REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)* 20-09-2010	**2. REPORT TYPE** Thesis	**3. DATES COVERED** *(From - To)* SEPT 2010 - OCT 2010
4. TITLE AND SUBTITLE Methods and Applications in Computational Protein Design		**5a. CONTRACT NUMBER** FA8720-05-C-0002
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jason Charles Biddle		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) MIT Lincoln Laboratory 244 Wood Street Lexington, MA 02420		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) ESC/CAA 20 Schilling Circle, Bldg 1305 Hanscom AFB, MA 01731		**10. SPONSOR/MONITOR'S ACRONYM(S)** ESC/CAA
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this thesis, we summarize our work on applications and methods for computational protein design. First, we apply computational protein design to address the problem of degradation in stored proteins. Specifically, we target cysteine, asparagine, glutamine, and methionine amino acid residues to reduce or eliminate a proteins's susceptibility to degradation via aggregation. deamidation, and oxidation. We demonstrate this technique on a subset of degradation-prone amino acids in phosphotriesterase, an enzyme that hydrolyzes toxic organophosphates including pesticides and chemical warfare agents. Second, we introduce BroMAP/A*, an exhaustive branch-and-bound search technique with enumeration. We compare performance of BroMAP/A* to DEE/A*, the current standard for conformational search with enumeration in the protein design community. When limited computational resources are available, DEE/A* sometimes fails to find the global minimum energy conformation and/or putational resources, we show how BroMAP/A* is able to solve large designs by efficiently dividing the search space into small, solvable subproblems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: U			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Zach Sweet
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	SAR	108	**19b. TELEPHONE NUMBER** *(include area code)* 781-981-5997

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

JA-GEN-6900 A

Methods and Applications in Computational Protein Design

by

Jason Charles Biddle

B.S., The Pennsylvania State University (2005)

Submitted to the School of Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

THIS MATERIAL HAS BEEN CLEARED
FOR PUBLIC RELEASE BY 66 ABG/PA

DATE: 20 Sep 10

CASE #: 66ABW-2010-1127

Author ....
School of Engineering
August 5, 2010

Certified by ....
Bruce Tidor
Professor of Biological Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Karen Willcox
Associate Professor of Aeronautics and Astronautics
Codirector, Computation for Design and Optimization Program

Methods and Applications in Computational Protein Design

by

Jason Charles Biddle

Submitted to the School of Engineering
on August 5, 2010, in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computation for Design and Optimization

Abstract

In this thesis, we summarize our work on applications and methods for computational protein design. First, we apply computational protein design to address the problem of degradation in stored proteins. Specifically, we target cysteine, asparagine, glutamine, and methionine amino acid residues to reduce or eliminate a protein's susceptibility to degradation via aggregation, deamidation, and oxidation. We demonstrate this technique on a subset of degradation-prone amino acids in phosphotriesterase, an enzyme that hydrolyzes toxic organophosphates including pesticides and chemical warfare agents. Second, we introduce BroMAP/A*, an exhaustive branch-and-bound search technique with enumeration. We compare performance of BroMAP/A* to DEE/A*, the current standard for conformational search with enumeration in the protein design community. When limited computational resources are available, DEE/A* sometimes fails to find the global minimum energy conformation and/or enumerate the lowest-energy conformations for large designs. Given the same computational resources, we show how BroMAP/A* is able to solve large designs by efficiently dividing the search space into small, solvable subproblems.

Thesis Supervisor: Bruce Tidor
Title: Professor of Biological Engineering and Computer Science

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Acknowledgments

First and foremost, I would like thank my thesis advisor, Professor Bruce Tidor, for his patience and guidance during my graduate studies. Bruce always set aside time to help me move forward with this work, even if it meant having to review basic chemistry or debug computer code. I greatly admire Bruce's passion towards his students and their work, and I am grateful for the opportunity to collaborate with Bruce over the past two years.

I would also like to thank former and current members of the Tidor Lab for their insights and comments on this research. Many of the issues I encountered were resolved as a direct result of their suggestions during afternoon discussions and lab meetings.

I would also like to thank the Computation for Design and Optimization (CDO) Program for providing the opportunity for me to develop a strong foundation in computational science and engineering at MIT. I especially thank Laura Koller for her many helpful email reminders and her assistance with administrative issues.

I would also like to thank MIT Lincoln Laboratory for funding my graduate studies through the Lincoln Scholars Program.

Finally, I would to like to dedicate this thesis to Briar and to my family. This accomplishment would not have been possible without their unwavering love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Computational protein design

Proteins perform many important roles in living organisms. One type of protein, called an enzyme, catalyzes biochemical reactions that are vital to metabolism. Other types of protein include signaling proteins, which help coordinate cellular activities. Structural proteins form scaffolds to maintain cell shape. Mechanical proteins play a variety of roles such as contraction of muscle fibers. Different proteins can also coordinate in the form of a structural complex to achieve a particular function.

Proteins are biomolecules made of a sequence of amino acids. There are 20 different amino acids that one can think of as protein building blocks. All amino acids contain a central alpha carbon atom connected to a hydrogen, an amino group, a carboxyl group, and a characteristic side chain. The 20 common side chains have different structural and chemical properties. Small side chains like glycine and alanine only have a few atoms. Large side chains like arginine and tryptophan have many atoms. Some side chains attract water (hydrophilic) while others repel water (hydrophobic). Some side chains are positively or negatively charged while others carry no charge.

Amino acids adjacent in sequence are chemically linked, or "chained", through peptide bond formation between the carboxyl group of one amino acid and the amino group of the following amino acid. This main chain is commonly referred to as a protein's backbone. The amino acid side chains then project out from the backbone. The sequence ordering and chemical properties of the side chains determine how the linear chain will fold into a three-dimensional structure. In turn, the protein's three-dimensional structure determines its function.

In computational protein design, we try to find amino acid sequences that fold to a predetermined structure with desirable properties. Developing novel proteins with certain catalytic, affinity, specificity, and stability properties is an active area of research in biology with broad applications. For example, Lippow *et al.* re-engineered multiple antibodies for significantly improved antigen binding affinity using computational design [1]. Antibody-affinity is important in the development of therapeutic agents. Chen *et al.* used computational design to re-engineer the nonribosomal peptide synthetase enzyme GrsA–PheA to bind substrates for which the wild-type enzyme has little or no specificity [2]. *De novo* design of proteins has also been demonstrated for small sequences. Dahiyat and Mayo computationally designed a 23-residue zinc finger protein [3]; Kuhlman *et al.* designed a 98-residue α/β protein with a novel fold [4]. Additional examples can be found in a recent review of progress in computational protein design by Lippow and Tidor [5].

We typically begin the computational protein design process by deciding on a set of desirable properties for a protein whose three-dimensional structure has been determined experimentally by X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy. The structure provides three-dimensional coordinates for all atoms in the protein. We can easily identify the backbone and side chains by combining these coordinates with the protein's amino acid sequence. The backbone atom coordinates define a backbone conformation. Similiarly, the side chain atom coordinates define one side chain conformation for each sequence position. The set of side chain conformations together with the backbone conformation define a global conformation for the protein. Next, we choose sequence positions to consider for mutation. These choices are typically driven by the set of desirable properties we aim to achieve in our design.

Once we have a three-dimensional structure and set of mutations, we enter the

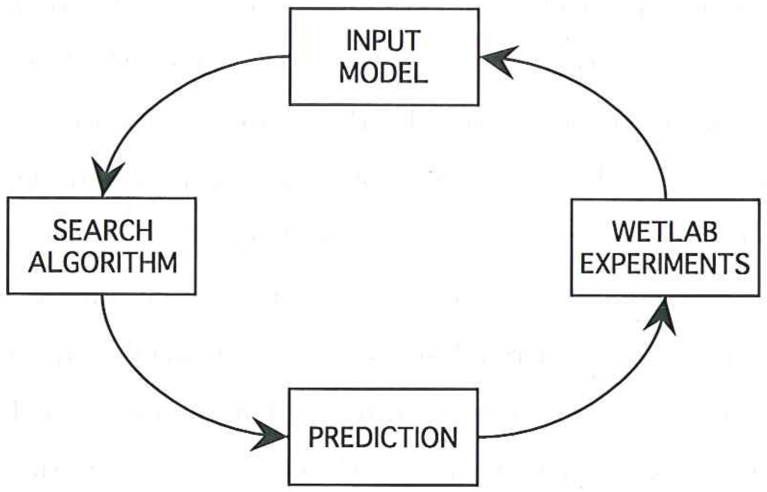


Figure 1-1: The protein design cycle.

protein design cycle shown in Figure 1-1. The input model specifies degrees of freedom (i.e., allowable backbone and side chain conformations) for the protein's structure in our calculations. It also contains an energy function used to evaluate each of the mutant structures. Next, a search algorithm identifies low-energy structures from the set of allowable mutations. Promising mutant sequences are then synthesized and tested to see if they achieve the desirable properties. Finally, the experimental results are used to validate and refine the input model for subsequent designs.

For the input model, we assume either a rigid or flexible backbone. With a rigid backbone, mutant amino acid side chains that clash with the backbone are assumed to destabilize the folded protein, and we typically discard them during the search procedure. On the other hand, we may allow some discrete or continuous backbone flexibility in the local three-dimensional neighborhood to accomodate a mutant side chain. A rigid backbone reduces the problem size because only side chains are allowed to move in the design. However, a flexible backbone more accurately represents backbone behavior in proteins. Backbones are not completely rigid, and they are known to change conformation during events like ligand binding.

We model the conformations of amino acid side chains using a rotamer library. Side chain rotamer libraries are built by systematically rotating bond angles around the carbon atoms for a particular amino acid. This search yields a discrete set of preferred, or low-energy, conformations for each type of amino acid relative to its central alpha carbon atom and amino group and carboxyl groups. The discrete treatment of side chain orientations is physically reasonable based on analyses of side chain conformations in proteins with determined structures. It has been shown that a small set of conformations occur more frequently than others for each type of amino acid [6]. By specifying side-chain rotamers relative to the amino acid's central alpha carbon and two backbone groups, we can efficiently insert any rotamer in the library at a particular sequence position independent of the other positions in the protein.

We can compute the potential energy of a given protein conformation using quantum or molecular mechanics. Quantum mechanical calculations are very accurate but computationally expensive. Explicit treatment of all atoms in a protein with quantum mechanics is computationally intractable; however, quantum calculations can be used to evaluate small regions of the protein (e.g., the active site). Conversely, molecular mechanics calculations are approximate but computationally efficient. Molecular mechanics models use a parameter set and force field to treat all atoms in a protein. The parameter set provides values for atom properties like net charge and molecular volume (i.e., van der Waals radius). It also provides a set of standard bond lengths, bond angles, and improper dihedrals for each amino acid type. These parameter set values are obtained by fits to experimental and theoretical data from quantum mechanics. The molecular mechanics force field is typically comprised of covalent and non-covalent energy terms. The covalent terms captures energy from bond lengths, bond angles, torsions, and improper dihedral terms. The non-covalent term captures van der Waal and electrostatic interactions between all pairs of atoms.

Once we have computed the energy terms for each possible side-chain rotamer in a design, we want to find the rotamer for each design position that produces the most energetically favorable protein conformation. For non-trivial designs, this discrete search problem is too large to solve with brute-force methods. For example, consider a design with 10 positions. Suppose we allow each position to mutate to any of the 20 amino acids, and each amino acid has 100 possible side-chain rotamers. In this example, there are 10^{33} conformations, or possible combinations of rotamers. In practice, designs can reach the scale of 10^{80}–10^{100} conformations (i.e., more combinations of rotamers than there are atoms in the known universe). Despite the combinatorial complexity of protein design problems, we can efficiently find exact and approximate solutions using guaranteed and non-guaranteed search methods, respectively.

Ideally, we would like to identify the optimal choice of rotamers, or global minimum energy conformation (GMEC), along with an ordered list up from the GMEC. Given unlimited computational resources and time, guaranteed search methods provably identify the GMEC. Some guaranteed methods also enumerate all of the conformations within an energy threshold of the GMEC. The current standard for guaranteed search with enumeration in protein design is Dead-End Elimination (DEE) with A* search, or DEE/A*. DEE reduces the search space. In addition to taking advantage of the fact that many rotamer combinations produce unstable protein conformations due to van der Waals clashes and unfavorable electrostatic interactions, DEE identifies rotamers and rotamer combinations inconsistent with being in the GMEC (i.e., the problem solution) [7–9]. DEE is applied iteratively until no additional rotamers can be pruned, and then A* search is used to find the GMEC along with an ordered list in the reduced conformational space [10]. Another more recent guaranteed method is branch-and-bound rotamer optimization using maximum-a-posteriori estimation (BroMAP) [11]. BroMAP recursively divides the search space into subproblems and keeps track of the tightest upper bound. If the subproblem is small enough in terms of conformational complexity, it's solved with DEE/A*. Otherwise, the subproblem is statistically bounded and compared to the upper bound to determine whether it can be eliminated or whether it requires further subdivision. Although BroMAP uses A* search to solve subproblems, it was not designed to enumerate conformations up from the GMEC. The final guaranteed method we mention is Integer Linear Programming (ILP). In ILP, the GMEC search problem is treated explicitly as an optimization problem with a binary decision variable for each individual rotamer [12]. Although ILP can exactly solve small protein designs, continuous problem relaxations are needed for large designs. These relaxations typically result in fractional, or approximate, solutions as opposed to integral solutions (i.e., the GMEC). Additionally, ILP does not provide an easy or efficient way to enumerate conformations up from the GMEC.

Non-guaranteed search methods identify conformations that are local minima based on an optimality criterion. For example, a local minimum might be a conformation where we cannot select a different rotamer at any one design position that reduces the overall energy. One common non-guaranteed search method is self-consistent mean-field (SCMF) theory, which iteratively refines a probability distribution over the rotamers at each position [13]. The distribution represents the likelihood that a particular rotamer belongs to the GMEC. SCMF is a deterministic method, which means it converges to the same conformation no matter how many times we run it on a given problem. Other non-guaranteed methods used in protein design include genetic algorithms [14] and Monte Carlo search [15]. Unlike SCMF, genetic algorithms and Monte Carlo are non-deterministic methods, which means they can converge to a different conformation each time we run them on a given problem.

Generally, non-guaranteed search methods are computationally faster than guaranteed methods. Search methods like DEE/A* and BroMAP must explicitly consider the entire conformational space to guarantee global optimality of a conformation, which requires considerable computational time and resources for large designs. The decision of whether to trade speed for accuracy in conformational search depends on one's design goals. If our goal is to quickly identify some low-energy conformation(s) or to upper bound the GMEC energy, then non-guaranteed methods are probably the better choice. However, if we want to reserve experimental resources and testing for only the lowest-energy conformations, then guaranteed search methods are the better choice. Additionally, guaranteed methods, particularly those that also enumerate all conformations within a given energy threshold of the GMEC, play an important role in the rational protein design cycle.

Suppose we have an order listed of the lowest-energy conformations within a given threshold of the GMEC. We then analyze the list and select a set of candidate mutant sequences for synthesis and experimental testing. At a later time, someone experimentally tests a mutant sequence that was not in our list of the lowest-energy mutations, and he or she finds that this new sequence is energetically more favorable than any of our candidates. Assuming the new sequence was represented in our search space, the finding suggests a deficiency in our input model (i.e., backbone assumption, rotamer library, and energy function). If the input model was accurate, then the new sequence would have been in the list returned by the guaranteed search algorithm. If instead we had used a non-guaranteed search for our predictions, then we could not immediately rule out the search algorithm as the cause. With non-guaranteed methods, one cannot say definitively that a lower-energy sequence does not exist in the search space.

As the above example demonstrates, guaranteed search methods with enumeration are a valuable tool in the rational protein design cycle. By using guaranteed search, we can be sure our predictions are optimal given our backbone assumption, rotamer library, and energy function. Discrepencies between computational predictions and experimental results can help us identify and fix gaps in our input model. By improving the accuracy of our input model, we improve the accuracy of predictions during subsequent iterations of the design cycle. In this thesis, we focus our attention on the application and development of guaranted search methods with enumeration for computational protein design.

1.2 Our work and contributions

First, we apply computational protein design to address the problem of degradation in stored proteins. Degradation occurs when a protein undergoes chemical or physical changes that alter its structure and function. Specifically, we target cysteine, asparagine, glutamine, and methionine amino acid residues to reduce or eliminate a protein's susceptibility to degradation via aggregation, deamidation, and oxidation. We demonstrate this technique on a subset of degradation-prone amino acids in phosphotriesterase (PTE), an enzyme that hydrolyzes toxic organophosphates including pesticides and chemical warfare agents. We use structure-based protein design to identify single mutations for Cys59, Cys227, Met314, Asn38, and Asn265 in PTE. We also discuss double and triple mutations predicted to significantly improve the packing stability of the neighborhood around Asn265.

Second, we introduce BroMAP/A*, an exhaustive branch-and-bound rotamer search technique with enumeration. Hong *et al.* developed BroMAP (branch-and-bound rotamer optimization using maximum-a-posteriori estimation), a guaranteed search method which indentifies the global minimum energy conformation (GMEC) [11]. We extend the framework of BroMAP to enumerate all conformations within an energy threshold of the GMEC. We compare performance of BroMAP/A* to DEE/A*, the current standard for conformational search with enumeration in the protein design community. When limited computational resources are available, DEE/A* sometimes fails to find the global minimum energy conformation and/or enumerate the lowest-energy conformations for large designs. Given the same computational resources, we show how BroMAP/A* is able to solve large designs by efficiently dividing the search space into small, solvable subproblems.

1.3 Thesis outline

In Chapter 2, we review preparation protocols and degradation pathways for stored proteins. We discuss how computational protein design can be used to limit degradation and improve the thermodynamic stability of stored proteins. We then apply this technique to phosphotriesterase and present mutations predicted to reduce its susceptibility to degradation and improve stability. Chapter 3 focuses on search algorithms for computational protein design. We discuss techniques for exhaustive search with enumeration and detail our developmental and characterization work on BroMAP/A*. We compare performance of DEE/A* and BroMAP/A* for a set of protein design test cases, and we show the advantage of using BroMAP/A* when limited computational resources are available. Finally, in Chapter 4, we conclude the thesis with a summary of our findings and suggestions for future research.

Chapter 2

Protein design for enhanced storage stability

2.1 Introduction

Proteins are stored in either an aqueous buffer or a lyophilized (freeze-dried) form. Aqueous formulations are easier to manufacture and to handle by the end-user. However, the solution must typically be kept at low temperature with minimal agitation [16, 17]. These conditions can be difficult to guarantee during shipping and extended storage; thus, aqueous formulations are typically used only for short-term storage in laboratory settings. Alternatively, proteins can be lyophilized to produce a dry powder that is generally more stable for long-term storage. Lyophilization is a three-stage process whereby the solvated protein is first frozen to produce large crystals. Then the frozen protein undergoes primary and secondary drying phases under vacuum to remove water. Primary drying removes most of the water ($\approx 95\%$) which is frozen as ice on the surface of the crystal. Secondary drying removes the remaining liquid water bound to the crystal.

In both liquid and dry storage formulations, proteins are susceptible to degradation. Degradation can also occur during preparation and manufacturing of the formulation. We say a protein is degraded when it undergoes chemical or physical changes that alter its structure and function. Chemical changes to individual amino acids or even slight physical changes to backbone conformation can significantly affect folding, three-dimensional structure, and activity. Environmental factors such as temperature, pH, and ionic strength play an important role in degradation. Certain conditions may reduce the probablility or frequency of chemical and physical changes while other conditions may accelerate degradation.

One method of mitigating degradation is the rational design of preparation and storage protocols for stable aqueous and lyophilized formulations. Given knowledge of the degradation mechanisms in proteins and the conditions under which they occur, one can carefully control the temperature, pH, and chemical composition of the formulation as well as add various stabilizers to limit degradation. This type of rational design is commonly used in developing stable protein pharmaceuticals. Reviews by both Wang and Carpenter *et al.* provide extensive guidance on designing stable aqueous and lyophilized formulations [18–20].

There are two limitations to rationally designing preparation processes and storage protocols. First, it may be difficult or impossible to limit exposure to certain temperature or pH levels during preparation, particularly for solid-state preparations. For example, during lyophilization extreme temperature levels are needed for the freezing and drying steps, and pH extremes often occur during lyophilization. Second, it may be unreasonable to expect stringent control of environmental factors such temperature, pH, or exposure to air for any type of applied use outside of laboratory or clinical settings. Ideally, we want the protein to remain stable and active even in the harshest environmental conditions.

We can use protein design as a tool to overcome these limitations. Suppose we know a particular type of amino acid is prone to degradation at high pH. Now suppose we use protein design to identify mutations for each instance of the degradation-prone amino acid in our target protein. If these mutations maintain or improve the protein's thermodynamic stability without affecting its activity, then the protein is no longer susceptible to the degradation pathway. Additionally, we are no longer restricted to lower pH levels during preparation and storage of the target protein (assuming high pH does not affect some other aspect of stability).

Along with the ability to replace degradation-prone amino acids, protein design gives us the ability to identify additional mutations to improve the overall thermodynamic stability of the protein. Any mutations that accelerate folding and/or stabilize the folded state are beneficial as long as they do not impair the protein's activity. For example, a single mutation could result in additional or stronger interactions with nearby residues compared to wild type, thereby improving local stability. We could also introduce multiple mutations to repack structurally significant regions of the protein. In both cases, we improve thermodynamic stability by increasing the energy required for the protein to go from a folded to an unfolded state.

In this chapter, we describe initial steps towards redesigning a protein for improved storage stability. First we review degradation pathways for proteins and discuss which amino acids to target with protein design. Then we introduce the protein of interest for our redesign, phosphotriesterase, and highlight its use for bioremidiation of toxic organophosphate chemicals. After describing our computational methods, we present mutations predicted to improve the thermodynamic stability of phosphotriesterase by reducing its susceptibility to degradation during preparation and storage.

2.2 Degradation of stored proteins

Degradation in proteins can be categorized into chemical or physical mechanisms [21]. Chemical degradation occurs when an amino acid residue is covalently modified to produce a new molecule. Physical degradation occurs when the protein's native three-dimensional structure is changed. Chemical degradation can lead to physical degradation and vice versa. In this section we review three degradation pathways in stored proteins: aggregation, deamidation, and oxidation. Aggregation is a form of physical degradation often driven by chemical disulfide bond formation between cysteine residues. Deamidation is a chemical process whereby aspargine or glutamine residues interact with the protein backbone resulting in either chemical or physical degradation. Oxidation is a form of chemical degradation resulting generally from interaction of cysteine or methionine residues with atmospheric oxygen dissolved in

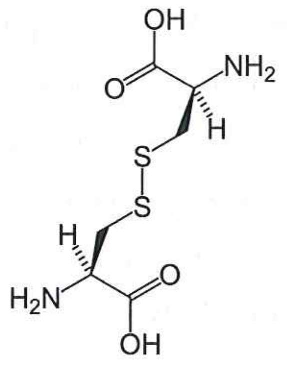


Figure 2-1: Disulfide bond between two cystine residues.

solution.

2.2.1 Aggregation

Cysteine residues play an important structural role in proteins. Cysteine contains a thiol group which can be oxidized to form a disulfide bond with a neighboring cysteine residue as illustrated in Figure 2-1. Disulfide bonds typically stabilize the three-dimensional structure of proteins. For single polypeptide chains, these bonds assist in folding by destabilizing the unfolded state through reduced chain entropy. Disfulfide bonds can also bridge two different polypeptide chains. In both cases, disulfide bonds between cysteine residues increase the thermodynamic stability of the protein or complex.

Although cysteines are structurally beneficial in many proteins, they can also introduce physical instability via aggregation when the protein is in an unfolded or partially folded state. Aggregation occurs when two cysteine residues that are not paired in the native three-dimensional structure of a protein form a disulfide bond. For example, two unpaired cysteine residues on the same polypeptide chain may bond and prevent the chain from properly folding. Another example is disfulfide bonding of unpaired cysteine residues from different chains, which can prevent either chain from properly folding. If multiple cysteines are present, additional chains can also link in a cascading fashion to form a daisy chain complex of inactive protein.

Protein concentration, temperature, and pH significantly affect protein aggregation [18]. Aggregation is more likely for highly concentrated protein formulations in which many polypeptide chains are in close proximity. High temperatures tend to induce protein unfolding, which can increase the likelihood of disulfide bond formation between unpaired cysteines. Finally, ionization of the free thiol group required for disulfide bond formation is accelerated at high pH (i.e., alkaline) conditions.

To mitigate aggregation using computational protein design, we must consider the structural role of all cysteine residues in the target protein. Given the three-dimensional structure of our protein, we can visually inspect all cysteine residues to determine whether they are close enough to form a disulfide bond. Initially, we may choose to remove the unpaired or "free" cysteines but keep the paired cystines forming a disulfide bond. If the paired cystines still result in aggregation, we may wish to remove them, too. Eliminating structurally significant disfulfide bonds will likely decrease thermodynamic stability. In this case, we may need to redesign the neighborhood around paired cystines to reintroduce stability.

2.2.2 Deamidation

Asparagine and glutamine residues are prone to a degradation reaction known as deamidation. Deamidation of asparagine occurs when the amide functional group reacts with the nearest carboxyl-side peptide bond nitrogen to produce ammonia and a cyclic five-membered imide intermediate. Hydroylsis of the intermediate results in either chemical modification of the original asparagine (i.e., aspartic acid) or modification of the local backbone conformation (i.e., isoaspartic acid). Figure 2-2 shows the complete deamidation reaction for aspargine. Deamidation of glutamine follows a similar reaction except it produces a six-membered glutarimide intermediate because glutamine has an additional methylene group compared to asparagine. The six-membered intermediate is less energetically favorable than its five-membered counterpart for asparagine. As a result, deamidation rates for glutamine residues are slower than for aspargine residues by a factor of roughly 100 [22].

The primary sequence of a protein plays an important role in deamidation. Deami-

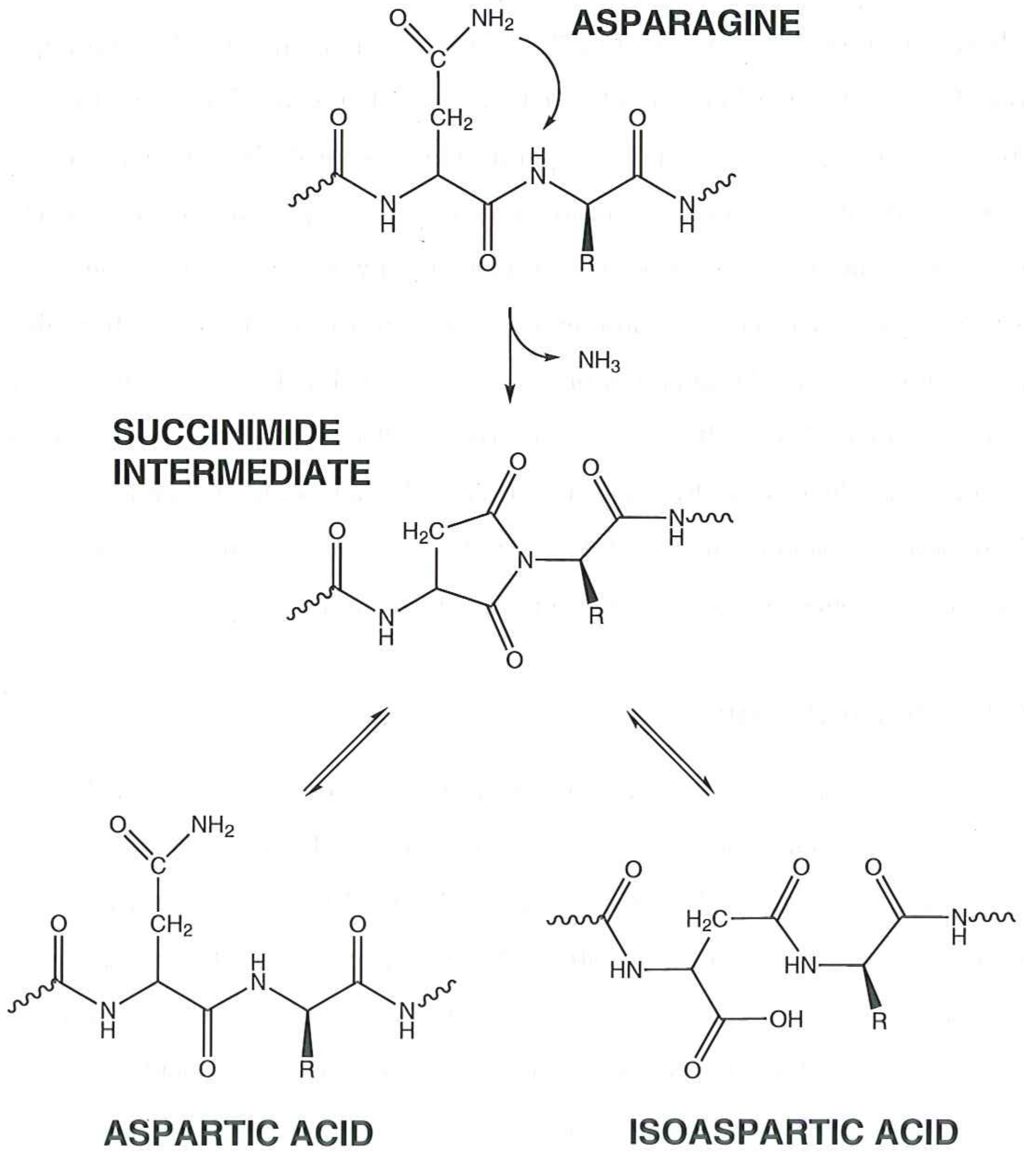


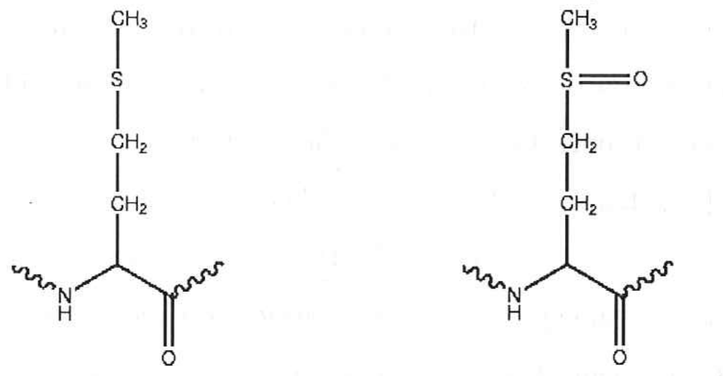
Figure 2-2: Deamidation of asparagine.

dation rates strongly depend on the amino acid immediately following the aspargine or glutamine (i.e., carboxyl-side residue). The preceeding amino acid (i.e., amino-side residue) also affects deamidation; however, the effect of the carboxyl-side residue is 20 times stronger [23]. Large residues such as phenylalanine, tyrosine, and tryptophan with their aromatic rings sterically hinder the deamidation reaction. Small residues such as glycine and alanine provide minimal steric protection from deamidation. Separate work by Capasso and Robinson propose equations to predict deamidation rates from primary sequence based on fits to experimental measurements of small polypetide chains containing asparagine and glutamine [23, 24].

The protein's three-dimensional structure also affects deamidation rates. Folding can bring amino acids in close proximity to an asparagine or glutamine which are not immediately adjacent in sequence. As with primary sequence, these neighboring residues in three-dimensional structure can sterically hinder the deamidation reaction. Neighboring residues can also increase deamidation rates; pockets of mostly acidic or mostly basic side chains can accelerate the deamidation reaction. Robinson proposes a method to predict deamidation rates from a protein's three-dimensional structure based on a fit to 22 proteins with known structure whose deamidation rates have been measured [25, 26]. However, these predictions are based on a fit to deamidation measurements under very specific experimental conditions (pH 7.4, 37 °C, and tris-HCl buffer), which may not generalize to protein conditions during preparation and long-term storage.

Robinson and Robinson have studied demidation rates under a wide range of experimental conditions [27]. Demidation rates take on a U-shaped distribution as a function of pH; rates are highest at pH extremes and lowest at slightly acidic pH ($\approx$ 6.5). Temperature also affects the deamidation, increasing its rate 1.2–2 fold per 10 °C in slightly basic environments. For aqueous solutions, phosphate buffers catalyze the deamidation reaction as opposed to Tris or histidine buffers. Additionally, deamidation rates were observed to increase with ionic strength.

We can use protein design to prevent deamidation reactions or to decrease its rate. Ideally, we want to mutate all asparagine and glutamine residues in the target protein



(a) Methionine (b) Methionine sulfoxide

Figure 2-3: Methionine and its oxidized form.

to eliminate the possibility of deamidation. If we have difficulty finding suitable mutations for particular positions, we can exploit the effect of primary sequence and three-dimensional structure on deamidation. For example, we might sterically protect an asparagine or glutamine from deamidation by mutating the carboxyl-side residue to a large amino acid like phenylalanine, tyrosine, or tryptophan. We might also repack the local neighborhood to introduce additional steric hindrance.

2.2.3 Oxidation

Methionine, cysteine, histidine, tyrosine, and tryptophan residues are all prone to oxidation; however, only methionine and cysteine are easily oxidized by dissolved atmospheric oxygen. We have already identified cysteine as a residue of interest based on its role in aggregation, so we focus our discussion here on methionine oxidation. Atmospheric oxygen reacts with the sulfur atom in methionine to form methionine sulfoxide. Figure 2-3 shows the native and sulfoxide forms of methionine.

We can think of oxidation as substitution of the non-polar methionine for a larger, more polar methionine sulfoxide residue. The effect of this substitution depends on the residue's local environment. The increased polarity may introduce water to a hydrophobic neighborhood. The oxygen in methionine sulfoxide may also hydrogen bond with a nearby residue. Moreover, the added bulk of the oxygen may disrupt side chain packing. These local changes in chemistry and structure can impact the overall thermodynamic stability of the protein.

Methionine oxidation has been observed in proteins in aqueous and lyophilized protein formulations exposed to atmospheric oxygen during preparation and storage [28]. To counteract oxidation, vials of stored protein are sometimes sealed with pure nitrogen instead of oxygen. Light exposure can also affect methionine oxidation rates. For some proteins such as lyophilized human insulin-like growth hormone (hIGI-I), exposure to light can increase the oxidation rate by up to a factor of 30 [29].

To mitigate oxidation using protein design, we should consider all methionine residues in the target protein. In the unfolded state, all methionine residues are susceptible to oxidation if exposed to atmospheric oxygen. In the folded state, methionine residues on the surface of the protein are more susceptible to oxidation than those buried in the protein core. Work by Kim *et al.* highlights this difference between surface and buried methionines, and it provides an interesting case study on designing oxidation-resistant proteins [30]. They compared the effect on stability of methionine oxidation versus mutagenesis for the protein staphylococcal nuclease. Staphylococcal nuclease contains four methionine residues, three of which are solvent exposed. They found mutation of the buried methionine was as disruptive as oxidation in terms of stability, whereas mutation of the solvent-exposed methionines was less disruptive than oxidation.

2.3 Phosphotriesterase

Phosphotriesterase (PTE) is an enzyme that catalyzes the hydrolysis of a variety of organophosphorus compounds. It belongs to the amidohydrolase superfamily [31]. PTE was first identified in *Flavobacterium* extracted from Philippine rice patties that had been treated with the pesticide diazinon [32]. Shortly thereafter, PTE was isolated from the soil bacterium *Pseudomonas diminuta* [33]. Although the natural substrate for PTE remains unknown, PTE has been shown to hydrolyze toxic organophosphate pesticides and nerve agents such as tabun (GA), sarin (GB), soman (GD), and VX [34, 35].

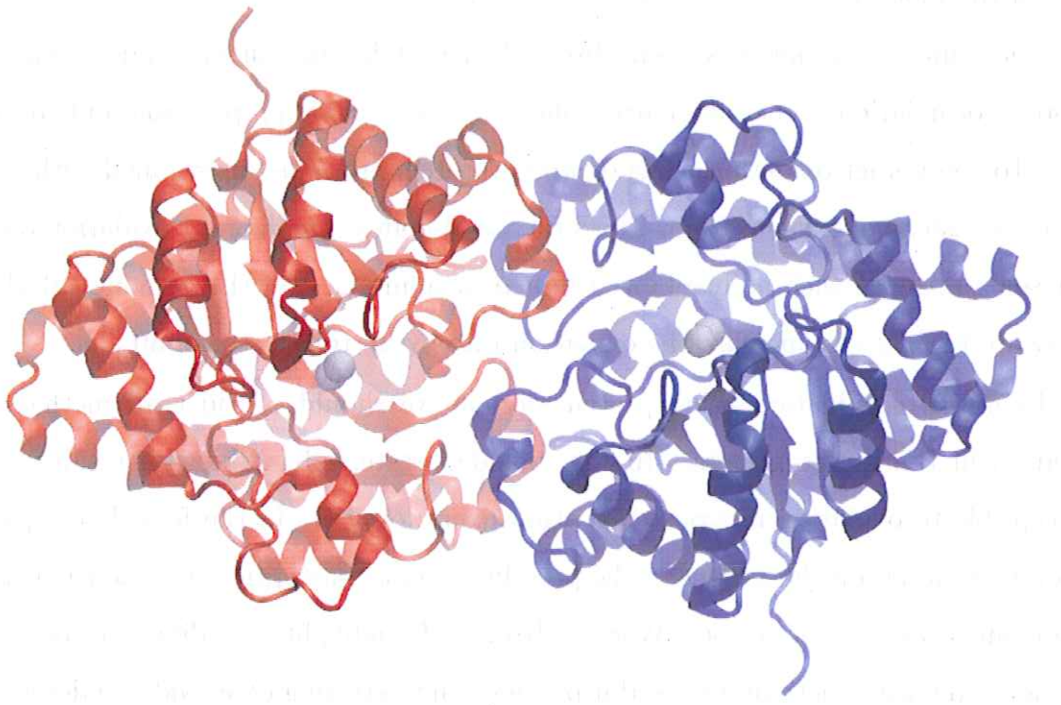


Figure 2-4: Crystal structure of the homodimeric enzyme phosphotriesterase with monomers colored red and blue and zinc ions colored grey (coordinates from PDB file 1HZY).

PTE is a homodimeric metalloprotein of known structure (see Figure 2-4) [36]. Each monomer has a binuclear catalytic center. Wild-type PTE contains two zinc ions per monomer, however PTE retains catalytic activity when the native zinc ions are replaced by cobalt, cadmium, mangenese, or nickel ions [37]. Figure 2-5 shows the active site of one monomer where five amino acids coordinate the two metal ions: Asp301, His55, His57, His201, and His230. A carbamylated lysine, Lys169 and a water molecule bridge the metal ions.

The catalytic mechanism of PTE proceeds via nucleophilic attack on the phosphorus center of the bound substrate [38]. The more solvent exposed zinc ion coordinated by His201 and His230 (rightmost ion in Figure 2-5) coordinates with the substrate phosphoryl oxygen to bind the organophosphate in the active site. This interaction weakens the binding of the bridging water to the zinc ion, polarizes the substrate P–O

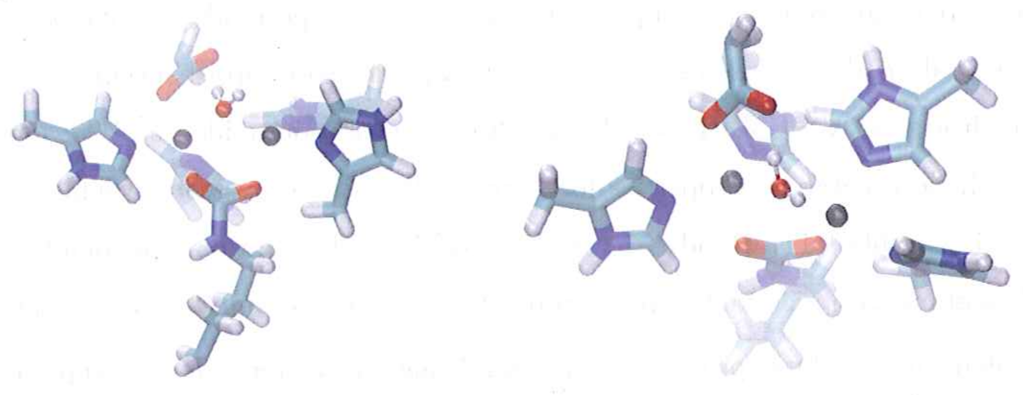

(a) Front view (b) Top view

Figure 2-5: Two views of the phosphotriesterase active site with zinc ions (colored grey) and bridging water molecule (coordinates from PDB file 1HZY).

bond, and makes the phosphorous center more electrophilic. Nucleophilic attack by the bridging water on the P–O bond is then assisted by a proton transfer to Asp301. Finally, a shuttle mechanism from Asp301 to His254 to Asp233 is thought to transfer the proton to bulk solvent [39].

The insecticide paraoxon is generally accepted as the preferred subtrate of PTE. Wild-type PTE hydrolyzes paraoxon with a turnover rate k_{cat} of $\approx 3000\ \text{s}^{-1}$ and a catalytic efficiency $k_{\text{cat}}/K_{\text{m}}$ of $5 \times 10^7\ \text{M}^{-1}\text{s}^{-1}$, which approaches the diffusion-controlled limit [37]. This efficiency is remarkable given paraoxon is a non-natural compound that has only existed since 1950 [40]. The catalytic efficiency of PTE for organophosphate nerve agents is significantly lower than for paraoxon with $k_{\text{cat}}/K_{\text{m}}$ of $\approx 1 \times 10^5\ \text{M}^{-1}\text{s}^{-1}$ for sarin (GB), $\approx 1 \times 10^4\ \text{M}^{-1}\text{s}^{-1}$ for soman (GD), and $\approx 1 \times 10^3\ \text{M}^{-1}\text{s}^{-1}$ for VX [41–43].

PTE is widely recognized for its use as an environmentally friendly bioremidation tool [44–48]. LeJeune *et al.* tested wild-type PTE in a blanket fire fighting foam for wide area chemical decontamination [49]. Yang *et al.* showed a solid lyophilized form of PTE can hydrolyze paraoxon vapors in a gas-phase bioreactor [50]. In the biosensor community, White *et al.* used PTE binding specificity for toxic organophosphates as the basis for an optical solid-state detector [51].

PTE has also been studied in the protein engineering community. Cho *et al.* used site-directed mutagenesis to improve hydrolysis of the pesticide methyl parathion [52]. Gopal *et al.* also used site-directed mutagenesis to improve specificity toward and hydrolysis of VX [53]. Hill *et al.* used directed evolution to identify PTE mutants with enhanced catalytic properties for sarin (GB) [54]. To overcome problems with expressing stable PTE, Roodveldt *et al.* used directed evolution and found a PTE triple-mutant with a 20-fold increase in functional expression in *Escherichia coli* [55].

Enhancing catalytic properties and stabilizing expression are key steps toward engineering PTE as a decontamination tool. Additionally, PTE must be stable under storage to be of practical use in environmental and military field settings. PTE degradation via aggregation, deamidation, or oxidation has not been reported in the literature. This is not surprising because experimental study of proteins takes place under conditions ideal for stability. It is difficult to predict PTE degradation rates for field settings where environmental factors such as temperature, pH, and atmospheric exposure are much harder to control.

Protein design provides an elegant approach to the storage stability problem. Suppose we use protein design to identify PTE mutations for all of the amino acids prone to degradation. If the mutations maintain or improve thermodynamic stability with respect to wild-type PTE without affecting catalysis, then we completely eliminate the enzyme's susceptibility to the aforementioned degradation mechanisms while maintaining its usefulness as a bioremidiation tool.

Table 2.1 lists the total number of cysteine, asparagine, glutamine, and methionine residues along with their sequence positions in each PTE monomer. We can heuristically evaluate which residues in each category are most prone to degradation. The two cysteines do not form disulfide bonds to stabilize the native three-dimensional structure, so they are of concern for aggregation. For asparagine deamidation, Asn265 is followed in sequence by alanine, which provides minimal steric protection from the reaction. For methionine oxidation, only Met293 lies on the protein surface; the remaining four residues are buried from solvent in the folded protein. In the following sections, we present computational methods and predictions towards redesigning PTE for enhanced storage stability. We chose a subset of these degradation-prone residues

Table 2.1: Amino acids in PTE prone to degradation and the associated mechanisms.

Degradation Mechanism	Total # Residues	Sequence Position
Cys Aggregation	2	59, 227
Asn Deamidation	6	38, 265, 300, 312, 321, 353
Gln Demidation	10	148, 155, 180, 206, 211,212, 278, 290, 295, 343
Met Oxidation	5	138, 293, 314, 317, 325

for initial investigation, and we used structure-based protein design to identify mutations predicted to maintain or improve the thermodynamic stability compared to wild-type PTE.

2.4 Methods

Our computational methods closely follow those developed by Lippow *et al.* with the exception of structure preparation steps specific to phosphotriesterase. For further details, we refer the reader to supplementary methods in recently published work by Lippow *et al.* [1].

2.4.1 Structure preparation

Multiple crystal structures of phosphotriesterase (PTE) from *Pseudomonas diminuta* exist in the Protein Data Bank. These structures include zinc-containing PTE with bound substrate analogs diethyl 4-methylbenzylphosphonate (PDB:1DPM), triethylphosphate (PDB:1EYW), and diisopropylmethyl phosphate (PDB:1EZ2) [56, 57]. The set also includes one high-resolution (1.3 Å) zinc-containing PTE structure (PDB:1HZY) and three high-resolution metal-substituted structures with no substrate analog: manganese-containing PTE (PDB:1IOB), cadmium-containing PTE (PDB:1JGM), and zinc/cadmium-containing PTE (PDB:1IOD) [58]. We chose the high-resolution zinc-containing PTE structure (PDB:1HZY) for our designs because it best represents naturally-occurring (i.e., zinc-binding) PTE under storage conditions, in which no substrates are typically present.

To prepare the 1HZY structure for calculations, we removed sodium ions and all 2-phenylethanol, 1,2-ethanediol, and ethylene glycol molecules remaining from the crystallization process. We assigned hydrogen-atom positions using the HBUILD facility in the program CHARMM [59, 60] with the PARAM27 all-atom topology and parameter sets for proteins [61, 62]. The 1HZY structure contains a carbamylated lysine (Kcx) at position 169 that bridges the two zinc ions in the active site of each monomer. We determined the partial atomic charges for Kcx169 with a geometry optimization and RESP-fitting procedure using the program Gaussian [63].

The 1HZY structure contains 764 water molecules. To eliminate non-bound waters, we removed all water molecules with fewer than 3 contacts with polar atoms within 3.3 Å (excluding contacts with other water molecules). We also removed any water molecule with a B-factor greater than 30. This elimination procedure reduced the total number of water molecules to 65, which we visually inspected to ensure they were structurally significant.

We resolved crystallographic nitrogen/oxygen uncertainties for asparagine and glutamine and we selected protonation states for histidine to optimize hydrogen bonding in the local environment. We visually inspected the neighborhood of all asparagine and glutamine residues and did not find better hydrogen bonding by exchanging the nitrogen and oxygen compared to the 1HZY coordinates (i.e., no Asn or Gln residues were "flipped"). For protonation of the seven histidine residues in the 1HZY structure, we used Hsd55, Hsd57, Hse123, Hse201, Hsd230, Hsp254, and Hse257 where Hsd, Hse, and Hsp refer to protonation of N_δ, protonation of N_ϵ, and doubly protonated histidine, respectively. We chose protonation states for Hsd55, Hsd57, Hse201, and Hsd230 based on their known roles coordinating the binuclear metal ions in the active site. We chose double protonation for Hsp254 based on its proposed coordination with Asp301 in shuttling excess protons to bulk solvent during hydrolysis. For Hse123, protonation of N_ϵ allowed hydrogen bonding with the oxygen of Glu148. For Hse257, protonation of N_ϵ allowed hydrogen bonding with oxygen of Asn300. For both Hse123 and Hse257, there were no potential hydrogen bonds within 5 Å for protonation of N_ϵ.

2.4.2 Conformational space

Our discrete rotamer library was based on the backbone-independent library by Dunbrack and Cohen (May 2002) expanded by $\pm 10°$ in the χ_1 and χ_2 angles [64]. Before expansion, three histidine rotamers were added for an unsampled ring flip, and two aspargine rotamers were added to increase sampling of the final dihedral angle rotation. Serine, threonine, and tyrosine hydroxyls were sampled every 10°. Our modified library contains a total of 4,025 side-chain rotamers. For crystallographic waters, we used a novel water library containing 61 rotamers. The first 60 rotamers correspond to fixing the oxygen atom based on its crystal structure coordinates and placing hydrogen atoms to create symmetric water molecule rotations. The final rotamer corresponds to removal of the particular water molecule from the structure [1].

For the design of single mutations, we chose one asparagine, cysteine, or methionine position and allowed it mutate to 18 other amino acids (excluding proline). We allowed rotamer flexibility for the local neighborhood of side chains around the mutant position. To redesign methionine residues, which have a long side chain, we allowed flexibility for any position within 5 Å of at least one side-chain atom of the target methionine. To redesign aspargine and cysteine residues, which have shorter side chains compared to methionine, we used a distance of 4.7 Å to select flexible positions. We added the wild-type side chain to the rotamer library for all mutant and flexible positions using the complete Cartesian coordinates of the side chains from the crystal structure. For higher-order mutations (double, triple, etc.), we took the union of the flexible neighborhoods around each mutant position.

2.4.3 Energy function and model

We determined folding stability for conformations using a hierarchical two-stage energy model. For both stages, we assumed a fixed backbone given by the 1HZY crystal structure. We used a "low-resolution," pairwise-additive energy function to calculate terms for all possible conformations of each sequence. We then identified the lowest-energy conformations of each sequence using conformational search. Finally, we reevaluated the lowest-energy conformations using a more accurate but more computationally expensive "high-resolution" energy function. This two-stage approach allowed us to trade accuracy for speed early in the design procedure, and it limited expensive computations to only the most promising mutations selected by conformational search.

For low-resolution energy evaluation, we used the CHARMM PARAM27 molecular mechanics force field with a 4r distance-dependent dielectric constant and with no cut-offs for non-bonded interactions. We included all available energy terms (i.e. bond, angle, Urey–Bradley, dihedral, improper, Lennard-Jones, and electrostatic). Our objective function for each conformation was the energy difference between the folded and unfolded state assuming a fixed backbone. For the unfolded state, we treated each side chain as being infinitely separated. The total energy of the unfolded state was the sum of energies from these isolated side chain compounds.

For high-resolution energy evaluation, we used a Poisson–Boltzmann continuum electrostatics model with implicit solvation to replace the distance-dependent electrostatics term from low-resolution evaluation. We used the PARSE parameters for partial atomic charges and radii [65]. We used a dielectric constant of 4 for the protein and for explicit water, and a dielectric constant of 80 for implicit solvent regions. We solved the linearized Poisson–Boltzmann equation using a locally modified version of the the DELPHI program [66]. We included a solvent-accessible surface area (SASA) term, calculated using the analytical surface routine in CHARMM, to capture the nonpolar component of the solvation free energy.

We also used a second model for the nonpolar component of the solvation free energy. This model separately considered cavity formation and solute–solvent van der Waals interactions. The cavitation term was linearly proportional to the solvent-excluding surface area, calculated using the program MSMS [67]. For continuum van der Waals interaction, we used the energy framework of Levy *et al.* with modifications for the CHARMM TIP3P water model and PARAM27 all-atom parameters [68]. We used a C++ program developed by Lippow *et al.* to solve the integration using atom-centered spherical volume elements [1].

2.4.4 Conformational search

During low-resolution energy evaluation, we eliminated all side-chain rotamers that clashed either with the rigid backbone atoms or with the side chains of positions not given conformational flexibility. Clashes were determined by examining the singleton energy of every possible rotamer. If a rotamer's singleton energy exceeded the lowest singleton energy at that sequence position by at least 50 kcal/mol, then we removed it from the search space.

We used dead-end elimination followed by A* search (i.e., DEE/A* [7–10]) on the low-resolution energies to find the global minimum energy conformation (GMEC) for each sequence. We kept only sequences whose GMEC energy was within 1000 kcal/mol of the lowest-energy sequence, excluding any sequence whose GMEC energy was greater than 100 kcal/mol. We used a 100 kcal/mol upper bound because a positive value for the objective function (i.e., the energy difference between the folded and unfolded state) means external energy is required to induce protein folding. Small positive values are allowed to account for any inaccuracies in our low-resolution energy model, but 100 kcal/mol is a high enough external energy requirement to rule out the sequence. To illustrate these thresholds, if the lowest-energy sequence in our design had a GMEC energy of -1000 kcal/mol, then we kept all sequences whose GMEC energies were less than 0 kcal/mol. On the other hand, if the lowest-energy sequence had a GMEC energy of -100 kcal/mol, then we only kept sequences whose GMEC energies were less than the 100 kcal/mol upper bound.

For each of the selected sequences, we kept up to 10 kcal/mol of lowest-energy conformations from the ordered-list returned by A*. Because of the additional 10° torsional sampling in our rotamer library, some of these conformations were not necessarily qualitatively different. We grouped conformations that only differed by a 10° dihedral angle rotation of one rotamer. If a particular sequence had fewer than 50 qualitatively different low-energy conformations within 10 kcal/mol of the sequence

GMEC energy, then all of its conformations were selected for high-resolution energy evaluation. Otherwise, we selected the 50 lowest-energy conformations for high-resolution evaluation.

2.4.5 Design of mutations

For our initial PTE designs, we chose the following subset of five degradation-prone amino acids from Table 2.1: Cys59, Cys227, Asn38, Asn265, and Met314. Both Cys59 and Cys227 are unpaired or "free", and they are the only cysteine residues in PTE. Both Asn38 and Asn265 lie on the protein surface. Asn38 is of particular concern because it is followed in sequence by alanine, which provides minimal steric hindrance to the deamidation reaction. Finally, Met314 lies in a solvent-exposed pocket near the protein surface, making it susceptible to oxidation by atmospheric oxygen.

For each of the five target amino acids, we performed independent single mutation designs. In each design, we considered the lowest-energy conformation for each mutant sequence after high-resolution evaluation. We filtered mutant sequences based on the predicted change in electrostatics from our high-resolution continuum electrostatics model. We eliminated any mutant sequence predicted to decrease electrostatic energy by greater than 2.0 kcal/mol compared to wild type. Electrostatic penalties greater than 2.0 kcal/mol indicate poor interaction with the local neighborhood as opposed to a deficiency in the modeling the side chain orientations (i.e., the resolution of dihederal angles in our discrete rotamer library).

For the remaining sequences, we calculated the folding stability relative to wild type based on our two models for the nonpolar component of solvation free energy. Both calculations included additive van der Waals and geometric (i.e., energy from bonds, angles, dihedrals, and impropers) terms from low-resolution energy evaluation along with the continuum electrostatic term from high-resolution evaluation. One calculation, $\Delta\Delta G_{\text{total}}^{\text{SASA}}$, included the solvent-accessible surface area (SASA) term from high-resolution evaluation. The other calculation, $\Delta\Delta G_{\text{total}}^{\text{CVDW}}$, included the cavitation and continuum van der Waals correction terms from high-resolution evaluation.

To select promising mutations, we tried to balance electrostatic and packing contributions to folding stability. Ideally, a mutation should have improved electrostatic and packing interactions compared to wild type; however, this rarely occurs in practice. We allowed for a tradeoff between small electrostatic penalities of less than 1.0 kcal/mol and improved packing, and vice versa. If a mutation decreased folding stability by less than 5.0 kcal/mol but significantly improved electrostatics by greater than 2.0 kcal/mol, then we attempted to repack the local environment by allowing mutation of neighboring residues. Our goal in this case was to maintain favorable electrostatics compared to wild type while reintroducing stability.

As previously mentioned, phosphotriesterase is a homodimer (i.e., a protein comprised of two monomers identical in sequence). The monomers are labeled chain A and chain B in the 1HZY crystal structure. When solving for the crystal structure, Benning *et al.* noted better ordering in the electron density map corresponding to chain B [58]. For each of the target residues, we performed separate, independent designs on each chain and compared the results. Energies and predictions presented in the following section correspond to designs on chain B. Although the energies differed slightly between the two chains, our recommended mutations would not change if instead we had presented predictions from designs on chain A.

2.5 Results and Discussion

Here we present results from our designs on Cys59, Cys227, Asn38, Asn265, and Met314 in phosphotriesterase. For all candidate mutations, we provide tables that list predicted packing stability relative to wild type for both solvation models, $\Delta\Delta G^{\mathrm{SASA}}_{\mathrm{total}}$ and $\Delta\Delta G^{\mathrm{CVDW}}_{\mathrm{total}}$, and the electrostatic contribution, $\Delta\Delta G_{\mathrm{elec}}$, from high-resolution energy evaluation[1]. In all tables, these three terms are measured in kcal/mol. Negative values correspond to favorable electrostatics or packing stability of the mutant relative to wild type. Positive values correspond to unfavorable electrostatics or packing stability of the mutant relative to wild type. We highlight promising single muta-

[1]The full set of energy terms from low-resolution and high-resolution evaluation can be found in Appendix A.

Table 2.2: Candidate mutations for Cys59.

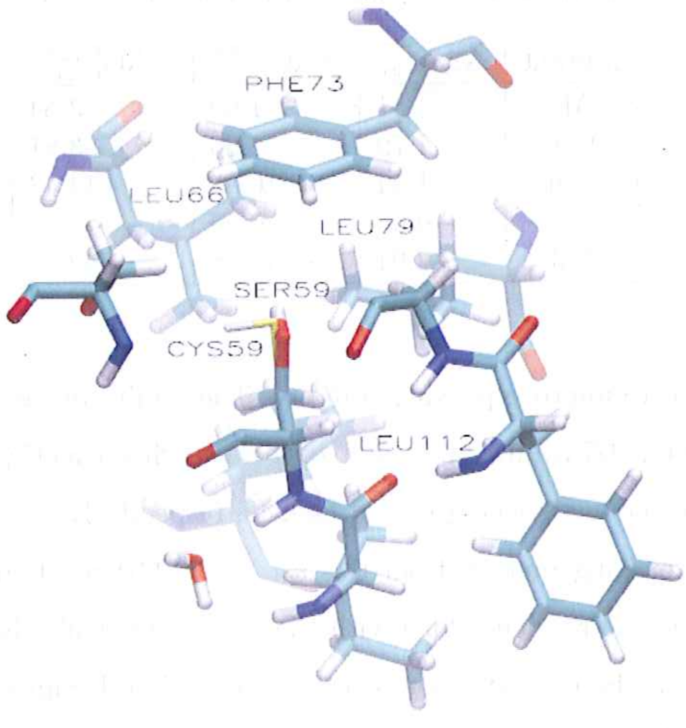
Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Ala	+0.25	+2.62	−0.88
Asn	+1.52	+2.48	+2.15
Gly	+0.40	+4.93	+8.61
Ser	+0.83	+4.46	+5.70

tions to eliminate Cys59, Cys227, Asn38, and Met314 as well as double and triple mutations to eliminate Asn265.

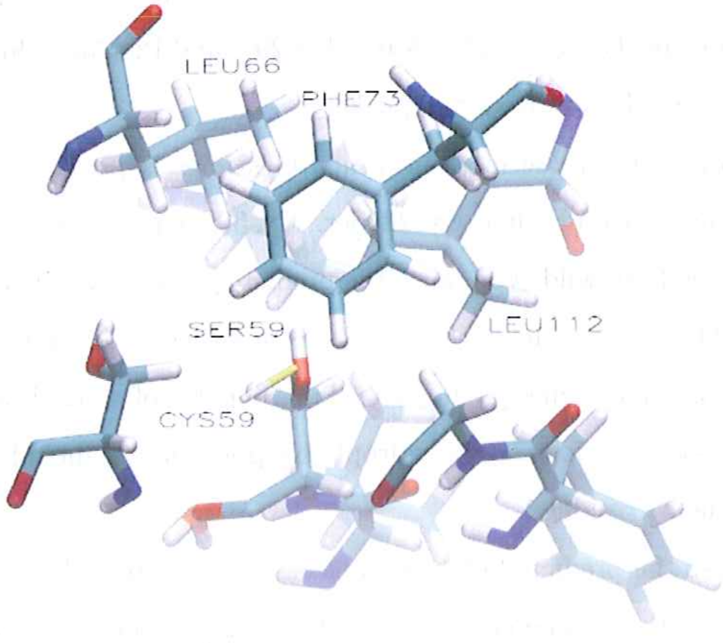
Table 2.2 lists the predicted change in electrostatic and packing stability for four Cys59 mutations. Other mutations were eliminated during search and filtering mainly due to clashes with Leu79 and Phe73 and unfavorable electrostatic interaction with Ser61. Of the four possible mutations, Cys59Ala and Cys59Gly pay the lowest electrostatic penalties compared to wild type. However, substitution of cysteine for a smaller alanine or glycine residue leaves a gap in the local neighborhood; both mutations pay a van der Waals penalty. Cys59Asn pays an electrostatic penalty of ≈ 1.5 kcal/mol. Additionally, a mutation to asparagine introduces a new residue prone to deamidation. This particular mutation is problematic because it is followed in sequence by Gly60, which provides minimal steric protection from deamidation.

The most promising mutant from this set is Cys59Ser. Cysteine and serine are structurally similar with the difference being cysteine contains a thiol group while serine contains a hydroxyl group. Figure 2-6 shows the mutant and wild-type residues along with their flexible neighborhoods. For both cysteine and the serine mutant, the flexible neighborhoods adopt the same conformation. The only difference between conformations is that the –OH group of serine points toward the local hydrophobic pocket formed by Leu66, Leu79, and Leu112, whereas the –SH group of cysteine is oriented away from the pocket.

Table 2.3 lists the predicted change in electrostatic and packing stability for five Cys227 mutations. As seen with Cys59, the electrostatically favorable alanine and glycine mutations leave a void in the local neighborhood. However, the alanine mu-



(a) Front view



(b) Top view

Figure 2-6: Two perspectives of the Cys59Ser mutation.

Table 2.3: Candidate mutations for Cys227.

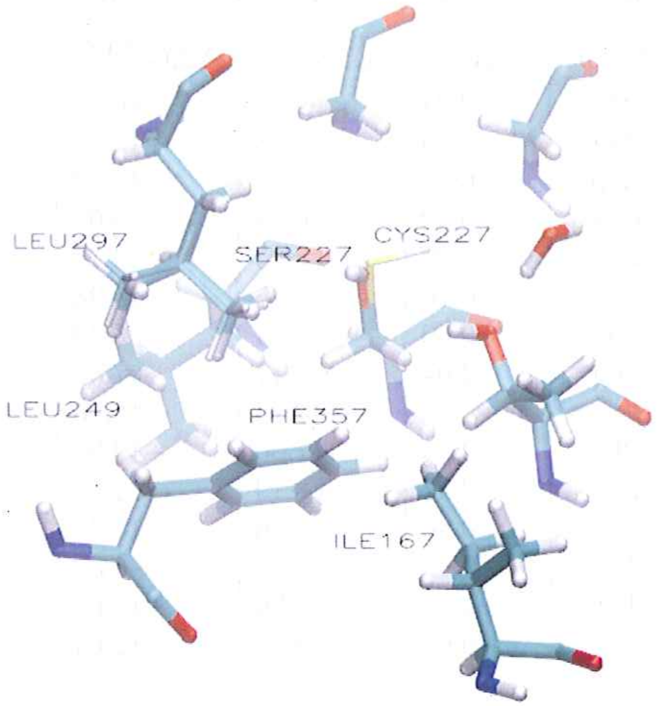
Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Ala	−1.51	−1.00	−3.54
Gly	−1.79	+2.06	+3.81
Leu	+1.34	+16.84	+14.43
Ser	−0.65	+0.96	+1.70
Val	−2.04	+43.25	+42.72

tation is prediced to improve packing stability. The significant destabilizing effect of Cys227Leu and Cys227Val are the result of a clash with a methyl group of Leu297 and a clash with the methylene group of Leu249, respectively.

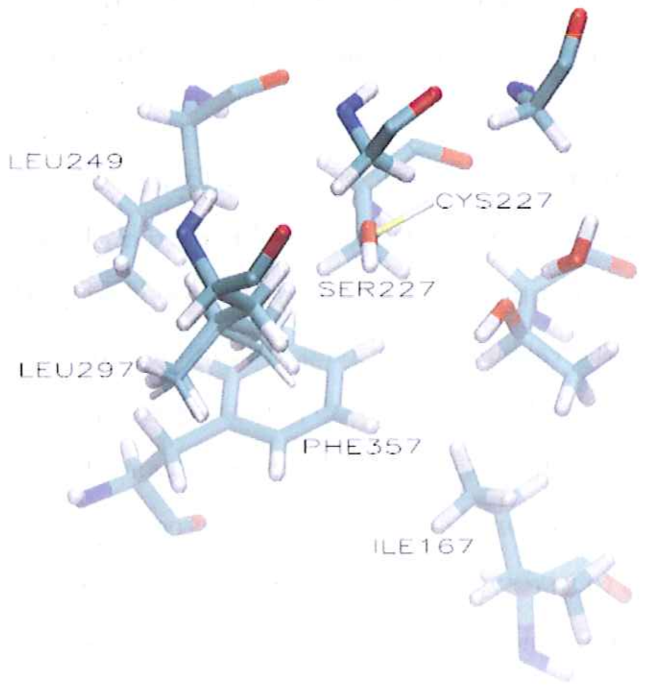
The most promising mutant from this set is Cys227Ser. Figure 2-7 shows the mutant and wild-type residues along with their flexible neighborhoods. Both neighborhoods adopt similar conformations except Leu297 slightly shifts for the Cys227Ser mutant, presumably to make a weak electrostatic interaction between its methyl group and serine's oxygen. As with Cys59Ser, the –OH group of serine points towards a local hydrophobic pocket formed by Ile167, Leu297, and Phe357, while the –SH group of cysteine is oriented away from the pocket.

Of the two cysteine to serine mutations, Cys59Ser is predicted to have a more destabilizing effect on PTE than Cys227Ser. Cys59Ser makes less favorable electrostatic interactions than wild type, and it is predicted to decrease packing stability by ≈ 5 kcal/mol. On the other hand, Cys227Ser is predicted to improve electrostatics by more than 0.5 kcal/mol while paying a packing penalty of ≈ 1.3 kcal/mol compared to wild type. Both mutations are in hydrophobic pockets that include a phenylalanine and multiple leucine residues.

Table 2.4 lists the predicted change in electrostatic and packing stability for possible Met314 mutations. The electrostatically favorable mutations to Leu, Ser, Thr, and Val all pay significant van der Waals penalties; these residues are too small to fill the void left by removing the larger methionine residue. A methyl group of nearby Leu303 (~5 Å from C_β of position 314) blocks Lys, Phe, Trp, and Tyr mutants from filling the void. Instead, the side chains of these mutant residues point out into



(a) Front view



(b) Top view

Figure 2-7: Two perspectives of the Cys227Ser mutation.

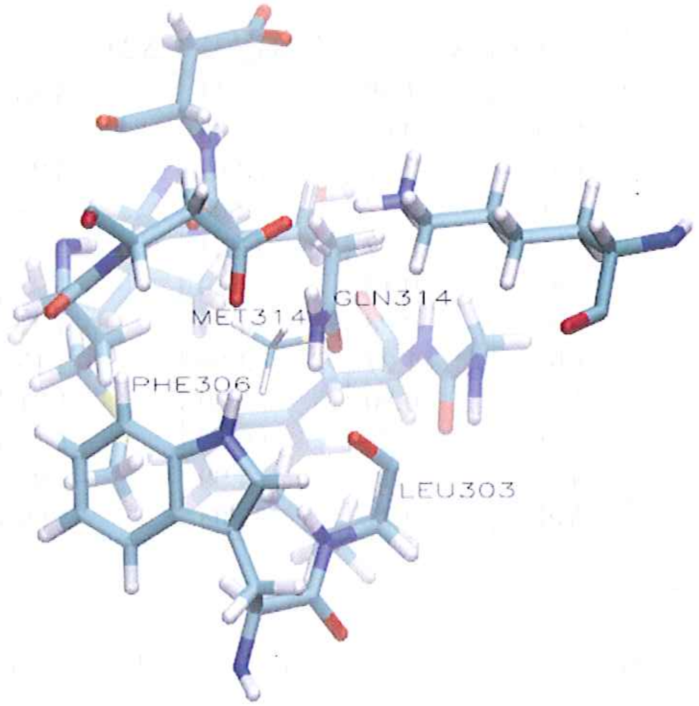
Table 2.4: Candidate mutations for Met314.

Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Ala	-2.84	$+5.43$	$+6.83$
Arg	$+0.33$	$+7.83$	$+12.42$
Asp	-6.37	$+7.54$	$+14.03$
Gln	$+0.29$	$+0.63$	$+0.61$
Gly	-4.22	$+8.33$	$+13.77$
Hsd	$+0.76$	$+5.25$	$+10.28$
Ile	$+0.60$	$+6.14$	$+6.42$
Leu	-3.04	$+5.17$	$+7.82$
Lys	$+1.29$	$+10.93$	$+15.56$
Phe	$+0.85$	$+7.50$	$+13.97$
Ser	-2.27	$+8.19$	$+11.25$
Thr	-0.40	$+8.46$	$+11.41$
Trp	$+0.11$	$+0.82$	$+6.52$
Tyr	$+1.31$	$+6.96$	$+13.67$
Val	-0.04	$+10.57$	$+12.64$

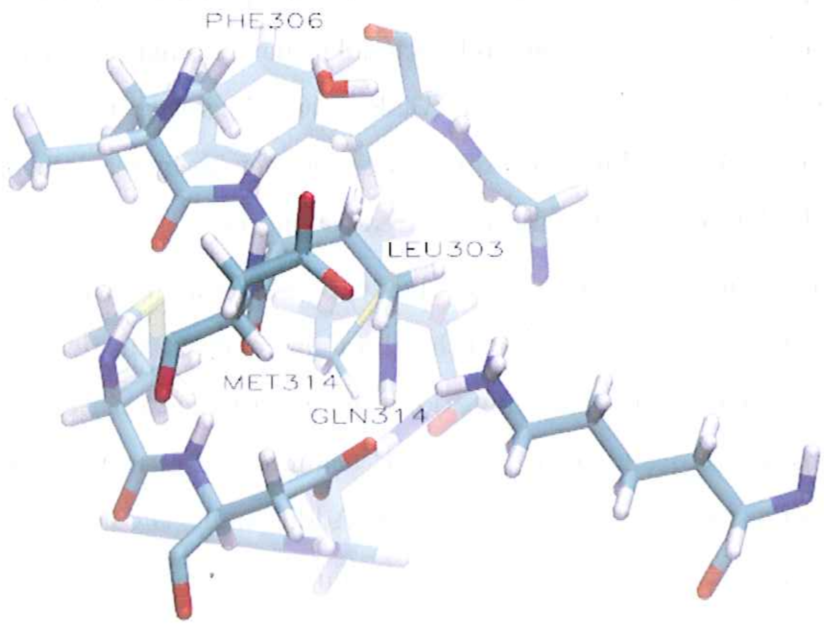
solvent. Similarily, the negatively charged side chain of Met314Asp points out into solvent to make a favorable electrostatic interaction, but it too pays a large van der Waals penalty.

The most promising mutant from this set is Met314Gln. Figure 2-8 shows the mutant and wild-type residues along with their flexible neighborhoods. For both methionine and the glutamine mutant, the flexible neighborhoods adopt the same conformation. For Met314Gln, the carbonyl functional group points toward the methylene group of Phe306, and its amide functional group points toward the negatively charged Asp318. Overall, the electrostatics of Met314Gln are less favorable by ≈ 0.3 kcal/mol, and both high-resolution solvation models predict a decrease in packing stability of ≈ 0.6 kcal/mol compared to wild type. Mutation from methionine to glutamine makes position 314 prone to deamidation. However, Met314Gln is followed in sequence by the negatively charged Asp315, and deamidation rates are extremely slow, on the order of 10^4 days, for GlnAsp peptides [69].

Table 2.5 lists the predicted change in electrostatic and packing stability for possible Asn38 mutations. Of the electrostatically favorable mutations, Ala, Gly, Ser, and



(a) Front view



(b) Top view

Figure 2-8: Two perspectives of the Met314Gln mutation.

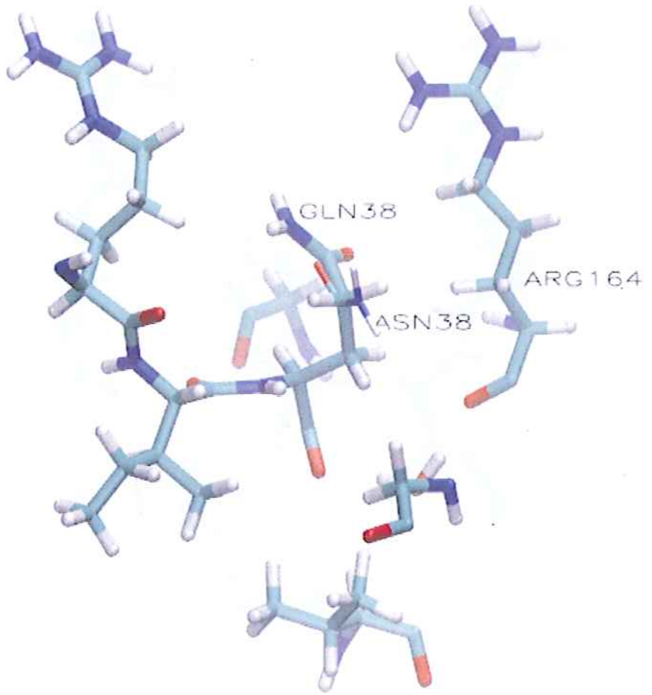
Table 2.5: Candidate mutations for Asn38.

Mutant	$\Delta\Delta G_{elec}$	$\Delta\Delta G^{SASA}_{total}$	$\Delta\Delta G^{CVDW}_{total}$
Ala	-1.80	-0.11	-3.86
Gln	$+0.65$	-2.69	-2.48
Gly	-2.27	$+3.62$	$+3.29$
Hse	$+0.72$	-2.06	-1.88
Ile	$+0.22$	$+11.61$	$+10.70$
Leu	-0.13	$+0.19$	-1.28
Met	-0.08	-4.79	-4.21
Phe	$+1.27$	-3.99	-4.15
Ser	-0.80	$+1.30$	$+1.91$
Thr	$+0.42$	$+0.87$	$+1.15$
Tyr	$+1.17$	-4.99	-4.90
Val	-0.77	$+5.08$	$+4.66$

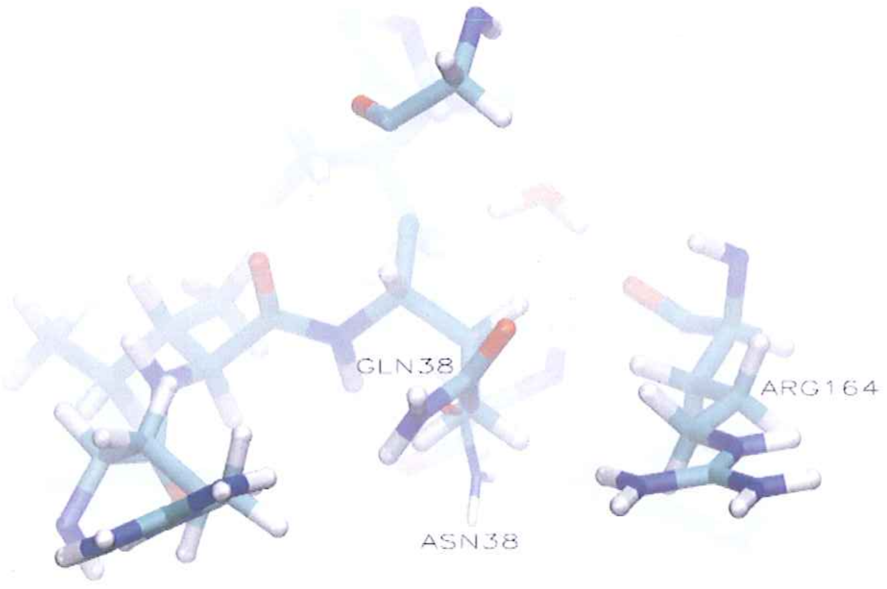
Val are too small to fill the void left by removing the larger aspargine residue. The larger Phe and Tyr mutations make poor interactions with solvent; each pays an electrostatic penalty of greater than 1 kcal/mol. One promising mutant from this set is Asn38Gln. Although mutations to Hse and Thr have similar electrostatics, Asn38Gln is predicted to make better van der Waals interactions with the local neighborhood. In fact, both high-resolution energy models predict an increase in packing stability by $\approx$ 2.5 kcal/mol for Asn38Gln compared to wild type.

Figure 2-9 shows the mutant and wild-type residues along with their flexible neighborhoods. For both asparagine and the glutamine mutant, the flexible neighborhoods adopt the same conformation. Mutation from asparagine to glutamine still leaves position 38 prone to deamidation. However, deamidation rates are significantly slower for glutamine as discussed in Section 2.2.2. Additionally, the first methylene group of Arg164 is likely to sterically hinder formation of the six-membered intermediate for the longer glutamine residue (~4 Å between C_β of positions 38 and 164).

The second promising mutant is Asn38Leu. Figure 2-10 shows the mutant and wild-type residues along with their flexible neighborhoods. For both asparagine and the leucine mutant, the flexible neighborhoods adopt the same conformation. Substitution of the hydrophilic asparagine side chain for the hydrophobic leucine is predicted

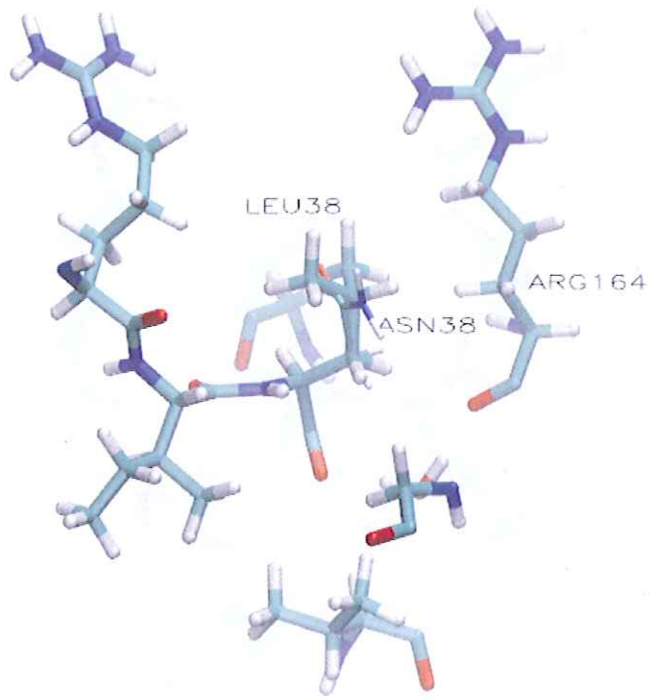


(a) Front view

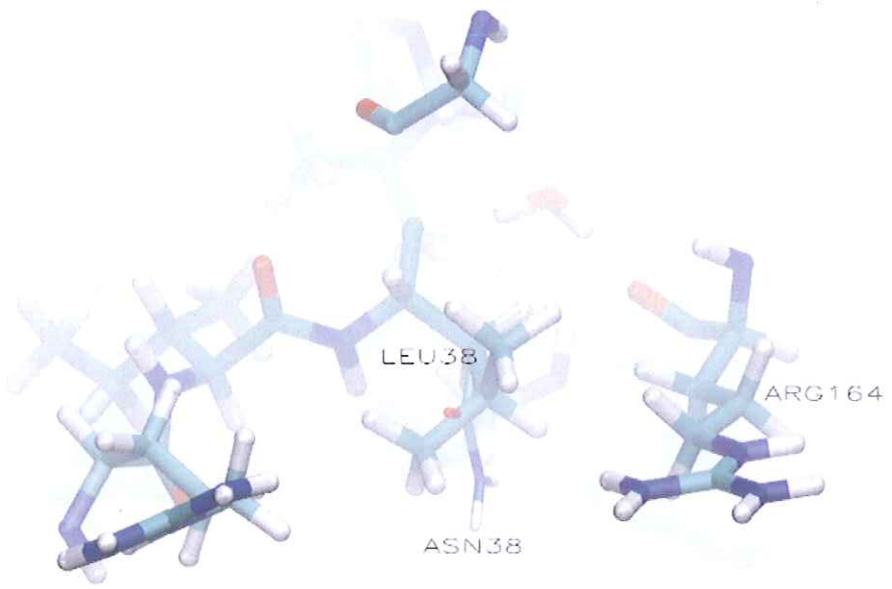


(b) Top view

Figure 2-9: Two perspectives of the Asn38Gln mutation.



(a) Front view



(b) Top view

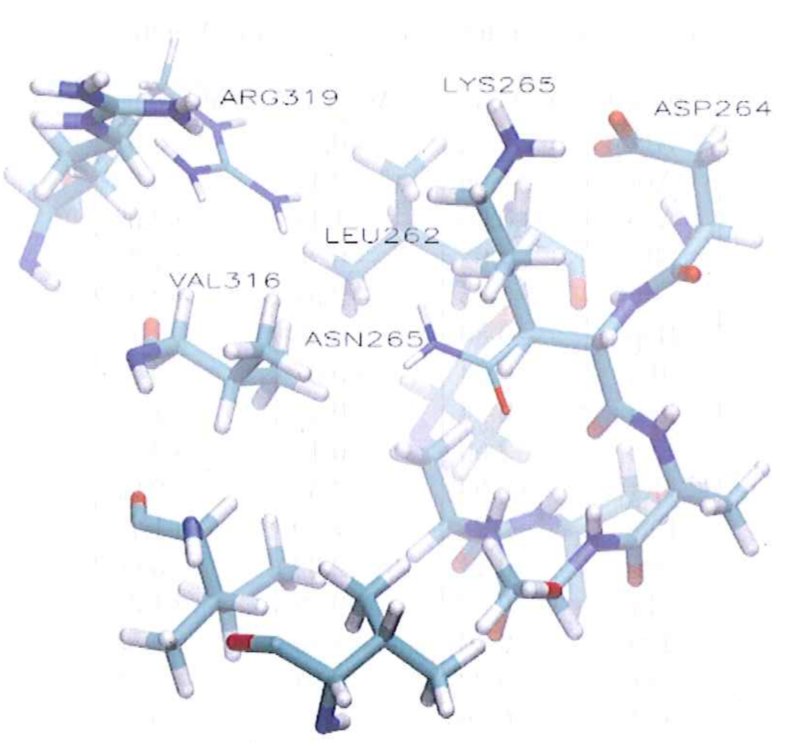
Figure 2-10: Two perspectives of the Asn38Leu mutation.

Table 2.6: Candidate mutations for Asn265.

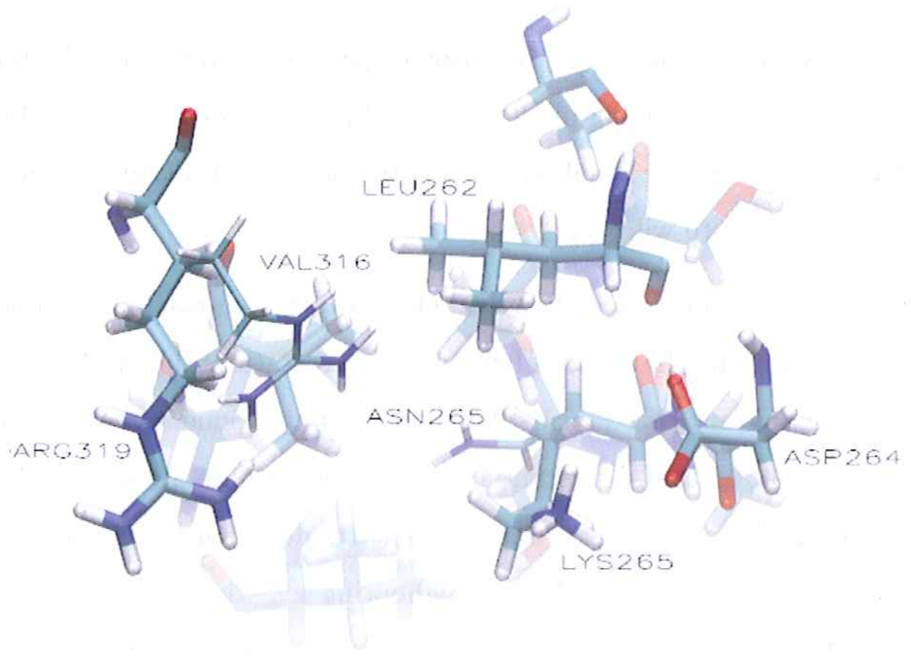
Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Ala	−1.02	+0.36	−2.57
Arg	+1.19	−2.17	−0.02
Asp	+0.09	+1.18	+2.09
Gln	+1.92	−0.70	−1.20
Gly	−1.35	+2.58	+3.37
Hsp	−0.80	−0.30	+3.12
Ile	+0.74	+0.07	+0.14
Leu	−0.14	−0.79	−1.20
Lys	−2.60	−0.11	+3.42
Met	+0.53	−2.39	−2.67
Phe	−0.25	−1.79	+1.19
Ser	+0.14	+2.39	+3.06
Thr	+0.88	+0.98	+1.35
Trp	+0.70	−2.40	+2.55
Tyr	−0.22	−1.91	+1.87
Val	+0.49	+0.53	+0.29

to slightly improve electrostatics. In terms of packing stability, our high-resolution continuum electrostatics model predicts a ≈ 0.2 kcal/mol decrease in stability, while our high-resolution van der Waals correction predicts a ≈ 1.3 kcal/mol improvement in stability compared to wild type.

Table 2.6 lists the predicted change in electrostatic and packing stability for possible Asn265 mutations. In terms of favorable electrostatic stability, Asn265Lys is the most promising mutant based on its 2.6 kcal/mol improvement over wild type. Figure 2-11 shows the Asn265Lys mutant and wild-type residues along with their flexible neighborhoods. The electrostatic improvement results from interaction of the positively charged lysine mutant with the neighboring negatively charged Asp264. For Asn265Lys, the two high-resolution energy models predict different changes in packing stability compared to wild type. The high-resolution calculation that includes a solvent-accessible surface area (SASA) term predicts a slight, ≈ 0.1 kcal/mol, improvement in packing stability. On the other hand, the calculation that includes cavitation and continuum van der Waals correction terms predicts a ≈ 3.5 kcal/mol



(a) Front view



(b) Top view

Figure 2-11: Two perspectives of the Asn265Lys mutation.

Table 2.7: Candidate mutations for Leu262 in the background of Asn265Lys.

Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Ala	−1.49	+1.02	+1.55
Arg	+1.14	−0.94	+3.33
Gln	+1.04	+1.20	+2.66
Glu	+0.89	+2.76	+2.69
Gly	−1.18	+2.21	+7.74
Hsd	+0.51	−1.68	−0.31
Ile	−1.33	+27.30	+30.05
Lys	−1.04	+1.52	+6.92
Phe	−2.00	−1.92	+2.71
Ser	−0.70	+3.92	+7.79
Thr	+0.07	+2.74	+6.16
Trp	−0.64	−4.02	−0.23
Tyr	+0.60	−3.01	−1.71
Val	−1.21	+37.29	+40.25

decrease in packing stability. In figure 2-11, we see that Arg319 in the flexible neighborhood changes orientation to reach out into solvent, thereby improving SASA for the Asn265Lys mutant structure. We also see the mutation from asparagine to lysine at position 265 leaves a gap in the vicinity of Leu262 and Val316. To maintain favorable electrostatics and improve packing stability, we independently considered mutations to Leu262 and Val316 in the background of Asn265Lys to fill the aforementioned gap.

Table 2.7 lists the predicted change in electrostatic and packing stability for possible Leu262 mutations in the background of Asn265Lys. Of the electrostatically favorable mutations, Gly, Ile, Lys, and Val all decrease folding stability relative to wild type by more than 5 kcal/mol. The four most promising mutants from this set are Hsd, Phe, Trp, and Tyr, which are all larger residues than leucine. Figure 2-12 shows these mutants along with Asn265Lys. Hsd and Tyr both have improved packing stability relative to wild type, but their electrostatic interactions are less favorable than wild type by ≈ 0.5 kcal/mol. The Phe mutant maintains significantly favorable electrostatics; however, the high-resolution energy calculation that includes cavitation and van der Waals correction terms predicts a decrease in packing stability relative

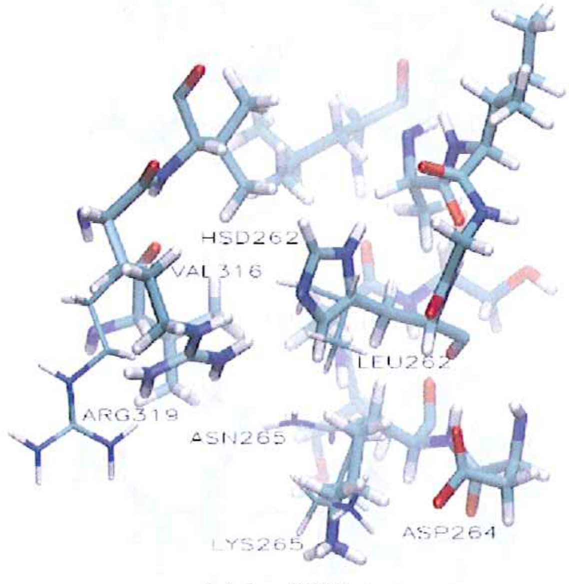
Table 2.8: Candidate mutations for Val316 in the background of Asn265Lys.

Mutant	$\Delta\Delta G_{elec}$	$\Delta\Delta G_{total}^{SASA}$	$\Delta\Delta G_{total}^{CVDW}$
Ala	-1.01	$+1.98$	-0.08
Asp	$+0.43$	$+2.15$	$+3.50$
Glu	$+1.81$	$+0.53$	$+0.83$
Gly	-1.20	$+6.71$	$+8.57$
Hse	$+0.18$	$+0.24$	$+2.48$
Ile	$+0.37$	-0.78	-0.78
Leu	-1.89	-0.48	$+1.76$
Phe	-1.02	-0.46	$+3.27$
Ser	$+0.12$	$+2.83$	$+4.64$
Thr	$+0.46$	$+1.20$	$+2.31$
Trp	$+0.91$	-3.52	-0.28
Tyr	-0.84	-1.40	$+2.55$

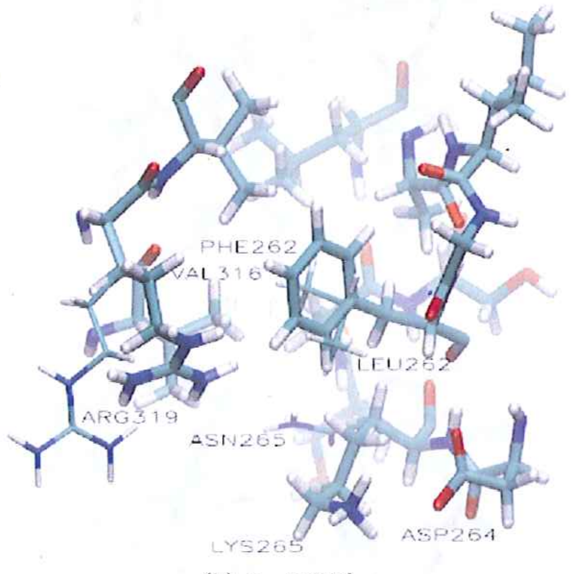
to wild type. Lastly, the Trp mutant is predicted to improve both electrostatic and packing stability relative to wild type.

Table 2.8 lists the predicted change in electrostatic and thermodynamic stability for possible Val316 mutations in the background of Asn265Lys. The five most promising mutants from this set are Ile, Leu, Phe, Trp, and Tyr. Figure 2-13 shows these mutants along with Asn265Lys. Leu, Phe, and Tyr all maintain improved electrostatics relative to wild type. For these three mutants, our high-resolution energy calculations that include a solvent-accessible surface area (SASA) term predict improved packing stability relative to wild type; however, the high-resolution calculations that include cavitation and continuum van der Waals correction terms predict a decrease in packing stability. On the other hand, for the Ile and Trp mutants, both high-resolution calculations predict improved packing stability at the cost of less favorable electrostatics compared to wild type.

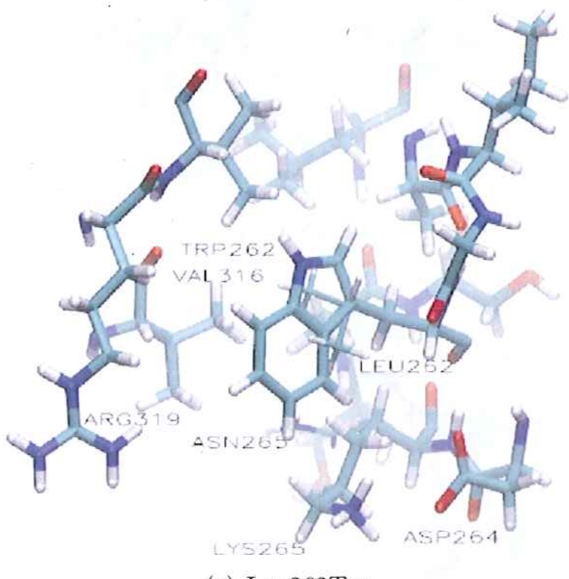
Finally, we considered a triple mutation comprised of all pairwise combinations of the set Leu262(Hsd, Phe, Trp, Tyr) with the set Val316(Ile, Leu, Phe, Trp, Tyr) in the background of Asn265Lys. Table 2.9 lists the predicted change in electrostatic and thermodynamic stability for the twenty combinations of Leu262 and Val316 mutations in the background of Asn265Lys. None of the triple mutant structures have improved



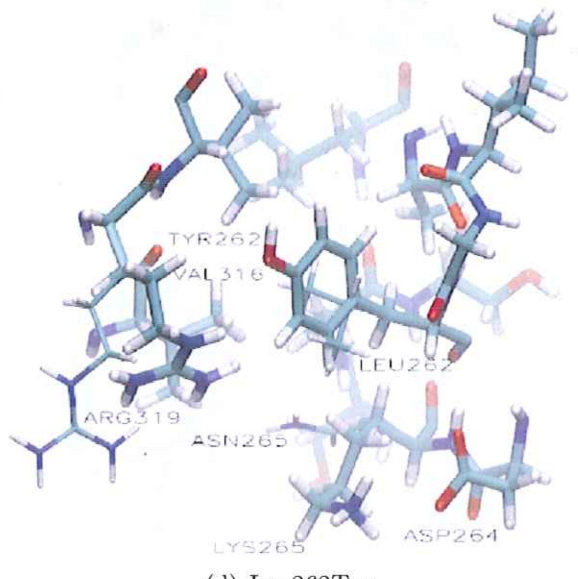
(a) Leu262Hsd



(b) Leu262Phe

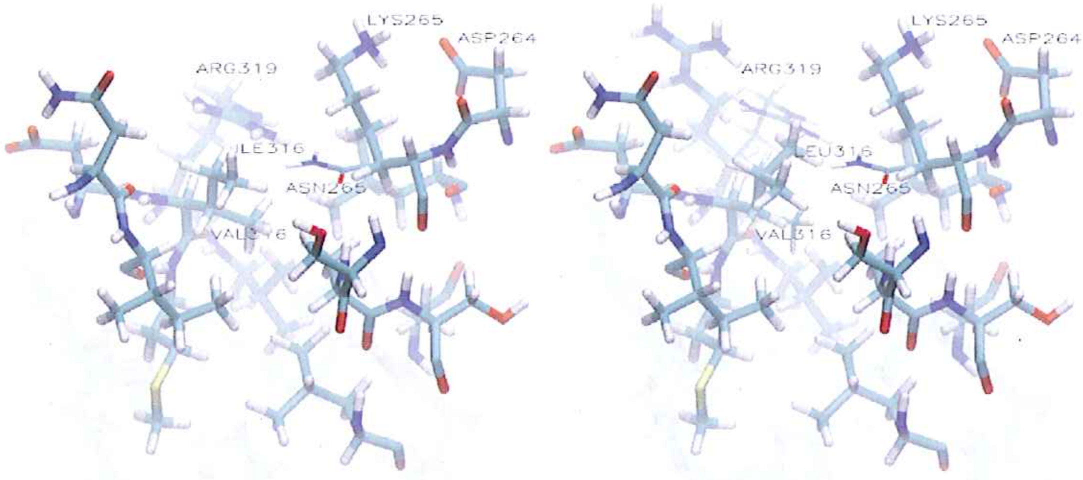


(c) Leu262Trp



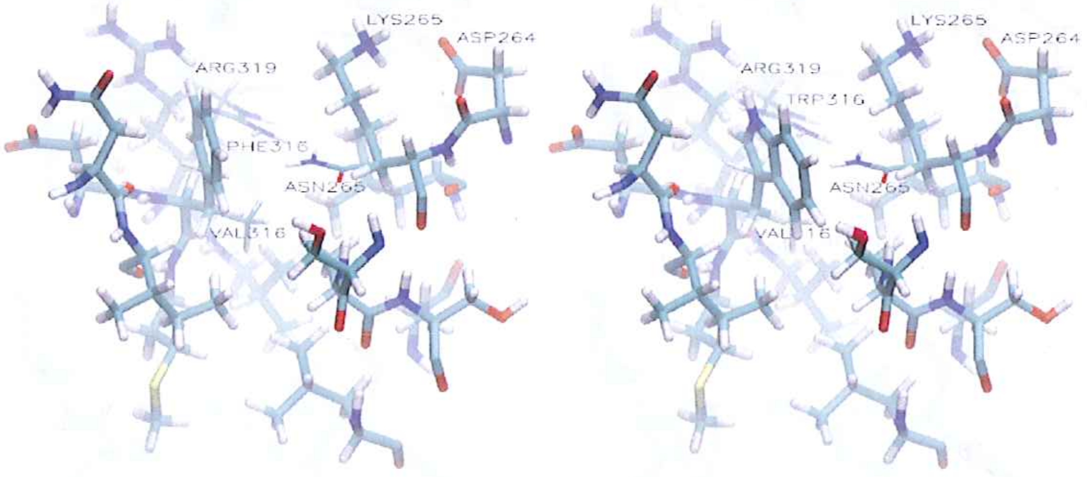
(d) Leu262Tyr

Figure 2-12: Four Leu262 mutations in the background of Asn265Lys.



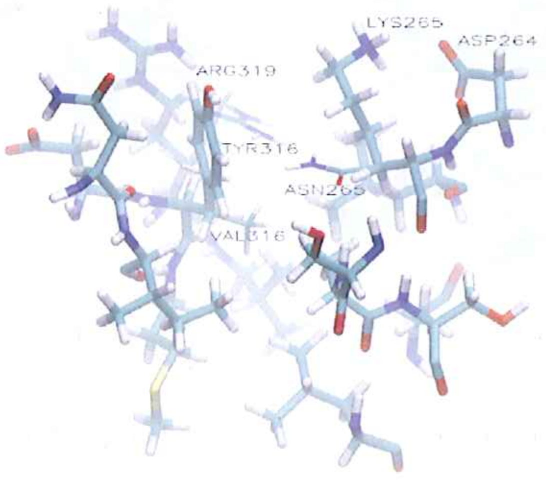
(a) Val316Ile

(b) Val316Leu



(c) Val316Phe

(d) Val316Trp



(e) Val316Tyr

Figure 2-13: Five Val316 mutations in the background of Asn265Lys.

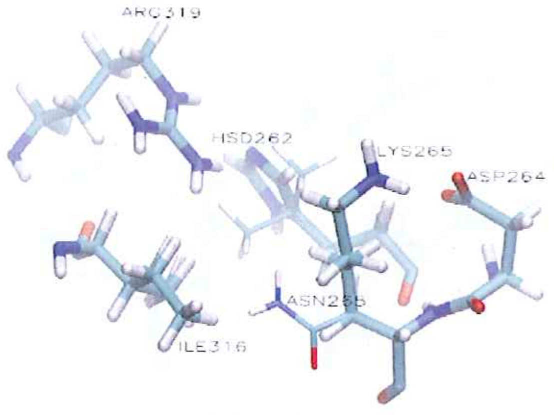
Table 2.9: Candidate mutations for Leu262 and Val316 in the background of Asn265Lys.

Leu262 Mutant	Val316 Mutant	$\Delta\Delta G_{\text{elec}}$	$\Delta\Delta G^{\text{SASA}}_{\text{total}}$	$\Delta\Delta G^{\text{CVDW}}_{\text{total}}$
Hsd	Ile	+0.82	−1.59	−0.91
Hsd	Leu	+1.17	+0.67	+2.51
Hsd	Phe	+1.87	−1.16	+2.22
Hsd	Trp	+3.72	−5.29	−2.18
Hsd	Tyr	+2.06	−2.13	+1.42
Phe	Ile	+0.87	−2.69	−2.24
Phe	Leu	+1.51	−1.09	+0.05
Phe	Phe	+1.67	−2.68	+0.21
Phe	Trp	+3.59	−5.95	−3.83
Phe	Tyr	+1.86	−3.64	−0.57
Trp	Ile	+3.44	−3.47	−2.97
Trp	Leu	+2.85	−4.51	−4.47
Trp	Phe	+0.65	−4.69	−1.07
Trp	Trp	+2.57	−7.75	−4.57
Trp	Tyr	+0.80	−5.65	−1.82
Tyr	Ile	+0.89	−4.00	−3.52
Tyr	Leu	+1.93	−1.80	−1.30
Tyr	Phe	+1.72	−3.96	−1.07
Tyr	Trp	+3.65	−7.23	−5.10
Tyr	Tyr	+1.92	−4.91	−1.85

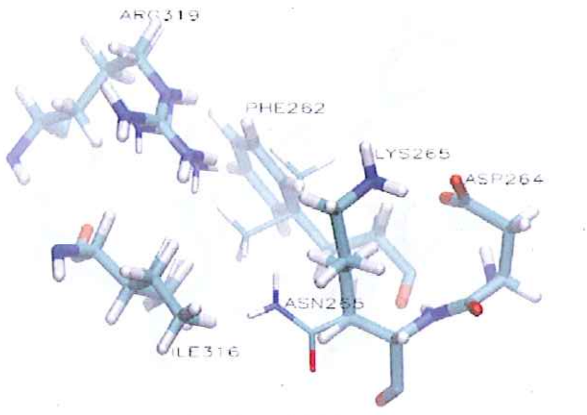
electrostatics relative to wild type, but many of them are predicted to significantly improve packing stability. Figure 2-14 shows the five triple mutants with the best electrostatics: Leu262Hsd–Val316Ile, Leu262Phe–Val316Ile, Leu262Trp–Val316Phe, Leu262Trp–Val316Tyr, and Leu262Tyr–Val316Ile. All five mutations have electrostatics less favorable than wild type by 0.6–0.9 kcal/mol. Of the five, Leu262Trp–Val316Phe and Leu262Trp–Val316Tyr have greatest improvement in packing stability relative to wild type. Figure 2-15 shows the five triple mutants predicted to improve packing stability relative to wild type by greater than 5 kcal/mol by either or both high-resolution energy models: Leu262Hsd–Val316Trp, Leu262Phe–Val316Trp, Leu262Trp–Val316Trp, Leu262Trp–Val316Tyr, and Leu262Tyr–Val316Trp. For these five mutations, electrostatics are predicted to be at least 2.5 kcal/mol less favorable than wild type, except for Leu262Trp–Val316Tyr, whose electrostatics are less favorable than wild type by only 0.8 kcal/mol. Of the twenty triple mutations, Leu262Trp–Val316Phe and Leu262Trp–Val316Tyr provide the best trade-off between destabilizing electrostatics and significantly improving packing stability.

In summary, we performed computational designs to replace Cys59, Cys227, Asn38, Asn265, and Met314 in PTE with amino acids less prone to degradation under storage. For Cys59, Cys227, and Met314, we recommend Cys59Ser, Cys227Ser, and Met314Gln mutants, respectively. For Asn38, we recommend Asn38Gln and Asn38Leu mutants. For Asn265, we recommend Asn265Lys, which is predicted to significantly improve electrostatics at the cost of decreased packing stability compared to wild type. We also recommend triple mutations from the set Leu262(Hsd, Phe, Trp)–Asn265Lys–Val316(Trp, Tyr), which are predicted to significantly improve packing stability compared to wild type.

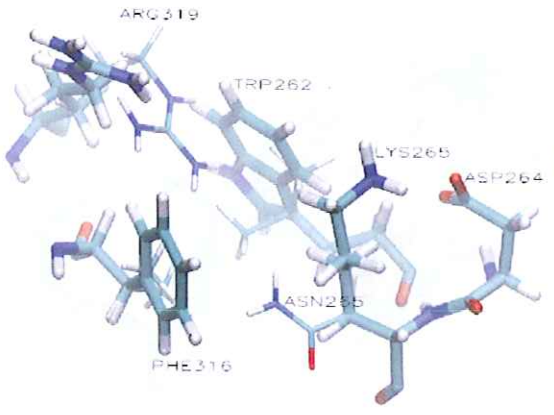
This computational design analysis has shown the multi-component nature of design and demonstrated that replacement of single amino acids often reduces packing stability with neighboring residues. Multiple mutations appear computationally to be capable of restoring packing stability. We await results of experimental testing to evaluate the effectiveness of these particular computational designs. This evaluation will also inform future computational designs for improved protein storage stability.



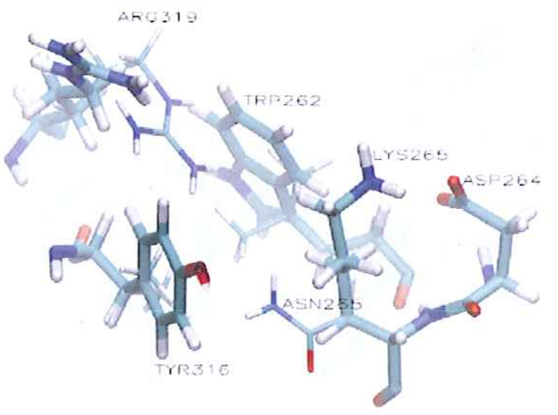
(a) Leu262Hsd–Val316Ile



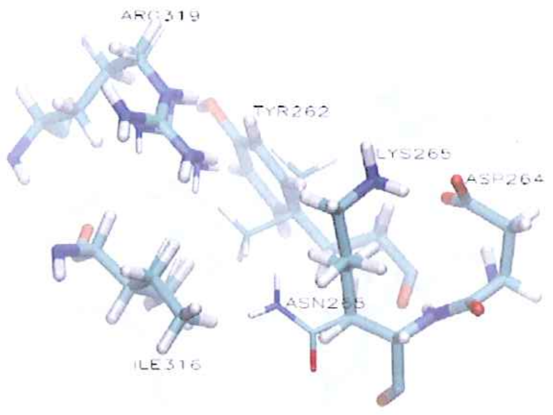
(b) Leu262Phe–Val316Ile



(c) Leu262Trp–Val316Phe

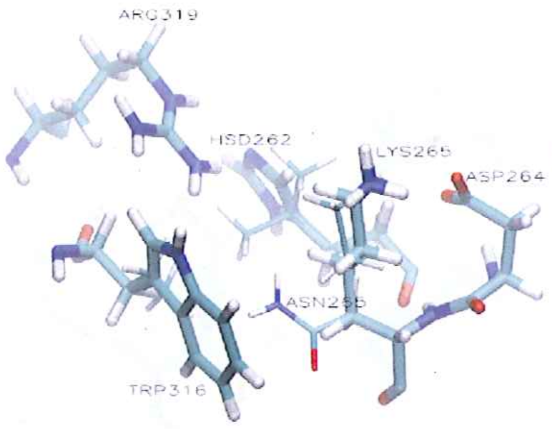


(d) Leu262Trp–Val316Tyr

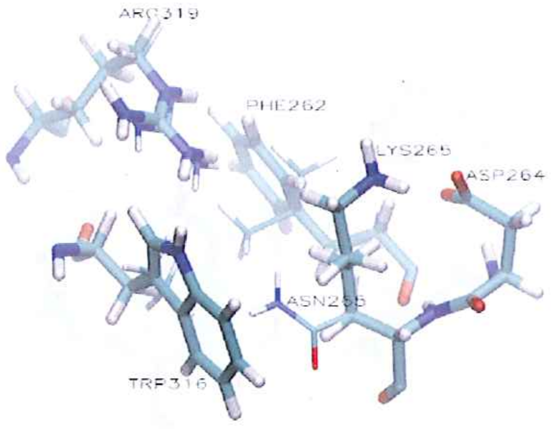


(e) Leu262Tyr–Val316Ile

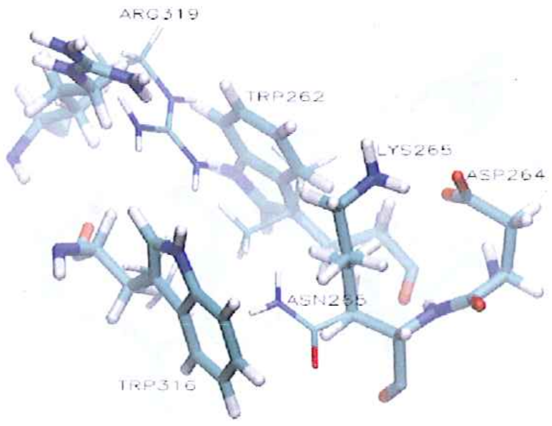
Figure 2-14: Leu262 and Val316 mutations in the background of Asn265Lys.



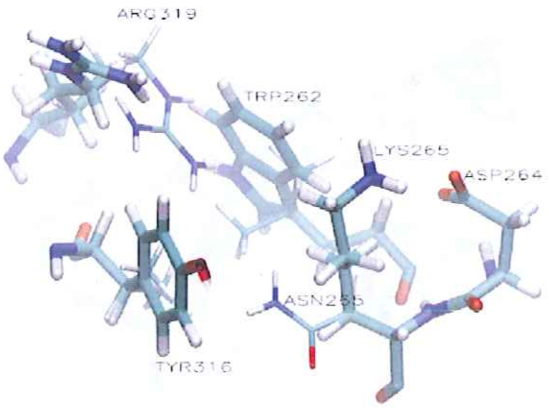
(a) Leu262Hsd–Val316Trp



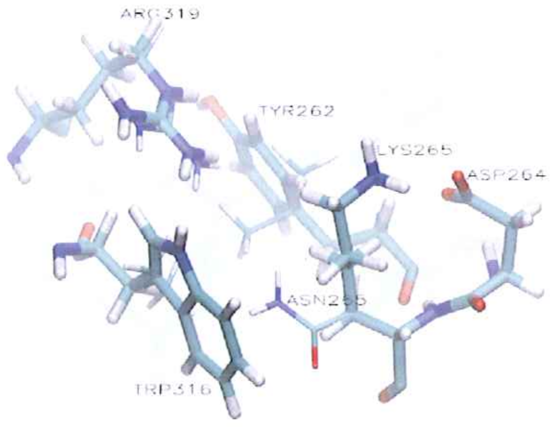
(b) Leu262Phe–Val316Trp



(c) Leu262Trp–Val316Trp



(d) Leu262Trp–Val316Tyr



(e) Leu262Tyr–Val316Trp

Figure 2-15: Leu262 and Val316 mutations in the background of Asn265Lys.

Chapter 3

Branch-and-bound rotamer search with enumeration

3.1 Introduction

The goal of computational protein design is to identify amino acid sequences that fold to a predetermined structure with desirable properties. In this chapter, we focus on search algorithms used in the protein design cycle. Specifically, we introduce BroMAP/A*, a new guaranteed search method for protein design. BroMAP/A* extends our branch-and-bound guaranteed search method (developed by Hong *et al.* [11]) by allowing enumeration. Given a matrix of single and pairwise energies with associated residue boundaries, BroMAP/A* returns a gap-free ordered list of the lowest energy conformations. This ordered list plays an important role in the protein design cycle. Analyzing the lowest energy conformations for structural and chemical similiarities improves predictions for synthesis and testing. Additionally, comparison of the ordered list to known low-energy conformations can help identify deficiencies in the input model (i.e., backbone assumption, rotamer library, and energy function).

The current standard for guaranteed search with enumeration in protein design is Dead-End Elimination (DEE) with A* search [70, 71]. DEE is a set of mathematical criteria that identify rotamers and rotamer combinations inconsistent with being in the ordered list of the lowest energy solutions [7–9, 72, 73]. After these rotamers and rotamer combinations are eliminated, A* acts on the reduced conformational space. A* is a best-first search technique that always follows the least-cost path from a starting node to a goal node representing the global minimum energy conformation (GMEC) [10]. Once A* reaches the GMEC, the ordered list of conformations is generated by a reverse walk along the least-cost path. Each node visited during the reverse walk represents the next lowest energy conformation in the search space. The walk terminates after enumerating all conformations within a predetermined energy threshold of the GMEC.

DEE/A* solves many protein design problems; however, it sometimes fails to identify the GMEC and/or enumerate the lowest energy conformations for large designs. In these cases, the remaining conformational space after DEE is too large for systematic search with A*. To determine the least-cost path, A* must retain the best-first search tree in memory. For complex search spaces, like those generated by difficult protein design cases, the best-first tree may contain an exponential number of nodes. If the number of nodes exceeds machine memory, then A* cannot complete the least-cost path. As an added complication, one cannot determine a priori if DEE/A* will fail based on the design's conformational complexity (assuming the conformational search space is larger than machine memory).

One way to resolve the memory issue is to replace A* with Iterative Deepening A* (IDA*) [74]. IDA* uses iterative depth-first search with a cutoff to save memory. With each iteration, the cutoff is increased, and the path length grows. IDA* only retains the seach tree up to the intermediate goal node based on the current cutoff, whereas A* must retain the entire search tree in memory. However, at each iteration, IDA* must rebuild the least-cost path up to the intermediate goal node from the previous iteration. Although IDA* has much lower memory requirements than A*, its computational expense is much greater. In practice, we find that IDA* is too slow for large designs. These cases usually require visiting an exponential number of nodes in the least-cost paths, which translates to an exponential number of iterations in IDA*.

An alternative approach to the memory issue is to subdivide the original search problem into small, manageable pieces for A* search. Two frameworks for the subdivision are divide-and-conquer and branch-and-bound. Georgiev *et al.* developed DACS, which uses traditional DEE/A* in a divide-and-conquer framework [75]. If DEE/A* cannot solve a particular design, DACS partitions the search space into subproblems whose union equals the original space. Each subproblem is then processed with DEE/A*, and subproblems that cannot be solved are again partitioned. The recursive partitioning continues until all subproblems have been solved. Divide-and-conquer methods such as DACS are able to solve large designs with limited memory resources; however, they must explicitly treat the entire search space with DEE/A*.

Branch-and-bound methods use statistical bounds on rotamers and rotamer pairs to guide subproblem creation and processing. Rotamers are distributed into subproblems based on their likelihood of belonging in the ordered list of the lowest energy conformations. Subproblems that are more likely to contain conformations in the ordered list are processed first in a depth-first manner. Solutions to these subproblems provide an upper bound on the GMEC, which can then be used to eliminate subproblems whose lower bound is outside the energy range of the ordered list. Unlike divide-and-conquer methods, branch-and-bound methods avoid performing DEE/A* on subproblems that provably do not contain any conformations in the ordered list, which reduces computational expenses.

Hong *et al.* developed BroMAP (branch-and-bound rotamer optimization using MAP estimation), a branch-and-bound method for protein design that can solve for the GMEC but not for an ordered, gapless list of near optimal solutions. They showed that BroMAP had similiar run-time performance to traditional DEE/A* across a large set of protein design test cases, and it solved many cases where DEE/A* failed. We have extended that work to allow enumeration of the lowest energy conformations with BroMAP. We call this new method BroMAP/A*. Given the same amount of computational resources, BroMAP/A* returns the gap-free ordered list for protein design cases where DEE/A* fails, which makes it a valuable tool in the protein design cycle.

In the following sections, we present our developmental and characterization work on BroMAP/A*. First we review theory for DEE/A* and BroMAP, and we describe modifications that allow enumeration of the lowest energy conformations in BroMAP/A*. Then we describe the implementations, computational resources, and protein design test cases used to compare BroMAP/A* and DEE/A*. Our results show a the clear advantage of using BroMAP/A* over traditional DEE/A* to enumerate the lowest energy conformations within limited computational resources.

3.2 Theory

A protein's function is determined by its three-dimensional structure, which in turn is determined by its amino acid sequence. In one common mode of computational protein design, a set of sequence design positions is chosen for a protein whose three-dimensional backbone structure is pre-selected. The goal is to determine the amino acid type and side chain placement for each design position that will result in a three-dimensional structure with desirable properties, often expressed as an energy to be minimized.

To convert the problem to a discrete space for optimization, only a finite number of fixed side chain placements, called rotamers, is allowed for each amino acid type. A global conformation of the protein is defined by choosing one rotamer for each design position. Consider a design with n positions, and let $m = \{m_1, m_2, ..., m_n\}$ represent one conformation of the protein where m_i represents a rotamer choice at position i. Assuming an energy function with only pairwise additive interactions contributing to the goal energy function, the value of the energy of conformation m can be calculated from

$$E_m = \sum_{i=1}^{n} E_{m_i}^{\text{self}} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E_{m_i,m_j}^{\text{pair}}, \tag{3.1}$$

where $E_{m_i}^{\text{self}}$ denotes the self energy of the rotamer at position i, including interactions with fixed atoms in the system, and $E_{m_i,m_j}^{\text{pair}}$ denotes the interaction energy of the rotamers at mobile positions i and j. These energies can be computed for the set of all wild-type and mutant rotamers using a molecular modeling package.

Given Equation 3.1, finding the GMEC can be formulated as a discrete optimization problem [7],

$$\mathbf{GMEC} = \min_{m \in M} E_m = \min_{m \in M} \left(\sum_{i=1}^{n} E^{\text{self}}_{m_i} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E^{\text{pair}}_{m_i,m_j} \right), \tag{3.2}$$

where M denotes the set of conformations defined by all possible combinations of rotamers at the n positions. In terms of theoretical complexity, the GMEC optimization problem is NP-hard. Pierce and Winfree proved the decision form of the GMEC problem (i.e., is there a rotamer at each position such that $E_m \leq C$ for a specified constant C) is NP-complete using a reduction from the satisfiability (SAT) problem [76].

Enumerating the conformations is an even more difficult optimization problem. Full enumeration of the conformational space is not practical (nor desirable) for most designs, so the amount of enumeration is typically limited by an energy threshold, e_{cut}. Let G be the global minimum energy in the conformational space. A guaranteed search method with enumeration must return a gap-free ordered list of all conformations whose energy is in the range $[G, G + e_{\text{cut}}]$. The GMEC optimization problem in Equation 3.2 is equivalent to the case where $e_{\text{cut}} = 0$.

3.2.1 DEE/A*

Next we review various Dead-End Elimination (DEE) criteria and A* search in the context of solving the optimization problem with enumeration. The DEE criteria identify rotamers and rotamer combinations that are mathematically inconsisent with being in the ordered list of the lowest energy conformations based on an energy threshold, e_{cut}. A* search returns the gap-free ordered list of all conformations whose energy is within e_{cut} of the global minimum energy.

Traditional Singles DEE

A trivial extension to the simple yet power DEE criterion of Desmet *et al.* removes rotamers that are guaranteed not to be in the ordered list [7]. For example, consider the DEE single rotamer elimination criterion. Let m_i^s and m_i^t represent two different rotamer choices for position i. If for some m_i^s there exists an m_i^t such that the following condition is met:

$$E_{m_i^s}^{\text{self}} + \sum_{j,j\neq i} \min_u \left(E_{m_i^s,m_j^u}^{\text{pair}}\right) > E_{m_i^t}^{\text{self}} + \sum_{j,j\neq i} \max_u \left(E_{m_i^t,m_j^u}^{\text{pair}}\right) + e_{\text{cut}}, \tag{3.3}$$

then m_i^s cannot be in the ordered list, and it can be pruned from the search space. One can think of rotamers s and t as competitors at position i. If the lowest energy rotamer s can produce when combined with the remaining conformational space is greater than the highest energy rotamer t can produce plus the energy threshold, then rotamer s cannot possibly belong to any conformation at position i in the ordered list.

When checking if rotamer s can be eliminated by some other rotamer at position i, the left-hand side of Equation 3.3 does not change. Similiarly, the right-hand side does not change when checking if rotamer t can eliminate some other rotamer at position i. To avoid redundant computations, the left-hand side and right-hand side of Equation 3.3 can be computed once and stored for each single rotamer. For a design with n positions and p rotamers on average per position, each side of Equation 3.3 requires $O(pn)$ pairwise evaluations. There are $O(pn)$ single rotamers, so the cost of one round of Traditional Singles DEE is $O(p^2n^2)$ pairwise evaluations.

Goldstein Singles and Pairs DEE

Goldstein developed a more powerful DEE single rotamer elimination criterion by simply rearranging the terms in Equation 3.3 [8]. Again, if for some m_i^s there exists an m_i^t such that the following condition is met:

$$E_{m_i^s}^{\text{self}} - E_{m_i^t}^{\text{self}} + \sum_{j,j\neq i} \left[\min_u \left(E_{m_i^s,m_j^u}^{\text{pair}} - E_{m_i^t,m_j^u}^{\text{pair}}\right)\right] > e_{\text{cut}}, \tag{3.4}$$

then m_i^s cannot be in the ordered list, and it can be pruned from the search space. The Goldstein criterion measures whether the total energy is increased or decreased by choosing rotamer t over s at position i when combined with the remaining conformational space. If the total energy is always decreased by at least the energy threshold in choosing rotamer t, then rotamer s cannot belong to any conformation at position i in the ordered list. For a design with n positions and p rotamers on average per position, Equation 3.4 requires $O(pn)$ pairwise evaluations for each combination of two rotamers at the same position. There are $O(p^2)$ rotamer combinations for each of the n positions, so the total cost of one round of Goldstein Singles DEE is $O(p^3n^2)$ pairwise evaluations. One round of Goldstein Singles DEE is more expensive than Traditional Singles DEE because the left-hand side of Equation 3.4 is unique for each combination of two rotamers.

Goldstein also proved this mathematical formulation for calculating how rotamer choices affect the total energy can be used to eliminate higher-order rotamer combinations (i.e., pairs, triplets, quadruplets, etc.) [8]. For example, consider the Goldstein rotamer pairs elimination criterion. Let (m_i^s, m_j^u) and (m_i^t, m_j^v) represent two different pairwise rotamer combinations for positions i and j. If for some rotamer pair (m_i^s, m_j^u) there exists a different rotamer pair (m_i^t, m_j^v) such that the following condition is met:

$$\begin{aligned}&\left(E^{\text{self}}_{m_i^s} + E^{\text{self}}_{m_j^u} + E^{\text{pair}}_{m_i^s,m_j^u}\right) - \left(E^{\text{self}}_{m_i^t} + E^{\text{self}}_{m_j^v} + E^{\text{pair}}_{m_i^t,m_j^v}\right) + \\ &\sum_{k,k\neq i\neq j} \min_{w} \left[\left(E^{\text{pair}}_{m_i^s,m_k^w} + E^{\text{pair}}_{m_j^u,m_k^w}\right) - \left(E^{\text{pair}}_{m_i^t,m_k^w} + E^{\text{pair}}_{m_j^v,m_k^w}\right)\right] > e_{\text{cut}},\end{aligned} \tag{3.5}$$

then the pair (m_i^s, m_j^u) cannot be in the ordered list, and it can be pruned from the search space. If the total energy is always decreased by at least the energy threshold in choosing rotamer t at position i and rotamer v at position j, then rotamers s and u cannot both belong to any conformation at positions i and j, respectively, in the ordered list. To be clear, pairwise elimination is not equivalent to eliminating both single rotamers using Equation 3.4. Elimination of the pair (m_i^s, m_j^u) means that if any conformation in the ordered list contains rotamer s at position i, then no conformation can contain rotamer u at position j, or vice versa. For a design with n positions and p rotamers on average per position, Equation 3.5 requires $O(pn)$ pairwise evaluations for each combination of two rotamer pairs. For any two positions, there are $O(p^4)$

ways to choose two out of the p^2 rotamer pairs. There are $O(n^2)$ pairs of positions to consider, so the total cost of one round of Goldstein Pairs DEE is $O(p^5n^3)$ pairwise evaluations.

Split-DEE

One limitation of Traditional and Goldstein singles DEE is that rotamer t must "outperform" rotamer s across the entire conformational space in order to prune it. Pierce et al. developed a means of splitting the conformational space such that multiple rotamers at position i can be used to eliminate rotamer s [9]. The simplest formulation of Split-DEE uses one split position ($s = 1$). Consider some position $k \neq i$, which has K rotamers. Suppose the space is split into K equally sized partitions; that is, each partition contains exactly one rotamer from position k. If for some m_i^s there exists an m_i^t in each partition such that the following condition is met for each rotamer v at split position k:

$$E^{\text{self}}_{m_i^s} - E^{\text{self}}_{m_i^t} + \sum_{j, j\neq k\neq i} \left[\min_u \left(E^{\text{pair}}_{m_i^s, m_j^u} - E^{\text{pair}}_{m_i^t, m_j^u} \right) \right] + \left(E^{\text{pair}}_{m_i^s, m_k^v} - E^{\text{pair}}_{m_i^t, m_k^v}\right) > e_{\text{cut}}, \quad (3.6)$$

then m_i^s cannot be in the ordered list, and it can be eliminated from the search space. For rotamer s at position i, if there is always some other rotamer choice for position i that decreases the total energy by at least e_{cut} in every partition, then rotamer s will never be chosen. For a design with n positions and p rotamers on average per position, one round of Split-DEE with one split position requires $O(p^3n^2)$ pairwise evaluations, which is the same cost as one round of Goldstein Singles DEE [9].

Multiple positions can be also be used for Split-DEE. For Split-DEE with two positions ($s = 2$), consider two positions h and k, which have H and K rotamers, respectively. The space is split into $H \times K$ equally sized partitions; that is, each partition contains exactly one rotamer from position h and one rotamer from position k. If for some m_i^s there exists an m_i^t in each partition such that the following condition is met for all rotamers v and w at split positions h and k:

$$E^{\text{self}}_{m_i^s} - E^{\text{self}}_{m_i^t} + \sum_{j, j \neq h \neq k \neq i} \left[\min_u \left(E^{\text{pair}}_{m_i^s, m_j^u} - E^{\text{pair}}_{m_i^t, m_j^u} \right) \right] + \\ (E^{\text{pair}}_{m_i^s, m_h^v} - E^{\text{pair}}_{m_i^t, m_h^v}) + (E^{\text{pair}}_{m_i^s, m_k^w} - E^{\text{pair}}_{m_i^t, m_k^w}) > e_{\text{cut}}, \tag{3.7}$$

then m_i^s cannot be in the ordered list, and it can be eliminated from the search space. For a design with n positions and p rotamers on average per position, one round of Split-DEE with two split positions requires $O(p^4 n^3)$ pairwise evaluations [9]. As the number of split positions increases, the number of partitions grows combinatorially, and Split-DEE evaluation becomes more costly.

Goldstein Magic Bullet Pairs Elimination

Testing all possible rotamer pairs with the Goldstein Pairs DEE criterion, as described in Equation 3.5, is computationally expensive. For a design with n positions and p rotamers on average per position, Goldstein Pairs DEE requires $O(p^5 n^3)$ pairwise evaluations to check all possible rotamer pairs. Through inspection of typical design problems, Gordon and Mayo found the number of dead-ending pairs is usually much smaller than the total number of pairs considered during Goldstein Pairs DEE [73]. They also observed that certain rotamer pairs, called "magic bullets", tend to eliminate most of these dead-ending pairs. Together, these observations form the basis for Goldstein Magic Bullet Pairs Elimination.

One of the most effective magic bullets is the rotamer pair for which the maximum interaction energy is least [73]. For a pair of positions, i and j, the magic bullet correponds to the rotamer pair identified by the following minimization:

$$\arg\min_{s,u} \left[E^{\text{self}}_{m_i^s} + E^{\text{self}}_{m_j^u} + E^{\text{pair}}_{m_i^s, m_j^u} + \sum_{k, k \neq i \neq j} \max_t \left(E^{\text{pair}}_{m_i^s, m_k^t} + E^{\text{pair}}_{m_j^u, m_k^t} \right) \right]. \tag{3.8}$$

After finding the magic bullet rotamer pair, all other rotamer pairs for positions i and j are compared to the magic bullet using the Goldstein Pairs DEE criterion from Equation 3.5. For a design with n positions and an average of p rotamers per position, comparing all other rotamer pairs for positions i and j to the magic bullet with Goldstein Pairs DEE requires $O(p^3n)$ pairwise evaluations. Finding the magic bullet for positions i and j requires $O(p^3n)$ pairwise evaluations. Goldstein Magic Bullet Pairs with one magic bullet for each of the $O(n^2)$ possible pairs of positions requires a total of $O(p^4n^3)$ pairwise evaluations, which is less than the $O(p^5n^3)$ pairwise evaluations required to test all rotamer pairs with one round of Goldstein Pairs DEE.

Logical Singles-Pairs Elimination

After applying DEE singles and pairs criteria, additional single rotamers can be logically eliminated as pointed out by Lasters *et al.* [72]. For example, consider rotamer r at position i. Suppose that all of the pairwise combinations between rotamer r at position i and each rotamer at another position j have been eliminated by a DEE pairs criterion such as Goldstein Magic Bullet Pairs. Any conformation that contains rotamer r at position i must interact with position j. Since every pairwise combination of rotamer r at position i with position j is inconsisent with being in the ordered list, rotamer r at position i cannot belong in the ordered list.

Logical elimination is also possible when a position has only one rotamer. For example, suppose position j has only one rotamer. Any conformation in the ordered list must contain that rotamer at position j, because no other choices are possible. If the pairwise combination of some rotamer r at position i and the one rotamer at position j has been eliminated by DEE pairs criteria, then rotamer r at position i cannot belong in the ordered list.

Position Unification

Higher-order DEE criteria, which test all possible rotamer pairs, triplets, etc., are often cost prohibitive due to the combinatorial nature of the GMEC problem. Goldstein introduced the concept of position unification, which allows for selective use of higher-order DEE criteria [8]. Unification involves merging the rotamers at two different positions into a new, unified position. Subsequent DEE pairs comparison between the unified position and any other position is equivalent to performing a triplet comparison in the non-unified search space. Similiarly, a DEE pairs comparison between two different unified positions is equivalent to a quadruplet comparison in the non-unified search space.

Position unification involves a trade-off between storage and eliminating power. Consider unification of positions i and j, which each have R_i and R_j rotamers, respectively. Before unification, $R_i + R_j$ storage is required for the single rotamers. After unification, $R_i \times R_j$ storage is required, since rotamer pairs in the non-unified space become single rotamers in the unified space. Now consider a third position k with R_k rotamers. Before unification, $R_k \times (R_i + R_j)$ storage is required for the rotamer pairs between positions i and k and between positions j and k. After unification, $R_k \times (R_i \times R_j)$ storage is required for the rotamer pairs between the unified position and position k.

There is no mathematically optimal criterion to choose positions for unification based on single rotamers and rotamer pairs. The effectiveness of a unified position in subsequent elimination depends on higher-order rotamer interactions, which are not captured in the individual pairwise energies. Therefore, heuristic criteria are used to select positions for unification. For example, one may choose to unify strongly interacting positions after initial DEE by considering the fraction of eliminated rotamer pairs for all pairwise combinations of positions [8].

A* Search

The DEE criteria discussed above eliminate rotamers and rotamer pairs that provably cannot belong in the ordered list of the lowest energy conformations. However, DEE does not necessarily eliminate every rotamer and rotamer pair that does not belong in the ordered list. After DEE has reduced the search space, brute-force enumeration can be used to generate the ordered list if the reduced space is sufficiently small. More elegantly and efficiently, A* is used to expand the conformation tree in order of conformational energies [10]. A* is a best-first search technique that always follows the least-cost path from a starting node to a goal node representing the global minimum energy conformation (GMEC). Once A* reaches the GMEC, the ordered list of conformations is generated by a reverse walk along the least-cost path.

Ideally, DEE significantly reduces the conformational space such that the A* search tree can fit in machine memory, in which case A* returns the gap-free ordered list. When the A* search tree becomes too large for memory, the problem cannot be solved. Two factors that affect the complexity of the reduced conformational space are the underlying design problem and the user-specified energy threshold for enumeration, e_{cut}. When the lowest energy conformations are driven by higher-order interactions between positions, such as three or more positions forming an energetically favorable pocket in the protein, branching in the A* search tree produces an exponential number of nodes, which exhausts memory [10]. The effect of the second factor, e_{cut}, is more obvious. As e_{cut} increases, fewer rotamers and rotamer combinations are inconsistent with being the ordered list of the lowest energy conformations; hence, they cannot be eliminated with DEE.

3.2.2 BroMAP

BroMAP is a branch-and-bound method for finding the global minimum energy conformation (GMEC) by dividing the search space into small, solvable subproblems based on available computational resources [11]. BroMAP maintains a global upper bound U, which is the lowest energy of any conformation found so far. To be clear, U is the closet known value of the GMEC energy, and the GMEC energy is always less than or equal to U at any point during processing. Starting with the original search space, BroMAP recursively performs the following steps until all subproblems have been processed:

1. Select a subproblem from the queue.

2. Can the subproblem be fully solved with limited time and memory? If so, (a) compute the minimum energy; (b) if the minimum energy is less than the current global upper bound U, then update U; (c) return to step (1).

3. Compute a lower bound and an upper bound on the minimum energy for this subproblem. If the upper bound is less than U, then update U.

4. If the lower bound exceeds the current global upper bound U, then discard (i.e., prune) the current subproblem and return to step (1).

5. When possible, exclude ineligible conformations from the search space.

6. Pick one residue and split its rotamers into two groups; define two child subproblems based on this split.

7. Add the child subproblems to the queue and return to step (1).

Statistical Bounds

BroMAP uses statistical bounds for subproblem creation, elimination, and processing. To calculate these bounds, the GMEC optimization problem in Equation 3.2 is formulated as a maximum-a-posteriori (MAP) estimation problem. For a random vector, $\mathbf{x} = (x_1, x_2, ..., x_n)$, and a probability distribution that maps each $\mathbf{x}$ in the sample space $\mathcal{X}$ to a probability $p(\mathbf{x})$, find the assignment $\mathbf{x}^*$ such that $\mathbf{x}^* \in \text{argmax}_{\mathbf{x} \in \mathcal{X}}\, p(\mathbf{x})$.

Consider a design with n positions. For each position, assign a discrete random variable, x_i, that ranges over R_i, the set of allowable rotamers at position i. Then, define a probability distribution $p(\mathbf{x})$ over $\mathcal{X} = R_1 \times R_2 \times ... \times R_n$ as follows:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-e(\mathbf{x})\}, \tag{3.9}$$

for a normalization constant Z and $e(\mathbf{x}) = \sum_{i=1}^{n} e_i(x_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} e_{ij}(x_i, x_j)$, where $e_i(r) = E^{\text{self}}_{m_i^r}$ for $r \in R_i$ and $e_{ij}(r, s) = E^{\text{pair}}_{m_i^r, m_j^s}$ for $(r, s) \in R_i \times R_j$. Note that $e(\mathbf{x})$ is just a rephrasing of Equation 3.1 in terms of the discrete random variables. The MAP assignment $\mathbf{x}$ that minimizes the energy, $e(\mathbf{x})$, maximizes the probability, $p(\mathbf{x})$. Finding the optimal MAP assignment is equally as difficult as finding the GMEC in the original optimization problem. However, many efficient approximation algorithms exist for MAP estimation.

BroMAP uses an adapted form of the tree-reweighted max-product algorithm (TRMP) on subproblems to compute lower and upper bounds on the energy by bounding the probability. TRMP, developed by Wainwright *et al.*, is a well known technique for MAP estimation using max-product belief propogation [77]. TRMP treats the MAP estimation problem as an undirected graphical model, and it computes bounds by building tree distributions with edge-based reparameterization updates. Hong *et al.* adapted TRMP to compute bounds for rotamers and rotamer pairs in the GMEC problem based on work by Kolmogorov [78]. Full mathematical details including proofs for adapted TRMP can be found in the original description of BroMAP [11]. In BroMAP, TRMP efficiently computes lower and upper bounds for entire subproblems; that is, the energy of any conformation in a subproblem is guaranteed to be in the range of the bounds. Additionally, TRMP computes bounds for all individual rotamers and rotamer pairs in a subproblem. The bounds for a particular rotamer specify the energy range for any conformation in the subproblem that contains the rotamer.

Eliminating Rotamers in Subproblems

During subproblem processing, Dead-End Elimination (DEE) and TRMP bounds are used to eliminate rotamers and rotamer pairs that are inconsistent with belonging to the minimum energy conformation for the subproblem (i.e., the local GMEC). The DEE criteria discussed in Section 3.2.1 are used without modification to reduce the conformational space of each subproblem. However, elimination of a rotamer in a particular subproblem does not extend to elimination of that same rotamer in other subproblems. The mathematical validity of DEE is local to each subproblem.

Rotamers and rotamer pairs are also eliminated by comparing individual bounds with the current global upper bound U. Let $L(i^s)$ represent the lower bound computed by TRMP for rotamer s at position i. The total energy of any conformation that contains rotamer s at position i will be at least $L(i^s)$. If $L(i^s)$ exceeds the current global upper bound U, then rotamer s at position i cannot belong in the GMEC. The same criterion applies for rotamer pairs. Let $L(i^s, j^u)$ represent the lower bound computed by TRMP for pairing of rotamer s at position i and u at position j. If $L(i^s, j^u)$ exceeds the current global upper bound U, then the pair cannot belong in the GMEC. As in DEE, rotamer pair elimination by TRMP bounds is not equivalent to elimination of both single rotamers. Either rotamer s at position i and or rotamer u at position j can belong in the GMEC, both both rotamers cannot simultaneously belong in the GMEC.

Solving Subproblems

In BroMAP, solving a subproblem means finding the minimum energy conformation for the subproblem (i.e., the local GMEC). Of the subproblems that can solved with limited time and memory, most are solved by A* search. However, the local GMEC can also be found during bounding by TRMP. TRMP includes an optimum specification (OS) criterion. If the MAP assignment at TRMP convergence satisfies the OS criterion, then it is guaranteed to be the optimal assignment. Recall that the optimal MAP assignment maximizes the probability, which minimizes the conformational energy.

Assuming the subproblem is not small enough for brute-force enumeration, there is no criterion that guarantees solving the subproblem within limited resources. Therefore, heuristic criteria are needed to decide whether to attempt A* search. We discuss the decision criteria in more detail later in this chapter. Subproblem evaluation is depicted in Figure 3-1. All subproblems undergo an initial pruning stage by DEE before the heuristic criteria are checked. If the reduced conformational space satisfies the criteria, then A* search is attempted. If A* search succeeds, then the subproblem is solved. If A* search fails, or if it is not attempted in the first place, then TRMP bounds are computed. If TRMP converges to an optimal MAP assignment, then the subproblem is solved. Otherwise, the subproblem lower bound is checked against the current global upper bound U to see if the subproblem can be eliminated. If the subproblem cannot be eliminated, then rotamers and rotamer pairs are eliminated using TRMP lower bounds before the subproblem is split.

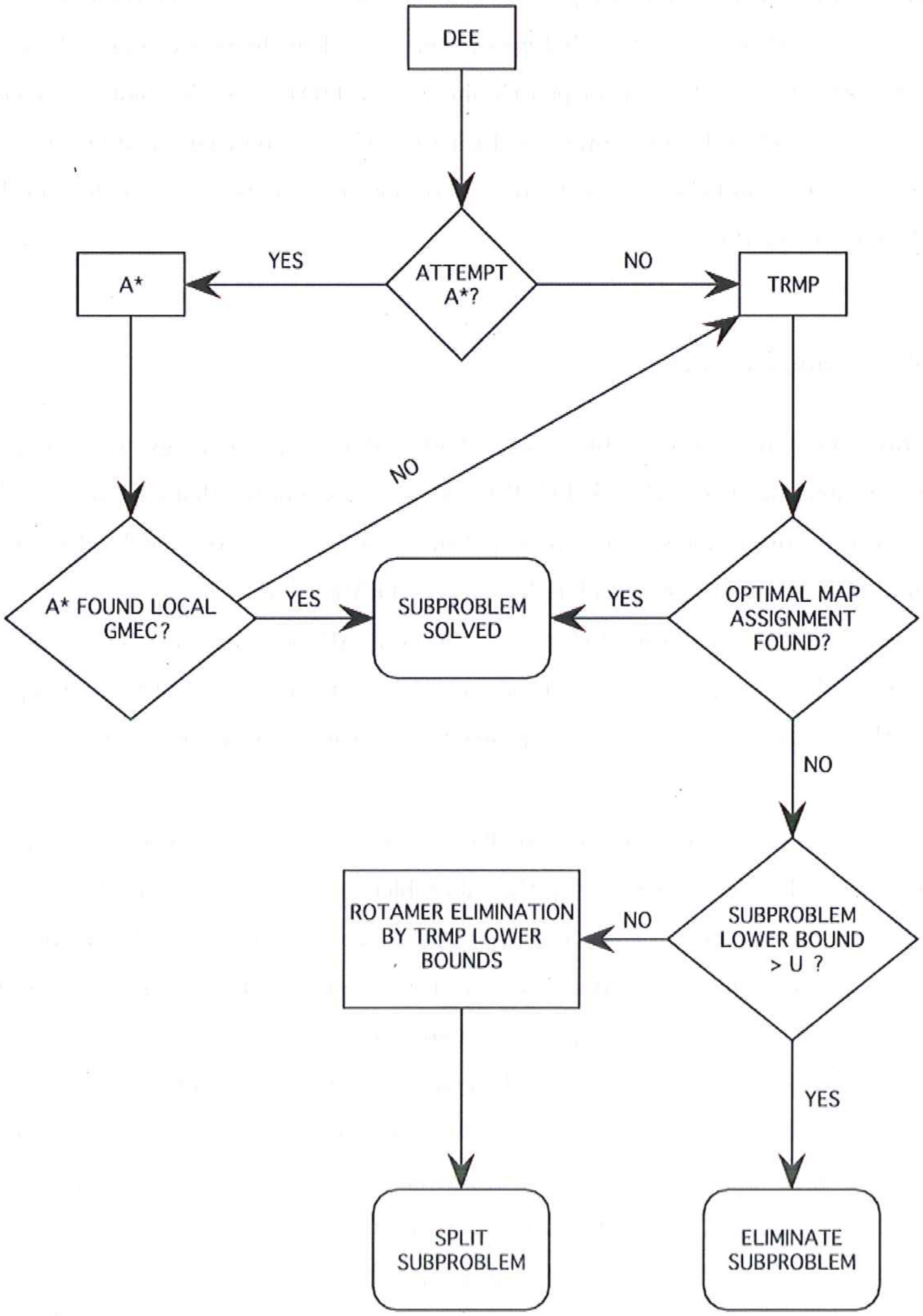


Figure 3-1: Subproblem evaluation in BroMAP.

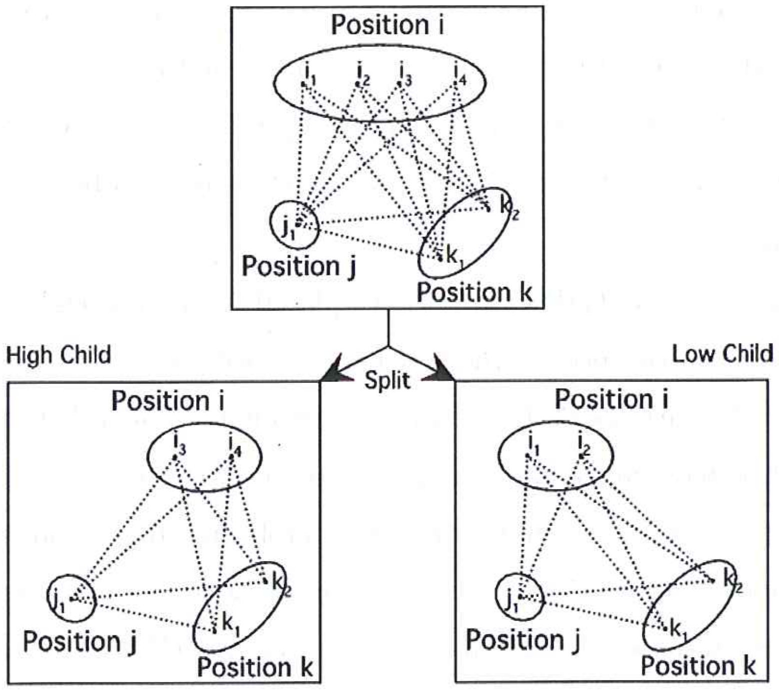


Figure 3-2: Example of subproblem splitting in BroMAP. Position i is the split position with TRMP rotamer lower bounds ordered as follows: $L(i_1) < L(i_2) < L(i_3) < L(i_4)$. Rotamers i_1 and i_2 are placed in the low child subproblem, and rotamers i_3 and i_4 are placed in the high child subproblem.

Subproblem Splitting and Ordering

During splitting, the parent subproblem is divided into two child subproblems of equal complexity using a split position. First, rotamers from the split position are ordered by their TRMP lower bounds. Then, the rotamers are divided equally into two sets. The set with higher-valued lower bounds (i.e., rotamers that are less likely to belong to the GMEC) are placed in the high child subproblem. The set with lower-valued lower bounds (i.e., rotamers that are more likely to belong to the GMEC) are placed in the low child subproblem. Figure 3-2 illustrates splitting for a simple example.

BroMAP uses a depth-first tree to order subproblems for evaluation. After a subproblem is split, the low child is always the next subproblem to be evaluated. Figure 3-2 illustrates subproblem ordering for a simple example. BroMAP's initial depth-first dive continues until a low child subproblem is solved. Solving the low child provides an update for the global upper bound U, which is used for subsequent rotamer and subproblem elimination. After the first low child is solved, the most recently generated high child is evaluated. If the high child is solved or eliminated, BroMAP proceeds up to the previous level in the depth-first tree; otherwise, the high child is split. BroMAP terminates when the high child generated by the first split is solved or eliminated.

Ideally, rotamers in the GMEC are always placed in the low child subproblem during splitting. When this occurs, the first low child subproblem solved during the depth-first dive will return the GMEC, and the global upper bound U becomes tight. BroMAP will then have maximum eliminating power when evaluating high child subproblems as it proceeds back up the tree of subproblems. High child subproblems whose lower bounds are nearly tight can be eliminated with minimal processing. On the other hand, the worst case occurs if the rotamers in the GMEC are placed in the high child when the first subproblem (i.e., original search space) is split. This can occur when the TRMP rotamer lower bounds are loose for the split position.

3.2.3 BroMAP/A*

BroMAP/A* extends BroMAP by providing enumeration of all conformations within an energy threshold, e_{cut}, of the GMEC. Starting with the original search space, BroMAP/A* recursively performs the following steps until all subproblems have been processed:

1. Select a subproblem from the queue.

2. Can the subproblem be fully solved with limited time and memory? If so, (a) enumerate conformations within e_{cut} of the minimum energy; (b) if the minimum energy is less than the current global upper bound U, then update U; (c) return to step (1).

3. Compute a lower bound and an upper bound on the minimum energy for this subproblem. If the upper bound is less than U, then update U.

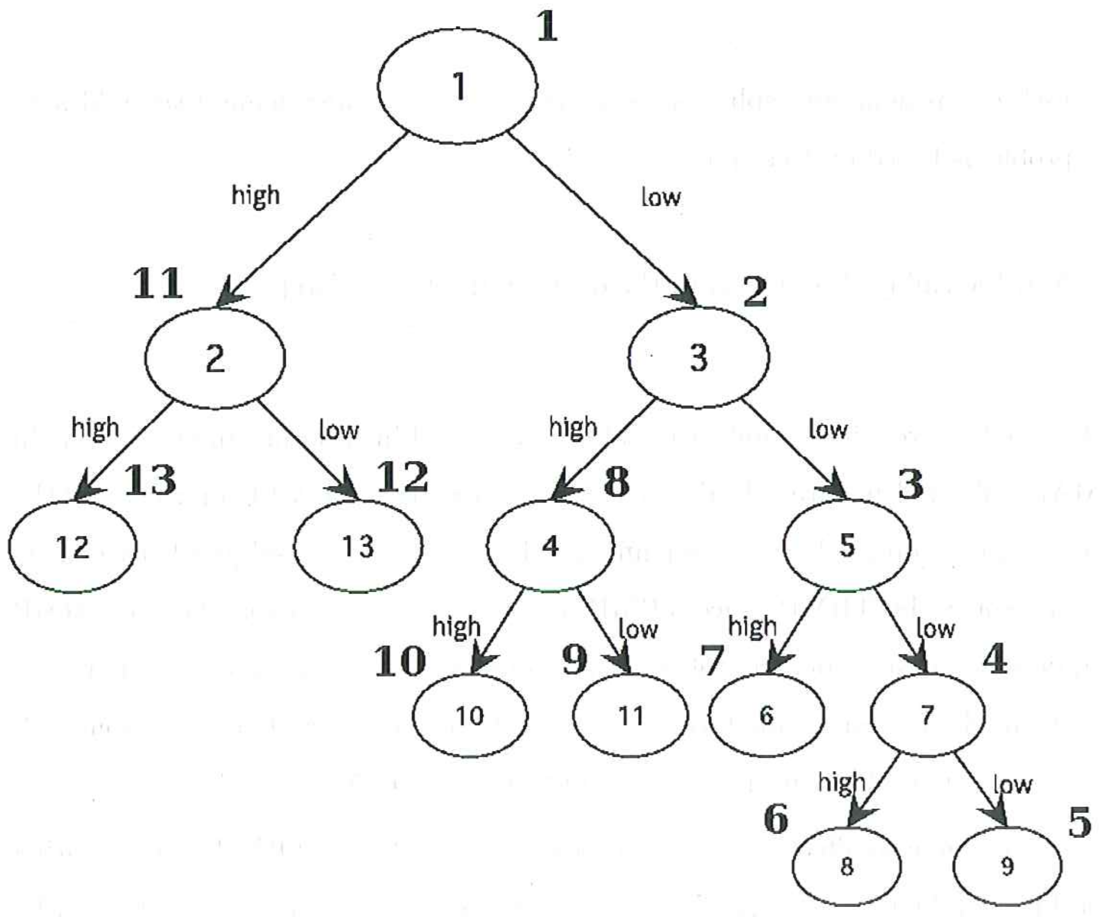


Figure 3-3: Example of depth-first subproblem ordering in BroMAP. Numbers inside of the circles correspond to subproblem indices, and numbers outside of the circles correspond to the order in which the subproblems are evaluated. In this example, the initial depth-first dive terminates at subproblem 9, the first low child that can be solved.

4. If the lower bound exceeds the current global upper bound U by e_{cut}, then discard (i.e., prune) the current subproblem and return to step (1).

5. When possible, exclude ineligible conformations from the search space accounting for e_{cut}.

6. Pick one residue and split its rotamers into two groups; define two child subproblems based on this split.

7. Add the child subproblems to the queue and return to step (1).

BroMAP/A* requires modification of the logic used in subproblem evaluation. In BroMAP, subproblems are solved when A* search finds the GMEC or when TRMP converges to an optimal MAP assignment. In BroMAP/A*, subproblems can no longer be solved by TRMP, since TRMP has no enumeration capability. If TRMP is required for a subproblem, the only two outcomes are subproblem splitting or subproblem elimination through comparison with the current global upper bound U. BroMAP/A* subproblem evaluation is depicted in Figure 3-4.

Enumeration also affects rotamer elimination by DEE and TRMP bounds during subproblem evaluation. As e_{cut} increases, fewer rotamers and rotamer pairs can be eliminated by DEE during the first step of subproblem evaluation. The DEE-reduced subproblem search space is larger than if no enumeration is required (i.e., $e_{\text{cut}} = 0$), and A* search is less likely to be attempted. Enumeration also changes rotamer and rotamer pair elimination by TRMP lower bounds; the criteria must now account for e_{cut}. Let $L(i^s)$ represent the lower bound computed by TRMP for rotamer s at position i. If $L(i^s) > U + e_{\text{cut}}$, then rotamer s at position i cannot belong in the ordered list. The same criterion applies for rotamer pairs. Let $L(i^s, j^u)$ represent the lower bound computed by TRMP for pairing of rotamer s at position i and u at position j. If $L(i^s, j^u) > U + e_{\text{cut}}$, then the pair cannot belong in the ordered list.

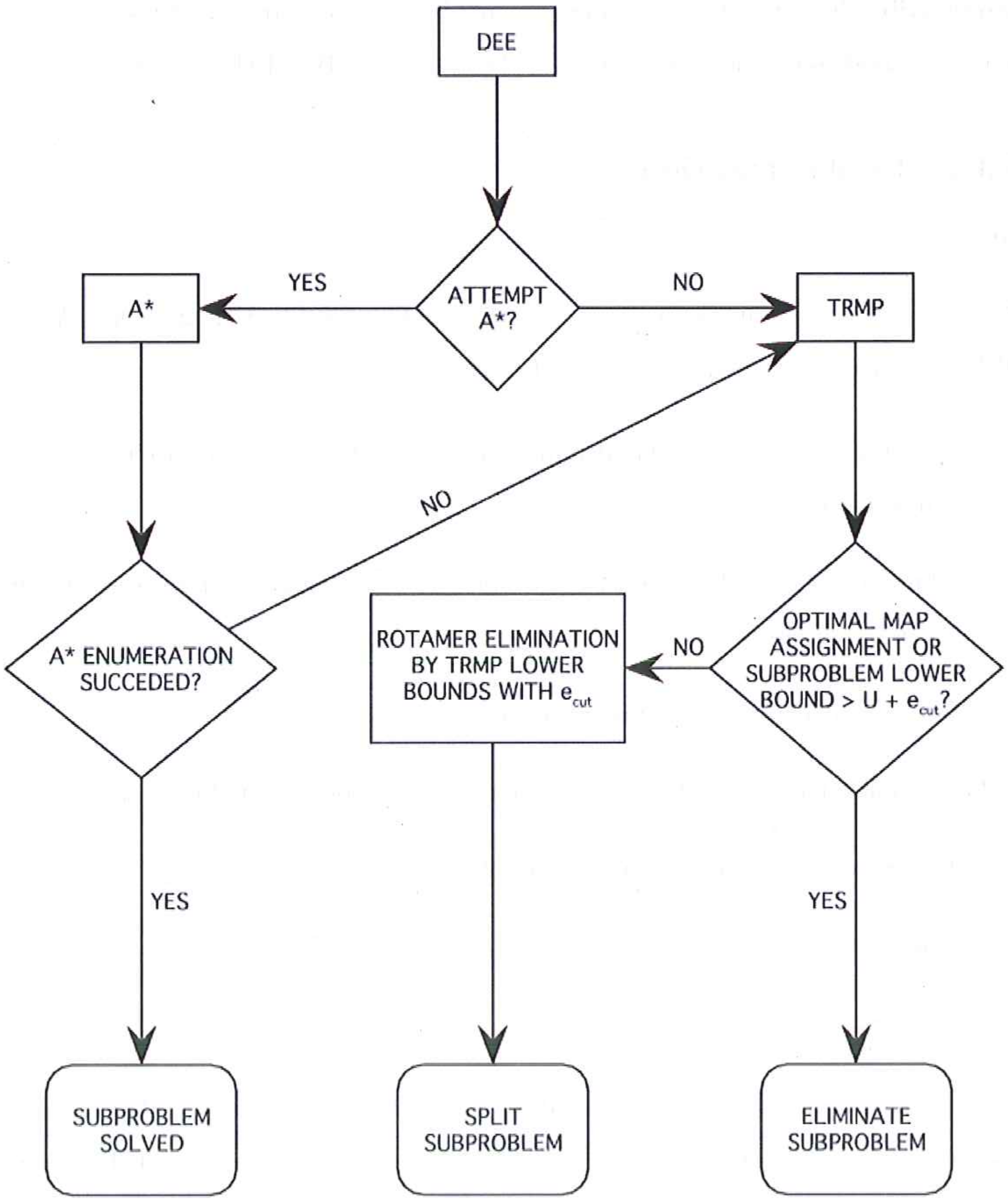

DEE
ATTEMPT A*?
YES
NO
A*
TRMP
NO
A* ENUMERATION SUCCEDED?
ROTAMER ELIMINATION BY TRMP LOWER BOUNDS WITH e_{cut}
NO
OPTIMAL MAP ASSIGNMENT OR SUBPROBLEM LOWER BOUND > U + e_{cut}?
YES
YES
SUBPROBLEM SOLVED
SPLIT SUBPROBLEM
ELIMINATE SUBPROBLEM

Figure 3-4: Subproblem evaluation in BroMAP/A*.

3.3 Methods

In this section, we discuss implementation details for DEE/A* and BroMAP/A*. We also describe the computational environment, limiting resources, and protein design test cases used to characterize and assess DEE/A* and BroMAP/A* performance.

3.3.1 Implementation

DEE/A*

We used a DEE/A* implementation written in C code by Altman [79]. We ran DEE/A* with the following options and order:

1. Eliminate singles using Goldstein's condition [8]. Repeat until elimination is unproductive.
2. Eliminate using split flags with one split position (s=1) [9]. Repeat until elimination is unproductive.
3. Do logical singles-pairs elimination [72].
4. Eliminate pairs using Goldstein's condition with one magic bullet [73].
5. Do logical singles-pairs elimination [72].
6. If unification is possible, perform unification [8], and go to step (1).
7. Perform A* search [10].

For Goldstein magic bullet pairs DEE, we determine one magic bullet for every pair of positions using Equation 3.8. Then, we use the Goldstein Pairs DEE criterion from Equation 3.5 to compare all other rotamer pairs with the magic bullet for each pair of positions.

Recall that position unification transforms rotamer pairs between two positions into single rotamers for a new unified position. Unification allows for the selective use of higher-order DEE criteria at the cost of increased storage. For unification, we choose the pair of positions with the largest fraction of flagged rotamer pairs (i.e., Goldstein's heuristic criterion [8]). However, we introduced a unification parameter, C_{uni}, in the criterion to limit the storage required for rotamer energies and to keep the cost reasonable during subsequent rounds of DEE. The C_{uni} parameter specifies a maximum value for the total number of single rotamers after unification. First, we check the pair of positions with the largest fraction of flagged rotamer pairs. If unifying this pair produces a total number of single rotamers that exceeds C_{uni}, then we check the pair of positions with the next-largest fraction of flagged rotamer pairs, and so on. When the value of C_{uni} is large, unified positions contain more single rotamers, more storage is required for rotamer energies, and DEE evaluation after unification is more expensive. Conversely, when the value of C_{uni} is small, unified positions contain fewer single rotamers, less storage is required for rotamer energies, and DEE evaluation after unification is less expensive.

For each case we considered multiple values for C_{uni}: 2000, 4000, 6000, 8000, 10000, 12000, 14000. Each of the seven values corresponds to a seperate DEE/A* run (i.e., C_{uni} does change during execution). We also ran DEE/A* with no unification (i.e., skip step (6) in the above algorithm).

BroMAP/A*

Our BroMAP/A* implementation is built from C++ code for BroMAP developed by Hong *et al.* [11]. Both BroMAP and BroMAP/A* use PEBBL, an object oriented branch-and-bound library [80]. PEBBL manages the queue and data structures for the depth-first subproblem tree. PEBBL also handles many high-level subproblem creation, evaluation, and elimination details. Code for DEE and A* search in BroMAP/A* subproblem evaluation is based on C code written by Altman [79].

In BroMAP/A*, DEE is the first step in subproblem evaluation. For DEE in BroMAP/A*, we used the same criteria and ordering as the DEE/A* implementation discussed in the previous section, namely, steps (1) through (6). As with DEE/A*, we used a unification parameter, C_{uni}, to limit the storage required for rotamer energies after unifying two positions. In our tests of BroMAP/A*, we used a constant value of $C_{\text{uni}} = 6000$ for all subproblems (i.e., only unify two positions if the subproblem search space after unification contains fewer than 6000 single rotamers).

After DEE, we used a heuristic criterion based on subproblem complexity to determine whether to attempt A* search. If the DEE-reduced subproblem search space contained less than 10^{28} conformations and fewer than 300,000 rotamer pairs, then A* search was attempted; otherwise, TRMP bounds were computed. These values were based on our prior experience with using DEE/A* for protein design problems. We have found that A* typically solves problems of this size given the computational resources we will discuss shortly.

During subproblem evaluation, rotamers and rotamer pairs are eliminated by DEE and TRMP bounds. DEE never eliminates a rotamer or rotamer pair that belongs to the local GMEC (i.e., the minimum energy conformation of a subproblem). On the other hand, rotamers and rotamer pairs that belong to the local GMEC can be eliminated by TRMP lower bounds if the local GMEC energy exceeds the current global upper bound U by at least e_{cut}. Immediate elimination of these particular rotamers from the subproblem increases the local GMEC energy. To avoid changing subproblem optimal values, we numerically enforced rotamer elimination by the TRMP lower bound, which was also used in BroMAP [11]. When a rotamer or rotamer pair can be eliminated by TRMP lower bounds, we set its energy to a very large value. This technique preserves the local GMEC of the parent subproblem while ensuring that the rotamers will be eliminated by DEE in the child subproblems.

To choose a position for subproblem splitting, we used a criterion that balances the need to create low child subproblems that contain rotamers that are more likely to belong to the GMEC with the need to quickly create solvable subproblems. TRMP computes lower bounds for all rotamers at each position. For each position, we computed the difference between the minimum and maximum lower bounds. Then, we selected the position with the largest difference, along with any other position whose difference was at least 90% of the largest difference. For example, if there were four position whose differences were 95, 20, 50, and 100, respectively, then positions 1 and 4 would be selected. Next, we determined which of the selected positions would produce the smallest child subproblems. Assuming an equal distribution of rotamers to the two child subproblems (i.e., a 50/50 split), we calculated the total number of rotamer pairs that would result from splitting at each of the selected positions. Finally, we chose the position that produced the smallest number of rotamer pairs. From the previous example, if splitting at positions 1 and 4 produced 100,000 and 200,000 subproblem rotamer pairs, respectively, then position 1 would ultimately be chosen for splitting, despite the fact that position 4 had a larger difference in rotamer lower bounds.

3.3.2 Computational Environment

We tested both search methods on 64-bit Linux workstations with a quad-core 2.8 GHz Intel Xeon E5440 processor and 4 GB of memory. The C/C++ codes for BroMAP/A* and DEE/A* were compiled using the Intel C/C++ Compiler Version 11.1 for Linux. Both executables were built using the same compiler optimization flags. All procedures were executed over a single core, and all times reported are CPU time.

3.3.3 Test Cases

To compare BroMAP/A* and DEE/A*, we chose the same set of 68 design cases used by Hong *et al.* to assess BroMAP [11]. These cases were derived from practical computational designs by Lippow *et al.* [1].

Each case corresponds to one of three different model systems:

1. FN3: derived from protein Fn3, the tenth human fibronectin type III domain. It is a 94-residue β-sheet protein with an immunoglobulin-like fold.

2. D44.1 and D1.3: antibodies that bind to different epitopes of hen egg-white lysozyme (HEL).

3. EPO: human erthropoietin (Epo) protein complexed to its receptor (EpoR).

Each case corresponds to one of three types of protein regions:

1. INT: protein–protein binding interface.
2. CORE: protein core (i.e., side chains that are not solvent-exposed).
3. CORE++: protein core plus boundary positions that are partially exposed to solvent.

Each case uses one of three sets of amino acid types for the design positions:

1. H: hydrophobic amino acids (Ala, Phe, Gly, Ile, Leu, Met, Trp, Val).
2. HP: hydrophobic plus polar amino acids (Ala, Phe, Gly, Ile, Leu, Met, Trp, Val, His, Asn, Gln, Ser, Thr, Tyr).
3. A: all types of amino acids, excluding Pro and Cys.

Each case uses one of two different rotamer libraries:

1. REG: standard rotamer library, which is based on the backbone-independent library by Dunbrack and Cohen (May 2002) [64]. The library was supplemented with three His rotamers for an unsampled ring flip, and two Asn rotamers to increase sampling of the final dihedral angle rotation.
2. EXP: expanded rotamer library. This library was created by expanding χ_1 and χ_2 of the rotamers in REG by $\pm 10°$. The hydroxls of Ser, Thr, and Tyr were sampled every 30°.

For all libraries and cases, the crystallographic wild-type rotamer for each design position was added to the library using the complete Cartesian representation of the side chain so that its bond lengths and bond angles, as well as torsions, would be maintained.

The singleton and pairwise rotamer energies were computed using the CHARMM PARAM22 molecular mechanics force field with a 4r distance-dependent dielectric constant and with no cut-offs for non-bonded interactions [59–61]. All available energy terms were included in the calculation (i.e., bond, angle, Urey–Bradley, dihedral, improper, Lennard-Jones, and electrostatic). Side-chain rotamers that clashed with fixed protein regions were eliminated during case generation. Clashes were determined by examining the singleton energy of every possible rotamer. If a rotamer's singleton energy exceeded the lowest singleton energy at that sequence position by at least 50 kcal/mol, then it was removed from the search space.

Table 3.1 lists the properties and complexity of each test case. In terms of problem size, the test cases range from small designs, 5 positions and 10^9 conformations, to much larger designs, 41 positions and 10^{78} conformations.

For each test case, we considered varying amounts of enumeration using e_{cut} values of 0, 1, 3, and 5 kcal/mol. We ran BroMAP/A* once for each test case and e_{cut} value. We ran DEE/A* eight times for each test case and e_{cut} value, one DEE/A* run without position unification and seven runs using the following unification parameter C_{uni} values: 2000, 4000, 6000, 8000, 10000, 12000, and 14000. In total, we performed 272 (68×4) BroMAP/A* runs, and 2176 ($68 \times 4 \times 8$) DEE/A* runs.

We did not impose a limit on CPU time for any of the runs; however, all runs had the same physical memory limit of 4 GB. For some test cases, DEE/A* failed because the storage required for the A* search tree exceeded physical memory. When this occurred, A* could not complete the lowest-cost path, and no solutions were returned. We still measured CPU time for these cases (i.e., the time spent on DEE plus the time for A* to fill up memory). On the other hand, BroMAP/A* always solved the test case given the same physical memory limit. If A* search failed during evaluation of a subproblem in the BroMAP/A* tree, the subproblem was recursively subdivided until it could be solved within the available memory resources.

Table 3.1: Protein design test cases. Each column represents (1) No: test case number, (2) Model: model system, (3) Region: protein region, (4) AA: amino acid types for the design positions, (5) Lib: rotamer library, (6) n: number of positions, (7) n_D: number of design positions (excluding mobile waters), (8) w: number of mobile water molecules, (9) Singles: total number of single rotamers, (10) Pairs: total number of rotamer pairs, (11) $\log_{10}$conf: conformational complexity.

No.	Model	Region	AA	Lib	n	n_D	w	Singles	Pairs	$\log_{10}$conf
1	fn3	core	HP	REG	14	14	0	743	2.5×10^5	23.4
2	fn3	core++	HP	REG	20	20	0	1778	1.5×10^6	38.2
3	fn3	core++	HP	REG	23	23	0	1894	1.7×10^6	42.8
4	fn3	core++	HP	REG	25	25	0	2048	2.0×10^6	46.6
5	fn3	core++	HP	REG	27	27	0	2083	2.1×10^6	49.0
6	fn3	core	HP	EXP	14	14	0	8774	3.5×10^7	38.5
7	D44.1	int	A	REG	7	4	0	476	8.5×10^4	9.4
8	D44.1	int	A	REG	7	7	0	822	2.8×10^5	14.3
9	D44.1	int	A	REG	8	8	0	965	4.0×10^5	16.3
10	D44.1	int	A	REG	9	9	0	1019	4.5×10^5	18.0
11	D44.1	int	A	REG	10	10	0	1133	5.6×10^5	19.8
12	D44.1	int	A	REG	11	11	0	1376	8.4×10^5	22.3
13	D44.1	int	A	REG	16	14	2	2020	1.9×10^6	32.6
14	D44.1	int	A	EXP	7	4	0	5026	9.5×10^6	16.5
15	D44.1	int	A	EXP	7	5	0	7019	1.9×10^7	18.0
16	D44.1	int	A	EXP	7	6	0	7910	2.6×10^7	19.3
17	D44.1	int	A	EXP	7	7	0	8771	3.2×10^7	21.6
18	D1.3	int	A	REG	6	4	2	450	8.3×10^4	18.5
19	D1.3	int	A	REG	11	8	3	767	2.6×10^5	18.5
20	D1.3	int	A	REG	23	7	9	1618	1.2×10^6	36.2
21	D1.3	int	A	EXP	6	4	2	3599	4.8×10^6	15.2
22	D1.3	int	A	EXP	7	5	2	3616	4.8×10^6	15.2
23	D1.3	int	A	EXP	8	6	2	4070	6.3×10^6	17.7
24	D1.3	int	A	EXP	11	4	3	4612	8.0×10^6	21.3
25	D1.3	int	A	EXP	11	6	3	4987	9.7×10^6	22.4
26	D1.3	int	A	EXP	11	7	3	5461	1.2×10^7	23.3
27	D1.3	int	A	EXP	11	7	3	5891	1.4×10^7	24.7
28	D1.3	int	A	EXP	11	8	3	6365	1.7×10^7	25.7
29	D1.3	core	H	REG	16	16	0	342	5.4×10^4	20.4
30	D1.3	core	H	REG	20	20	0	430	8.6×10^4	25.0
31	D1.3	core	H	REG	26	26	0	503	1.2×10^5	30.3
32	D1.3	core	H	REG	34	34	0	567	1.5×10^5	36.7
33	D1.3	core	HP	REG	16	16	0	980	4.4×10^5	27.6
34	D1.3	core	HP	REG	20	20	0	1228	7.1×10^5	33.9
35	D1.3	core	HP	REG	26	26	0	1431	9.7×10^5	41.8
36	D1.3	core	HP	REG	34	34	0	1582	1.2×10^6	50.7

(*continued*)

Table 3.1 – (*continued*)

No.	Model	Region	AA	Lib	n	n_D	w	Singles	Pairs	$\log_{10}$conf
37	D1.3	core	H	EXP	13	13	0	1844	1.5×10^6	26.5
38	D1.3	core	H	EXP	16	16	0	2734	3.5×10^6	35.0
39	D1.3	core	H	EXP	20	20	0	3370	5.3×10^6	41.9
40	D1.3	core	H	EXP	26	26	0	3894	7.1×10^6	50.6
41	D1.3	core	H	EXP	34	34	0	4444	9.4×10^6	63.8
42	epo	int	A	REG	5	5	0	466	7.1×10^4	8.8
43	epo	int	A	REG	6	6	0	419	6.8×10^4	9.5
44	epo	int	A	REG	11	11	0	1005	4.4×10^5	19.2
45	epo	int	A	REG	21	11	3	1503	1.0×10^6	31.4
46	epo	int	A	REG	21	15	3	1999	1.9×10^6	36.7
47	epo	int	A	REG	21	18	3	2138	2.1×10^6	39.5
48	epo	int	A	EXP	5	5	0	5001	8.4×10^6	14.2
49	epo	int	A	EXP	6	6	0	4170	6.8×10^6	14.5
50	epo	int	A	EXP	8	8	0	7544	2.3×10^7	23.0
51	epo	int	A	EXP	9	9	0	8724	3.2×10^7	26.1
52	epo	core	H	REG	17	17	0	291	3.9×10^4	19.9
53	epo	core	H	REG	22	22	0	395	7.4×10^4	26.5
54	epo	core	H	REG	28	28	0	433	8.9×10^4	29.7
55	epo	core	H	REG	33	33	0	573	1.6×10^5	37.2
56	epo	core	H	REG	41	41	0	727	2.6×10^5	46.2
57	epo	core	HP	REG	17	17	0	827	3.2×10^5	27.7
58	epo	core	HP	REG	22	22	0	1103	5.8×10^5	36.4
59	epo	core	HP	REG	28	28	0	1208	7.0×10^5	42.0
60	epo	core	HP	REG	33	33	0	1615	1.3×10^6	52.1
61	epo	core	HP	REG	36	36	0	1827	1.6×10^6	57.6
62	epo	core	HP	REG	38	38	0	1956	1.9×10^6	61.2
63	epo	core	HP	REG	41	41	0	1999	1.9×10^6	64.0
64	epo	core	H	EXP	17	17	0	2307	2.4×10^6	33.8
65	epo	core	H	EXP	22	22	0	3006	4.2×10^6	45.0
66	epo	core	H	EXP	28	28	0	3213	4.8×10^6	50.3
67	epo	core	H	EXP	33	33	0	4322	8.9×10^6	62.9
68	epo	core	H	EXP	41	41	0	5712	1.6×10^7	78.3

3.4 Results and Discussion

Among the 68 test cases, DEE/A* solved 50, 39, 32, and 24 cases for e_{cut} values 0, 1, 3, and 5, respectively. Given the same computational resources, BroMAP/A* solved every case for all four e_{cut} values. Thus, there were no cases DEE/A* solved that BroMAP/A* was unable to solve. For $e_{\text{cut}} = 0$ (i.e., solving only for the GMEC), the

Table 3.2: Test cases solved by DEE/A*.

e_{cut}	Percentage Solved without Unification	Percentage Solved with Unification
0	34% (23/68)	74% (50/68)
1	26% (18/68)	57% (39/68)
3	25% (17/68)	47% (32/68)
5	18% (12/68)	35% (24/68)

same test cases were unsolved by DEE/A* here as found in previous computational experiments by Hong *et al.* [11]. Additionally, our results show that the number of cases solved by DEE/A* decreases significantly once enumeration is required.

Position unification was key in the ability of DEE/A* to solve test cases. Table 3.2 summarizes the number of cases solved with and without unification. Compared to DEE/A* with unification, the number of cases solved by DEE/A* decreased by a factor of two when unification was not allowed. Overall, DEE/A* run times were significantly shorter when unification was not allowed. The mean run time across all cases and e_{cut} values was 60 minutes, and the longest run time was 300 minutes. Compared to the run times we will see shortly, DEE/A* without unification is relatively inexpensive to attempt, but it is unlikely to solve most designs.

Overall, BroMAP/A* solved all of the test cases for the four e_{cut} values considered. However, BroMAP/A* required a significant amount of run time, greater than one month, for some cases. Table 3.3 lists the test cases and e_{cut} values for which BroMAP/A* required more than one month of running time. These cases represent particularly challenging designs given the computational resources used in our tests. DEE/A* did not solve any of these cases. Table 3.3 also lists the total time spent on failed DEE/A* attempts across all seven unification parameter values. The total time represents a worst-case scenario where the user attempts DEE/A* with all seven C_{uni} values in a serial manner, only to find that DEE/A* runs out of memory each time. For most of the challenging designs, the total time for DEE/A* was on the order of 14 days.

BroMAP/A* solved 64, 62, 57, and 53 cases for e_{cut} values 0, 1, 3, and 5, re-

Table 3.3: Summary of test cases solved by BroMAP/A* that required greater than one month of running time. None of these cases were solved by DEE/A*. The total time for seven failed DEE/A* attempts, computed as the sum of time for DEE/A* to run out of memory for each C_{uni} value, is also listed for each case.

Test Case	e_{cut}	Total Time (days) for failed DEE/A* attempts
13	0	4.6
46	0	11.9
47	0	14.4
68	0	12.3
13	1	6.3
46	1	12.8
47	1	14.9
62	1	18.6
63	1	20.6
68	1	19.1
13	3	3.7
40	3	14.6
41	3	15.6
46	3	12.2
47	3	16.7
60	3	15.1
61	3	18.7
62	3	17.6
63	3	17.4
67	3	13.3
68	3	21.8
12	5	7.4
13	5	4.3
35	5	11.9
39	5	9.5
40	5	14.9
41	5	16.0
46	5	10.4
47	5	17.4
51	5	1.4
60	5	13.2
61	5	17.4
62	5	17.4
63	5	17.5
67	5	11.8
68	5	18.5

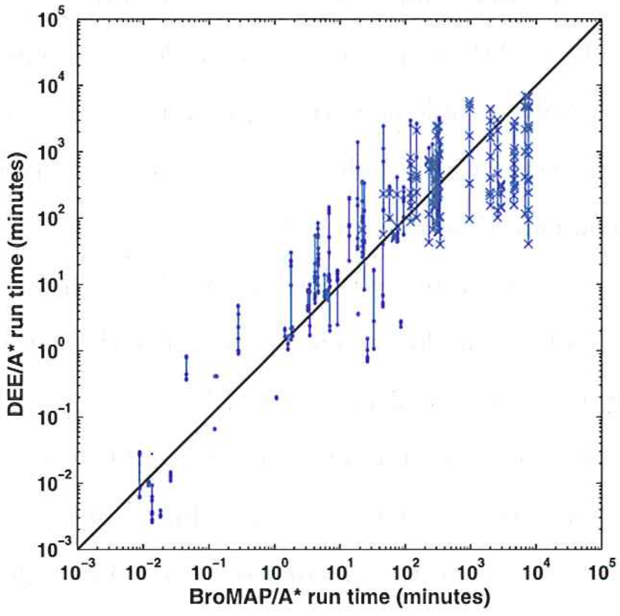
spectively, in less than one month. We focused on these cases for our more detailed comparison of BroMAP/A* and DEE/A* run time performance. For $e_{\text{cut}} = 0$ (i.e., no enumeration required), Figure 3-5(a) shows the range of DEE/A* run times across all seven C_{uni} values compared to BroMAP/A* for each case. DEE/A* run time varied significantly as a function of C_{uni}, spanning more than two orders of magnitude, for many cases. For a few cases that DEE/A* solved in less than 10 minutes, the run times were more tightly clustered. If DEE/A* solved a case for any C_{uni} value, then we plot the minimum run time in Figure 3-5(b). If DEE/A* failed to solve a case for every C_{uni} value, then we plot the total time for DEE/A* to run out memory in Figure 3-5(b), computed as the sum of run times for each C_{uni} value. In comparing BroMAP/A* to DEE/A*, the former condition for solved cases gives an advantage to DEE/A*. It assumes the user knows which of the seven C_{uni} values will solve a case in the shortest time. The latter condition for failed cases represents a user who must serially test all seven C_{uni} values with DEE/A* to find that it runs out of memory each time. Figures 3-6, 3-7, and 3-8 show the same comparisons for e_{cut} values 1, 3, and 5, respectively.

Among the cases solved by both methods, BroMAP/A* had similar run times to the best-case DEE/A* run times for all four e_{cut} values. As e_{cut} increased, DEE/A* was able to solve fewer cases within limited memory. In nearly all of the cases where DEE/A* failed, BroMAP/A* successfully enumerated the lowest energy conformations in less time than it took for DEE/A* to run out of memory across all seven C_{uni} values. These result points out an important caveat about using DEE/A*. DEE/A* is only guaranteed to return the ordered list if the A* search tree can fit in memory. For nontrivial designs, one cannot determine whether or not DEE/A* will fail without actually running it. Additionally, DEE/A* may not fail quickly. In many of our tests, DEE/A* ran for days or weeks before it failed. Our results show that for a user trying to solve one of these cases by repeatedly attempting DEE/A* with different parameters, that CPU time would be better spent on BroMAP/A*.

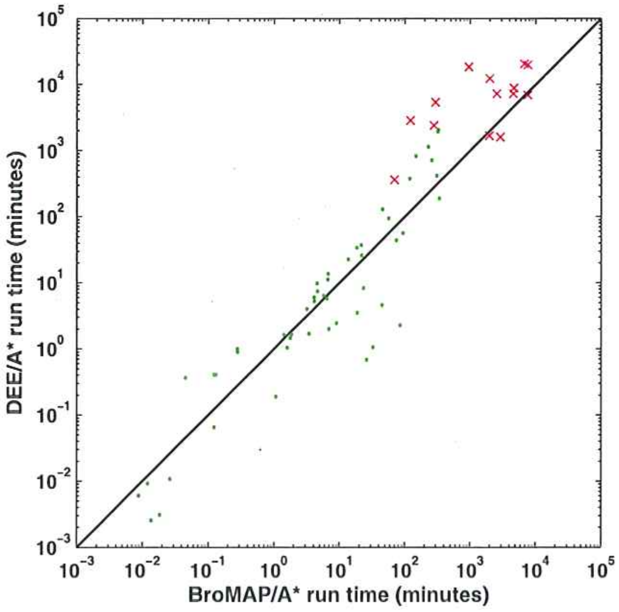
In summary, BroMAP/A* is an effective alternative to DEE/A* for enumerating protein design problems within limited computational resources. BroMAP/A* uses

TRMP in a branch-and-bound framework to divide complex problems into small, solvable subproblems. BroMAP/A* provides a valuable search capability for protein design. It allows one to solve problems with any amount of memory resources. This capability makes protein design accessible to new users who only have access to the resources found in a common desktop computer.

Additional work remains to improve BroMAP/A*. The efficiency of BroMAP/A* is driven by its ability to keep the lowest energy conformations in low child subproblems generated during the initial depth-first dive. For the cases where BroMAP/A* required more than one month of running time, it would be useful to analyze the distribution of these conformations among high and low nodes in the tree of subproblems. Understanding why low-energy conformations end up in high child subproblems would help us develop more accurate statistical bounds and better subproblem splitting techniques. These improvements would allow BroMAP/A* to efficiently solve more difficult protein design problems that cannot be solved DEE/A*.

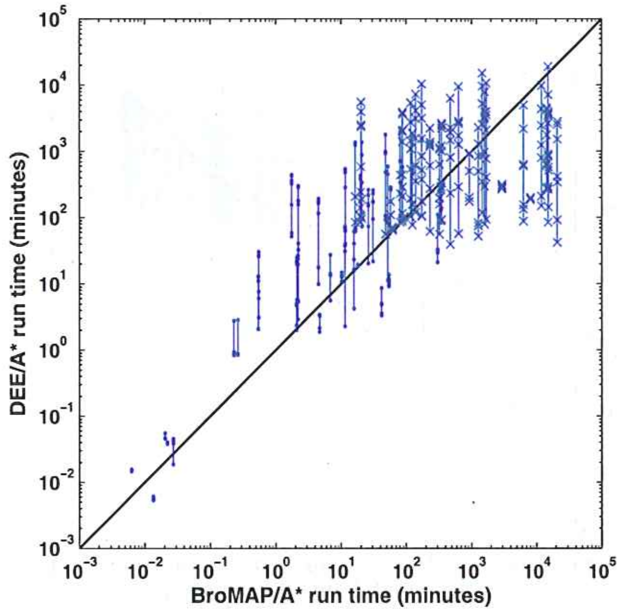


(a) Vertical lines show the range of DEE/A* run times across all seven C_{uni} values for each test case. For each C_{uni} value, a dot indicates DEE/A* solved the test case and an X indicates DEE/A* that failed.

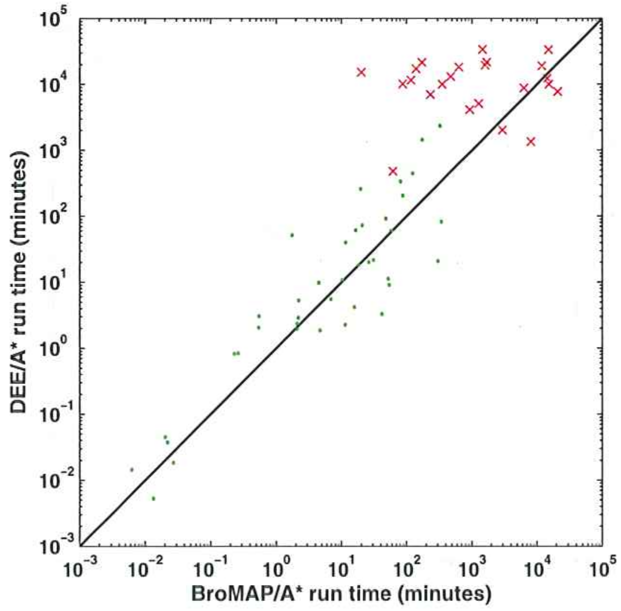


(b) For cases solved by DEE/A*, a dot shows the minimum running time among the seven C_{uni} values. For cases that DEE/A* failed to solve, an X shows the sum of run times for tests of all seven C_{uni} values.

Figure 3-5: For $e_{\text{cut}} = 0$, BroMAP/A* and DEE/A* run time performance for test cases solved in less than one month by BroMAP/A*. Cases 13, 46, 47, and 68 are not included (see Table 3.3 for details).

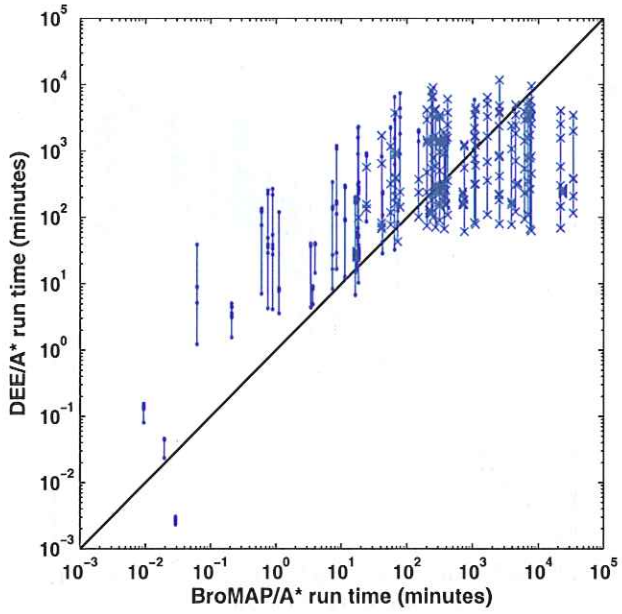


(a) Vertical lines show the range of DEE/A* run times across all seven C_{uni} values for each test case. For each C_{uni} value, a dot indicates DEE/A* solved the test case and an X indicates DEE/A* that failed.

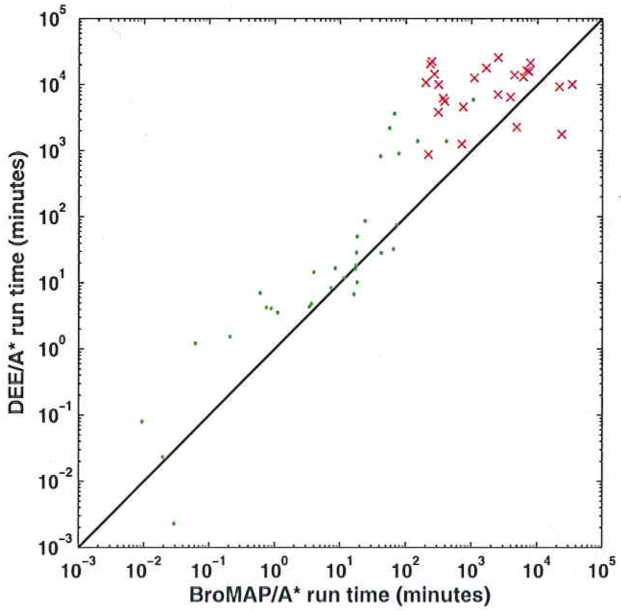


(b) For cases solved by DEE/A*, a dot shows the minimum running time among the seven C_{uni} values. For cases that DEE/A* failed to solve, an X shows the sum of run times for tests of all seven C_{uni} values.

Figure 3-6: For $e_{\text{cut}} = 1$, BroMAP/A* and DEE/A* run time performance for test cases solved in less than one month by BroMAP/A*. Cases 13, 46, 47, 62, 63, and 68 are not included (see Table 3.3 for details).

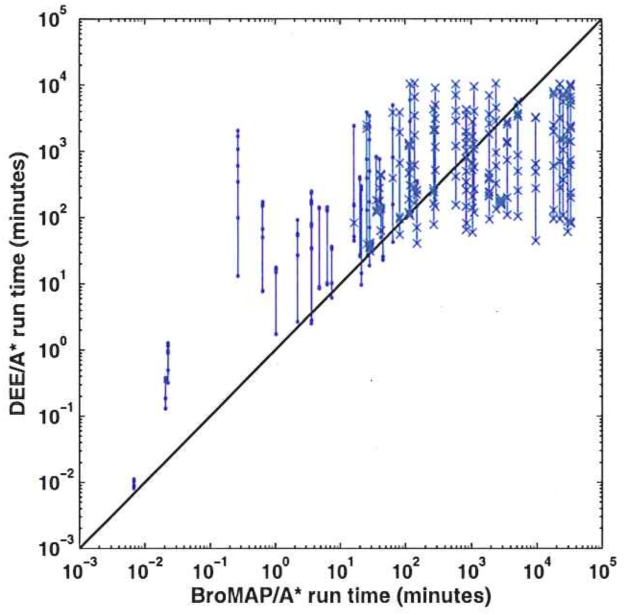


(a) Vertical lines show the range of DEE/A* run times across all seven C_{uni} values for each test case. For each C_{uni} value, a dot indicates DEE/A* solved the test case and an X indicates DEE/A* that failed.

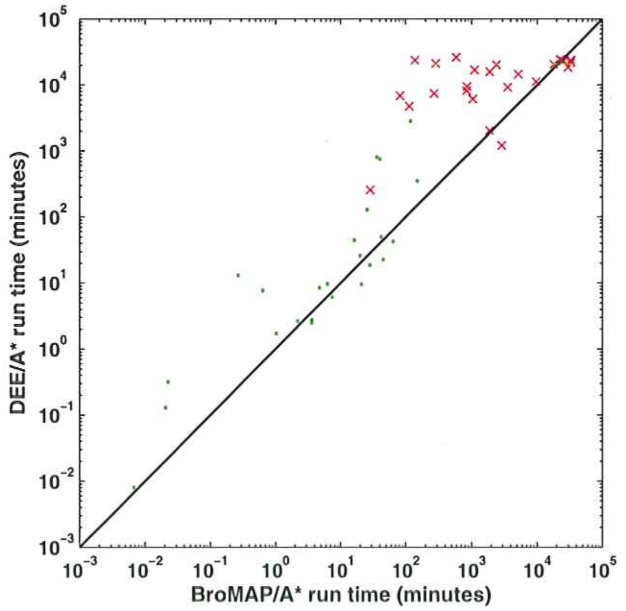


(b) For cases solved by DEE/A*, a dot shows the minimum running time among the seven C_{uni} values. For cases that DEE/A* failed to solve, an X shows the sum of run times for tests of all seven C_{uni} values.

Figure 3-7: For $e_{\text{cut}} = 3$, BroMAP/A* and DEE/A* run time performance for test cases solved in less than one month by BroMAP/A*. Cases 13, 40, 41, 46, 47, 60, 61, 62, 63, 67 and 68 are not included (see Table 3.3 for details).



(a) Vertical lines show the range of DEE/A* run times across all seven C_{uni} values for each test case. For each C_{uni} value, a dot indicates DEE/A* solved the test case and an X indicates DEE/A* that failed.



(b) For cases solved by DEE/A*, a dot shows the minimum running time among the seven C_{uni} values. For cases that DEE/A* failed to solve, an X shows the sum of run times for tests of all seven C_{uni} values.

Figure 3-8: For $e_{\mathrm{cut}} = 5$, BroMAP/A* and DEE/A* run time performance for test cases solved in less than one month by BroMAP/A*. Cases 12, 13, 35, 39, 40, 41, 46, 47, 51, 60, 61, 62, 63, 67, and 68 are not included (see Table 3.3 for details).

Chapter 4

General conclusions

In conclusion, this thesis presents novel methods and applications in computational protein design. In Chapter 2, we discuss how to improve the stability of stored proteins using computational protein design. Specifically, one can target cysteine, asparagine, glutamine, and methionine amino acid residues to reduce or eliminate a protein's susceptibility to degradation via aggregation, deamidation, and oxidation. One can also repack local regions of the protein to improve thermodynamic stability. We use computational protein design on the enzyme phosphotriesterase (PTE), and we provide predictions for a subset of the degradation-prone residues in PTE. We await results of experimental testing to evaluate the effectiveness of these particular computational designs. This evaluation will also inform future computational designs for improved protein storage stability.

Additionally, we point out a general need for experimental protocols that simulate long-term storage on much a shorter time scale. Consider protein degradation via deamidation, which may take years to occur. Suppose one has a protein with one asparagine residue and a redesigned, asparagine-free mutant. It's impractical to wait a year or more to compare the stability of the deamidated wild type protein to the mutant. By manipulating environmental factors such as buffer, temperature, pH, and ionic strength, one should be able to induce degradation processes that occur during long-term storage on the order of days or weeks. Reducing the time scale is critical for the ability to refine predictions through iterations in the protein design cycle.

In Chapter 3, we introduce BroMAP/A*, an exhaustive branch-and-bound rotamer search technique with enumeration. By efficiently dividing the search space into small, solvable subproblems, BroMAP/A* can successfully enumerate design problems for which DEE/A* fails. BroMAP/A* provides a "set it and forget it" capability for conformational search. Even though BroMAP/A* may take days or weeks of running time for a particular problem, it will return the ordered list of the lowest energy conformations upon termination. On the other hand, multiple DEE/A* attempts may be needed to find the unification heuristic or higher-order DEE criteria that produce a reduced search space that is solvable by A*. These failures can be frustrating to the end-user, especially if each DEE/A* attempt requires days or weeks of running time.

We have identified four main areas where BroMAP/A* can be improved. First, it would be useful if BroMAP/A* continuously reported an estimated time to completion as it progresses through the search tree. It's not immediately obvious how one might calculate this estimate, but the calculation could be based on information such as the current depth and branching of the search tree, distribution of solved subproblems in the tree, subproblem lower bounds, and conformational complexity of the remaining subproblems. The estimate need not be exact, and it's more useful for large designs. For example, an estimated time to completion of 6 months, even with error bars of ± 3 months, may drive the end-user to scale back the complexity of the design for the sake of forward progress.

Second, BroMAP/A* can be primed with a global upper bound computed by an inexpensive non-guaranteed search technique. If one starts with an upper bound close to the GMEC energy, then many rotamers and rotamer pairs can be eliminated by TRMP bounds starting with the very first subproblem. Overall, this should reduce both the evaluation cost and the number of subproblems. Non-guaranteed methods like Monte Carlo, greedy search, and genetic algorithms, can return reasonably tight upper bounds in significantly less time than it takes to solve a problem exactly. It would be useful to assess the run-time performance of BroMAP/A* as the initial upper bound approaches the GMEC energy. It would also be useful to analyze changes in the overall branch-and-bound search tree as the initial upper bound improves.

Third, BroMAP/A* could benefit from an alternative subproblem splitting criterion. BroMAP/A* currently uses the same subproblem splitting method as BroMAP. After a design position is chosen for splitting, the rotamers at that position are distributed equally to the two child subproblems. When solving for the GMEC with BroMAP, this 50/50 split is sensible. It balances the need to create low child subproblems that contain rotamers that are more likely to belong to the GMEC with the need to quickly create solvable subproblems. However, it's not clear that a 50/50 split is the best choice for enumeration with BroMAP/A*. With enumeration, it may be advantageous to distribute rotamers to the low child subproblem based on the energy threshold, e_{ecut}, for the ordered list. For example, the low child subproblem could contain any rotamer at the split position whose lower bound is within e_{ecut} of the minimum lower bound. As e_{ecut} increases, this criterion would create larger low child subproblems. However, the smaller high child subproblems would be less likely to contain rotamers in the ordered list. In turn, high child subproblems would be more easily eliminated by TRMP bounds. Further analysis is needed to determine if cost of solving larger low child subproblems is lower than the cost of having to solve many high child subproblems for a few conformations using the current 50/50 split method.

Fourth, we point out that the branch-and-bound framework and the independence of subproblems in BroMAP/A* allows for parallel processing of high child subproblems. After the first subproblem is solved exactly in the depth-first dive, BroMAP/A* could begin processing multiple high child subproblems, starting from the deepest levels of the search tree. Parallel BroMAP/A* would be most effective when the first subproblem solved during in the depth-first dive contains the GMEC. When the GMEC is in a subproblem somewhere in the middle of the branch-and-bound tree, parallel BroMAP/A* may end up expanding high child subproblems that would not have been expanded by serial BroMAP/A*. However, the additional cost is distributed among multiple threads. The most important performance measure for the end-user is wall time. If the user has access to many processors/cores, parallel BroMAP/A* should decrease the wall time required to solve a design.

To conclude, we note that the energy functions used to evaluate protein models are continually being improved. Looking ahead, protein models will more accurately reflect nature, and computational design will become more widely adopted as a means for developing novel proteins. As researchers try to solve new and more complex problems using computational design, there will be an increased need for efficient and effective search methods such as BroMAP/A*.

Appendix A

Energy tables for protein design predictions

Here we provide the full set of energy terms from low-resolution and high-resolution evaluation of the mutant structures discussed in Chapter 2. The objective function for each conformation, $\Delta G_{\text{folding}}$, was the energy difference between the folded and unfolded state assuming a fixed backbone. For the unfolded state, we treated each side chain as being infinitely separated. The total energy of the unfolded state was the sum of energies from these isolated side chain compounds. Let $\Delta G^{\text{wt}}_{\text{folding}}$ and $\Delta G^{\text{mut}}_{\text{folding}}$ represent the energy difference between the folded and unfolded state for the wild-type and mutant conformations, respectively. The folding stability of the mutant conformation relative to wild type for each energy term was calculated as follows:

$$\Delta\Delta G = \Delta G^{\text{wt}}_{\text{folding}} - \Delta G^{\text{mut}}_{\text{folding}}. \tag{A.1}$$

For high-resolution evaluation, we used a Poisson–Boltzmann continuum electrostatics model with two different implicit solvation models. One calculation captured the nonpolar component of the solvation free energy with a solvent-accessible surface area (SASA) term,

$$\Delta\Delta G^{\text{SASA}}_{\text{total}} = \Delta\Delta G_{\text{vdw}} + \Delta\Delta G_{\text{geo}} + \Delta\Delta G_{\text{elec}} + \Delta\Delta G_{\text{sasa}}. \tag{A.2}$$

Table A.1: Candidate mutations for Cys59.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+4.60	−2.37	+0.25	+0.15	−0.86	−2.49	+2.62	−0.88
Asn	−0.40	+1.45	+1.52	−0.09	+0.21	−0.63	+2.48	+2.15
Gly	+6.98	−2.83	+0.40	+0.38	−1.59	+5.65	+4.93	+8.61
Ser	+2.72	+0.82	+0.83	+0.10	−0.45	+1.79	+4.46	+5.70

Table A.2: Candidate mutations for Cys227.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+2.84	−2.51	−1.51	+0.19	−1.41	−0.94	−1.00	−3.54
Gly	+5.47	−1.98	−1.79	+0.37	−1.85	+3.96	+2.06	+3.81
Leu	+17.66	−1.87	+1.34	−0.29	+0.91	−3.61	+16.84	+14.43
Ser	+2.47	−0.97	−0.65	+0.11	−0.87	+1.72	+0.96	+1.70
Val	+46.97	−1.56	−2.04	−0.11	+0.14	−0.78	+43.25	+42.72

The other calculation separately considered cavity formation and solute-solvent van der Waals interactions,

$$\Delta\Delta G_{\mathrm{total}}^{\mathrm{CVDW}} = \Delta\Delta G_{\mathrm{vdw}} + \Delta\Delta G_{\mathrm{geo}} + \Delta\Delta G_{\mathrm{elec}} + \Delta\Delta G_{\mathrm{cvdw}} + \Delta\Delta G_{\mathrm{cav}}. \qquad (A.3)$$

Table A.3: Candidate mutations for Met314.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+10.97	−3.24	−2.84	+0.53	−2.16	+4.09	+5.43	+6.83
Arg	+7.28	+0.23	+0.33	−0.01	+3.00	+1.58	+7.83	+12.42
Asp	+13.58	−0.66	−6.37	+0.99	+0.38	+7.10	+7.54	+14.03
Gln	+1.46	−1.21	+0.29	+0.09	−0.48	+0.55	+0.63	+0.61
Gly	+12.84	−1.06	−4.22	+0.77	−2.75	+8.95	+8.33	+13.77
Hsd	+5.70	−1.34	+0.76	+0.13	+2.01	+3.15	+5.25	+10.28
Ile	+6.40	−0.92	+0.60	+0.06	−0.06	+0.40	+6.14	+6.42
Leu	+8.15	−0.07	−3.04	+0.13	+1.32	+1.45	+5.17	+7.82
Lys	+10.86	−1.11	+1.29	−0.11	+2.94	+1.58	+10.93	+15.56
Phe	+6.92	−0.79	+0.85	+0.53	+3.83	+3.16	+7.50	+13.97
Ser	+11.21	−1.27	−2.27	+0.53	−1.83	+5.42	+8.19	+11.25
Thr	+9.88	−1.31	−0.40	+0.29	−0.65	+3.88	+8.46	+11.41
Trp	+1.80	−1.36	+0.11	+0.27	+4.88	+1.10	+0.82	+6.52
Tyr	+6.06	−0.88	+1.31	+0.46	+4.40	+2.78	+6.96	+13.67
Val	+11.34	−0.92	−0.04	+0.19	−0.82	+3.08	+10.57	+12.64

Table A.4: Candidate mutations for Asn38.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+3.22	−1.94	−1.80	+0.41	−3.53	+0.19	−0.11	−3.86
Gln	−3.33	+0.19	+0.65	−0.20	−0.29	+0.30	−2.69	−2.48
Gly	+5.90	−0.63	−2.27	+0.61	−4.60	+4.88	+3.62	+3.29
Hse	−2.49	−0.03	+0.72	−0.25	+1.59	−1.66	−2.06	−1.88
Ile	+9.86	+1.70	+0.22	−0.17	−0.12	−0.96	+11.61	+10.70
Leu	−0.88	+1.44	−0.13	−0.24	+0.01	−1.72	+0.19	−1.28
Met	−4.57	+0.15	−0.08	−0.29	+0.34	−0.06	−4.79	−4.21
Phe	−5.14	+0.34	+1.27	−0.46	+2.86	−3.48	−3.99	−4.15
Ser	+2.06	−0.30	−0.80	+0.34	−2.07	+3.01	+1.30	+1.91
Thr	+0.54	−0.23	+0.42	+0.14	−1.25	+1.67	+0.87	+1.15
Tyr	−5.89	+0.23	+1.17	−0.50	+3.39	−3.80	−4.99	−4.90
Val	+5.59	+0.24	−0.77	+0.02	−0.99	+0.59	+5.08	+4.66

Table A.5: Candidate mutations for Asn265.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+2.74	−1.64	−1.02	+0.29	−1.28	−1.36	+0.36	−2.57
Arg	−2.94	−0.40	+1.19	−0.02	+2.17	−0.04	−2.17	−0.02
Asp	+0.76	+0.17	+0.09	+0.16	+0.11	+0.96	+1.18	+2.09
Gln	−2.63	+0.10	+1.92	−0.09	+0.09	−0.69	−0.70	−1.20
Gly	+4.79	−1.34	−1.35	+0.49	−2.22	+3.50	+2.58	+3.37
Hsp	−0.03	+0.27	−0.80	+0.26	+1.64	+2.05	−0.30	+3.12
Ile	−0.51	−0.07	+0.74	−0.08	+0.54	−0.56	+0.07	+0.14
Leu	−0.80	+0.24	−0.14	−0.09	+0.16	−0.67	−0.79	−1.20
Lys	+1.82	+0.47	−2.60	+0.20	+1.95	+1.78	−0.11	+3.42
Met	−3.05	+0.29	+0.53	−0.17	+0.65	−1.10	−2.39	−2.67
Phe	−1.56	+0.01	−0.25	+0.01	+3.10	−0.11	−1.79	+1.19
Ser	+2.18	−0.15	+0.14	+0.22	−0.63	+1.52	+2.39	+3.06
Thr	+0.21	−0.20	+0.88	+0.09	−0.10	+0.56	+0.98	+1.35
Trp	−3.09	+0.10	+0.70	−0.10	+5.08	−0.24	−2.40	+2.55
Tyr	−2.16	+0.45	−0.22	+0.03	+3.65	+0.16	−1.91	+1.87
Val	+0.06	−0.01	+0.49	−0.02	−0.15	−0.12	+0.53	+0.29

Table A.6: Candidate mutations for Leu262 in the background of Asn265Lys.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+3.72	−1.72	−1.49	+0.50	+0.13	+0.91	+1.02	+1.55
Arg	−2.25	+0.31	+1.14	−0.13	+4.47	−0.34	−0.94	+3.33
Gln	−0.10	+0.34	+1.04	−0.08	+1.58	−0.19	+1.20	+2.66
Glu	+1.33	+0.72	+0.89	−0.19	+0.63	−0.88	+2.76	+2.69
Gly	+5.00	−2.37	−1.18	+0.75	−0.82	+7.11	+2.21	+7.74
Hsd	−2.21	+0.13	+0.51	−0.10	+2.11	−0.85	−1.68	−0.31
Ile	+27.99	+0.54	−1.33	+0.09	+1.56	+1.27	+27.30	+30.05
Lys	+1.94	+0.59	−1.04	+0.02	+4.41	+1.01	+1.52	+6.92
Phe	−0.65	+0.73	−2.00	−0.00	+4.60	+0.03	−1.92	+2.71
Ser	+2.68	+1.53	−0.70	+0.42	+0.72	+3.57	+3.92	+7.79
Thr	+2.08	+0.29	+0.07	+0.29	+1.00	+2.70	+2.74	+6.16
Trp	−3.33	+0.23	−0.64	−0.28	+5.24	−1.74	−4.02	−0.23
Tyr	−3.45	+0.24	+0.60	−0.40	+3.66	−2.76	−3.01	−1.71
Val	+37.83	+0.47	−1.21	+0.20	+0.99	+2.17	+37.29	+40.25

Table A.7: Candidate mutations for Val316 in the background of Asn265Lys.

Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Ala	+2.99	−0.18	−1.01	+0.18	−0.29	−1.59	+1.98	−0.08
Asp	−0.32	+2.16	+0.43	−0.12	+1.84	−0.60	+2.15	+3.50
Glu	−3.00	+1.99	+1.81	−0.28	+1.59	−1.56	+0.53	+0.83
Gly	+5.36	+2.15	−1.20	+0.39	−1.21	+3.46	+6.71	+8.57
Hse	−1.61	+2.05	+0.18	−0.38	+2.68	−0.81	+0.24	+2.48
Ile	−2.44	+1.66	+0.37	−0.37	+2.48	−2.84	−0.78	−0.78
Leu	−1.15	+2.66	−1.89	−0.11	+2.99	−0.86	−0.48	+1.76
Phe	−0.28	+1.07	−1.02	−0.22	+5.30	−1.79	−0.46	+3.27
Ser	+1.21	+1.42	+0.12	+0.08	+0.89	+1.01	+2.83	+4.64
Thr	−0.62	+1.37	+0.46	−0.02	+0.88	+0.20	+1.20	+2.31
Trp	−6.04	+2.06	+0.91	−0.44	+6.50	−3.71	−3.52	−0.28
Tyr	−1.33	+1.05	−0.84	−0.28	+5.70	−2.02	−1.40	+2.55

Table A.8: Candidate mutations for Leu262 and Val316 in the background of Asn265Lys.

Leu262 Mutant	Val316 Mutant	VDW	GEO	ELEC	SASA	CVDW	CAV	SASA total	CVDW total
Hsd	Ile	−3.86	+1.83	+0.82	−0.39	+3.44	−3.15	−1.59	−0.91
Hsd	Leu	−1.75	+1.66	+1.17	−0.42	+3.36	−1.93	+0.67	+2.51
Hsd	Phe	−3.40	+0.91	+1.87	−0.54	+6.00	−3.16	−1.16	+2.22
Hsd	Trp	−10.02	+1.71	+3.72	−0.70	+7.33	−4.92	−5.29	−2.18
Hsd	Tyr	−4.49	+0.89	+2.06	−0.59	+6.37	−3.41	−2.13	+1.42
Phe	Ile	−4.81	+1.79	+0.87	−0.54	+4.44	−4.54	−2.69	−2.24
Phe	Leu	−5.89	+3.65	+1.51	−0.36	+3.58	−2.80	−1.09	+0.05
Phe	Phe	−4.52	+0.86	+1.67	−0.68	+6.94	−4.74	−2.68	+0.21
Phe	Trp	−10.46	+1.86	+3.59	−0.94	+7.96	−6.77	−5.95	−3.83
Phe	Tyr	−5.60	+0.85	+1.86	−0.74	+7.32	−5.00	−3.64	−0.57
Trp	Ile	−8.99	+2.80	+3.44	−0.71	+5.52	−5.74	−3.47	−2.97
Trp	Leu	−9.93	+3.16	+2.85	−0.59	+3.72	−4.27	−4.51	−4.47
Trp	Phe	−5.47	+0.82	+0.65	−0.68	+8.74	−5.81	−4.69	−1.07
Trp	Trp	−11.23	+1.82	+2.57	−0.91	+9.88	−7.61	−7.75	−4.57
Trp	Tyr	−6.52	+0.81	+0.80	−0.74	+9.15	−6.05	−5.65	−1.82
Tyr	Ile	−6.04	+1.75	+0.89	−0.61	+4.81	−4.94	−4.00	−3.52
Tyr	Leu	−7.33	+4.04	+1.93	−0.45	+3.35	−3.29	−1.80	−1.30
Tyr	Phe	−5.76	+0.83	+1.72	−0.75	+7.28	−5.15	−3.96	−1.07
Tyr	Trp	−11.70	+1.82	+3.65	−1.00	+8.31	−7.18	−7.23	−5.10
Tyr	Tyr	−6.84	+0.81	+1.92	−0.81	+7.67	−5.41	−4.91	−1.85

Bibliography

[1] Lippow, S. M.; Wittrup, K. D. and Tidor, B., *Nat Biotechnol*, 2007, **25**, 1171–1176.

[2] Chen, C.; Georgiev, I.; Anderson, A. C. and Donald, B. A., *Proc Natl Acad Sci USA*, 2009, **106**, 3764–3769.

[3] Dahiyat, B. I. and Mayo, S. L., *Science*, 1997, **278**, 82–87.

[4] Kuhlman, B.; Dantas, G.; Ireton, G. C.; Varani, G.; Stoddard, B. L. and Baker, D., *Science*, 203, **302**, 1364–1368.

[5] Lippow, S. M. and Tidor, B., *Curr Opin Biotech*, 2007, **18**, 305–311.

[6] Branden, C. and Tooze, J., In *Introduction to Protein Structure*, Garland Publishing, New York, NY, 1999; chapter 1, pages 3–12; 2nd ed.

[7] J. Desmet, M. de Maeyer, B. H. and Lasters, I., *Nature*, 1992, **356**, 539–542.

[8] Goldstein, R. F., *Biophys J*, 1994, **66**, 1335–1340.

[9] Pierce, N. A.; Spriet, J. A.; Desmet, J. and Mayo, S. L., *J Comput Chem*, 2000, **21**, 999–1009.

[10] Leach, A. R. and Lemon, A. P., *Proteins*, 2007, **25**, 1171–1176.

[11] Hong, E.; Lippow, S. M.; Tidor, B. and Pérez, T. L., *J Comput Chem*, 2009, **30**, 1923–1945.

[12] Kingsford, C. L.; Chazelle, B. and Singh, M., *Bioinformatics*, 2005, **21**, 1028–1036.

[13] Koehl, P. and Delarue, M., *J Mol Biol*, 1994, **239**, 249–275.

[14] Pokala, N. and Handel, T. M., *J Mol Biol*, 2005, **347**, 203–227.

[15] Dahiyat, B. I. and Mayo, S. L., *Protein Sci*, 1996, **5**, 895–903.

[16] Shahrokh, Z., In *Therapeutic Protein and Peptide Formulation and Delivery*, Shahrokh, Z.; Sluzky, V.; Cleland, J. L.; Shire, S. J. and Randolph, T. W., Eds., American Chemical Society, 1997; chapter 1, pages 1–28.

[17] Sluzky, V.; Klibanov, A. M. and Langer, R., *Biotechnol Bioeng*, 2004, **40**, 895–903.

[18] Wang, W., *Int J Pharm*, 2000, **203**, 1–60.

[19] Carpenter, J. F.; Pikal, M. J.; Chang, B. S. and Randolph, T. W., *Pharm Res*, 1997, **14**, 969–975.

[20] Carpenter, J. F.; Chang, B. S.; Garzon-Rodriguez, W. and Randolph, T. W., *Pharm Biotechnol*, 2002, **13**, 103–133.

[21] Lai, M. C. and Topp, E. M., *J Pharm Sci*, 1999, **88**, 489–500.

[22] Robinson, N. E. and Robinson, A. B., In *Molecular Clocks: Demidation of Asparaginyl and Glutaminyl Residues in Peptides and Proteins*, Althouse Press, Cave Junction, OR, 2004; chapter 6, pages 51–95.

[23] Capasso, S., *J Peptide Res*, 2000, **55**, 224–229.

[24] Robinson, N. E. and Robinson, A. B., *J Peptide Res*, 2004, **63**, 437–448.

[25] Robinson, N. E. and Robinson, A. B., *Proc Natl Acad Sci USA*, 2001, **98**, 4367–4372.

[26] Robinson, N. E., *Proc Natl Acad Sci USA*, 2002, **99**, 5283–5288.

[27] Robinson, N. E. and Robinson, A. B., In *Molecular Clocks: Demidation of Asparaginyl and Glutaminyl Residues in Peptides and Proteins*, Althouse Press, Cave Junction, OR, 2004; chapter 8, pages 103–116.

[28] Johnson, O. L., In *Protein Formulation and Delivery*, McNally, E. J., Ed., Marcel Dekker, New York, NY, 2000; chapter 8, pages 237–239; 1st ed.

[29] Fransson, J.; Florin-Robertsson, E.; Axelsson, K. and Nyhlen, C., *Pharm Res*, 1996, **13**, 1252–1257.

[30] Kim, Y. H.; Berry, A. H.; Spencer, D. S. and Stites, W. E., *Protein Eng*, 2001, **14**, 343–347.

[31] Seibert, C. M. and Raushel, F. M., *Biochemistry*, 2005, **44**, 6383–6391.

[32] Sethunathan, N. and Yoshida, T., *Can J Microbiol*, 1973, **19**, 873–875.

[33] Munnecke, D. M., *Appl Environ Microb*, 1976, **32**, 7–13.

[34] Dumas, D. P.; Caldwell, S. R.; Wild, J. R. and Raushel, F. M., *J Biol Chem*, 1989, **264**, 19659–19665.

[35] Donarski, W. J.; Dumas, D. P.; Heitmeyer, D. H.; Lewis, V. H. and Raushel, F. M., *Biochemistry*, 1989, **28**, 4650–4655.

[36] Benning, M. M.; Kuo, J. M.; Raushel, F. M. and Holden, H. M., *Biochemistry*, 1994, **33**, 15001–15007.

[37] Omburo, G. A.; Kuo, J. M.; Mullins, L. S. and Raushel, F. M., *J Biol Chem*, 1992, **267**, 13278–13283.

[38] Raushel, F. M., *Curr Opin Microbiol*, 2002, **5**, 288–295.

[39] Aubert, S. D.; Yinchun, L. and Raushel, F. M., *Biochemistry*, 2004, **43**, 5707–5715.

[40] Schrader, G., *Angew Chem*, 1950, **62**, 471–473.

[41] Dumas, D. P.; Durst, H. D.; Landis, W. G.; Raushel, F. M. and Wild, J. R., *Arch Biochem Biophys*, 1990, **277**, 155–159.

[42] Kolakowski, J. E.; DeFrank, J. J.; Harvey, S. P.; Szafraniec, L. L.; Beaudry, W. T.; Lai, K. and Wild, J. R., *Biocatal Biotransform*, 1997, **15**, 297–312.

[43] Efremenko, E. M. and Sergeeva, V. S., *Russ Chem B+*, 2001, **50**, 1826–1832.

[44] LeJeune, K. E.; Dravis, B. C.; Yang, F.; Hetro, A. D.; Doctor, B. P. and Russell, A. J., *Ann N Y Acad Sci*, 1998, **864**, 153–170.

[45] Russell, A. J.; Berberich, J. A.; Drevon, G. F. and Koepsel, R. R., *Annu Rev Biomed Eng*, 2003, **5**, 1–27.

[46] Ang, E. L.; Zhao, H. and Obbard, J. P., *Enzyme Microb Tech*, 2005, **37**, 486–496.

[47] Ghanem, E. and Raushel, F. M., *Toxicol Appl Pharm*, 2005, **207**, 459–470.

[48] Singh, S.; Kang, S. H.; Mulchandani, A. and Chen, W., *Curr Opin Biotech*, 2008, **19**, 437–444.

[49] LeJeune, K. E. and Russell, A. J., *Biotechnol Bioeng*, 1999, **62**, 659–665.

[50] Yang, F.; Wild, J. R. and Russell, A. J., *Biotechnol Prog*, 1995, **11**, 471–474.

[51] White, B. J. and Harmon, H. J., *Biosens Bioelectron*, 2005, **20**, 1977–1983.

[52] Cho, C. M.; Mulchandani, A. and Chen, W., *Protein Eng Des Sel*, 2006, **19**, 99–105.

[53] Gopal, S.; Rastogi, V.; Ashman, W. and Mulbry, W., *Biochem Bioph Res Co*, 2000, **279**, 516–519.

[54] Hill, C. M.; Li, W.; Thoden, J. B.; Holden, H. M. and Raushel, F. M., *J Am Chem Soc*, 2003, **125**, 8990–8991.

[55] Roodveldt, C. and Tawfik, D. S., *Protein Eng Des Sel*, 2005, **18**, 51–58.

[56] Vanhooke, J. L.; Benning, M. M.; Raushel, F. M. and Holden, H. M., *Biochemistry*, 1996, **35**, 6020–6025.

[57] Benning, M. M.; Hong, S. B.; Raushel, F. M. and Holden, H. M., *J Biol Chem*, 2000, **275**, 30556–30560.

[58] Benning, M. M.; Shim, H.; Raushel, F. M. and Holden, H. M., *Biochemistry*, 2001, **40**, 2712–2722.

[59] Brooks, B. R.; Bruccoleri, R. E.; Olafson, D. J.; States, D. J.; Swaminathan, S. and Karplus, M., *J Comput Chem*, 1983, **4**, 187–217.

[60] Brooks, B. R.; Brooks, III, C. L.; Mackerell, Jr., A. D.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M. and Karplus, M., *J Comput Chem*, 2009, **30**, 1545–1614.

[61] MacKerell, Jr., A. D.; Bashford, D.; Bellott, M.; Dunbrack Jr., R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D. T.; Prodhom, B.; Reiher, III, W. E.; Roux, B.; Schlenkrich, M.; Smith, J.; Stote, R.; Straub, J.; Watanabe, M.; Wiorkiewicz-Kuczera, J.; Yin, D. and Karplus, M., *J Phys Chem B*, 1998, **102**, 3586–3616.

[62] MacKerell, J. A. D.; Feig, M. and Brooks, C. L., I., *J Comput Chem*, 2004, **25**, 1400–1415.

[63] Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Montgomery, Jr., J. A.; Vreven, T.; Kudin, K. N.; Burant, J. C.; Millam, J. M.; Iyengar, S. S.; Tomasi, J.; Barone, V.; Mennucci, B.; Cossi, M.; Scalmani, G.; Rega, N.; Petersson, G. A.; Nakatsuji, H.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Klene, M.; Li, X.; Knox, J. E.; Hratchian, H. P.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Ayala, P. Y.; Morokuma, K.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Zakrzewski, V. G.; Dapprich, S.; Daniels, A. D.; Strain, M. C.; Farkas, O.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Ortiz, J. V.; Cui, Q.; Baboul, A. G.; Clifford, S.; Cioslowski, J.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Challacombe, M.; Gill, P. M. W.; Johnson, B.; Chen, W.; Wong, M. W.; Gonzalez, C. and Pople, J. A., Gaussian 03, Gaussian, Inc., Wallingford, CT, 2004.

[64] Dunbrack, J. R. L. and Cohen, F. E., *Protein Sci*, 1997, **6**, 1661–1681.

[65] Sitkoff, D.; Sharp, K. A. and Honig, B., *J Phys Chem*, 1994, **98**, 1978–1988.

[66] Sharp, K. A. and Honig, B., *Ann Rev Biophys Bio*, 1990, **19**, 301–332.

[67] Sanner, M. F.; Olson, A. J. and Spehner, J. C., *Biopolymers*, 1996, **38**, 305–320.

[68] Levy, R. M.; Zhang, L. Y.; Gallicchio, E. and Felts, A. K., *J Am Chem Soc*, 2003, **125**, 9523–9530.

[69] Robinson, N. E. and Robinson, A. B., In *Molecular Clocks: Demidation of Asparaginyl and Glutaminyl Residues in Peptides and Proteins*, Althouse Press, Cave Junction, OR, 2004; chapter 6, pages 51–95.

[70] Gordon, D. B. and Mayo, S. L., *Structure*, 1999, **7**, 1089–1098.

[71] Safi, M. and Lilien, R. H., *J Comput Chem*, 2009, **31**, 1207–1215.

[72] Lasters, I.; de Maeyer, M. and Desmet, J., *Protein Eng*, 1995, **8**, 815–822.

[73] Gordon, D. B. and Mayo, S. L., *J Comput Chem*, 1998, **19**, 1505–1514.

[74] Korf, R. E., *Artif Intell*, 1985, **27**, 97–109.

[75] Georgiev, I.; Lilien, R. H. and Donald, B. R., *Bioinformatics*, 2006, **22**, 174–183.

[76] Pierce, N. A. and Winfree, E., *Protein Eng*, 2002, **15**, 779–782.

[77] Wainwright, M. J.; Jaakkola, T. S. and Willsky, A. S., *IEEE Trans Inform Theory*, 2005, **51**, 3697–3717.

[78] Kolmogorov, V., *IEEE Trans Pattern Anal*, 2006, **28**, 1568–1583.

[79] Altman, M. D. *Computational ligand design and analysis in protein complexes using inverse models, combinatorial search, and accurate solvation modeling* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

[80] Eckstein, J.; Phillips, C. A. and Hart, W. E. In *Proc Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, pages 219–265. Elsevier Scientific Series on Studies in Computational Mathematics, 2001.