel, P., Aronheim, A., Kavanagh, E., Balda, M., Matter, K., Bunney, T., and Marshall, C. (2004) RalA interacts with ZONAB in a cell density-dependent manner and regulates its transcriptional activity. The EMBO Journal **24**, 54–62

22. Bodemann, B. and White, M. (2008) Ral GTPases and cancer: linchpin support of the

tumorigenic platform. Nature Reviews Cancer **8**, 133–140. ISSN 1474-175X

23. Fenwick, R., Campbell, L., Rajasekar, K., Prasannan, S., Nietlispach, D., Camonis, J., Owen, D., and Mott, H. (2010) The RalB-RLIP76 Complex Reveals a Novel Mode of Ral-Effector Interaction. Structure **18**, 985–995. ISSN 0969-2126

24. Fukai, S., Matern, H., Jagath, J., Scheller, R., and Brunger, A. (2003) Structural basis of the interaction between RalA and Sec5, a subunit of the sec6/8 complex. The EMBO Journal **22**, 3267–3278

25. Jin, R., Junutula, J., Matern, H., Ervin, K., Scheller, R., and Brunger, A. (2005) Exo84 and Sec5 are competitive regulatory Sec6/8 effectors to the RalA GTPase. The EMBO Journal **24**, 2064–2074

26. Leaver-Fay, A., Tyka, M., Lewis, S., Lange, O., Thompson, J., Jacak, R., Kaufman, K., Renfrew, P., Smith, C., Sheffler, W., et al. (2011) ROSETTA3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. Methods in enzymology **487**, 545. ISSN 1557-7988

27. Rohl, C., Strauss, C., Misura, K., and Baker, D. (2004) Protein structure prediction using Rosetta. Methods in enzymology **383**, 66–93. ISSN 0076-6879

28. Allen, B. and Mayo, S. (2006) Dramatic performance enhancements for the FASTER optimization algorithm. Journal of computational chemistry **27**, 1071–1075. ISSN 1096-987X

29. Leaver-Fay, A., Liu, Y., and Snoeyink, J. (2004) Faster placement of hydrogen atoms in protein structures by dynamic programming. In 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)

30. Leaver-Fay, A., Snoeyink, J., and Kuhlman, B. (2008) On-the-fly rotamer pair energy evaluation in protein design. In Proceedings of the 4th international conference on Bioinformatics research and applications, 343–354. Springer-Verlag. ISBN 3540794492

31. Gray, J., Moughon, S., Wang, C., Schueler-Furman, O., Kuhlman, B., Rohl, C., and Baker, D. (2003) Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. Journal of Molecular Biology **331**, 281–299. ISSN 0022-2836

32. Cantor, S., Urano, T., and Feig, L. (1995) Identification and characterization of Ral-

binding protein 1, a potential downstream target of Ral GTPases. Molecular and cellular biology **15**, 4578. ISSN 0270-7306

33. Moskalenko, S., Tong, C., Rosse, C., Mirey, G., Formstecher, E., Daviet, L., Camonis, J., and White, M. (2003) Ral GTPases regulate exocyst assembly through dual subunit interactions. Journal of Biological Chemistry **278**, 51743. ISSN 0021-9258

34. Fleishman, S. J., Leaver-Fay, A., Corn, J. E., Khare, S. D., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., and Baker, D. (2011) RosettaScripts: an XML-like interface to the Rosetta macromolecular modeling suite. PLoS One **submitted**

35. Crooks, G., Hon, G., Chandonia, J., and Brenner, S. (2004) WebLogo: a sequence logo generator. Genome research **14**, 1188. ISSN 1088-9051

36. Fromer, M., Yanover, C., and Linial, M. (2010) Design of multispecific protein sequences using probabilistic graphical modeling. Proteins: Structure, Function, and Bioinformatics **78**, 530–547. ISSN 1097-0134

37. Nautiyal, S., Woolfson, D., King, D., and Alber, T. (1995) A designed heterotrimeric coiled coil. Biochemistry **34**, 11645–11651. ISSN 0006-2960

38. Grigoryan, G., Zhou, F., Lustig, S., Ceder, G., Morgan, D., and Keating, A. (2006) Ultra-fast evaluation of protein energies directly from sequence. PLoS Comput Biol **2**, e63

39. Fromer, M. and Shifman, J. (2009) Tradeoff between stability and multispecificity in the design of promiscuous proteins. PLoS Comput Biol **5**, e1000627

40. Keating, A., Malashkevich, V., Tidor, B., and Kim, P. (2001) Side-chain repacking calculations for predicting structures and stabilities of heterodimeric coiled coils. Proceedings of the National Academy of Sciences of the United States of America **98**, 14825

Chapter 5

Conclusion

The goal of protein design programs is to produce sequences that will fold and perform some desired function. Most successes to date in protein design have been obtained using positive design only. In positive design, sequences are optimized to fit well for one structure and one function alone. Multistate design is a new approach to protein design which optimizes sequences simultaneously for the desired structure and against competing, undesired structures. This dissertation describes the use of multistate design for two problems in protein design: the *de novo* design of stable and soluble proteins and protein-protein interaction specificity. For the first problem, negative design is used against the aggregated state via the development of the hpatch score. Havranek and Harbury also applied negative design in this way, i.e. by using a modified scoring scheme, to disfavor the aggregated state(1). The second way negative design is used in this thesis is for improving binding specificity during protein interface design. The previous chapter showed that *in silico* multistate design does a better job than single state design in destabilizing undesired interactions in redesigns of the Ral signaling network. In this chapter, I discuss how the hpatch score fits in with the other Rosetta energy function terms and another way the score might be used in the future. This chapter will conclude with some of the future directions and exciting applications of computational protein design on the horizon.

5.1 Remarks about the hpatch score

Many different forms of the hpatch score were tested before arriving at the implementation described in chapter 3. Early versions of the score were residue-centric, in that each surface-exposed residue was assigned a score. The first version of the score looked only at the number of hydrophobic residues around the residue being scored. The big flaw with this version was that it gave a residue surrounded by four alanines the same score as a residue surrounded by four tryptophanes. Subsequent versions of the score looked at the identities of the neighboring side chains. The hpatch-fast score was one of the final residue-centric scores tested. Although this score is good at identifying residues with greater-than-native amounts of hydrophobic surface area surrounding them, designing with the score was not able to prevent patches from forming. The key to obtaining a good score for favoring or penalizing some protein characteristic is having a good measure of that characteristic. In the best case, there will be a significant difference in that measure between native and designed proteins. The reason patches were still able to form when using the hpatch-fast score is because most of the positions in designed proteins had scores similar to those of natives. The score was looking at surface hydrophobicity within a certain distance, and, in most cases, the value was not significantly different from what is seen in native proteins. It failed to see that patches could form over the area surrounding multiple residues. Only by implementing a true patch-finding approach, as used by the hpatch-SASA score, was it possible to keep patches from forming on designed proteins.

5.1.1 How the hpatch score fits in with solvation energy

The solvation energy of a protein is the change in free energy that occurs when transferring a protein from vacuum to water. Typically the solvation energy is divided into nonpolar and polar terms. The nonpolar term represents the cost of forming a solute-sized cavity in water, solvent rearrangement and solute-solvent dispersion interactions, while the polar term describes the energy of electrostatic interactions between the solute and the solvent(2). Most programs estimate the nonpolar contribution to the solvation energy using surface-area dependent models, pioneered by Eisenberg et al.(3). With these approaches, the assumption is that the solvation energy of a protein can be obtained by summing up the contributions of all atomic groups. The contribution of each atom to the solvation energy depends on the solvent accessible surface area (SASA) of that atom and an atomic solvation parameter, a value derived from amino acid transfer free energies. Because calculating SASA is time-consuming, Rosetta uses the much faster Lazaridis-Karplus (LK) solvation model to calculate the nonpolar contribution of the solvation energy. The LK model, also called EEF1, is based on volume exclusion, and not on atom SASA(4). EEF1 estimates the solvation free energy by taking the solvation free energy of a group i in a fully solvent-exposed reference state and subtracting some energy to account for neighboring desolvating groups. The total solvation free energy for the protein is then obtained by summing over all groups in the protein. How much energy is subtracted from the reference state energy is determined by looking at how much volume is excluded by each neighbor j around group i. The Lazaridis-Karplus method for calculating solvation energy is very fast because it does not require the calculation of SASA.

In general, solvation energy terms penalize the burial of polar atoms and favor the burial of hydrophobic residues. Why then does the LK model not penalize the formation of hydrophobic patches? The answer lies in the reference state for the LK model. For most positions in a globular protein, the folded state will be more desolvated than in the reference state. Therefore, design of hydrophobic residues at these positions will be favored because of the energetic bonus of desolvating a hydrophobic residue. To avoid designing all hydrophobic surfaces, Rosetta uses the amino acid reference energies which exert their greatest effect to surface positions. The reference energies favor the design of polar residues and offset the bias to design hydrophobic residues by the LK model.

In some respects, the use of the hpatch score with the Lazaridis-Karplus solvation model works similarly to a surface-area dependent solvation model. The LK model favors the burial of nonpolar surface area and penalizes the burial of polar surface area. The hpatch score complements the LK term and penalizes hydrophobic patches on the surface, not to be confused with a penalty for hydrophobic surface area. Mayo and coworkers have experimented with the use of a nonpolar surface area penalization term in their energy function(5; 6; 7). The concern with such a term is what effect it has on overall hydrophobic surface area. Protein surfaces do have hydrophobic surface area, whether its for stability or for function, and a general nonpolar exposure penalty may result in designed proteins with artificially low amounts of hydrophobic surface area. Even if the penalty on this term was set to be low, clustering of hydrophobic surface area into patches could still occur. The advantage of the hpatch score is that it penalizes only large patches of hydrophobic surface area, not the total overall hydrophobic surface area. The disadvantage of the score is that is requires the calculation of SASA which slows down design simulations approximately 20-fold. For this reason, the hpatch score will most likely be used as a filter in protein design protocols. Designs will continue to be created with the standard Rosetta energy function and the hpatch score will be used during post-processing to remove designs with large hydrophobic patches.

5.1.2 Decoy discrimination with the hpatch score

Predicting the three-dimensional structure of a protein from its primary sequence is still an unsolved problem. Most structure prediction programs work by sampling protein conformation space and then ranking candidate structures using a scoring function(8). These protocols can generate thousands of models, called decoys, during a simulation. These decoys are then typically clustered in some way and the lowest energy or best scoring structures are submitted as the prediction. The key to obtaining the correct prediction comes down to the ability to identify the native structure from thousands of other decoys, assuming the native structures conformation was sampled during the protocol. In the world of protein structure prediction, this recognition step is referred to as decoy discrimination. Protein folding program energy functions are commonly trained and tested with decoy discrimination tests. In these tests, the native protein, several relaxed natives, and thousands of incorrect structures are combined and the energy function is asked to separate the native structures from the decoys. The better an energy function can determine native from non-native, the better it should be able to predict structure from sequence alone.

By scoring a feature not currently captured by the Rosetta energy function, the hpatch score may be useful in *ab initio* structure prediction. To see if the hpatch score provides any

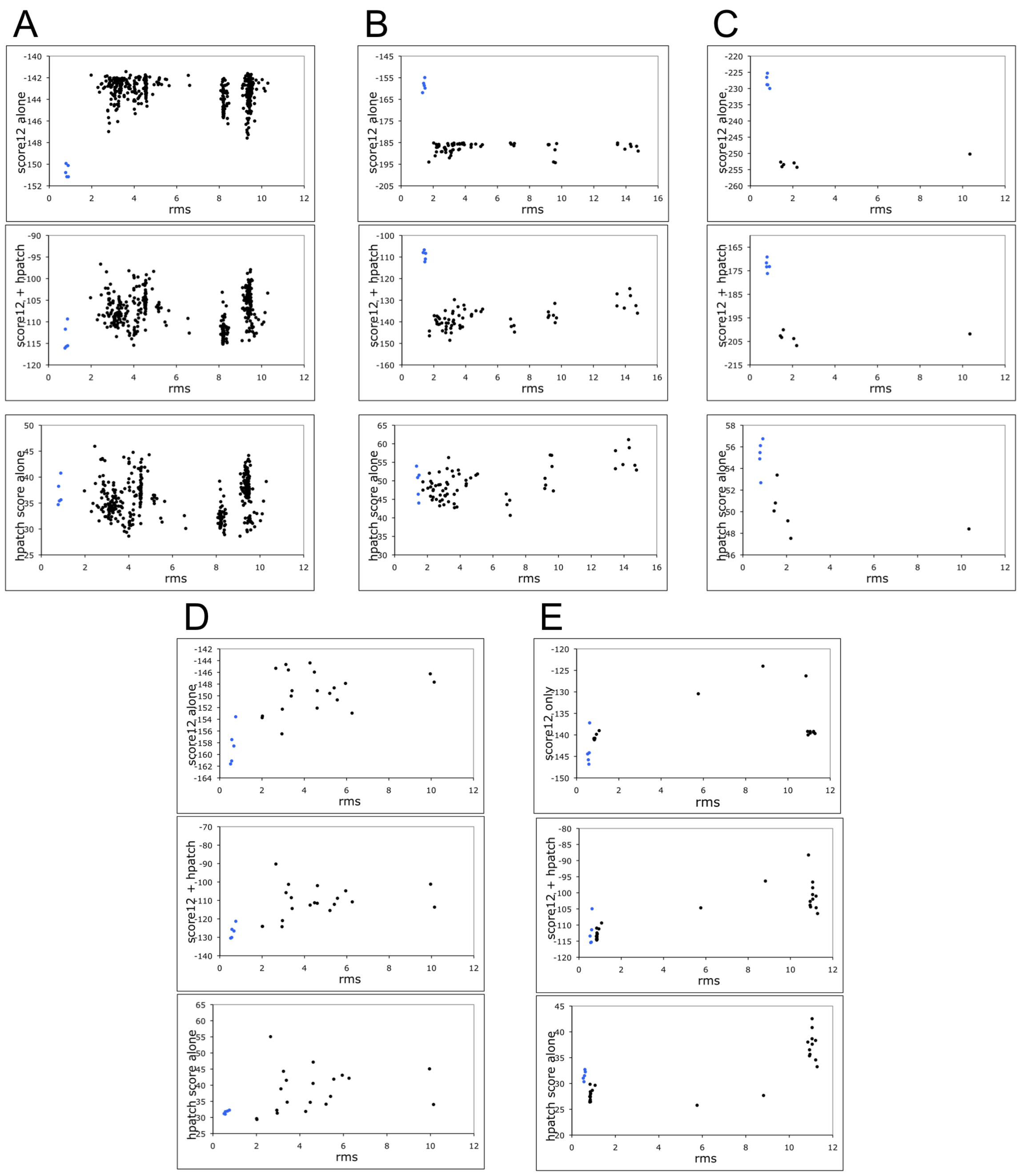


Figure 5.1: Decoy discrimination tests using the pre-hpatch score. The scatterplots show the Rosetta score, Rosetta with pre-hpatch score, and pre-hpatch score alone versus RMSD to the native structure for 5 different proteins. Decoys are indicated with black dots, relaxed natives are indicated with blue dots. These targets have been very hard to predict successfully because the high-RMSD decoys have scores as good as the low-RMSD decoys and relaxed natives. The pre-hpatch score seems to provide some amount of discrimination for proteins B and E, as can be seen from the shift of the high-RMSD decoy points to higher scores.

benefit in decoy discrimination, we used a preliminary version of the score (pre-hpatch) by itself and in conjunction with the Rosetta energy function to identify the native structure in a set of five CASP(9) targets that have been very difficult for Rosetta to predict successfully (Mike Tyka, personal communication). The decoys in these sets of structures have native-like Rosetta energies but high RMSD to the native structure. Therefore, finding a way to detect these structures as being non-native is of high interest. Plots of Rosetta score, Rosetta with pre-hpatch score, and pre-hpatch score alone versus RMSD for all five targets are shown in Figure 5.1. In two of the five targets, the pre-hpatch score gives the high-RMSD decoys higher energies than the native and low-RMSD decoys (Fig. 5.1B, E). It will be interesting to see how the final version of the hpatch score and the optimized energy functions perform in these tests.

5.2 Challenges for multistate design

Many design tasks taken on in the future will likely make use of multistate design. For example, in a recent study by Suarez et al., the authors wanted to design an enzyme capable of catalyzing two reactions(10). By using multistate design, in which sequences were optimized for folding free energy and the *de novo* catalytic activity, the authors were able to redesign *E. coli* thioredoxin so that it had esterase activity and retained the native oxidoreductase activity. The biggest limitation currently for multistate design is the lack of high-resolution structures for the modeling of negative states. One way to sidestep this issue is by using currently available protein modeling tools to predict the negative states. Although structure prediction algorithms are not perfect, *de novo* designed sequences can be fed into these programs to predict how they will fold(11). If the folding trajectory converges on the design model, that gives support that the design model will indeed adopt the desired fold. If the folding trajectory converges to other structures, those structures can be used as negative states during a following round of sequence design. This iterative approach to *de novo* design should increase the probability for obtaining the target state fold. This idea also applies to the design of specific protein-protein interfaces. Docking programs can be used to create negative states for interface design problems, as was done in Chapter 4.

In the near future, it will be desirable to compare different multistate design approaches with experimental results to see which methods perform best. Since multistate design is applied most to changing protein binding specificities, large protein-protein interaction data sets will be needed. Obtaining these affinities one at a time even for small interface design problems is not practical. Experimental methods that can quantitatively test large numbers of interactions quickly, such as protein microarrays(12), will be needed to validate multistate design algorithms. Grigoryan et al. recently described the use of coiled coil arrays to test their implementation of multistate design(13). These protein-protein interaction data sets will also be useful in training and testing energy functions for protein interface design.

5.3 Future directions for protein design

Computational protein design will likely be combined with directed evolution in future design projects. Already a number of studies have used computational design with library screening to rapidly obtain proteins with the desired function(14; 7; 15). In one of the first examples of directed library screening, Hayes et al. succeeded in making β-lactamase mutants highly resistant to the antibiotic cefotaxime(14). Treynor et al. used several computational methods to design libraries of GFP variants and screened them for fluorescence(7). Guntas et al. used a computationally directed library to engineer a protein-protein interface(15). Using Rosetta to design a library, they were able to create E3 ubiquitin-ligase E6AP variants that had nanomolar affinities for a variant of Ubc12. For new design projects, the library approach will probably be taken a step further with the addition of directed evolution to optimize protein properties. Computational protein design and directed evolution are very complementary in that protein design energy functions can sample a much larger space than is accessible with even the best screening methods in directed evolution, and directed evolution is not limited by the inaccuracies that are present in protein design energy functions. The way these two methods would be used for interface design would be that computational design would be used create weakly interacting interfaces followed by directed evolution to increase affinity.

Karanicolas et al. used exactly this type of approach to create a pair of proteins that interact with an affinity of 180 picomolar(16)! Computational design was able to create an interface with 130 nM affinity, and directed evolution was able to find point mutants that increased affinity 1000-fold.

Another area of future work in protein interface design lies in how sampling is performed. A number of approaches currently under development are borrowing information from existing protein-protein interfaces to increase the likelihood of success. Anchored design is an approach where 1-3 continuous residues important for binding in a protein-protein interface, the anchor, and placed in the same orientation into a scaffold protein. The residues surrounding this anchor in the scaffold protein are then optimized for binding the target protein. Building the interface around 1-3 residues that are known to interact with the target in some other protein ensures that at least some affinity will be obtained for the target. Another way to make protein interface design a little easier is to only go after certain types of interfaces. For example, a number of interfaces are formed by two proteins pairing exposed βstrands. Backbone-backbone hydrogen bonds largely determine how two βstrands will come together. Targeting proteins with an exposed β-strand using scaffolds that also have exposed β-strands ensures the proteins will, at the least, interact in the desired orientation. How strongly they interact depends on how complementary the rest of the proteins are. Designing metal atom binding sites at protein-protein interfaces is a third way of reducing the conformational space of protein-protein interface design. In this approach, metal coordinating residues such as cysteine and histidine are introduced on both sides of an interface in the proper geometry for metal binding. The rest of the interface is designed as usual. If the interface is designed well, addition of metal to the solution should lead to the proteins interacting. This approach has already been used with some success(17; 18).

The scoring functions used by protein design programs to rank sequences are not perfect. Identifying where and why the scoring functions fail and fixing them is critical for computational protein design to continue to have success. Scoring functions can be improved by adjusting the parameters of the underlying score terms and/or optimizing the weights of the various terms. Haidar et al. trained an energy function on a set of experimentally characterized enzyme/inhibitor complex point mutants and then used it to design a mutant T-cell receptor (TCR) with 99-fold higher affinity to its MHC complex than the wild type TCR(19). Sharabi et al. optimized the ORBIT energy function for side chain rotamer recovery and then used this energy function in various interface design tests(20). Alternatively, new scoring terms can be added as our understanding of protein folding and design improves. Sheffler et al. developed a new score term for Rosetta which measures the quality of packing in the interior of proteins and protein-protein interfaces(21). Very important to this long-term effort will be a way to see how energy function changes affect performance in the various areas Rosetta is used. For example, changes to the energy function that improve protein design may worsen the quality of predictions made during *ab initio* structure prediction. Having a system in place that benchmarks performance in different areas will be key to further improvements of the energy function.

5.4 New applications of computational protein design

Numerous applications of computational protein design have been discussed in this thesis. Two uses of designed proteins not previously discussed are as therapeutics and biosensors. A number of protein therapeutics are already in use and biologics represent the fastest growing class of therapeutics being approved by the FDA(22). Most of these biologics are monoclonal antibodies that are targeted to various extracellular receptors. Protein therapeutics for targets inside the cell have largely been avoided because of the difficulty of getting drugs inside cells. Liu et al. recently described an approach for delivering functional proteins into mammalian cells. In the study, they were able to deliver active Cre recombinase to 5 different cell lines by fusing the protein to GFP variants with high positive charge(23; 24). Other approaches for delivering proteins into cells have been reviewed(25). Intracellular delivery of designed proteins has the potential to revolutionize medicine. Computational protein design also offers a unique way to study cellular processes *in vivo*. A very active area of work currently lies in the design of biosensors, designed proteins that can detect changes in cells(26). Wu et al. were able to design a Rac1 fusion protein that they used to study Rac1 interactions in living cells

(27). By fusing the protein to the photoactivatable LOV domain, they were able to activate Rac1 spatially and temporally to observe the active Rac1 phenotype very precisely. Finally, not only can designed proteins be used to learn about signaling networks, someday they might even be used to create synthetic cellular networks(28; 29).

References

1. Havranek, J. and Harbury, P. (2002) Automated design of specificity in molecular recognition. Nature Structural & Molecular Biology **10**, 45–52

2. Dzubiella, J., Swanson, J., and McCammon, J. (2006) Coupling nonpolar and polar solvation free energies in implicit solvent models. The Journal of chemical physics **124**, 084905

3. Eisenberg, D. and McLachlan, A. (1986) Solvation energy in protein folding and binding. Nature

4. Lazaridis, T. and Karplus, M. (1999) Effective energy function for proteins in solution. Proteins: Structure, Function, and Bioinformatics **35**, 133–152. ISSN 1097-0134

5. Dahiyat, B. and Mayo, S. (1997) Probing the role of packing specificity in protein design. Proceedings of the National Academy of Sciences of the United States of America **94**, 10172

6. Street, A. and Mayo, S. (1998) Pairwise calculation of protein solvent-accessible surface areas. Folding and Design **3**, 253–258. ISSN 1359-0278

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

8. Bradley, P., Misura, K., and Baker, D. (2005) Toward high-resolution de novo structure prediction for small proteins. Science **309**, 1868. ISSN 0036-8075

9. Moult, J., Pedersen, J., Judson, R., and Fidelis, K. (1995) A large-scale experiment to assess protein structure prediction methods. Proteins: Structure, Function, and Bioinformatics **23**, ii–iv. ISSN 1097-0134

10. Suarez, M., Tortosa, P., Garcia-Mira, M., Rodríguez-Larrea, D., Godoy-Ruiz, R., Ibarra-Molero, B., Sanchez-Ruiz, J., and Jaramillo, A. (2010) Using multi-objective computational design to extend protein promiscuity. Biophysical chemistry **147**, 13–19. ISSN 0301-4622

11. Bazzoli, A., Tettamanzi, A., and Zhang, Y. (2011) Computational Protein Design and Large-Scale Assessment by I-TASSER Structure Assembly Simulations. Journal of Molecular Biology ISSN 0022-2836

12. MacBeath, G. and Schreiber, S. (2000) Printing proteins as microarrays for high-throughput function determination. Science **289**, 1760. ISSN 0036-8075

13. Grigoryan, G., Reinke, A., and Keating, A. (2009) Design of protein-interaction specificity gives selective bZIP-binding peptides. Nature **458**, 859–864. ISSN 0028-0836

14. Hayes, R., Bentzien, J., Ary, M., Hwang, M., Jacinto, J., Vielmetter, J., Kundu, A., and Dahiyat, B. (2002) Combining computational and experimental screening for rapid optimization of protein properties. Proceedings of the National Academy of Sciences of the United States of America **99**, 15926

15. 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. ISSN 0027-8424

16. Karanicolas, ., J, Corn, J., Chen, I., Joachimiak, L., Dym, O., Peck, S., Albeck, S., Unger, T., Hu, W., Liu, G., Delbecq, S., Montelione, G., Spiegel, C., Liu, D., and Baker, D. (2011) A De Novo Protein Binding Pair By Computational Design and Directed Evolution. Molecular Cell

17. Salgado, E., Radford, R., and Tezcan, F. (2010) Metal-directed protein self-assembly. Accounts of chemical research **43**, 661–672. ISSN 0001-4842

18. Salgado, E., Ambroggio, X., Brodin, J., Lewis, R., Kuhlman, B., and Tezcan, F. (2010) Metal templated design of protein interfaces. Proceedings of the National Academy of Sciences **107**, 1827. ISSN 0027-8424

19. Haidar, J., Pierce, B., Yu, Y., Tong, W., Li, M., and Weng, Z. (2009) Structure-Based Design of a T Cell Receptor Leads to Nearly 100-Fold Improvement in Binding Affinity for pepMHC. Proteins **74**, 948

20. Sharabi, O., Yanover, C., Dekel, A., and Shifman, J. (2011) Optimizing energy functions for protein–protein interface design. Journal of Computational Chemistry ISSN 1096-987X

21. Sheffler, W. and Baker, D. (2009) RosettaHoles: Rapid assessment of protein core pack-

ing for structure prediction, refinement, design, and validation. Protein Science **18**, 229–239. ISSN 1469-896X

22. Belsey, M., Harris, L., Das, R., and Chertkow, J. (2006) Biosimilars: initial excitement gives way to reality. Nature Reviews Drug Discovery **5**, 535–536. ISSN 1474-1776

23. McNaughton, B., Cronican, J., Thompson, D., and Liu, D. (2009) Mammalian cell penetration, siRNA transfection, and DNA transfection by supercharged proteins. Proceedings of the National Academy of Sciences **106**, 6111

24. Cronican, J., Thompson, D., Beier, K., McNaughton, B., Cepko, C., and Liu, D. (2010) Potent Delivery of Functional Proteins into Mammalian Cells in Vitro and in Vivo Using a Supercharged Protein. ACS Chemical Biology ISSN 1554-8929

25. Grdisa, M. (2011) The Delivery of Biologically Active (Therapeutic) Peptides and Proteins into Cells. Current medicinal chemistry ISSN 1875-533X

26. Hahn, K. and Toutchkine, A. (2002) Live-cell fluorescent biosensors for activated signaling proteins. Current opinion in cell biology **14**, 167–172. ISSN 0955-0674

27. Wu, Y., Frey, D., Lungu, O., Jaehrig, A., Schlichting, I., Kuhlman, B., and Hahn, K. (2009) A genetically encoded photoactivatable Rac controls the motility of living cells. Nature **461**, 104–108. ISSN 0028-0836

28. Guido, N., Wang, X., Adalsteinsson, D., McMillen, D., Hasty, J., Cantor, C., Elston, T., and Collins, J. (2006) A bottom-up approach to gene regulation. Nature **439**, 856–860. ISSN 0028-0836

29. Ro, D., Paradise, E., Ouellet, M., Fisher, K., Newman, K., Ndungu, J., Ho, K., Eachus, R., Ham, T., Kirby, J., et al. (2006) Production of the antimalarial drug precursor artemisinic acid in engineered yeast. Nature **440**, 940–943. ISSN 0028-0836

Appendix A

The Rosetta all-atom energy function

The functional form of each term in the Rosetta all-atom energy function. Equations reproduced from Rohl et al. (1) and Kuhlman et al. (2).

Lennard Jones

$$\sum_i \sum_{j>i} \begin{cases} \left[\left(\frac{r_{ij}}{d_{ij}}\right)^{12} - 2\left(\frac{r_{ij}}{d_{ij}}\right)^{6}\right] e_{ij}, & if \frac{d_{ij}}{r_{ij}} < 0.6 \\ \left[-8759.2\left(\frac{d_{ij}}{r_{ij}}\right) + 5672.0\right] e_{ij}, & else \end{cases} \tag{A.1}$$

where i, j = residue indices, d = interatomic distance, e = geometric mean of atomic well depths, and r = summed van der Waals radii

Hydrogen bonding

$$\sum_i \sum_j (-\ln[P(d_{ij}|h_j ss_{ij}] - \ln[P(\cos\phi_{ij}|d_{ij}h_j ss_{ij}] - \ln[P(\cos\psi_{ij}|d_{ij}h_j ss_{ij}]) \tag{A.2}$$

where i = donor residue index, j = acceptor residue index, d = acceptor-proton interatomic distance, h = hybridization (sp^2, sp^3), ϕ = proton-acceptor-acceptor base bond angle, ψ =donor-proton-acceptor base bond angle

Solvation

$$\sum_i \left[\Delta G_i^{ref} - \sum_j \left(\frac{2\Delta G_i^{free}}{4\pi^{3/2}\lambda_j r_{ij}^2} e^{-d_{ij}^2 V_j} + \frac{2\Delta G_j^{free}}{4\pi^{3/2}\lambda_j r_{ij}^2} e^{-d_{ji}^2 V_i}\right)\right] \tag{A.3}$$

where i, j = atom indices, d = interatomic distance, r = summed van der Waals radii, λ = correlation length, V = atomic volume, $\Delta G^{ref}, \Delta G^{free}$ = energy of a fully solvated atom

Amino acid self-energy

$$\sum_i -\ln\left[\frac{P(aa_i|\phi_i,\psi_i)}{P(aa_i)}\right] \tag{A.4}$$

where i = residue index, aa = amino acid type, ψ, ψ =backbone torsion angles

Residue pair interactions

$$\sum_i \sum_{j>i} -ln\left[\frac{P(aa_i, aa_j|d_{ij}, env_i, env_j)}{P(aa_i|d_{ij}, env_i)P(aa_j|d_{ij}, env_j)}\right] \tag{A.5}$$

where i, j = residue indices, d = distance between residues, aa = amino acid type, $env_{i,j}$ = enviroment of residue i or j

Ramachandran torsion preferences

$$\sum_i -\ln[P(\phi_i, \psi_i|aa_i, ss_i)] \tag{A.6}$$

where i = residue index, ψ, ψ = backbone torsion angles (36^o bins), aa = amino acid type, ss = secondary structure

Rotamer self-energy

$$\sum_i -\ln\left[\frac{P(rot_i|\phi_i,\psi_i)P(aa_i|\phi_i,\psi_i)}{P(aa_i)}\right] \tag{A.7}$$

where i = residue index, rot = Dunbrack backbone-dependant rotamer, aa = amino acid type, ψ, ψ = backbone torsion angles

Reference energy

$$\sum_{aa} n_{aa} \tag{A.8}$$

where aa = amino acid type, n = number of residues

References

1. Rohl, C., Strauss, C., Misura, K., and Baker, D. (2004) Protein structure prediction using Rosetta. Methods in enzymology **383**, 66–93. ISSN 0076-6879

2. Kuhlman, B., Dantas, G., Ireton, G., Varani, G., Stoddard, B., and Baker, D. (2003) Design of a novel globular protein fold with atomic-level accuracy. Science **302**, 1364. ISSN 0036-8075