Protein and Drug Design Algorithms Using Improved Biophysical Modeling

by

Mark A. Hallen

Department of Computer Science
Duke University

Date: __________________________

Approved: __________________________

Bruce R. Donald, Supervisor

Carlo Tomasi

Mauro Maggioni

Jane S. Richardson

David N. Beratan

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2016
Abstract

Protein and Drug Design Algorithms Using Improved Biophysical Modeling

by

Mark A. Hallen

Department of Computer Science
Duke University

Date: ____________________
Approved: ____________________

___________________________
Bruce R. Donald, Supervisor

___________________________
Carlo Tomasi

___________________________
Mauro Maggioni

___________________________
Jane S. Richardson

___________________________
David N. Beratan

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University 2016
Abstract

This thesis focuses on the development of algorithms that will allow protein design calculations to incorporate more realistic modeling assumptions. Protein design algorithms search large sequence spaces for protein sequences that are biologically and medically useful. Better modeling could improve the chance of success in designs and expand the range of problems to which these algorithms are applied. I have developed algorithms to improve modeling of backbone flexibility (DEEPer) and of more extensive continuous flexibility in general (EPIC and LUTE). I’ve also developed algorithms to perform multistate designs, which account for effects like specificity, with provable guarantees of accuracy (COMETS), and to accommodate a wider range of energy functions in design (EPIC and LUTE).
Dedicated to Ying
Contents

Abstract iv
List of Tables xi
List of Figures xiv
Acknowledgements xxix

1 Introduction 1
  1.1 The voxel paradigm ........................................ 3
    1.1.1 Overview ........................................... 3
    1.1.2 Previous algorithms based on the voxel paradigm ...... 5
    1.1.3 OSPREY and its successes ................................ 6
    1.1.4 Alternative paradigms for protein design ............... 7
  1.2 Backbone flexibility .......................................... 8
  1.3 Interactions with solvent ...................................... 9
  1.4 Quantum-chemical energy modeling ............................. 11
  1.5 The legend of Perseus ......................................... 13
  1.6 Outline of thesis ............................................. 14

2 Dead-End Elimination with Perturbations (“DEEPer”):
   A provable protein design algorithm with continuous sidechain and
   backbone flexibility 17
  2.1 Introduction ................................................. 19
  2.2 Methods .................................................. 26
2.2.1 Perturbations ............................................. 27
2.2.2 Types of perturbations ................................. 28
2.2.3 Implementation and selection of perturbations ........... 34
2.2.4 Residue conformations (RCs) .......................... 36
2.2.5 Indirect pruning ........................................ 41
2.2.6 Choice of pruning zones ................................. 52
2.2.7 Proof of single-RC pruning condition ................. 54
2.2.8 Proof of tuples pruning condition ..................... 57
2.2.9 Tests and validation .................................... 60

2.3 Results and Discussion .................................. 61
2.3.1 Comparisons to previous algorithms with less flexibility .... 61
2.3.2 Tests of larger backbone conformational changes .......... 67
2.3.3 Tests of ensemble generation ........................... 69
2.3.4 Modeling entropy ....................................... 70
2.3.5 Summary ................................................ 73

2.4 Conclusions ................................................. 74

3 EPIC (Energy as Polynomials in Internal Coordinates): A compact representation of continuous energy surfaces for more efficient protein design 76
3.1 Introduction .................................................. 77
3.2 Methods ..................................................... 84
3.2.1 Preliminaries ........................................... 84
3.2.2 Basic least-squares method ............................. 87
3.2.3 Modified least-squares method ......................... 89
3.2.4 A fast algorithm for modified least-squares fitting ....... 93
3.2.5 Sparse atom-pair energies (SAPE) ..................... 96
4.4 Conclusions ................................................................. 150

5 Fast gap-free enumeration of conformations and sequences for protein design .......................... 151

5.1 Methods ................................................................. 152

5.1.1 Background: The A* algorithm for the protein design problem ........................................ 152

5.1.2 Ordered A* trees ...................................................... 154

5.1.3 Benchmarking methods ............................................ 159

5.2 Results ................................................................. 161

5.2.1 Variable ordering ................................................... 161

6 Crystal structure, conformational fixation and entry-related interactions of mature ligand-free HIV-1 Env ................................................................. 166

6.1 Methods ................................................................. 168

6.1.1 Computation of antibody-epitope r.m.s. deviation, volume overlap and epitope presence. .... 168

6.1.2 Structural compatibility analysis. ................................ 169

6.2 Results ................................................................. 170

6.2.1 Structural compatibility and appropriate target conformation ............................................ 170

6.2.2 Appropriate target antigenicity for an Env-vaccine antigen .............................................. 172

7 LUTE (Local Unpruned Tuple Expansion): Accurate continuously flexible protein design with general energy functions and rigid-rotamer-like efficiency ........................................ 175

7.1 Introduction ............................................................. 177

7.2 Methods ................................................................. 182

7.2.1 The discrete expansion ............................................ 182

7.2.2 Calculation of expansion by least squares ................................................................. 186

7.2.3 Selection of tuples to include .................................... 187

7.2.4 Conformational and sequence search with LUTE ....................................................... 188
List of Tables

2.1 Continuous flexibility in DEE-based protein design algorithms . . . . . 24
2.2 Types of perturbations ................................................. 30
2.3 Pairwise energies for RCs in the toy example. Pruning of RC \(i_r\) using \(i_t\) will be attempted, with pruning zone \(Z = \{i, j\}\) ........................................... 52

3.1 GMEC calculation test cases. EPIC with SAPE solved each of these cases; in most cases, A* without EPIC was slower or did not finish within the time limit (\(\dagger\)). Minimization ("Min.") speedup (ratio of single-conformation minimization time without EPIC vs. time with EPIC) is reported for EPIC with and without SAPE. Similarly, A* speedup denotes the ratio of total A* time without EPIC vs. time with EPIC. Fit times (times to calculate the EPIC polynomials) and total calculation times (Tot. time) are reported for EPIC calculations with SAPE, in minutes, calculated without parallelization of fitting. "GMEC energy error" is the absolute difference between GMEC energies calculated with and without EPIC, and is reported in kcal/mol. "Fit DNF" means the time limit was exceeded during polynomial fit precomputation, and "Bad fit" means that fits for EPIC without SAPE, even at the maximum allowed polynomial degree, did not accurately represent the energy. This problem is rescued by SAPE for these cases. When neither A* without EPIC nor EPIC without SAPE was successful, the A* speedup without SAPE cannot be calculated and is listed as “n/a.” EPIC was performed with minimization of partial conformations in all cases. ................................................. 110

3.2 Partition function calculation results. EPIC was performed with SAPE in all cases. As in Table 3.1, EPIC solved all cases, and cases for which A* without EPIC did not finish are denoted by \(\dagger\). \(a\) Ratio of total A* time without EPIC vs. total A* time with EPIC and minimization of partial conformations. \(b\) Ratio of total A* time with EPIC but no minimization of partial conformations vs. total A* time with EPIC and minimization of partial conformations. ................................................. 112
4.1 Protein design test cases with continuous flexibility, as described in Section 3 of the main text. Type is “aff” for designs for affinity, “stab” for designs for stability robust to force field choice, “spec” for designs to be specific to one complex, “multi” for designs to be multispecific to more than one, and “red” or “ox” for designs to favor either a reduced or oxidized state (each type is described further in Section 4.2.6). As described in Section 3 of the main text, $s$ is the number of states, $N$ is the number of sequences in the designed states, $m$ is the number of sequence tree nodes created, and $g$ is the number of state GMECs computed. Subscripts 1 and 5 denote calculation of the best sequence or enumeration of the best 5 sequences respectively. $N_e$ denotes the number of sequences enumerated (all runs were set to enumerate the five best sequences, but $N_e < 5$ if less than five sequences in the design space satisfy the constraints). Table continues on next page. *BPTI set: 3btd, 3bte, 3btf, 3btg, 3bth, 3btk, 3btm, 3btq, 3btt, 3btw . . . . . 145

4.2 Protein design test cases without continuous flexibility. Columns as in Table 4.1. Table continues on next page. . . . . . . . . . . . . . . . 148

5.1 Summary of A* method terminology . . . . . . . . . . . . . . . 154

7.1 Single-state protein design test cases. Each design was run with continuous sidechain flexibility, and some also included continuous backbone flexibility. Designs were performed with and without LUTE, using EPIC$^{64}$ in both cases. Designs not finishing without LUTE (after three weeks of computation time) are marked DNF. $n_L$ and $n_n$ denote the number of nodes in the A* tree after enumeration of the GMEC (or of the last conformation if several conformations closely spaced in energy were calculated; see Ref. 64), with and without LUTE respectively. The residual of the pairwise LUTE least-squares fit is denoted $r_p$. The final least-squares fit residual is denoted $r_f$. $r_f = r_p$ for designs with $r_p < 0.01$, but otherwise we computed a LUTE matrix with sparse triples and report the residual from the triples fit as $r_f$. These residuals are reported for the second iteration of iMinDEE, if two iterations were required (iMinDEE always finishes after at most two iterations).$^{40}$ The residual from the first round of iMinDEE is typically much lower because of the greater amount of pruning in the first iteration. Table continues on next page. . . . . . . . . . . . . . . 204
7.2 Multistate protein design test cases. These designs systems were taken from Ref. 61 and were run using the COMETS multistate design algorithm. Each design was run both with continuous flexibility, using LUTE, and without continuous flexibility. Type is “aff” for designs for affinity, “stab” for designs for stability robust to force field choice, and “multi” for designs to be multispecific to two different complexes (the types of designs, and the COMETS algorithm in general, are described further in Ref. 61). In each design, $N$ is the number of sequences in the search space and $k$ is the number of sequences enumerated (we enumerate either the top sequence or the top 5). $m_L$ and $m_d$ are the numbers of sequences explicitly considered using LUTE and in discrete search respectively. Similarly, $g_L$ and $g_d$ are the numbers of full conformational optimizations performed using LUTE and in discrete search respectively. “WC” indicates that the wild-type protein was found to have a clash unavoidable by discrete conformational search, and “NF” indicates that the discrete search provably cannot find five sequences that satisfy the design constraints. Table continues on next page.
List of Figures

1.1 Some continuous degrees of protein flexibility: the first sidechain dihedral ($\chi_1$) of each residue, and the backrub backbone perturbation (see Chapter 2). ................................................................. 4

2.1 DEEPer begins with the selection of a set of perturbations to use (this step may be manual or automated). These perturbations define a set of RCs for each residue. RCs that cannot be in low-energy conformations are pruned, using the new indirect pruning algorithm (Section 2.2.5) as well as previous pruning algorithms. A* with minimization (see Introduction), as in iMinDEE,\textsuperscript{40} is then used to output low-energy conformations. The result is the GMEC and a gap-free list of all conformations and sequences within a user-specified interval $E_w$ of the GMEC. This list can be used to select mutant sequences to synthesize for experimental testing, either by (i) selection of the sequences with the lowest-energy conformations, or (ii) by a provably-good approximation algorithm to calculate the binding affinities via the $K^*$ software module.\textsuperscript{48,108} ......................................................... 27

2.2 In DEEPer, the conformation of a protein is defined by multiple degrees of freedom. Sidechain dihedrals, such as the $\chi_1$ dihedrals for the residues shown, affect the conformation of only one residue. Perturbations, such as the backrub shown, can affect the conformation of several residues (three in this case; black balls denote the boundaries of the backbone region affected by the backrub). ................................. 28

2.3 Types of perturbations implemented in DEEPer. Each perturbation is from red backbone and orange sidechains to blue backbone and purple sidechains. Black balls denote the boundaries of the backbone region affected by the perturbation. (a) Backrub. (b) Shear. (c) Loop closure adjustment. (d) Secondary structure adjustment. (e) Partial structure switch. (f) Full structure switch. Two crystal structures of the motor protein Ncd related by a major conformational change are shown (PDB codes 2NCD\textsuperscript{144} and 1N6M\textsuperscript{174}). (g) Proline flip. ............................. 29
2.4 Geometry of multi-step perturbations. As in Fig. 2.3, each perturbation is from red backbone and orange sidechains to blue backbone and purple sidechains. (a) The two steps of a backrub perturbation, starting from an ideal poly-Ala β sheet. First, the dipeptide is rotated by the specified amount (here, +10°) around the C$_{\alpha1}$-C$_{\alpha3}$ axis (left). Then, each individual peptide is rotated by some fraction (70% by default, here corresponding to -7.0° and -5.6°) of the amount needed to restore the two carbonyl oxygens to their original positions (right). These same rotations, albeit with very different angles, are used for loop closure adjustments and secondary adjustments. (b) The three steps of a shear perturbation, starting from an ideal poly-Ala α helix. First, the N-terminal peptide is rotated by the specified amount (here, +5°) around C$_{\alpha1}$ in the C$_{\alpha1}$-C$_{\alpha2}$-C$_{\alpha3}$ plane (left). Then, the C-terminal peptide is rotated around C$_{\alpha2}$ in the C$_{\alpha2}$-C$_{\alpha3}$-C$_{\alpha4}$ plane such that the original C$_{\alpha2}$-C$_{\alpha3}$ distance (dotted line) is maintained (middle). Finally, the original middle peptide is rotated and translated to fit into the gap and to match the original carbonyl orientation (right).

2.5 The shear backbone motion. (a) Simple C$_{\alpha}$-only representation of shear, viewed down the axis of an ideal α helix (light colors). Shears of 5° (darker) and 10° (darkest) swing the central C$_{\alpha2}$-C$_{\alpha3}$ peptide (cyan) sideways by coordinated rotations of the C$_{\alpha1}$-C$_{\alpha2}$ peptide (blue) and C$_{\alpha3}$-C$_{\alpha4}$ peptide (green). (b) All-atom representation of shear, viewed from the side of the ideal α helix (i.e. rotated 90° from (a)). Shears of 2° over a 10° range are shown. The central carbonyl is notably displaced parallel to the central peptide. One endpoint conformation is marked by balls and line segments colored as in (a).

2.6 iMinDEE can be viewed as a global minimization algorithm over many degrees of freedom (DOFs) that searches a set of energy wells. (a, b) RCs represent likely locations for energy wells of particular residues. The notation $i_r$ is used to represent the $r^{th}$ RC of residue $i$. (c) This leads to an $n$-dimensional lattice for the possible energy well locations for $n$ residues. (d) We prune RCs and tuples of RCs using iMinDEE. Then A* is used to identify the energy wells with the lowest lower bounds, and local minimization is used to search these wells for the GMEC.
2.7 Parametric incompatibility of RCs. Suppose residue $i$ and $j$ are affected by the same backrub. RCs $i_t$ and $j_s$ are parametrically compatible because there exists an overall conformation of the protein in which residue $i$ is in $i_t$ and residue $j$ is in $j_s$. Similarly, $i_w$ and $j_s$ are parametrically compatible. But $i_r$ and $j_s$ are parametrically incompatible because there is no overall conformation of the protein in which residue $i$ is in $i_r$ and residue $j$ is in $j_s$.

2.8 Parametric incompatibility impedes DEE pruning but not indirect pruning. (a) Normally, DEE is able to compare conformations containing a candidate RC $i_r$ to those containing a competitor RC $i_t$ and prune $i_r$ if the conformations containing $i_r$ are always higher in energy. We can determine this by comparing all energy terms involving residue $i$: its internal energy plus its interaction energies with all other residues. Solid lines denote this summed energy for $i_r$ (red) and for $i_t$ (purple). (b) If $i_r$ and $i_t$ have different parameter intervals for a perturbation, then the conformations of other residues affected by the perturbation cannot be directly compared by the DEE criterion. Thus, pruning using this criterion is impossible, regardless of the energetics of these RCs. (c) This problem is alleviated if we compare all energy terms involving a multi-residue pruning zone rather than just the single residue position $i$. Indirect pruning works by comparing the sets of all conformations of the pruning zone containing $i_r$ (red) to all conformations of the pruning zone containing $i_t$ (purple). But even for a given set of conformations for all residues outside the pruning zone, each of these sets of conformations will have a range of energies (shown as colored regions). We use bounds (red and blue lines) to avoid the expensive process of considering each conformation of the pruning zone. We seek a lower bound on the energy difference between the best conformation involving $i_t$ (the blue line is an upper bound on this conformation’s energy) and the best conformation involving $i_r$ (the red line is a lower bound on this conformation’s energy, so the difference between the red and blue lines represents a lower bound on the energy difference). Dashed lines indicate the energy gap between conformational states involving $i_r$ and those involving $i_t$. 

xvi
2.9 The $R_j(i_r)$ notation. Suppose residues $i$ and $j$ are affected by the same backrub. Then $R_j(i_r)$ (blue voxels on the right) can consist only of RCs with the same parameter range for this backrub as $i_r$. RCs can also be excluded because of pruned pairs: even though $j_s$ has the same parameter range as $i_r$, $j_s \notin R_j(i_r)$, because the pair $(i_r,j_s)$ has been pruned (either by a steric check, a previous round of indirect pairs pruning, or another pruning algorithm). RCs of $j$ that are not in $R_j(i_r)$ (those that cannot be found simultaneously with $i_r$ in a conformation of the protein) are shown in green.

2.10 DEEPer GMEC designs compared to designs by previous algorithms with less flexibility on 67 test systems. (a) Decrease in energy from the iMinDEE GMEC to the DEEPer GMEC. (b) Decrease in energy from the BD GMEC to the DEEPer GMEC. Two systems are not shown because BD pruned all rotamers, indicating a steric clash for all rigid rotamers that was resolved by continuous minimization in iMinDEE and DEEPer. These systems are Chinese cobra phospholipase A2 (PDB code 1POA) and Pyrococcus furiosus hypothetical protein PF0899 (PDB code 2PK8). One system, the $\alpha$ subunit of human S-adenosylmethionine synthetase 2 (PDB code 2P02) had a higher DEEPer than BD GMEC energy. A dotted red line is shown in both (a) and (b) at the thermal energy at room temperature, 0.592 kcal/mol, as a rough measure of the significance of energy differences.

2.11 DEEPer GMEC designs compared to designs by previous algorithms with less flexibility on 67 test systems, continued. (a) Decrease in energy from the BD to the DEEPer GMEC versus decrease in energy from the iMinDEE to the DEEPer GMEC. (b) Numbers of test systems with given numbers of sidechain rotamer changes between the iMinDEE and DEEPer GMECs.
2.12 Comparisons of DEEPer GMECs (blue) to iMinDEE GMECs (red) and BD GMECs (green) for four systems. (a) Porcine pancreatic spasmolytic polypeptide (PDB code 2PSP), residues 52-58. Residue 54 is a tyrosine in the DEEPer GMEC but a lysine in the iMinDEE and BD GMECs. A steric clash (pink spikes, generated using Probe\textsuperscript{169}) persists after fixed-backbone, flexible-sidechain energy minimization of the rotamers of the DEEPer GMEC (magenta), making a tyrosine at residue 54 infeasible without backbone flexibility. (b-c) *Bacillus subtilis* cephalosporin D deacetylase (PDB code 1L7A), residues 100-107. The DEEPer, iMinDEE, and BD GMECs all adopt different histidine rotamers at residue 100 (b); the DEEPer rotamer is the only one in the crystal structure. Residue 105 is a tryptophan in the DEEPer GMEC but an alanine in the iMinDEE and BD GMECs (c); a steric clash persists after flexible-backbone, rigid-rotamer energy minimization of the rotamers of the DEEPer GMEC (magenta), making a tryptophan at residue 105 infeasible without continuous sidechain flexibility. (d) The protease penicillopepsin from *Penicillium janthinellum* (PDB code 2WEA), residues 300-305. The iMinDEE and BD GMEC backbones are very similar; the DEEPer GMEC adopts a different backbone and achieves a lower energy. (e) The Zβ domain of the human RNA editing enzyme ADAR1 (PDB code 1XMK), residues 306-311. The iMinDEE and DEEPer GMECs are virtually identical; the BD GMEC adopts a different backbone but still has a higher energy due to a lack of sidechain flexibility.

2.13 The lever effect for a backrub: the sidechain atoms move the most because they are farthest from the axis of rotation. A 5° backrub is shown. The view on the left looks down the axis of rotation, while the view on the right shows the axis of rotation as a black line.

2.14 Examples of DEEPer GMEC searches. (a) A sequence-design run on structure 2BGX (AmiD from *E. coli*). The GMEC backbone moved away from the starting conformation for residues 126-131. The lack of continuous flexibility in this run allows display of all searched backbone conformations. (b) A conformational-search run on structure 2IXT (sphericase). From the starting conformation, a partial structure switch allowed the GMEC to change the backbone to that of a crystallographic alternate conformation for residues 37-42, where its sidechain rotamers also matched. Starting structure, black/gray; complete searched ensemble, purple; GMEC, pink. Green balls demarcate flexible-backbone regions; sidechains outside these regions are omitted for visual clarity.
2.15 DEEPer ensembles are dependent on structural and sequence contexts. 
(a) The low-energy ensemble of computed models was fairly wide at residues 157-160 of sphericase (structure 2IXT) and spanned the crystallographic alternates. The GMEC was on the fringe of the ensemble. 
(b) Residues 238 and 240-243 also have alternates in structure 2IXT, but the low-energy ensemble from DEEPer for the G242S mutant, including the GMEC, was very tight around alternate A. One low-energy model made a significant excursion via a > 90° peptide flip, executed by a loop closure adjustment and backrubs. (c) Residues 35-38 in structure 1UBQ (ubiquitin) have a single well-ordered conformation. Correspondingly, the low-energy ensemble from DEEPer is very compact: the biggest departure is a single proline flip perturbation. Starting structure, black/gray; low-energy ensemble, blue; GMEC, pink. Green balls demarcate flexible-backbone regions; sidechains outside these regions are omitted for visual clarity.

3.1 (A) The energy of each residue is represented by EPIC as a polynomial in the internal coordinates, such as sidechain dihedrals \( \chi \). Low-degree, inexpensive polynomials (blue) are tried first, and the degree is increased as needed to achieve a good fit (black) to the actual energy function (red). These polynomials are then used for design in place of the full energy function. (B) Interactions between pairs of residues are represented in terms of both residues’ internal coordinates.

3.2 The number of energy wells in a protein system scales exponentially with the number of flexible residues, leading to an exponential number of energy function calls, but EPIC can replace most of these calls with quick evaluations of low-degree polynomials. (Top) A protein may have an energy well for every combination of rotamers (rainbow) at different residues. The global minimum-energy conformation (GMEC) of a protein may be in any of these wells. We model the energy as a sum of pairwise energy terms. Each pairwise term will have wells for pairs of rotamers, but there are far fewer wells of this kind—a number quadratic in the number of residues. We can easily afford the energy function calls needed to characterize each pairwise well. (Bottom) By precomputing a polynomial representation (blue) of the energy within each well of each pairwise term (red), we enable computation of any pairwise term in any pairwise well, and thus of the full protein energy in any energy well of the protein, solely by a quick evaluation of polynomials.
3.3 (A) For each energy value $E'$, there is a range of “ideal” values for the EPIC fit (green). For the energies below cutoff $b_1$, which may be found in favorable conformations, this range is just the energy (the range has zero width). For higher energies, the range is defined using the cutoffs $b_1$ and $b_2$. For fitting purposes, EPIC fit values are penalized by the amount they lie outside the ideal range (the purple point represents a sample conformation for a given EPIC fit incurring the penalty indicated in red). (B) Example of curves satisfying these conditions. The EPIC fit matches the energy closely up to the cutoff $b_1$, after which it deviates from the energy, but stays in the target region shown in A, by staying below the energy. Once the energy is over $b_2$, the EPIC fit can be either above or below the true energy without leaving the target region.

3.4 Mutatable residues in the redesign of the surface of the HIV surface protein gp120 in complex with the broadly neutralizing antibody NIH45-46 (PDB code 3u7y$^{27}$). This design finished only when EPIC was used. Mutatable residues, blue backbone and pink sidechains; gp120, black backbone; NIH45-46 heavy chain, green backbone; NIH45-46 light chain, brown backbone.

3.5 (A) A* times with and without EPIC. Five designs that did not finish without EPIC are shown on the right in red. (B) Proportions of each type of fit (see Section 3.2.6) required in EPIC calculations. The “quartic*” category includes both full quartic fits and quadratic fits with quartic terms added for $D_{10}$ or for $D_{100}$. Fits were all made as high-degree as needed to obtain a residual below 0.0001, as described in Section 3.2.6. Some fits have substantially lower residuals, especially quadratic fits without SAPE, since no lower fit degrees were allowed. (C) Speedups due to different EPIC methods compared to A* based on pairwise lower-bound energies; standard EPIC includes both SAPE and minimization of partial conformations. PF denotes partition function calculations; the others are GMEC calculations.

3.6 Intra-residue energy calculated for Phe 2 of aspartame using Hartree-Fock theory with a STO-3G basis set, and quadratic EPIC fit, as a function of the two sidechain dihedrals. The fit is very close to the energy surface, though a slight discrepancy is visible in the upper right-hand corner ($\chi_1 \approx 70^\circ$, $\chi_2 \approx 100^\circ$).
4.1 Flexible and mutable residues in a design for specificity. The apoptotic regulator CED4 forms two different dimers, one to block apoptosis (left; PDB id 2a5y\textsuperscript{172}) and one to induce it (right; PDB id 3lqr\textsuperscript{133}). We want to design for specificity (to block apoptosis), so we allow mutations to some residues in the binding site (blue). To accurately model the conformational changes induced by the mutations, we also model as flexible the residues on the opposite side of each interface that interact with the mutable residues (orange, pink). Analysis of this calculation and others is described in Section 4.3.

4.2 Expansion steps during node processing generate nodes with partially (e.g., VXXXX or AXXXX) and then fully (e.g., VFYWI) defined sequences. Once a node has a fully defined sequence, conformational trees are built for it for all states. Then conformational tree expansions lead to fully processed nodes. X, unassigned amino acid or RC; V, Val; A, Ala; F, Phe; Y, Tyr; W, Trp; I, Ile.

4.3 comets is a sequence of node-processing operations.

4.4 Expansion of the state conformational trees for the node with sequence V in the toy example. (A) Each state conformational tree starts with a single node, in which all conformational degrees of freedom are undefined. (B) When the sequence tree node is fully processed, the lowest-scoring node in each conformational tree fully defines a conformation. (Only the lowest-scoring node in each tree is shown here: each tree will also have higher-scoring nodes, with varying numbers of unrestricted conformational degrees of freedom). (C) The expansion of the conformational tree for the T-bound state. As in single-state A*, nodes are chosen for expansion based on their scores, which are lower bounds on the energies of conformations in their conformational spaces.

4.5 Number \( g \) of state GMECs calculated in comets runs with (A) rigid or (B) continuous flexibility, compared to the number \( sN \) of state GMECs in the entire design space (\( sN \) is the number of sequences in the design space times the number of states). Results are shown both for calculation of the best sequence and for enumeration of the best five, when possible under the design constraints. Exhaustive search would have to calculate all state GMECs (green curve).
4.6 Speedup due to reduced explicit consideration of sequences in COMETS, compared to exhaustive search (green line), for designs with rigid rotamers. $m$: number of sequence tree nodes created in COMETS. $N$: number of sequences in the design space. Magnifying this speedup, COMETS handles sequences that it considers explicitly very efficiently (Fig. 4.5).

4.7 For the design run optimizing the difference in energy between the reduced and oxidized states of angiotensinogen (PDB ids 2wxy and 2wxx, respectively), the single mutation Y12I (blue to green) was found to fit well into the reduced state, but to cause steric clashes (pink) in the oxidized state. The selection of this single mutant as optimal required explicit multistate design to destabilize the undesired oxidized state, while maintaining the stability of the reduced state: optimization of either the reduced or oxidized state alone yielded aromatic residues at position 12.

5.1 **Tree representation of protein conformation space.** (A) A toy example of three serine residues (shown in orange, blue and grey) belonging to the antibody VRC07 (PDB id: 4OLZ\textsuperscript{143}), partially shown in white cartoon. (B) A 2D representation of (A), and for the purposes of this toy example, we allow each residue to mutate to only two rotamers (shown here as a star and a circle). (C) Protein design algorithms compute pairwise interactions between rotamers based on an input energy function, and these are shown here in matrices between residue pairs. For simplicity, all internal rotamer energies are zero, and the pairwise energies not shown have a zero value. (D) The protein conformation and sequence space can be represented as a tree. In a tree representation, each level represents a residue, each inner node (each of the nodes between the root of the tree, $r$, and the leaves of the tree) represents a partially assigned conformation, and each child assigns a rotamer choice for the next residue. Each leaf represents a fully assigned conformation. A naïve approach to solve the protein design problem would explore this tree completely. The optimal path is shown in red. (E) Branch-and-bound algorithms such as A* explore a small part of the tree by computing energy lower bounds (called $f$-scores and shown next to each node) on the possible conformations allowed at each inner node. A* expands nodes in order of their $f$-score and guarantees that the optimal solution is found (shown in red).
5.2 Toy example that demonstrates the effect of residue position 
reordering on an A* search. (A) Toy design problem where seven 
residue positions are each allowed to mutate to two rotamers (repre-
sented by a star and a circle). Each residue position is colored by a 
unique color: orange (position 0), cyan (position 1), grey (position 2),
purple (position 3), maroon (position 4), green (position 5), and black 
(position 6). (B) Diagram showing the pairwise energies between all 
rotamers in this toy example. For simplicity, assume that all intra-
rotamer energies are zero and can be ignored, and that the interactions 
between pairs that are not joined by an edge are zero. (C-E) The A* 
algorithm explores only part of the full conformation tree to compute 
the optimal conformation. A* iteratively expands the node with the 
lowest $f$-score (shown by the dotted red path for the nodes in the 
optimal conformation path) until a leaf is reached. Each expansion 
results in the creation of new nodes representing the children of the 
expanded node. To compute the optimal conformation efficiently, it 
is desirable to expand the fewest number of nodes. The number of 
nodes expanded can be dramatically reduced by changing the order-
ing of the tree. (C) The traditional A* algorithm for protein design 
(Trad-A*) sorts residues in the arbitrary sequential order given by 
the protein sequence. The bounds on the energies for each inner node 
in the optimal conformation are shown in red, and the path that leads 
to the optimal conformation is marked in a thick, red, dashed line. 
In this toy example, Trad-A* expands 33 nodes, and creates 67 nodes 
(the 33 expanded nodes plus their children). (D-E) Large speedups 
in A* can be achieved by a rational ordering of nodes. The energies 
of each node in the optimal conformation are shown. (D) In a static 
reordering, residue levels are reordered once before A* runs. In this 
toy example, A* with static reordering must only expand 13 nodes 
and create 25 nodes to compute the optimal conformation. (E) In a 
dynamic reordering, the next level is chosen independently for each 
path “on the fly” (i.e., as the A* algorithm expands nodes). In this 
dynamic reordering example, at depth $m = 2$ the solution path ex-
pands position 3 (purple) while the alternative path expands position 
4 (maroon). A* with dynamic ordering must expand only 9 nodes, 
and create a total of 17 nodes, to compute the optimal conformation. 
(F) The optimal conformation for this example is shown.
5.3 The number of expanded A* tree nodes is greatly reduced by improved variable ordering methods. **Top:** The total number of conformations A* had to search through for 31 difficult side-chain placement problems. The size of the conformation space shown is the number of conformations remaining after dead-end elimination pruning. **Bottom:** The number of A* nodes expanded by three different A* orderings for the 31 side-chain placement problems. Data is shown for the sequential residue ordering used in Trad-A* (red circles), the StaticHMean static variable ordering (purple pentagons), and DynHMean dynamic variable ordering (green squares). The $x$-axis is labeled by the PDB id used for each side-chain placement problem. All of the runs used the Trad-A* $f$-score. Trad-A* failed to solve 21 problems (right of the red vertical line) and StaticHMean failed to solve 5 of the problems. For visual clarity the $x$-axis is ordered first by the number of nodes expanded by Trad-A*, second by StaticHMean, and finally by DynHMean.
6.1 Ligand-free HIV-1 Env trimer is structurally compatible with epitopes of broadly neutralizing, but not ineffective, antibodies. (a) Superposition of ligand-free and antibody-bound HIV-1–Env structures. Left, ligand-free gp120 core monomer shown in ribbon representation, with regions of less (or greater) than 2 Å r.m.s. deviation upon antibody binding shown in green (or magenta) and representative antibody-bound structures in gray. Middle and right, ligand-free and antibody-bound HIV-1–Env trimers. At right, antibodies PGT122 and 35O22 are shown in gray semitransparent surface, and the rear protomer has been removed for clarity. (b) Breadth-potency plot of broadly neutralizing (green) and ineffective (magenta) antibodies on a diverse 170 HIV-1–isolate panel. (c) Structural compatibility of ligand-free trimer by antibody epitope. The ligand-free Env structure is displayed as Cα ribbon, with antibody-epitope residues colored green (structurally compatible) or magenta (incompatible) or gray for nonepitope regions. r.m.s. deviation (solid fill) and volume overlap (striped fill) with the indicated antibody–Env complexes are shown in bar graph, with two linear scales split at r.m.s. deviation and antibody-antigen–volume overlap cutoffs of 2 and 500 Å³, respectively; bars below the respective cutoffs are colored green and magenta otherwise. Antibody labels are colored green if the epitope is structurally compatible, magenta if incompatible and gray if not present in the structure. (d) Ligand-free-trimer structural compatibility versus antibody breadth. Volume overlap (left), r.m.s. deviation (middle) and antigenic structural compatibility (ASC) score (right), plotted versus antibody breadth on a diverse 170 HIV-1–isolate panel. P values for Spearman correlations provided (n = 14 antibodies).
7.1 **LUTE** makes continuously flexible design efficient by representing continuous flexibility using local, discrete energy terms. (A) Protein design with discrete flexibility searches over a discrete (albeit large) conformational space (“Conf”), looking for low-energy (“E”) conformations. Highly efficient algorithms like DEE/A* are available for this problem. (B) Protein design with continuous flexibility must search over a large space of voxels (blue) in a continuous conformational space, but we are usually interested only in the minimum-energy point of each voxel. We thus want a way to search combinatorially over these minimum-energy points. (C) The minimized energy of a voxel in protein conformational space depends on all rotamers in the voxel (arrow 1). But we can expand this minimized energy as a sum of local contributions from low-order tuples (e.g., pairs) of residues (arrows 2, 3). (Minimized conformations shown in red, ideal rotamers in blue). (D) This expansion, known as LUTE, gives us a discrete combinatorial search problem of the same form as protein design with discrete flexibility (arrows 4, 5). But this new discrete problem searches over the minimum-energy points (red) of voxels in continuous conformational space (blue). We can solve this problem very efficiently. Figure shows Leu 29, Leu 51, Phe 55, and Lys 59 of the Atx1 metallochaperone (PDB id 1CC8).

7.2 **Competitor pruning reduces the set of RCs we need to consider as competitors during iMinDEE pruning.** In iMinDEE pruning, we take a “candidate” RC (or RC pair or triple) $i_r$ and compare it to another RC (or pair or triple) $i_t$ by evaluating the pruning checksum $E_\bigotimes(i_t) - E_\bigotimes(i_t) + \sum_{j \neq i} \min E_\bigotimes(i_r, j_s) - E_\bigotimes(i_t, j_s)$. We compare this checksum to the pruning interval $I$, where $E_\bigotimes$ is a lower bound on a one-body or pairwise interaction energy. If the checksum exceeds $I$, then we can eliminate $i_r$ as a possible constituent of the GMEC. Without competitor pruning (top), all RC tuples (blue) need to be considered as candidates or competitors, and thus the pruning condition (green) must be evaluated for every pair of tuples. But with competitor pruning (bottom), we first perform a modified pruning protocol to quickly eliminate those tuples that will not be effective as competitors. Then, when performing our actual pruning (to eliminate non-GMEC RC tuples), we can do so more quickly because we have less competitors to consider.
7.3 Continuous pruning increases pruning power by using a tighter lower bound on conformational energy. iMinDEE pruning is based on lower bounds on conformational energy, which are constructed by adding up lower bounds on pairwise interaction energies (red arrows) and one-body energies (circles). Without continuous pruning (left), each flexible residue and each flexible residue pair has its lower-bound energy computed separately. But with continuous pruning (right), if we add the continuous contribution $c$ for a pair of RCs, then we effectively replace the three lower-bound terms for that pair (the two one-body energies and the pairwise interaction energy) with a single lower bound on the energy of the pair. This tighter bound can significantly increase pruning power, because if a residue pair has an unavoidable clash, the lower bound on the pair’s internal energy is likely to show this even if each of the three constituent energies is individually capable of continuously minimizing to a better value.

7.4 LUTE markedly reduces the cost of continuously flexible conformational search. Ratios (without LUTE:with LUTE) of the number of nodes in the A* tree before enumeration of the GMEC (or of the last conformation if several conformations closely spaced in energy were calculated; see Ref. 64), versus number of flexible residues. A 20-residue sidechain placement with node ratio $2 \times 10^5$ is not shown because it would break the scale.

7.5 LUTE accurately represents continuously minimized energies. Residuals for LUTE ((kcal/mol)$^2$) on the cross-validation data set, measuring the difference between the EPIC energy and a pairwise expansion (blue) or one with sparse triples (red; computed only if pairwise residual exceeded 0.01). $x$ axis: number of flexible residues. Inset: All the same data plotted on a linear scale.
7.6 **LUTE enables very large provably accurate protein designs with continuous flexibility (left) and with Poisson-Boltzmann energy functions (right).** Left: previously, protein designs with continuous flexibility only finished when performed with significantly fewer flexible residues, compared to designs with discrete rotamers. Even 20-residue designs were often intractable. But LUTE solved a sidechain placement problem with continuous flexibility in which 40 residues (purple) were made flexible in the Atx1 metallochaperone (PDB id 1CC8). Right: previous designs using the Poisson-Boltzmann energy function could not optimize this function directly, but only used Poisson-Boltzmann energies to rerank top hits from optimization of a simpler, pairwise energy function. But LUTE can optimize the Poisson-Boltzmann energy function directly—e.g., in a sidechain placement of 20 residues (purple) of Atx1.

7.7 **LUTE brings rigid-rotamer-like efficiency to provably accurate multistate designs with continuous flexibility.** Top: The number of sequences considered explicitly in continuously flexible designs using LUTE with COMETS (blue) is similar to the number considered in discrete COMETS designs on the same system (red) and far less than the total number of sequences in the search (green). Bottom: In the same designs, the number of full conformational optimizations needed is also similar for the continuous (LUTE, blue) and discrete (red) designs. Usually, only the optimal sequence (or top 5 if enumerating 5 sequences) required full conformational optimization, while exhaustive search must fully conformationally optimize every sequence in every state in the multistate design (green). Design test cases taken from Ref. 61. * denotes enumeration of the top 5 sequences. Missing discrete calculations mean that discrete search was unable to find a non-clashing conformation for the wild-type protein, or that 5 sequences satisfying the design constraints were not available in the discrete search space.
Acknowledgements

I would like to thank all my colleagues in the Donald lab, especially Bruce and my collaborators Kyle, Pablo, and JJ. I’d also like to thank Kyle and Pablo for providing PDB files and scripts for testing my algorithms. I’d like to thank my committee members for all their advice and willingness to learn about my project: Bruce, Jane Richardson, David Beratan, Carlo Tomasi, Mauro Maggioni, Pei Zhou, and Michael Fitzgerald. I’d also like to thanks the Richardsons and Daniel Keedy for working with me on DEEPer. I’m also very grateful for the support of my family—Ying, my parents, my brothers, my grandparents—and my friends, especially Zack, Sarah, and David. I would like to thank the NIH both for funding and for hosting me for an internship, especially Peter Kwong and Ivelin Georgiev, and also the DoD, PhRMA Foundation, and Liebmann Foundation for funding. And I’d like to thank Duke, the NSF, and the Biophysical Society for travel grants.
The biological and chemical sciences have made enormous advances in the past several decades. The discovery of new chemicals and new uses for old ones have revolutionized both medicine and industry. However, we still face very significant challenges. For example, many diseases, notably cancer and antibiotic-resistant bacterial infections, are still often difficult to treat effectively. New chemicals—especially proteins—could be very helpful in addressing these issues.

The space of chemicals that humans have never synthesized—even just protein sequences, or just druglike small molecules—is far greater than the space of natural products and current synthetic chemicals combined. We have only begun to scratch the surface of this space, even in computational studies. Even small subsets of this space—like protein sequences with mutations limited to an active site—are too large to search exhaustively. To address this issue, we need computational methods that can optimize the property we want—for example, the affinity of a designed protein to its desired ligand—by a method much faster than exhaustive search, while still computing the optimum fairly accurately. Previous work, in particular Refs. 108 and 48, have given us a framework that addresses this need fairly directly, if implemented
with sufficiently efficient and accurate algorithms. I’ll refer to this framework as the *voxel paradigm*.

In this work, I’ll focus mostly on the design of proteins, which are the most versatile known class of chemicals in terms of what functions they can perform. In addition to being useful for many applications (e.g., see Section 1.1.3), proteins are an ideal model system for developing design methods, for two reasons. First, their conformational and energetic behavior is quite well characterized, including their rotamers, sterics, energy functions, etc. Second, synthesizing proteins of novel sequence is relatively easy compared to other classes of molecules with a similar, combinatorially large number of design options. However, most of my algorithmic work is applicable to designing other chemicals as well as proteins, as long as the design problem can be posed in the voxel paradigm. In particular, three of my four main algorithms—COMETS (Chapter 4), EPIC (Chapter 3), and LUTE (Chapter 7)—are agnostic to the specifics of molecular structure.

The main goals of this thesis are to further develop the voxel paradigm, in a way that allows more accurate biophysical modeling, and to develop efficient algorithms for it. As such, the bulk of this thesis will describe the four main algorithms I have developed in my graduate studies, all of which are meant to enable enhanced modeling—DEEPer (Chapter 2), COMETS, EPIC, and LUTE. I will now describe the voxel paradigm in detail (Section 1.1), to provide the mathematical background in which these algorithms are formulated. I will also provide background on some parts of the biophysical model that I am trying to enhance: backbone flexibility (Section 1.2), quantum-chemical energy modeling (Section 1.4), and interactions with solvent (Section 1.3). In addition, each chapter of this thesis will review literature of particular relevance to the new research presented in that chapter. Finally, I will discuss how the Greek myth of Perseus inspires me during my research in protein design (Section 1.5), and will provide an outline of this thesis (Section 1.6).
1.1 The voxel paradigm

1.1.1 Overview

To introduce the voxel paradigm, let’s consider a simple protein design problem. We wish to optimize the sequence of some protein B to optimize its binding to another protein A. That is, we wish to minimize the dissociation constant $K_d = \frac{[A][B]}{[A \cdot B]}$, where $[\cdot]$ denotes the concentration of a chemical species in solution, and A·B denotes the complex of A and B. We know that $-RT\ln K_d = G(A) + G(B) - G(A \cdot B)$, where $G(\cdot)$ denotes the free energy of a chemical species in solution.

Since we are optimizing with respect to the sequence of $B$, $G(B)$ and $G(A \cdot B)$ are actually functions of the sequence $s$. Since B is a protein, it is composed of small chemical units called residues that can have one of twenty chemical structures (called amino acids). $s$ is a sequence of amino-acid types, one for each mutable residue in B. The mutable residues are chosen by the user (based on what set of mutations he or she wants to search), but must be defined consistently between B and A·B.

To apply this paradigm to molecules that are not proteins, we need an analogous notion of residue, and each residue must have a set of residue types to which it can mutate, such that the sequence $s$ defines the chemical structure (element types and bonds between them) of the molecule. For example, in RNA, the residues could be nucleotides, with types A, C, G, and U.

Now we can set up our optimization problem. To minimize $K_d$, we want to minimize

$$G(A \cdot B, s) - G(A) - G(B, s)$$

with respect to the sequence $s$, where $G(\cdot, s)$ denotes the free energy as a function of the sequence of B.

To proceed further, we must have some method of estimating free energies as a function of sequence, and this is where structural and conformational modeling
come into play. We will assume that A and B each have zero or more conformational
degrees of freedom: for example, sidechain dihedrals (Fig. 1.1). Furthermore, we
will assume that the feasible conformational space of each residue can be modeled
as a union of boxes, or voxels, which we will call residue conformations or RCs.
For example, sidechain dihedral values for each amino-acid type are known to fall
in a small number of clusters, called rotamers.\textsuperscript{81, 115} Thus, if we are modeling only
sidechain dihedral flexibility, we may create an RC for each sidechain rotamer, and
the center of the box can be the modal dihedral values for that rotamer.

Now we have a tractable representation of the conformational space of a protein:
it is a union of voxels, where each voxel corresponds to a list $r$ of RCs, one for each
residue being modeled as flexible. Letting $R$ be the set of all such lists and $V(r)$ be
the voxel corresponding to $r$, our conformational space is $\bigcup_{r \in R} V(r)$.

At this point we also need to start making some assumptions about the energy.
To a very good approximation, for any protein molecule, there exists an “energy
function” $E(c)$ of the molecular conformation $c$, such that the probability density
$P(c)$ of finding the molecule in conformation $c$ is proportional to $\exp\left(\frac{-E(c)}{RT}\right)$. The
free energy of the molecule is then given by

$$G = -RT \ln \int \exp\left(\frac{-E(c)}{RT}\right) dc = -RT \ln \sum_{r \in R} \int_{c \in V(r)} \exp\left(\frac{-E(c)}{RT}\right) dc \quad (1.2)$$

where the integral is over all conformations, $R$ is the universal gas constant, and $T$
is the temperature. Because this integral can be difficult to estimate accurately, it is sometimes approximated as

$$\min_c E(c) = \min_{r \in R} \min_{c \in V(r)} E(c)$$  \hspace{1cm} (1.3)

(i.e., neglecting entropy). A better approximation is introduced in Refs. 108 and 48:

$$-RT \ln \sum_{r \in R} \exp \left( - \frac{\min_{c \in V(r)} E(c)}{RT} \right)$$  \hspace{1cm} (1.4)

Whichever approximation we use, we now have a way to evaluate free energy as a function of sequence, so we can minimize Eq. (1.1) with respect to sequence to get the sequences that optimize binding. A similar approach works for other protein design problems: for example, to maximize the stability of the protein B, we can simply minimize $G(B, s)$ with respect to $s$.

1.1.2 Previous algorithms based on the voxel paradigm

The voxel paradigm itself, including continuous flexibility, and modeling of entropy, was introduced in the context of the $K^*$ algorithm. $^{48, 108}$ $K^*$ optimizes binding of one molecule to another (as in the example described in Section 1.1.1), by approximating the free energy using Eq. (1.4), for the complex and each unbound molecule. It does this by approximating the sum

$$\sum_{r \in R} \exp \left( - \frac{\min_{c \in V(r)} E(c)}{RT} \right)$$  \hspace{1cm} (1.5)

from Eq. (1.4) within a relative error of at most $\varepsilon$, where $\varepsilon$ is adjustable but often taken to be 3%. It includes some techniques to prune sequences that prevent it from being entirely exhaustive over sequences, while still performing a provably accurate optimization (with respect to the biophysical assumptions used).
In order to efficiently approximate Eq. (1.5), $K^*$ enumerates conformations $r \in R$ in order of a lower bound on the minimized conformation. This lower bound is the sum of lower bounds on pairwise interaction energies. Enumerating conformations in order of this lower bound is a computational problem of the same form as enumerating conformations in a discrete-rotamer conformational space in order of energy, and thus the DEE/A* algorithm\textsuperscript{99} for discrete-rotamer search is used for the enumeration. The DEE pruning conditions are adapted for the continuous case, and this modified DEE/A* is known as minDEE.\textsuperscript{48}

minDEE can be used to minimize over a conformational space that includes rotamers from multiple amino acids at one or more residues. This would then compute the sequence with the lowest optimized conformational energy, which is appropriate for stabilizing a protein. This optimization is sometimes also used to optimize the stability of a protein complex, as a proxy for computing the free energy of binding.\textsuperscript{34, 48, 87}

Variants of minDEE that include backbone flexibility instead of continuous sidechain flexibility were also developed. BD\textsuperscript{44} allows continuous motions in any of the backbone dihedrals, with the changes being propagated all the way down the backbone. brDEE\textsuperscript{46} allows non-overlapping, discrete backrub\textsuperscript{23} motions, which move the backbones of three residues each in a direction that has been observed in many crystallographic alternates.

More recently, a version of minDEE with much greater pruning ability, known as iMinDEE, has been derived.\textsuperscript{40} Most of my work in this thesis builds directly on iMinDEE, significantly extending both its modeling capabilities and its efficiency.

1.1.3 OSPREY and its successes

The Donald lab’s protein design software package—OSPREY\textsuperscript{41, 48, 49}—has always been based on the voxel paradigm, and included the original implementations of
I have also implemented my algorithms in it, and am leading a refactoring effort, which is based largely on the concepts of the voxel paradigm.

**OSPREY** has been used for numerous protein designs that performed well experimentally—*in vitro*\textsuperscript{13,36,42,54,139,143,151} and *in vivo*\textsuperscript{54,136,139,143} as well as in non-human primates.\textsuperscript{143} It has been used to predict mutations that lead to antibiotic resistance in *Staphylococcus aureus*, by maintaining binding to the substrate (in the form of the enzyme-substrate complex that is an intermediate in catalysis) while exhibiting reduced binding to inhibitor.\textsuperscript{36,136} It has also been used to develop peptide inhibitors,\textsuperscript{139} to alter the sequence of antibody to bind its antigen more tightly,\textsuperscript{143} and to switch the substrate of an enzyme from phenylalanine to leucine.\textsuperscript{13} These designs could have therapeutic applications for cystic fibrosis,\textsuperscript{139} HIV,\textsuperscript{143} and cancer.\textsuperscript{54}

### 1.1.4 Alternative paradigms for protein design

The majority of protein design work does not use the voxel paradigm *per se*, but generally falls in two other classes.

One class resembles the voxel paradigm, but with each residue's conformational space restricted to discrete rotamers. DEE/A* was originally developed in this context.\textsuperscript{99} However, various methods without guarantees of accuracy are often used to solve it, such as Metropolis Monte Carlo and simulated annealing.\textsuperscript{93,102} These methods can be run on very large systems, but unfortunately they add an additional layer of error to the drastic simplification of the conformational space and to all the other errors that are present even in the voxel paradigm. Since all these errors are typically amplified when many simultaneous mutations are modeled, the benefits of the large sequence space that discrete rotamers and simulated annealing allow are likely to be offset by the high levels of error in the search. Discrete rotamers can be seen as a special case of the voxel paradigm: the case where the backbone is rigid.
and each voxel is a single point.

The other class estimates free energy (or binding behavior directly) by simulating protein motion over time, thus avoiding any restrictions on protein conformational space, but now having to perform a separate energy computation for each sequence. This type of simulation is known as molecular dynamics.\textsuperscript{134}

The voxel paradigm is effectively a tradeoff between these extremes. Its coverage of conformational space is not yet as extensive as that of molecular dynamics, but in principle could be expanded to cover any portion of conformational space. For the time being, molecular dynamics is still able to model phenomena that cannot be handled in the voxel paradigm. However, the voxel paradigm can provide provable guarantees of accuracy when searching the conformational space that it does model, and it need not exhaustively search over sequences.

1.2 Backbone flexibility

On an atomic level, the structure of a protein consists of a “main chain” of atoms with various substituents. The main chain, which is considered part of the backbone, is a repeating sequence of atoms: N, C, C, N, C, N, C, C,... Each repeat (N, C, C) is a different residue. For each residue, we can define three dihedral angles: the C-N-C-C dihedral is called $\phi$, N-C-C-N is called $\psi$, and C-C-N-C is called $\omega$.

The backbone flexibility of a protein residue is dominated by two degrees of freedom, the $\phi$ and $\psi$ dihedrals. Other internal coordinates of the backbone—the $\omega$ dihedral, and the bond lengths and angles—can only move slightly. Furthermore, for each residue, most values of the pair $(\phi, \psi)$ are not biophysically feasible. For each amino-acid type, feasible values of $(\phi, \psi)$ can be computed from statistics on known high-resolution crystal structures,\textsuperscript{114} and are known as Ramachandran-allowed. Thus, when modeling backbone flexibility, we are primarily concerned with modeling changes in the $\phi$ and $\psi$ dihedrals that are Ramachandran-allowed.
If we have a wild-type crystal structure, our goal is usually to predict conformational changes that arise in response to a mutation. In many cases, the mutation will induce only a small conformational change, and thus we need only model a small region of backbone conformational space near the wild-type backbone conformation. Analysis of crystallographic alternates and of large classes of mutants suggests that protein backbone flexibility sometimes is clustered along certain directions, such as a three-residue motion known as a backrub, rather than being equally distributed in all directions in \((\phi, \psi)\) space.\textsuperscript{23,88}

1.3 Interactions with solvent

Proteins are bathed in solvent, typically mostly water, and this drastically affects their behavior. Proteins are even known to have significantly different behavior (especially binding) as a function of ion concentrations. Modeling interactions with solvent is probably the biggest difficulty for the voxel paradigm compared to molecular dynamics, because of the difficulty of modeling explicit solvent molecules (whether water or ions). However, some approaches to address this problem have been explored.

First, rotamers have been modeled that have a water molecule interacting with the amino acid. These are known as solvated rotamers.\textsuperscript{82}

Second, we can “factor out” the interactions with water so that they are treated as part of the energy function. Starting from Eq. (1.2), the free energy of a molecule is given by

\[
G = -RT \ln \int \exp \left( -\frac{E(c)}{RT} \right) dc = -RT \ln \int \int \exp \left( -\frac{E((c_{H2O}, c_p))}{RT} \right) dc_p dc_{H2O} 
\]

(1.6)

where \(c_p\) denotes the conformation of the protein, \(c_{H2O}\) denotes the conformation of the solvent (the position and orientation of each water molecule, and any salt ions,
etc.), and thus the overall conformation $c$ is the pair $(c_{\text{H}_2\text{O}}, c_p)$. We can thus define a “protein-only” energy function

$$E_p(c_p) = -RT \ln \int \exp \left( -\frac{E((c_{\text{H}_2\text{O}}, c_p))}{RT} \right) dc_{\text{H}_2\text{O}}, \quad (1.7)$$

and if we know how to evaluate this, we need only integrate over the conformation of the protein to get the full free energy:

$$G = -RT \ln \int \exp \left( -\frac{E_p(c_p)}{RT} \right) dc_p \quad (1.8)$$

Eq. (1.7) could be evaluated by sampling methods like molecular dynamics or Monte Carlo. But due to the considerable computational expense of such calculations, it is often approximated by “implicit-solvent” methods, which treat the solvent as a polarizable continuum, i.e., a continuous bath of charge. A common implicit solvent method is to calculate the enthalpic component of the solvation energy using the Poisson-Boltzmann equation, and the entropic component as proportional to solvent-accessible surface area (SASA).\(^{148}\) The Poisson-Boltzmann equation models a polarizable continuum directly according to the laws of electrostatics. It is Gauss’ law for electricity adapted to a setting with variable dielectric and with salt ions distributed according to a Boltzmann distribution in the electric potential. Using the Poisson-Boltzmann equation, one can calculate the electric potential around a protein in the presence of solvent, and the energy of the protein in this potential plus the empirical SASA-based correction for solvent entropy can be used as $E_p(c_p)$. Several efficient algorithms have been derived both to solve the Poisson-Boltzmann equation\(^{123,141}\) and the SASA.\(^{100,176}\) However, since even fast algorithms for the Poisson-Boltzmann calculation are still substantially slower than evaluating a molecular-mechanics force field, faster empirical methods for handling implicit solvation, like generalized Born and EEF1,\(^{97}\) are commonly used.
1.4 Quantum-chemical energy modeling

To a very high level of accuracy, the energy function $E(c)$ from Section 1.1.1 is given by the ground-state energy of the quantum-mechanical system formed by the electrons of the molecule when the positions of the atomic nuclei are given by the conformation $c$. That is, if we let $x_i$ denote the coordinates of electron $i$, and we let $a_j(c)$ and $Z_j$ denote the nuclear coordinates and atomic number of atom $j$ respectively, then $E(c)$ is the minimal $E$ such that

$$-\frac{\hbar^2}{2m_e} \nabla^2 \psi(x) + Ce^2 \left( \sum_{i,j} \frac{-Z_j}{||x_i - a_j||} + \sum_{i<k} \frac{1}{||x_i - x_k||} \right) \psi(x) = E\psi(x) \quad (1.9)$$

for some function $\psi(x)$ of the electron coordinates, known as the wavefunction, where $C$ is the Coulomb constant, $m_e$ the mass of the electron, and $e$ the elementary charge. Eq. (1.9) is called the Schrödinger equation. This differential equation only admits known analytical solutions for a few very simple molecules (notably the hydrogen atom and the H$_2^+$ ion), and its high dimensionality precludes the use of standard numerical methods like finite differences for most molecules as well. However, several specialized methods have been developed to solve it approximately. These methods offer a range of tradeoffs between speed and accuracy. At the least accurate end are semi-empirical models like CNDO$^{132}$ that neglect correlation between positions of electrons and even most of the electron-electron repulsive interactions, scaling the remaining interactions to fit empirical data. At the most accurate end, the full configuration interaction method$^{147}$ solves Eq. (1.9) by a spectral method in a finite basis, whose basis functions are roughly based on the wavefunction of the hydrogen atom. Unfortunately, this multi-electron basis consists of all cross products of the hydrogen wavefunction-based single-electron basis functions, and thus the size of this multi-electron basis scales exponentially with the number of electrons. Coupled-cluster methods,$^{15}$ which solve a system of equations formed by a smaller number
of projections of Eq. (1.9) (scaling only polynomially with the number of electrons), can achieve very close agreement with experiment.\textsuperscript{20} But this is only practical for fairly small systems, because the number of single-electron basis functions and the degree of the polynomial generally need to be fairly large to achieve this agreement.

Quantum-chemical calculations play an important role in parameterizing molecular-mechanics force fields.\textsuperscript{18} Nevertheless, these force fields may fall short of the quantum-chemistry methods on which they were based, in three ways. First, for a given molecule, the dependence of that molecule’s energy on its conformation is not exactly given by any function of the form that is assumed by the force fields, due to effects like electronic polarization. This can be alleviated to some degree by giving the force field a more flexible functional form, e.g., by employing a polarizable force field.\textsuperscript{85,116} Second, for a given type of interaction (e.g., Coulombic), the parameters (e.g., charges) governing that interaction may depend in a complex way on the chemical composition of the molecule, possibly leading to errors in estimating the interaction energy between two atoms unless the parameters have been computed specifically for the pair of molecules containing those atoms. Third, even for a particular force field and a given set of variable parameters, the parameters used in a standard force field may not be those that best fit the system we are using. For example, the AMBER\textsuperscript{18,127,168} and CHARMM\textsuperscript{9} force field have very similar functional forms and sets of variable parameters, but often return significantly different values, which cannot both be right. Inclusion of quantum-mechanical calculations in energy calculations can significantly improve agreement with experiment.\textsuperscript{180} Improvements in energy modeling can be achieved in part by learning from empirical data, but quantum-mechanical calculations are also important since they give a much more complete (if possibly less accurate) picture of energetics at the atomic level.

Fortunately, significant progress has been made towards performing accurate quantum chemistry calculations on larger systems, like biomolecules.\textsuperscript{52} These ap-
proaches typically rely in some way on locality of interactions. Correlation between the positions of different electrons, as least as modeled by coupled cluster, is mostly between electrons in the same functional group. Furthermore, although most quantum chemistry methods require precomputation of a matrix of electron repulsion integrals whose size scales as the fourth power of the number of single-electron basis functions, this scaling can be reduced for larger systems if one neglects matrix elements exceeding a fixed cutoff. Large systems have also been treated with divide-and-conquer approaches, including the Molecular Fraction with Conjugate Caps approach, which breaks up a protein into covalently “capped” residues, whose interactions with a ligand are each calculated separately. As the accuracy and efficiency of large-system quantum chemistry advances, it will become an even more useful tool for building better energy functions for design.

1.5 The legend of Perseus

Protein design calculations are vulnerable to many sources of error, each easily capable of causing designs to fail. Addressing this issues can seem like fighting a formidable monster. In dealing with this, I suggest we take inspiration from a famous legend about a hero who faced multiple monsters.

According to myth, Perseus set out to slay the Gorgon Medusa, a humanoid monster with snakes instead of hair who turned every mortal who looked upon her to stone. Armed with gifts from the gods—flying sandals from Hermes, and crucially, a reflective shield from Athena—he slew the monster by looking only at her reflection in the shield. He severed her head and used it to turn the enormous sea monster Cetus to stone, saving the princess Andromeda, and finally to petrify his evil stepfather. He then gave the head to Athena, who fixed it to her shield. Likewise, when we want to do protein design, the difficulty of accurate modeling leads us into twin pitfalls—either “blind” search that makes drastic approximations and lacks provable
guarantees of accuracy, like a failed Medusa-hunter who keeps his eyes closed to avoid being petrified, or extremely computationally expensive methods that try to “look upon” the full complexity of the biophysics, and as a result are “petrified” by an impractically slow rate of computation. Fortunately, we too have gifts from others that can help us in our endeavor: advances in computing—both hardware and software—and our fairly comprehensive understanding of the underlying physics. Like Perseus viewing Medusa clearly in his shield, we can closely approximate the physics of protein and other chemical design if we use sufficiently efficient algorithms.

So let’s face the one monster of accurately modeling biophysics for purposes of designing chemicals, and if we may slay it, many others will fall easily, and we can fix its head to the shield of the entire scientific community to face all similar challenges in the future.

1.6 Outline of thesis

1. Chapter 2 introduces a provably accurate algorithm for protein design with continuous sidechain and backbone flexibility. This work has been published:

Mark A. Hallen, Daniel A. Keedy, and Bruce R. Donald. Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility.


2. Chapter 3 introduces a compact, polynomial-based representation of continuous energy surfaces that significantly reduces the number of energy function calls needed in protein design with continuous flexibility. This work has also been published:

Mark A. Hallen, Pablo Gainza, and Bruce R. Donald. A compact
representation of continuous energy surfaces for more efficient protein

3. Chapter 4 introduces a provably accurate algorithm for multistate protein de-
sign. This work has been presented at the RECOMB conference and also
published as a journal version:

Mark A. Hallen and Bruce R. Donald. comets (Constrained Op-
timization of Multistate Energies by Tree Search): A provable and
efficient protein design algorithm to optimize binding affinity and
specificity with respect to sequence. Journal of Computational Biol-

4. Chapter 5 introduces an algorithmic improvement for A* search in protein
design. This work has been published as part of a larger project led by Kyle
Roberts and Pablo Gainza:

Kyle E. Roberts, Pablo Gainza, Mark A. Hallen, and Bruce R. Don-
ald. Fast gap-free enumeration of conformations and sequences for
protein design. Proteins: Structure, Function, and Bioinformatics,

5. Chapter 6 introduces an antigen structural compatibility (ASC) score that mea-
sures the suitability of a protein construct as a potential HIV vaccine candidate,
based on structural information. This work has been published as part of a
larger project that included the first unliganded crystal structure of the HIV
envelope trimer, led by Young Do Kwon, Marie Pancera, Priyamvada Acharya,
and Ivelin Georgiev:

6. Chapter 7 introduces a machine learning method to faithfully represent advanced modeling features, like continuous flexibility and Poisson-Boltzmann solvation energies, in a form suitable for input to efficient discrete protein design algorithms like DEE/A*. It has been accepted to the RECOMB conference:

When I was rotating in Bruce Donald’s wet lab before joining his group, he mentioned at a lab meeting how Daniel Keedy was characterizing a new protein backbone motion found in crystallographic alternates, known as the shear. Bruce asked if anyone wanted to try incorporating this motion into our design framework, and I took on this project. I ended up building a general framework for modeling backbone perturbations, which became DEEPer. I initially considered the name perDEE, along the same lines as previous algorithms like minDEE\textsuperscript{48} and brDEE,\textsuperscript{46} but I realized that it sounded cool when reversed (“DEEPer”), and this started a pattern for me of using semantically meaningful acronyms to name my algorithms. We published DEEPer in Proteins:

Mark A. Hallen, Daniel A. Keedy, and Bruce R. Donald. Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility. Proteins:
Summary  Computational protein and drug design generally require accurate modeling of protein conformations. This modeling typically starts with an experimentally-determined protein structure and considers possible conformational changes due to mutations or new ligands. The DEE/A* algorithm provably finds the GMEC (global minimum-energy conformation) of a protein assuming the backbone does not move and the sidechains take on conformations from a set of discrete, experimentally-observed conformations called rotamers. DEE/A* can efficiently find the overall GMEC for exponentially many mutant sequences. Previous improvements to DEE/A* include modeling ensembles of sidechain conformations and either continuous sidechain or backbone flexibility. We present a new algorithm, DEEPer (Dead-End Elimination with Perturbations), that combines these advantages and can also handle much more extensive backbone flexibility and backbone ensembles. DEEPer provably finds the GMEC or, if desired by the user, all conformations and sequences within a specified energy window of the GMEC. It includes the new abilities to handle arbitrarily large backbone perturbations and to generate ensembles of backbone conformations. It also incorporates the shear, an experimentally-observed local backbone motion never before used in design. Additionally, we derive a new method to accelerate DEE/A*-based calculations, indirect pruning, that is particularly useful for DEEPer. In 67 benchmark tests on 64 proteins, DEEPer consistently identified lower-energy conformations than previous methods did, indicating more accurate modeling. Additional tests demonstrated its ability to incorporate larger, experimentally-observed backbone conformational changes and to model realistic conformational ensembles. These capabilities provide significant advantages for modeling protein mutations and protein-ligand interactions.
2.1 Introduction

Accurate computational protein and drug design requires a biophysically reasonable representation of protein conformations and an efficient method to search protein conformational and sequence space. The full conformational and sequence space available to a protein is too vast to search completely. One approach to this problem is to model the overall fold of a protein on a coarse level, and then to introduce additional detail as needed; this has allowed the de novo design of proteins with a desired overall fold. Alternatively, a protein or protein-ligand complex similar to an experimentally-determined structure can be modeled by searching conformations similar to the empirical structure, allowing design of novel proteins or protein-binding drugs. In this case, the conformational search space can be represented by choosing a set of flexible residues, whose conformations are likely to change in response to mutations or new ligands. For protein design, the search can be performed over sequence space as well because multiple amino acid types can be considered for each flexible residue. The conformation and sequence of the entire protein is then represented as a tuple of conformations and amino acid types for the flexible residues. Since the flexibility of protein sidechains is mostly in their dihedral angles and these angles are usually found in discrete clusters called rotamers, previous algorithms have typically discretized the conformational space of a residue into sidechain rotamers. Using an energy function, which assigns an energy to each conformation, this scheme casts protein design into a combinatorial optimization problem. Empirical molecular-mechanics energy functions are typically used for this purpose.

The sequence and structure of a native or designed protein or complex can be modeled by finding its Global Minimum-Energy Conformation (GMEC): the sequence and conformation of the protein that minimizes the energy function. The
GMEC, like any sequence and conformation, can be represented as a tuple of conformations and amino acid types of individual residues. However, consideration of the GMEC alone neglects important entropic effects, because proteins in solution populate many conformations. Consequently, considering an ensemble of low-energy conformations results in more accurate predictions of binding and catalysis. This has led to three main types of search methods over protein conformational and sequence space: (a) stochastic, heuristic methods that try to find the GMEC, (b) methods to provably find the GMEC, and (c) methods to generate conformational ensembles. All these methods require a biophysical model as input. The biophysical model specifies not only the conformations and sequences available to the protein, but also the energy function used to score those conformations and sequences. Often, the space of allowed conformations and sequences is defined using (i) a starting structure, (ii) a set of residues allowed to differ from the starting structure, and (iii) a library of rotamers for those residues to populate. The allowed sequence space to search may be just one sequence, or an astronomical number of combinatorial possibilities. Methods of type (c) may or may not have provable guarantees of accuracy. Our lab has developed methods of types (b) and (c) with provable guarantees of accuracy and successfully tested them in vitro and in vivo, generating mutants and ligands that may aid in synthetic biology and in the treatment of bacterial diseases, cystic fibrosis, HIV, and leukemia.

Because finding the GMEC is NP-hard, several heuristic algorithms have been developed for this problem. These algorithms can find a relatively low-energy conformation fairly rapidly, albeit without any guarantees of completeness or accuracy. (We will refer to an algorithm as complete if it is guaranteed to consider its entire search space and as accurate if its answers are always correct for the algorithm’s input, at least within a given margin of error). Metropolis Monte
Carlo-based methods have been used to search the space of rotamers. These methods can incorporate continuous flexibility by applying the Monte Carlo with minimization method, and have also incorporated some backbone flexibility. The GMEC has also been approximated by molecular-dynamics simulation of protein folding, offering the added benefit of modeling kinetics. But molecular dynamics is rarely computationally tractable for protein design due to the large sequence space and large number of folding intermediate states that must typically be considered. Unlike DEEPer, the algorithm introduced in this work, molecular dynamics only considers one sequence at a time. Genetic algorithms and numerical global minimization methods have also been used to search for the GMEC, and the heuristic FASTER method, which combines deterministic and stochastic steps, has been developed specifically for this search as well.

On the other hand, algorithms to provably find the GMEC are also available. Because they are guaranteed to find the optimal conformation and sequence given a biophysical model, they consider all functionally significant conformational and sequence changes allowed by the model, and thus they will generally produce more empirically accurate results than heuristic algorithms using the same model. Provably algorithms are also useful for evaluating and improving models because they ensure that any discrepancies between experimental results and the predictions of the algorithm are due solely to the inadequacies of the model, and not to the algorithm. Most provable methods for identifying the GMEC start by eliminating rotamers that cannot participate in the GMEC, by using Dead-End Elimination (DEE). This step is typically followed by the A* search algorithm, which finds the GMEC using the unpruned rotamers. If A* is continued after the GMEC is found, it will output the other conformations of the protein gap-free in ascending order of energy. Thus, it can be used to identify all conformations whose energy is within a specified energy interval $E_w$ of the GMEC energy.
This DEE/A* framework has been extended to run more efficiently, to model ensembles, and to include continuous flexibility, enhancements which come with provable guarantees of accuracy and are applicable to the simultaneous search of mutant sequence space and conformational space.\textsuperscript{28}

First, DEE has been extended to eliminate more rotamers as well as tuples of rotamers at different residues.\textsuperscript{53} The related Bounds\textsuperscript{55} and conformational-splitting DEE\textsuperscript{130} algorithms prune some rotamers that ordinary DEE cannot prune. The HERO algorithm\textsuperscript{55} combines these improvements into a single algorithm by applying them successively and iteratively.

Next, DEE/A* has also been extended to allow continuous flexibility within each rotameric state, producing “minimization-aware” algorithms whose pruning is still provable even with this continuous flexibility. In these methods, the conformation is no longer uniquely defined by a tuple of rotamers, but once a rotamer has been assigned to each residue, the GMEC can be found by local minimization. Correspondingly, the “minimization-aware” pruning is followed by a modified A* search that enumerates tuples of rotamers in order of a lower bound on the conformational energy, rather than in order of the conformational energy. The true conformational energies are then obtained by local minimization. The minDEE algorithm\textsuperscript{48} allows the sidechain dihedrals to vary continuously within a specified interval about the modal values for a given rotamer, while the BD\textsuperscript{44} and brDEE\textsuperscript{46} algorithms allow small backbone conformation adjustments while modeling discrete sidechain rotamers. In BD, the backbone is continuously flexible within a voxel. In brDEE, the backbone states are discrete but systematically and closely spaced; the backbone states for each residue are represented using a single parameter. iMinDEE,\textsuperscript{40} a significantly more efficient version of minDEE, provides a more powerful minimization-aware pruning criterion. \(i_r\) denotes a rotamer \(r\) at residue position \(i\) in the protein, where \(r\) is chosen from a set \(R\), and \(1 \leq i \leq n\), where \(n\) is the number of residues in the protein.
It prunes a rotamer $i_r$ at a residue $i$ if, for any other rotamer $i_t$ at residue $i$,

\[ E_{\Theta}(i_r) - E_{\Theta}(i_t) + \sum_{j \neq i} \min_{j_s}(E_{\Theta}(i_r, j_s) - E_{\Theta}(i_t, j_s)) > E_w + I \quad (2.1) \]

where $E_{\Theta}$ is a lower bound on the internal energy of a rotamer or the interaction energy for a pair of rotamers (see 48 and Section 2.2.5). $I$, the “pruning interval,” is an upper bound on the energy difference between the GMEC and the lowest lower-bound on a conformational energy, calculated by summing the single-rotamer and pairwise lower-bound energies. This equation will prune rotamers not found in conformations within a user-specified energy interval $E_w$ of the GMEC. If only the GMEC is desired, $E_w$ can be set to 0. In summary, methods based on DEE/A* are now available to provably and efficiently find the GMEC in a search space defined by sidechain rotamers as well as a small amount of continuous flexibility in either the sidechains or the backbone.

Finally, the $K^*$ algorithm 28, 48, 108 compares the conformational ensemble of a protein with a ligand to its ensemble without the ligand. This is done by approximating their partition functions within a provably guaranteed margin of error (given the biophysical model). $K^*$ is an extension for DEE/A* that can be applied to any of the minimization-aware variants of DEE/A*, allowing the modeling of continuous flexibility in computing partition functions. Unlike GMEC-based analysis, this method accounts for entropic effects, allowing substantial improvements in the accuracy of binding predictions. $K^*$ can identify tight-binding sequences from a large sequence space efficiently and with provable accuracy. It has been shown to optimize design for affinity 13, 28, 48, 108, 139.

Further increases in accuracy for modeling mutant or new ligand-bound conformations require including more backbone flexibility, simultaneously with continuous sidechain flexibility (Table 2.1). Three lines of evidence support this need. First,
Table 2.1: Continuous flexibility in DEE-based protein design algorithms

<table>
<thead>
<tr>
<th>Backbone flexibility</th>
<th>Sidechain flexibility</th>
<th>Continous</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>DEE&lt;sup&gt;25&lt;/sup&gt;</td>
<td>minDEE&lt;sup&gt;,48&lt;/sup&gt; iMinDEE&lt;sup&gt;,40&lt;/sup&gt;</td>
</tr>
<tr>
<td>Discrete</td>
<td>brDEE&lt;sup&gt;46&lt;/sup&gt;</td>
<td>DEEPer</td>
</tr>
<tr>
<td>Continuous</td>
<td>BD&lt;sup&gt;,44&lt;/sup&gt;</td>
<td>DEEPer</td>
</tr>
</tbody>
</table>

mutations have often been found to induce substantial backbone conformational changes.<sup>32,112,174</sup> Second, modeling of continuous sidechain flexibility has been found to substantially enhance the accuracy of rotamer assignments and the identification of low-energy sequences.<sup>40</sup> Third, backbone as well as sidechain degrees of freedom contribute significantly to conformational ensembles.<sup>86,181</sup> Thus, we present a new algorithm, Dead-End Elimination with Perturbations (DEEPer), that incorporates continuous sidechain and backbone flexibility in a provably complete search. This search always finds the GMEC or a gap-free list of the lowest-energy conformations, depending on the user’s preference, and it interfaces directly with $K^*$ to predict binding affinity using ensembles, maintaining $K^*$’s provable guarantees on accuracy but giving it the novel ability to account for some backbone conformational entropy. This means that it can estimate binding affinity more accurately. To maintain search efficiency as this additional flexibility is added, increased pruning is necessary. We show that a novel pruning algorithm, *indirect pruning*, can alleviate the increased computational cost that is incurred with substantial backbone flexibility, while maintaining a provable guarantee of accuracy (Section 2.2.5). Indirect pruning combines the minimization awareness of iMinDEE<sup>,40</sup> with features of Goldstein singles and tuples pruning<sup>,53</sup> and Bounds pruning<sup>,55</sup>

DEEPer is also well adapted for including backbone perturbations based on information from previous experiments. These can include backbone conformational
changes previously observed for the protein of interest, if available, but we also feature two commonly observed forms of backbone flexibility “moves,” the shear and backrub. While the backrub has been incorporated in previous design algorithms, the shear has not. Both moves were identified in high-resolution crystal structures by examination of anisotropic electron density and of crystallographic alternates, which are multiple conformations of the same segment of a protein that are each observed in some portion of the molecules in a single crystal. Backrubs also observably accommodate sequence changes in natural proteins, thus validating their use for accommodating engineered mutations in protein design. Shears are expected to similarly aid searching of combined conformational and sequence space. Shears and backrubs are relatively small backbone perturbations, so they are suitable for modeling backbone perturbations within helix and sheet secondary structures, where the backbone motion is limited. Together with the five other types of perturbations implemented in DEEPer (Table 2.2), shears and backrubs allow a wide range of backbone flexibility in all types of secondary structure.

Thus, by introducing DEEPer, this paper makes the following contributions:

1. A provable algorithm for searching a continuously-flexible conformational space, including both sidechain and backbone degrees of freedom, as well as simultaneous combinatorial search of mutant sequence space.

2. The extension of iMinDEE to minimize over backbone as well as sidechain degrees of freedom.

3. A general, provable method for introducing backbone perturbations based on previous studies of protein backbone flexibility, such as the shear motion, into protein design calculations.

4. A provable method for modeling ensembles of conformations in which confor-
mations are allowed to vary with respect to sidechain and backbone degrees of freedom, during a search over combined sequence and conformational space.

5. An implementation of DEEPer in our laboratory’s open-source osprey protein-design software package,\textsuperscript{13,36,48,49} available by request as free software.


7. Computational tests on 64 proteins to demonstrate that DEEPer identifies lower-energy structures and sequences than previous methods, and tests to demonstrate that it computes biophysically reasonable ensembles of backbone conformations.

2.2 Methods

DEEPer builds three new components onto iMinDEE.\textsuperscript{40} First, we introduce a new representation of backbone flexibility, which we call perturbations (Section 2.2.1). We introduce seven diverse types of perturbations, some continuous and some discrete (Section 2.2.2). The second component is the concept of a residue conformation, or $RC$ (Section 2.2.4). We can represent protein conformations by assigning a residue conformation to each residue. This is analogous to the way that rigid-backbone methods like iMinDEE represent protein conformations by assigning a sidechain conformation to each residue. Taken together, the concepts of perturbations and RCs provide a general method for introducing degrees of freedom affecting multiple residues into the framework of dead-end elimination. The third component, indirect pruning (Section 2.2.5), is an enhancement of the dead-end elimination algorithm designed to be especially helpful in the presence of perturbations. These topics will be discussed in the following subsections. The general flow of the DEEPer algorithm is illustrated in Fig. 2.1.
DEEPer begins with the selection of a set of perturbations to use (this step may be manual or automated). These perturbations define a set of RCs for each residue. RCs that cannot be in low-energy conformations are pruned, using the new indirect pruning algorithm (Section 2.2.5) as well as previous pruning algorithms. A* with minimization (see Introduction), as in iMinDEE, is then used to output low-energy conformations. The result is the GMEC and a gap-free list of all conformations and sequences within a user-specified interval $E_w$ of the GMEC. This list can be used to select mutant sequences to synthesize for experimental testing, either by (i) selection of the sequences with the lowest-energy conformations, or (ii) by a provably-good approximation algorithm to calculate the binding affinities via the $K^*$ software module.

2.2.1 Perturbations

DEEPer searches the space of sidechain dihedrals within a specified interval around the modal dihedral value for each rotamer. This interval is $\pm 9^\circ$ by default, except for proline, whose dihedrals cannot rotate freely. All other flexibility is represented in the form of perturbations, which can be any conformational adjustments that commute with rotations of the non-proline sidechain dihedrals (Fig. 2.2). The effect of each perturbation is quantified using a single, scalar perturbation parameter. Some perturbations can feasibly be represented using a continuous range of perturbation parameter values, while other perturbations admit only discrete values of the perturbation parameter (see Section 2.2.2). These will be referred to as continuous and discrete perturbations, respectively. Each perturbation has a defined set of residues that it can affect. Perturbations should be chosen to keep these sets of residues small whenever possible, since the tractability of DEEPer relies on relatively few perturbations affecting each residue.
Figure 2.2: In DEEPer, the conformation of a protein is defined by multiple degrees of freedom. Sidechain dihedrals, such as the $\chi_1$ dihedrals for the residues shown, affect the conformation of only one residue. Perturbations, such as the backrub shown, can affect the conformation of several residues (three in this case; black balls denote the boundaries of the backbone region affected by the backrub).

2.2.2 Types of perturbations

We implemented DEEPer with seven types of perturbations (Table 2.2, Fig. 2.3).

First of all, backrubs (Fig. 2.3a, Fig. 2.4a) are applied as in brDEE. Briefly, if the backrub affects residues $i$ through $i + 2$, the section of protein chain between residue $i$’s C$_\alpha$ and residue $i + 2$’s C$_\alpha$ is rotated about the axis defined by those two C$_\alpha$s, and the rotation angle, i.e. the “primary backrub angle,” is the perturbation parameter. Then, in order to reduce bond-angle distortion and maintain any pre-existing hydrogen bonds, a smaller counter-rotation is applied about the peptide plane’s C$_\alpha$-C$_\alpha$ axis for both of the peptide planes in this section of protein chain. The angle of each counter-rotation is 70% of the angle that minimizes the displacement of the carbonyl oxygen caused by the backrub. In addition to moving the backbone, backrubs reorient the sidechain of residue $i + 1$ in a direction perpendicular to the local chain direction. Backrubs tend to be found in extended conformations: $\beta$ sheets
Figure 2.3: Types of perturbations implemented in DEEPer. Each perturbation is from red backbone and orange sidechains to blue backbone and purple sidechains. Black balls denote the boundaries of the backbone region affected by the perturbation. (a) Backrub. (b) Shear. (c) Loop closure adjustment. (d) Secondary structure adjustment. (e) Partial structure switch. (f) Full structure switch. Two crystal structures of the motor protein Ncd related by a major conformational change are shown (PDB codes 2NCD and 1N6M). (g) Proline flip.
Table 2.2: Types of perturbations

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Continuous?</th>
<th>Residues affected</th>
<th>Where usually found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shear *</td>
<td>Primary shear angle</td>
<td>Yes</td>
<td>4</td>
<td>Helices</td>
</tr>
<tr>
<td>Backrub</td>
<td>Primary backrub angle</td>
<td>Yes</td>
<td>3</td>
<td>β sheets, loops</td>
</tr>
<tr>
<td>Loop closure adjustment *</td>
<td>Solution set from discrete set</td>
<td>No</td>
<td>3</td>
<td>Loops</td>
</tr>
<tr>
<td>Secondary structure</td>
<td>Solution set from discrete set</td>
<td>No</td>
<td>3-4</td>
<td>Loop-helix or loop-sheet borders</td>
</tr>
<tr>
<td>adjustment *</td>
<td>Structure from discrete set</td>
<td>No</td>
<td>As desired</td>
<td>Anywhere</td>
</tr>
<tr>
<td>Partial structure switch</td>
<td>Structure from discrete set</td>
<td>No</td>
<td>All</td>
<td>Entire protein</td>
</tr>
<tr>
<td>Full structure switch</td>
<td>Structure from discrete set</td>
<td>No</td>
<td>All</td>
<td>Entire protein</td>
</tr>
<tr>
<td>Proline flip</td>
<td>Boolean: flip or no flip</td>
<td>No</td>
<td>1</td>
<td>Prolines</td>
</tr>
</tbody>
</table>

*These perturbations cannot, to our knowledge, be performed by any previous protein design algorithm. DEEPer also offers novel combinations of perturbations that have been modeled previously; for example, no previous provable algorithm modeled overlapping backrubs. DEEPer is also novel in combining these perturbations with continuous sidechain flexibility, which is represented as orthogonal to perturbations in conformational space.

or loops in which the mainchain is relatively extended.23

On the other hand, shears (Fig. 2.3b, Fig. 2.5, Fig. 2.4b) move the central peptide of a three-peptide segment parallel to the chain direction; they occur primarily in α helices, particularly at the termini (Jane and David Richardson, personal communication). Shears have previously been proposed by Smith and Kortemme.149

The shear motion, if affecting residues $i$ through $i + 3$, rotates the mainchain atoms from residue $i$’s C$_α$ to residue $i + 1$’s C$_α$ as well as residue $i + 1$’s sidechain about residue $i$’s C$_α$ in the plane defined by residue $i$’s C$_α$, residue $i + 1$’s C$_α$, and residue $i + 2$’s C$_α$. The angle of this rotation, i.e. the “primary shear angle,” is the perturbation parameter. Then residue $i + 2$’s sidechain and the mainchain segment from residue $i + 2$’s C$_α$ to residue $i + 3$’s C$_α$ are rotated as a rigid body about residue $i + 3$’s C$_α$ in the plane defined by residue $i + 1$’s C$_α$, residue $i + 2$’s C$_α$, and residue $i + 3$’s C$_α$. 

30
C\textsubscript{\(\alpha\)}. The angle of rotation is calculated to keep the distance between residue \(i + 1\)’s C\textsubscript{\(\alpha\)} and residue \(i + 2\)’s C\textsubscript{\(\alpha\)} at its unperturbed value. Finally, the mainchain atoms between residue \(i + 1\)’s C\textsubscript{\(\alpha\)} and residue \(i + 2\)’s C\textsubscript{\(\alpha\)} are rotated about the axis defined by those C\textsubscript{\(\alpha\)}s to make residue \(i + 1\)’s carbonyl C–O bond vector as close as possible to its unperturbed direction. As in the case of backrubs, this counter-rotation is intended to maintain hydrogen bonds and reduce bond-angle strain.

Thus, although they are local, affecting 3-4 residues each, backrubs and shears together allow a given C\textsubscript{\(\alpha\)} atom to move in any direction by a small amount. The component rotations of shears and backrubs are illustrated in Fig. 2.4.

**Loop closure adjustments** (Fig. 2.3c) are also perturbations affecting three residues, but unlike backrubs, they do so without changing any bond angles or lengths. The space of such motions is described in\textsuperscript{19} and the algorithm from that work is used to implement them. It solves a system of polynomial equations, and each solution is associated with a possible conformation of the tripeptide. Given a residue \(i\), these equations find the different possible segments of protein chain from residue \(i\)’s C\textsubscript{\(\alpha\)} to residue \(i + 2\)’s C\textsubscript{\(\alpha\)} that maintain specified bond angles, lengths, and \(\omega\) dihedrals without changing the conformation of any other part of the protein at all. For DEEPer, these specified values are taken from the pre-perturbation conformation. The perturbation parameter is a discrete index specifying which solution of the polynomial equations is used to create the perturbed conformation. Loop closure adjustments can be generated using the same component rotations as backrubs (Fig. 2.4a), though the angles of these rotations and the method by which the angles are determined are completely different (backrubs are biophysically feasible because the distortions in bond angles are small, while loop closure adjustments are biophysically feasible because the rotation angles are computed to avoid any bond angle changes).

**Secondary structure adjustments** (Fig. 2.3d) are perturbations that change secondary structure. If one of the residues moved by a loop closure adjustment motion
Figure 2.4: Geometry of multi-step perturbations. As in Fig. 2.3, each perturbation is from red backbone and orange sidechains to blue backbone and purple sidechains. (a) The two steps of a backrub perturbation, starting from an ideal poly-Ala β sheet. First, the dipeptide is rotated by the specified amount (here, +10°) around the Cα1-Cα3 axis (left). Then, each individual peptide is rotated by some fraction (70% by default, here corresponding to -7.0° and -5.6°) of the amount needed to restore the two carbonyl oxygens to their original positions (right). These same rotations, albeit with very different angles, are used for loop closure adjustments and secondary adjustments. (b) The three steps of a shear perturbation, starting from an ideal poly-Ala α helix. First, the N-terminal peptide is rotated by the specified amount (here, +5°) around Cα1 in the Cα1-Cα2-Cα3 plane (left). Then, the C-terminal peptide is rotated around Cα2 in the Cα2-Cα3-Cα4 plane such that the original Cα2-Cα3 distance (dotted line) is maintained (middle). Finally, the original middle peptide is rotated and translated to fit into the gap and to match the original carbonyl orientation (right).
Figure 2.5: The shear backbone motion. (a) Simple Cα-only representation of shear, viewed down the axis of an ideal α helix (light colors). Shears of 5° (darker) and 10° (darkest) swing the central Cα2–Cα3 peptide (cyan) sideways by coordinated rotations of the Cα1–Cα2 peptide (blue) and Cα3–Cα4 peptide (green). (b) All-atom representation of shear, viewed from the side of the ideal α helix (i.e. rotated 90° from (a)). Shears of 2° over a 10° range are shown. The central carbonyl is notably displaced parallel to the central peptide. One endpoint conformation is marked by balls and line segments colored as in (a).

is at the beginning or end of a helix or sheet while the other two are in a loop, then the secondary structure of the helix or sheet residue can effectively be changed to loop structure. Alternatively, a similar motion can be used to convert the first residue in a loop to the secondary structure of the residues preceding it, or to convert the last residue in a loop to the secondary structure of the residues following it. The backbone dihedrals of the residue to be changed are adjusted to be appropriate for the helix or sheet by copying the dihedrals of the adjacent helix or sheet residue. Then, the tripeptide on the other side of the residue from the helix or sheet is placed into the structure using the loop closure equations from.\(^{19}\) The perturbation parameter is a discrete index identifying a solution to the equations, as for a loop closure adjustment.

Partial structure switches (Fig. 2.3e) take a section of backbone from another
structure and replace the corresponding section of the structure being modeled with it. If needed, the new section of backbone is rotated and translated to fit into the gap. The new and old sections must be very similar in end-to-end length, because otherwise it is impossible to insert the new section without changing its backbone conformation significantly. However, the implementation offers two methods to slightly adjust end-to-end length: scaling coordinates uniformly or using the tripeptide closure method from.\textsuperscript{19} Adjusting the length by more than 0.2 Å is set to raise an error in the implementation of DEEPer. Partial structure switches can be used to incorporate different ligand conformations, positions, and orientations as well. This perturbation allows the incorporation of experimentally-observed alternate backbone conformations—e.g. crystallographic alternates, NMR ensembles, or homologous structures—or computational predictions like docking poses or computationally-derived ensembles of loop conformations (e.g. from the POOL algorithm\textsuperscript{156}). For a \textit{full structure switch} (Fig. 2.3f), the entire system is modeled with a different crystal structure (or other structure) for the perturbed RCs, so this affects all residues of the system.

\textit{Proline flips} (Fig. 2.3g) are perturbations that flip the ring pucker of proline from \textit{endo} to \textit{exo} or vice versa, thus flipping the signs of all the sidechain dihedrals. Switching the proline peptide conformation from \textit{cis} to \textit{trans}, on the other hand, could be accomplished by a partial structure switch.

\textit{2.2.3 Implementation and selection of perturbations}

To flip a proline ring’s pucker, the $C_\gamma$ atom is moved, holding all bond lengths as well as the $C_\alpha$-$C_\beta$-$C_\gamma$ angle constant. This limits the $C_\gamma$ position to at most two positions; if there is a position other than the current one (as there essentially always is for a valid conformation), then the flip consists of moving the $C_\gamma$ to that position. This is similar to the method of Ho et al.\textsuperscript{77} Then the hydrogens are placed
to induce two-fold rotational symmetry at each of the four tetrahedral carbons in
the sidechain, using the hydrogen-carbon-hydrogen angles from Allen et al.\textsuperscript{4} Unlike
other perturbations, this does affect the sidechain dihedrals, but these dihedrals
are not continuously-flexible and are not changed by minDEE or by the normal
sidechain-dihedral adjustments that DEEPer inherits from minDEE. This is because
the dihedrals cannot be changed without affecting the bond lengths and angles.

After perturbations are applied, the sidechain is moved as a rigid body to put
the C\textsubscript{\textbeta} in its ideal position (as in the sidechain idealization feature\textsuperscript{114} in KiNG\textsuperscript{14})
and to return the \( \chi_1 \) dihedral to its value before the perturbation. The latter step is
to make sure perturbations commute with sidechain dihedral adjustments and is not
applied for proline, so proline flips are not reversed at this stage. Proline is, however,
further idealized by imposing an ideal C\textsubscript{\textdelta}-N-C\textsubscript{\textalpha} angle and then placing the C\textsubscript{\gamma} and the
hydrogens in the same manner as for a proline flip, except choosing the C\textsubscript{\gamma} position
that does not flip the ring pucker. Backbone conformations that do not allow any
valid proline ring conformation are assigned infinite energy, preventing them from
being considered as GMEC candidates or low-energy ensemble members. In light of
these modifications, the DEEPer version of OSPREY now supports mutations to and
from proline.

When computing energy lower bounds and final minimized conformations, contin-
uous perturbations are minimized along with sidechain dihedrals using the steepest
descent-based local minimization algorithm currently implemented in OSPREY.

We have implemented an automatic perturbation selection module for OSPREY,
which generates shears, backrubs, loop closure adjustments, secondary structure ad-
justments, and proline flips. Partial and full structure switches can be included if the
user provides alternate-structure information. The set of flexible residues is taken
as input, as in iMinDEE. For any set of consecutive flexible residues with suitable
secondary structure and the correct number of residues for a given perturbation type,
one perturbation of that type is generated. Perturbed backbone conformations are run through a Ramachandran filter with a used-specified cutoff. Ramachandran data from\textsuperscript{114} were used, with glycine, proline, and pre-proline residues all treated as special cases. If specified by the user, perturbed backbone conformations are also run through a backbone RMSD filter. The latter rejects a perturbed backbone conformation (and its associated RCs) if it is too similar to another backbone conformation that is being considered, as measured by backbone heavy-atom RMSD. The RMSD filter can widen the diversity of sampled backbone conformations per unit computational cost. Parameter intervals for shears and backrubs can be specified by the user. Proline flips are placed at any residues where prolines are allowed. This automatic perturbation selector can be applied straightforwardly to any protein system, though sometimes better results might be obtained by manual adjustment of perturbations, particularly when experimental data on alternate backbone conformations is available. For example, crystallographic alternates can be used for this purpose. Another possibility is to use crystal structures of different conformations of the same or similar proteins, such as the different complementarity-determining region loop structures found for antibodies.\textsuperscript{3} The automatic perturbation selector can incorporate manually selected perturbations and then select additional perturbations automatically.

2.2.4 Residue conformations (RCs)

To use the perturbations described above, we require a new way to represent possible conformational states. To meet this need, DEEPer introduces residue conformations (RCs), which are analogous to rotamers in traditional DEE, but also incorporate the perturbations as additional degrees of freedom. In this section, we first introduce the philosophy and nomenclature behind RCs. We then lay out DEEPer’s strategy for representing RCs in the presence of perturbations affecting multiple residues, performing energy minimization with the correct order of operations even with non-
Figure 2.6: iMinDEE can be viewed as a global minimization algorithm over many degrees of freedom (DOFs) that searches a set of energy wells. (a, b) RCs represent likely locations for energy wells of particular residues. The notation $i_r$ is used to represent the $r^{th}$ RC of residue $i$. (c) This leads to an $n$-dimensional lattice for the possible energy well locations for $n$ residues. (d) We prune RCs and tuples of RCs using iMinDEE. Then $A^*$ is used to identify the energy wells with the lowest lower bounds, and local minimization is used to search these wells for the GMEC.

commutative perturbations, and adding special wild-type rotamers to some RCs.

Like rotamers in traditional DEE or iMinDEE, $^{40}$ RCs are prunable, enumerable states of a residue. Specifically, consider a residue $i$ with $s$ sidechain dihedrals that is affected by $c$ continuous perturbations and $d$ discrete perturbations. Then each RC of $i$ may be represented as a $(2s + 2c + d + 1)$-tuple whose elements are (1) a maximum and minimum value for each sidechain dihedral angle (following minDEE, we have implemented this as the modal dihedral for the rotamer $\pm 9^\circ$), (2) a maximum and minimum parameter value for each continuous perturbation, (3) a single parameter
value for each discrete perturbation, and (4) an amino acid type. For a residue with
a particular amino acid type whose conformation is described by a set of sidechain
dihedral angles and perturbation parameter values, we say that the residue is in
a given RC when its amino acid type and discrete perturbation parameters match
those of the RC and its sidechain dihedrals and continuous perturbation parameters
each fall between the minimum and maximum values for the RC. Let us use the
notation $i_r$ for an RC affecting residue $i$, by analogy to the notation for rotamers.$^{25}$

Like a rotamer in minDEE,$^{40,48}$ an RC represents a set of closely-related confor-
mations. The sets of conformations are meant to be small enough that the GMEC
can be found by local minimization once an RC has been assigned to each residue
(Fig. 2.6). In general, RCs are handled analogously to rotamers in minDEE but can
account for backbone as well as sidechain flexibility.

An important property of perturbations is that if a perturbation is applied to
a protein, the perturbation must have the same parameter value for all affected
residues. Thus, if a perturbation $p$ affects residue $i$ and $j$, and two RCs $i_r$ and $j_s$
have different parameter value intervals for $p$, then the pairwise energy of $i_r$ and
$j_s$ can be set to $\infty$, and the pair should always be pruned. Such a pair is called
parametrically incompatible (Fig. 2.7). For example, if a shear perturbation (Section
2.2.2) in a protein moves both residues 60 and 61, there can be no conformation
in which residue 60 is in an RC with shear parameter 3-5° but residue 61 is in an
unsheared RC. RCs at different residues with different but overlapping parameter
value intervals for the same perturbation are not allowed, but this restriction does
not limit the conformational space available to DEEPer; it only limits the choice of
RCs used to represent this conformational space.

Pairs of RCs that are not parametrically incompatible will be referred to as
parametrically compatible.

Having defined the set of RCs for each residue position and pruned parametrically
Figure 2.7: Parametric incompatibility of RCs. Suppose residue $i$ and $j$ are affected by the same backrub. RCs $i_t$ and $j_s$ are *parametrically compatible* because there exists an overall conformation of the protein in which residue $i$ is in $i_t$ and residue $j$ is in $j_s$. Similarly, $i_u$ and $j_s$ are parametrically compatible. But $i_r$ and $j_s$ are *parametrically incompatible* because there is no overall conformation of the protein in which residue $i$ is in $i_r$ and residue $j$ is in $j_s$.

Incompatible pairs, we are ready to search for the GMEC or low-energy ensemble, starting with the same minimization-aware pruning methods as iMinDEE,\(^\text{40}\) which can be applied without modification if parametrically incompatible pairs are assigned an infinite pairwise energy. This is possible because the derivation of iMinDEE\(^\text{40}\) does not assume anything about the geometry of the flexibility. It only assumes that lower bounds for single-rotamer and pairwise interaction energies are available and that there is a method to minimize conformations once rotamers have been assigned.

Thus, the iMinDEE algorithm can be applied to RCs with continuous sidechain and backbone flexibility just as it can be applied to rotamers with only sidechain flexibility (see Fig. 2.6). Pruning is followed by A* search and enumeration of conformations; as in minDEE\(^\text{48}\) and iMinDEE,\(^\text{40}\) the A* search outputs unminimized conformations in order of lower-bound minimized energy, and local minimization is used to identify the GMEC or the desired ensemble. These steps are also part of the iMinDEE protocol.\(^\text{40}\)
The conformation of the protein is always uniquely defined by the sequence and the values of all perturbation parameters and sidechain dihedrals. This conformation is constructed from the original, wild-type conformation by performing any necessary mutations, rotating sidechains to obtain the correct dihedrals, and then applying all perturbations with the specified parameters. When there are overlapping perturbations, the precise geometry of the RC will typically depend on the order in which the perturbations are applied, so DEEPer must assign an ordering to all the perturbations before beginning calculations. This ordering can be specified by the user, but a default ordering is provided by the automatic perturbation selector (Section 2.2.2).

If the initial conformational state to which a perturbation \( b \) is applied depends on the state of a previously applied perturbation \( a \), then any residue affected by \( b \) is also considered to be affected by \( a \) for purposes of defining RCs and minimization, since \( a \) affects the final perturbed conformations of that residue. It is useful to choose an ordering that reduces the number of residues affected by each perturbation and thus the number of RCs. Measures to accomplish this include performing small perturbations after larger ones and applying perturbations in a minimal number of “layers,” where a layer is a set of non-overlapping perturbations applied consecutively.

Like iMinDEE, DEEPer requires minimization of pairwise energies to compute lower-bound energies as well as minimization of the total energy of the protein to obtain final minimized conformations. DEEPer minimizes pairwise energies with respect to all continuous perturbation parameters affecting either of the residues in the pair, in addition to the residues’ sidechain dihedrals. The total energy is minimized with respect to all continuous perturbation parameters and sidechain dihedrals. To change the parameter value for a perturbation \( p \), \( p \) can be undone and then reapplied with the correct parameter. To ensure that perturbations are all applied in the correct order, perturbations that were applied after \( p \) may have to be undone before \( p \) is undone and then reapplied after \( p \) is reapplied. As a result,
it is best to apply continuous perturbations after discrete ones, so that discrete perturbations do not need to be re-applied during minimization.

At the discretion of the user, the space of RCs in DEEPer can be augmented by introducing wild-type “rotamers” for each residue. These are rotamers generated by taking the sidechain conformation from the starting structure (including the original bond lengths and angles) and then allowing ±9° minimization of sidechain dihedrals. Wild-type rotamers are added to the calculation along with the generic rotamers from the rotamer library (which, in the Penultimate rotamer library\textsuperscript{115} we use, correspond to clusters of dihedral angles from high-resolution structures in the PDB).

2.2.5 Indirect pruning

By construction, DEEPer allows searching a larger conformational space than any previous provable protein design algorithm can search. This additional flexibility can lead to additional computational complexity. In this section, we show that a new algorithm, indirect pruning, can be used to alleviate this complexity. To explain this new algorithm, we first introduce the concept of the pruning zone, and then provide additional machinery that will be needed to describe the algorithm formally. Next, we provide theorems establishing the correctness of indirect pruning and an analysis of its computational complexity. Finally, we provide a simple “toy” example of indirect pruning to illustrate the functioning of the algorithm.

In principle, DEEPer could be implemented using only previous minimization-aware DEE pruning algorithms, such as iMinDEE\textsuperscript{40} and minBounds.\textsuperscript{47} For these algorithms, RCs are used instead of rotamers, and pairwise energies of parametrically incompatible RC pairs are set to $\infty$. There is some additional cost due simply to the larger number of RCs than rotamers. However, there is also a source of inefficiency in the DEE pruning step by previous algorithms that arises specifically due to the DEEPer perturbation model (Fig. 2.8a,b). Let $i_r$ and $i_t$ be any two RCs
that include a different parameter value interval for some perturbation \( p \), where \( p \) affects both residue \( i \) and some other residue \( k \). In such a case, \( i_r \) is parametrically incompatible with some RC of \( k \) that is parametrically compatible with \( i_t \). But \( i_t \) is parametrically incompatible with some other RC of \( k \) that is parametrically compatible with \( i_r \). Consequently, the left-hand side of the iMinDEE pruning condition Eq. (2.1) will necessarily be \(-\infty\), since the term in the sum for \( j = k \) will be \(-\infty\) regardless of whether we are trying to prune \( i_r \) using \( i_t \) or the other way around. This lack of pruning would require every combination of parameter values for different perturbations to be enumerated in the A* step. The number of such combinations can grow exponentially with the number of perturbations, quickly leading to intractable calculations.

To avoid this problem, we derive a new pruning algorithm, indirect pruning, that can prune in such cases. This method is designed to take advantage of already-pruned pairs of RCs. These can be parametrically incompatible pairs, pairs with steric clashes, pairs that have been pruned by other pruning algorithms, and pairs pruned by previous iterations of indirect pruning. Indirect pruning is intended for use in addition to previous techniques, rather than as a replacement for them, in order to obtain optimal pruning.

**Pruning zones.** We will now introduce the concept of a pruning zone, a key novel feature of our method.

We designate a set of residues as the pruning zone, denoted as \( Z \). In principle, \( Z \) may be chosen to be any set of residues in the system (good choices of \( Z \) are discussed below). We then attempt to prune RCs for each residue in \( Z \). In DEE pruning, a “candidate” rotamer is said to be pruned using a “competitor” rotamer, with “witness” rotamers at other residues considered in the calculation. Suppose we wish to prune the candidate RC \( i_r \) using the competitor RC \( i_t \). Because some RCs
elsewhere in the pruning zone may be parametrically incompatible with \( i \) but not \( i_r \) (or vice versa), we may not be able to find witness RCs at those residues. As a result, it is not feasible to prune \( i_r \) by considering only the energy terms involving residue \( i \), as DEE does.

Instead, we try to prune all conformations of \( Z \) that contain \( i_r \) (Fig. 2.8). Since we represent each conformation of \( Z \) as a tuple of RCs, the pruning condition for tuples of RCs (Eq. (2.4) below) could be used here; this pruning condition considers the energy terms involving all residues in \( Z \). However, pruning each tuple individually would be very time consuming, because the number of conformations of \( Z \) grows exponentially with respect to \( z = |Z| \).

To avoid this problem, we use bounds to “indirectly” prune all the tuples containing \( i_r \) at once. First, we bound the energy differences that we need to prune conformations of \( Z \) containing \( i_r \). Then, we try to identify a conformation of \( Z \) that includes \( i_t \) and can prune all conformations of \( Z \) that include \( i_r \). We still consider the energy terms involving all residues in \( Z \), and we break up the pruning condition into terms of the form Eq. (2.2) to allow efficient computation of the bounds. This leads to the maximization of a minimum in Algorithm 1 below: we perform a minimization in step 2 to bound the energy associated with \( Z \) when residue \( i \) is in the RC \( i_r \), and this creates the overall conformation \( u \) of \( Z \) that will be “hardest” to prune. Then we find the overall conformation \( v \), consisting of \( i_t \) and its companions, that will at least approximately maximize the energy difference \( q \) (see Algorithm 1) and thus make pruning most feasible. We use a greedy maximization strategy that is not provably optimal to avoid an exponential running time; however, pruning is still provable. Like conformational-splitting DEE,\(^{130}\) this algorithm effectively partitions the search space of “witness” RCs at residues other than that of the RC we are trying to prune. However, by implicitly constructing conformations of the entire pruning zone, it can more comprehensively exclude infeasible conformations from the

43
pruning and thus prune more powerfully than conformational splitting, which only splits conformations based on the RC choice at one or two residue positions.

Unlike previous algorithms, indirect pruning can remove pruned competitor-witness as well as candidate-witness pairs from analysis. Indirect pruning has been implemented first using all flexible residues as the pruning zone, followed by additional pruning zones as described in section 2.2.6.

**Notation.** Some of the notation presented here is shared with previous work on DEE-based algorithms, such as, but some new notation is also required for the explanation of indirect pruning. We define $E_{\sigma}(i_r)$ to be the minimum intra-residue energy that the residue can have while in the RC $i_r$, and given some other RC $j_s$ ($j \neq i$), we define $E_{\sigma}(i_r, j_s)$ as the minimum pairwise interaction energy that residues $i$ and $j$ can have while $i$ is in $i_r$ and $j$ is in $j_s$. These quantities will be used to calculate lower bounds for conformational energies. Following normal usage, we call an RC or a tuple of RCs at different residues pruned if it has been determined that its residue(s) cannot all be in the specified RC(s) in the GMEC or in any conformation of the desired ensemble. Next, we provide a notation for tuples of RCs. Suppose we have an $m$-tuple $r$ of RCs, where each RC $i_r \in r$ applies to a different residue $i$; we then define $M_r$ to be the set of residues to which the RCs in $r$ apply. For $i \in M_r$, let $i_r$ denote the RC in $r$ corresponding to $i$. In other words, we will use the notation $i_r$ to refer to elements in a tuple, indexed by residue. Also, let $N$ denote the set of residues that are not in the pruning zone.

Now, we provide a notation for the RCs that we are searching over. Let $R_j$ be the set of unpruned RCs at residue $j$. For each pair $(i,j)$ of residues, we can construct a set $R_j(i_r) \subseteq R_j$ containing all RCs of $j$ that may be found in a conformation that also contains $i_r$ (Fig. 2.9). To do this, for each $i \neq j$, we stipulate that an RC $j_s \in R_j$ is in $R_j(i_r)$ if and only if $j_s$ has not been pruned and the pair $(i_r, j_s)$
Figure 2.8: Parametric incompatibility impedes DEE pruning but not indirect pruning. (a) Normally, DEE is able to compare conformations containing a candidate RC $i_r$ to those containing a competitor RC $i_t$ and prune $i_r$ if the conformations containing $i_r$ are always higher in energy. We can determine this by comparing all energy terms involving residue $i$: its internal energy plus its interaction energies with all other residues. Solid lines denote this summed energy for $i_r$ (red) and for $i_t$ (purple). (b) If $i_r$ and $i_t$ have different parameter intervals for a perturbation, then the conformations of other residues affected by the perturbation cannot be directly compared by the DEE criterion. Thus, pruning using this criterion is impossible, regardless of the energetics of these RCs. (c) This problem is alleviated if we compare all energy terms involving a multi-residue pruning zone rather than just the single residue position $i$. Indirect pruning works by comparing the sets of all conformations of the pruning zone containing $i_r$ (red) to all conformations of the pruning zone containing $i_t$ (purple). But even for a given set of conformations for all residues outside the pruning zone, each of these sets of conformations will have a range of energies (shown as colored regions). We use bounds (red and blue lines) to avoid the expensive process of considering each conformation of the pruning zone. We seek a lower bound on the energy difference between the best conformation involving $i_t$ (the blue line is an upper bound on this conformation’s energy) and the best conformation involving $i_r$ (the red line is a lower bound on this conformation’s energy, so the difference between the red and blue lines represents a lower bound on the energy difference). Dashed lines indicate the energy gap between conformational states involving $i_r$ and those involving $i_t$. 

45
Figure 2.9: The $R_{ij}(i_r)$ notation. Suppose residues $i$ and $j$ are affected by the same backrub. Then $R_{ij}(i_r)$ (blue voxels on the right) can consist only of RCs with the same parameter range for this backrub as $i_r$. RCs can also be excluded because of pruned pairs: even though $j_s$ has the same parameter range as $i_r$, $j_s \notin R_{ij}(i_r)$, because the pair $(i_r, j_s)$ has been pruned (either by a steric check, a previous round of indirect pairs pruning, or another pruning algorithm). RCs of $j$ that are not in $R_{ij}(i_r)$ (those that cannot be found simultaneously with $i_r$ in a conformation of the protein) are shown in green.

has not been pruned. In other words, $R_{ij}(i_r)$ consists of all the RCs at position $j$ that are compatible with $i_r$. If the indirect pruning algorithm is to be applied to a given pruning zone multiples times, $R_{ij}(i_r)$ needs to be recomputed each time. This is analogous to the way newly pruned rotamers are excluded from analysis at each iteration in other forms of dead-end elimination. Since $i_r$ is compatible with itself, we also define $R_{ii}(i_r) = \{i_r\}$.

Similarly, let us define $R_{ij}(r)$ to specify the set of RCs of residue $j$ that are compatible with all RCs in $r$. Unpruned RCs at position $j$ can be excluded from $R_{ij}(r)$ based either on pruned pairs or larger tuples. To make this rigorous, for $j \notin M_r$, $j_s \in R_j$ is in $R_{ij}(r)$ if and only if no tuple of RCs $g$ specifying the conformation of $W \cup \{j\}$ has been pruned, where $W \subseteq M_r$, $j_g = j_s$, and $h_g = h_r$ for each residue $h \in W$. If no tuples larger than pairs have been pruned, then this condition can be be simplified: $j_s \in R_j$ is in $R_{ij}(r)$ if and only if no pair $(i_r, j_s)$ has been pruned, for $i \in M_r$. 46
RCs that are part of \( r \) are compatible with \( r \), so for \( j \in M_r \), \( R_j(r) = \{ j_r \} \).

**Formal description of algorithm.** Indirect pruning of individual RCs proceeds as follows.

**Algorithm 1. Pruning individual RCs.**

1. For each residue \( i \) in \( Z \), and for each ordered pair of RCs \( (i_r, i_t) \) at \( i \) with \( i_r \neq i_t \), calculate \( K(i_r, i_t) \), where

\[
K(h_a, h_b) = E_\otimes(h_a) - E_\otimes(h_b) + \sum_{h' \in Z, h' < h} \left( \min_{h'_a \in R_{h_a}(h_a)} E_\otimes(h_a, h'_a) - \max_{h'_b \in R_{h_b}(h_b)} E_\otimes(h_b, h'_b) \right)
\]

\[
+ \sum_{j \in N, j \in R_j} \min_{j_s \in R_j} \left( E_\otimes(h_a, j_s) - E_\otimes(h_b, j_s) \right).
\]

(2.2)

2. For each pair \( (i_r, i_t) \):

(a) Create a variable \( q \) and set \( q = 0 \).

(b) For each \( h \in Z \), iterating in increasing order of \( h \), find the RC \( h_v \) that maximizes \( \min_{h_a \in R_{h}(i_r)} K(h_a, h_v) \) subject to the conditions \( h_v \in R_{h}(i_t) \) and \( h_v \in R_{h}(j_v) \) for each \( j \in Z \) with \( j < h \); add this maximum value to \( q \).

(c) Prune \( i_r \) if and only if \( q > E_w + I \), where \( E_w \) is the desired energy window and \( I \) is the pruning interval (see Introduction).

This pruning condition, unlike the iMinDEE pruning condition Eq. (2.1), is broken down into the RC pair terms \( K(h_a, h_b) \) (see Eq. (2.2)) first rather than into pairwise and single-residue terms, to facilitate the algorithm. Eq. (2.2) in turn is broken down into single-residue terms, pairwise terms within the pruning zone, and finally pairwise terms between residues inside and outside the pruning zone.

The indirect pruning strategy can also be applied to pairwise or higher-order pruning (pruning of pairs or larger tuples of RCs). Let \( r \) be a tuple of RCs. We will
try to prune tuples of residues in the pruning zone, i.e., try to prune some tuples \( r \) such that \( M_r \subseteq Z \). To prune tuples, we must first precompute \( K(i_r, i_t) \) for each ordered pair of RCs \((i_r, i_t)\) with \( i_r \neq i_t \) at each \( i \) in \( Z \). This is the same precomputation as is required for pruning single RCs (Algorithm 1). Our implementation prunes single RCs before pairs and reuses the precomputed \( K(i_r, i_t) \) values for the pairs pruning. Then, to attempt pruning of the tuple \( r \) using the competitor tuple \( t \), where \( M_r = M_t \) (so corresponding elements of \( r \) and \( t \) are at the same residue), we apply the following algorithm:

**Algorithm 2. Pruning a tuple of RCs.**

1. Set \( q = 0 \).

2. For each \( h \in Z - M_r \), iterating in increasing order of \( h \), find the RC \( h_v \) that maximizes \( \min_{h_a \in R_h(r)} K(h_a, h_v) \) subject to the conditions \( h_v \in R_h(r) \) and \( h_v \in R_h(j_v) \) for each \( j \in Z \) with \( j < h \); add this maximum value to \( q \).

3. Prune \( r \) if and only if \( q + K(r, t) > E_w + I \) where

\[
K(r, t) = \sum_{i \in M_r} \left( E_{\bigoplus}(i_r) - E_{\bigoplus}(i_t) + \sum_{h' \in M_r, h' < i} (E_{\bigoplus}(i_r, h'_r) - E_{\bigoplus}(i_t, h'_t)) \right)
+ \sum_{h' \in Z - M_r} \left( \min_{h'_a \in R_{h'_r}(r)} \sum_{i \in M_r} (x_{h'_r, i} E_{\bigoplus}(i_r, h'_a)) - \max_{h'_b \in R_{h'_t}(t)} \sum_{i \in M_r} (x_{h'_t, i} E_{\bigoplus}(i_t, h'_b)) \right)
+ \sum_{j \in N, j_s \in R_j} \sum_{i \in M_r} \left( E_{\bigoplus}(i_r, j_s) - E_{\bigoplus}(i_t, j_s) \right).
\]  

A proof of the correctness of Algorithm 1 is provided in section 2.2.7, and a proof of the correctness of Algorithm 2 is provided in section 2.2.8.

**Complexity.** Indirect pruning, like regular DEE, runs in polynomial time. We can characterize the complexity in terms of \( n \), the number of residues in the system; \( z \),
the size of the pruning zone, with \( z \leq n \); and \( r \), the maximum number of RCs at any residue in the pruning zone:

**Lemma 2.2.1.** Indirect pruning of single RCs for a pruning zone \( Z \) runs in \( O(z^2r^4 + zn^3 + z^3r^3) \) time.

**Proof.** The time complexity of pruning single RCs for a pruning zone is the sum of the time to calculate \( K(i_r, i_t) \) using Eq. (2.2) for each \( i \) in \( Z \), and for each ordered pair of RCs \((i_r, i_t)\) at \( i \) with \( i_r \neq i_t \), plus the time to evaluate the pruning condition (step 2 of Algorithm 1) for each such pair. At each residue, there are at most \( r^2 \) ordered pairs of RCs at the same residue including repeats \((i_r, i_r)\), so there are less than \( zr^2 \) pairs of RCs to consider in total.

For a given RC pair \((h_a, h_b)\), it takes constant time to evaluate \( E_\Theta(h_a) - E_\Theta(h_b) \). It takes \( O(r) \) time to evaluate \( \min_{h'_a \in R_{h'}(h_a)} E_\Theta(h_a, h'_a) \) or \( \max_{h'_b \in R_{h'}(h_b)} E_\Theta(h_b, h'_b) \) for some \( h' \in Z \), because at most \( r \) RCs need to be considered to find the maximum or minimum, so it takes \( O(zr) \) time to evaluate \( \sum_{h' \in Z, h' < h} (\min_{h'_a \in R_{h'}(h_a)} E_\Theta(h_a, h'_a) - \max_{h'_b \in R_{h'}(h_b)} E_\Theta(h_b, h'_b)) \).

Finally, it takes \( O(r) \) time to evaluate \( \min_{j \in R_j} (E_\Theta(h_a, j) - E_\Theta(h_b, j)) \), and therefore it takes \( O(nr) \) time to evaluate \( \sum_{j \in R_j} \min_{j \in R_j} (E_\Theta(h_a, j) - E_\Theta(h_b, j)) \). Thus, it takes \( O(1 + zr + nr) = O(nr) \) time to evaluate Eq. (2.2), and so the computation of \( K(i_r, i_t) \) for each of the \( O(zr^2) \) pairs \((i_r, i_t)\) takes \( O(znr^3) \) time.

Next, for each residue \( h \in Z \), evaluating the maximum of a minimum in step 2 of the algorithm will require performing less than \( r^2 \) comparisons of terms \( K(h_a, h_b) \) and less than \( zr \) checks of the condition \( h_\nu \in R_h(j_\nu) \) and so will take \( O(r^2 + zr) \) time. Because there are \( z \) residues at which this must be performed, the time required for step 2 for a given RC ordered pair \((i_r, i_t)\) is \( O(zr^2 + z^2r) \), and so the total time to evaluate the pruning condition for all \( O(zr^2) \) pairs is \( O(z^2r^4 + z^3r^3) \).
Consequently, the total time required to perform pruning of single RCs for a pruning zone \( Z \) is \( O(z^2r^4 + xnr^3 + z^3r^3) \).

**Lemma 2.2.2.** Indirect pruning of \( m \)-tuples of RCs for a pruning zone \( Z \) runs in \( O(z^m r^{2m+1}(n + z^2 + zr)) \) time.

**Proof.** Let us first assume that we have already precalculated \( K(i_r, i_t) \) for each \( i \) in \( Z \), and for each ordered pair of RCs \((i_r, i_t)\) at \( i \) with \( i_r \neq i_t \); the cost of this precomputation will be considered later.

To prune \( m \)-tuples in \( Z \) we must consider each of the \( \binom{z^m}{m} = O(z^m) \) possible \( m \)-tuples of residues in \( Z \), and assign a pair of RCs for each residue out of the \( O(r^2) \) pairs available, so that we have one tuple \( r \) to prune using another \( t \). This gives us \( O(z^m r^{2m}) \) pairs of tuples to evaluate for pruning. It takes \( O(r^2 + zr) \) time to evaluate the maximum of a minimum term for \( h \in Z \), and so overall it takes \( O(zr^2 + z^2r) \) time to perform step 2 of the algorithm. This leaves the \( K(r, t) \) term, which requires evaluating Eq. (2.3) once. We require constant time (with respect to \( n, r, \) and \( z \) to evaluate \( \sum_{i \in M_r} \left( E_\Theta(i_r) - E_\Theta(i_t) + \sum_{h' \in M_r, h' < i} (E_\Theta(i_r, h'_r) - E_\Theta(i_t, h'_t)) \right) \). It takes \( O(r) \) time to evaluate \( \min_{h'_r \in R_{h_r}(r)} \sum_{i \in M_r} \left( \chi_{h'_r < i} E_\Theta(i_r, h'_r) \right) \) or \( \max_{h'_r \in R_{h_r}(t)} \sum_{i \in M_r} \left( \chi_{h'_r < i} E_\Theta(i_t, h'_r) \right) \) for \( h' \in Z - M_r \), so it takes \( O(zr) \) time to evaluate the \( \sum_{h' \in Z - M_r} \left( \min_{h'_r \in R_{h_r}(r)} \sum_{i \in M_r} \left( \chi_{h'_r < i} E_\Theta(i_r, h'_r) \right) \right) \) term. Finally it takes \( O(r) \) time to evaluate \( \min_{j \in N} \sum_{j \in R_j} \left( E_\Theta(i_r, j_s) - E_\Theta(i_t, j_s) \right) \) for \( j \in N \), and thus \( O(nr) \) time to evaluate \( \sum_{j \in N} \min_{j \in R_j} \sum_{i \in M_r} \left( E_\Theta(i_r, j_s) - E_\Theta(i_t, j_s) \right) \). Adding these together, the cost of evaluating Eq. (2.3) is \( O(nr + zr + 1) = O(nr) \), and so the cost of trying to prune one \( m \)-tuple using another is \( O(nr + zr^2 + z^2r) \). Thus the overall cost of trying to prune with all \( O(z^m r^{2m}) \) pairs of \( m \)-tuples is \( O(z^m r^{2m+1}(n + z^2 + zr)) \). For any \( m > 1 \), this dominates the time required to prune single RCs even with the \( K(i_r, i_t) \) precomputation, so the cost of
pruning $m$-tuples is still $O(z^m r^{2m+1} (n + z^2 + zr))$ even if the precomputation cost is added.

Note that pruning all $m$-tuples takes exponential time with respect to $m$ using any form of DEE simply because the number of possible $m$-tuples to prune grows exponentially with respect to $m$. The time cost to prune a single $m$-tuple is polynomial with indirect pruning or any previous DEE-based algorithm. Unlike the original DEE algorithm, indirect pruning has nontrivial space complexity, because it must calculate the RC-pair terms $K(i_r, i_t)$ for each residue pair $(i_r, i_t)$. However, because there are $O(zr^2)$ such pairs, the memory cost is just $O(zr^2)$, which is negligible compared to A* or even to the cost of storing a boolean for each tuple indicating whether it is pruned, which is $O(n^m r^m)$.

Because it can prune RCs using RCs in different backbone states, indirect pruning could also be useful in multistate protein design, which designs sequences that prefer one backbone state over another.\textsuperscript{173}

**Toy example of indirect pruning.** The indirect pruning condition can be illustrated using the toy example whose energies are listed in Table 2.3. We are using the pruning zone $Z = \{i, j\}$ and trying to prune $i_r$ using $i_t$. We will not consider any continuous flexibility, so we can just consider well-defined energies $E$ instead of energy lower-bounds $E_{\ominus}$ and let $I = 0$. Suppose residues $i$ and $j$ are affected by a discrete perturbation, which may have parameter value either 0 or 1. The RCs $i_r$ and $j_s$ have parameter value 0 for this perturbation, while $i_t$ and $j_{s'}$ have parameter value 1; thus $R_j(i_r) = \{j_s\}$, $R_j(i_t) = \{j_{s'}\}$, $R_i(j_s) = \{i_t\}$, and $R_i(j_{s'}) = \{i_t\}$. All these RCs have alanine as their amino acid type. Since alanine has no sidechain dihedrals, none of the RCs include bounds for sidechain dihedrals. Furthermore, suppose residue $k$ is not affected by the perturbation but has two rotamers, treated as rigid here: the $m$ rotamer of valine and the $t0$ rotamer of aspartate (the other rotamers of valine
Table 2.3: Pairwise energies for RCs in the toy example. Pruning of RC $i_r$ using $i_t$ will be attempted, with pruning zone $Z = \{i, j\}$.

<table>
<thead>
<tr>
<th>Residue conformation</th>
<th>Perturbation parameter value</th>
<th>Intra-energy</th>
<th>Pairwise energies</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_r$</td>
<td>0</td>
<td>2</td>
<td>$j_s$ $j_{s'}$ $k_u$ $k_{u'}$</td>
</tr>
<tr>
<td>$i_t$</td>
<td>1</td>
<td>$\infty$</td>
<td>-3 -5 5</td>
</tr>
<tr>
<td>$j_s$</td>
<td>0</td>
<td>-2</td>
<td>20 3</td>
</tr>
<tr>
<td>$j_{s'}$</td>
<td>1</td>
<td>0</td>
<td>2 3</td>
</tr>
<tr>
<td>$k_u$</td>
<td>n/a</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$k_{u'}$</td>
<td>n/a</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

and aspartate having been pruned already). The RCs $k_u$ and $k_{u'}$ correspond to the two rotamers of $k$. They have the amino acid types and dihedral angles of their corresponding rotamers and no perturbation parameters.

We first need to calculate $K(i_r, i_t) = E(i_r) - E(i_t) + E(i_r, k_{u'}) - E(i_t, k_{u'}) = 2$, and also we need $K(j_s, j_{s'}) = E(j_s) - E(j_{s'}) + E(i_r, j_s) - E(i_t, j_{s'}) + E(j_s, k_{u'}) - E(j_{s'}, k_{u'}) = -1$. When we apply our pruning condition, $K(i_r, i_t) + K(j_s, j_{s'}) > E_w$, we are pruning all conformations of $Z$ that include $i_r$ (in this case, there is just one such possible conformation of $Z$, defined by $i_r$ and $j_s$). Therefore, we prune if $2 - 1 = 1 > E_w$.

Since we are looking for all conformations within energy $E_w$ of the GMEC, if we want only the GMEC, we set $E_w = 0$ and thus prune $i_r$.

2.2.6 Choice of pruning zones

Making the pruning zone $Z$ be the entire system will ensure that all pruned pairs are contained within $Z$ (i.e. for each pruned pair $(i_r, j_s), i, j \in Z$), which may result in additional pruning relative to previous methods. The time and space cost of indirect pruning for $Z$ scales as $z$ but is expected to be much smaller than that of A* regardless. However, indirect pruning becomes less powerful than Goldstein DEE when few of the RC pairs in the pruning zone are pruned. To avoid this problem, one may construct a much smaller, “minimal” pruning zone by the following protocol:
1. Select a perturbation and put all the residues it affects in $Z$.

2. While there is a perturbation that affects at least one residue in $Z$ and at least one in $N$, add all the residues it affects to $Z$.

If $Z$ is a minimal pruning zone, then there will be no parametrically incompatible pairs $(i_r, j_s)$ for $i \in Z$ but $j \in N$, which would lead to a lack of pruning. But there is no proper subset of $Z$ with this property, so removing more residues from $Z$ would be imprudent. In other words, using a minimal pruning zone will take advantage of all the parametrically incompatible pairs resulting from the perturbations. Also, it will make sure that no RCs are impossible to prune just because they have different parameter values than other RCs at the same residue. But it will put no more residues in $Z$ than necessary to satisfy those two conditions.

Optimal pruning can be obtained by using multiple rounds of indirect pruning, one with the entire system as the pruning zone and one for each possible minimal pruning zone (there are at most as many as minimal pruning zones as there are perturbations); these can be applied in sequence during each pruning cycle along with all the other DEE methods in our lab’s osprey software\cite{osprey, osprey2} (Goldstein singles, pairs, etc.). The cycle is repeated until it does not prune any RCs. Thus, the number of cycles cannot be greater than the total number of RCs at all residues, but in practice it will tend to be far smaller than that. Using both the entire protein as a pruning zone and the minimal pruning zones achieves a balance between the benefits of large and small pruning zones. Because the pruning is provable for any sequence of pruning zones, more could be included, though the amount of additional pruning will likely be much diminished. In fact the first pruning zone used, which in the current implementation is the entire protein, tends to give most or all of the pruning in test runs. Multiple iterations with that first pruning zone are usually needed for pruning to complete, though, just as for other forms of DEE.
2.2.7 Proof of single-RC pruning condition

Let us show the indirect pruning algorithm for pruning single RCs, algorithm 1, is correct. First of all, we must consider the DEE pruning condition for a conformation \( u \) of \( Z \), where \( u \) is a \( z \)-tuple of RCs. We use the iMinDEE minimization-aware version\(^{40}\) of Goldstein\(^{53}\)'s DEE pruning condition for tuples.

**Lemma 2.2.3.** Let \( u \) be a \( z \)-tuple of RCs with \( Z = M_u \). If there exists a \( z \)-tuple \( v \) of RCs specifying a conformation of \( Z \) such that

\[
\sum_{h \in Z} \left( E_\Theta(h_u) - E_\Theta(h_v) + \sum_{h' \in Z, h' < h} (E_\Theta(h'_u) - E_\Theta(h'_v, h'_v)) \right) + \sum_{j \in N} \min_{j_s, j'_s} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j'_s) \right) > E_w + I, \tag{2.4}
\]

then \( u \) can be pruned, meaning that \( Z \) is not found in the conformation \( u \) in any overall protein conformation whose energy is within \( E_w \) of the GMEC.

**Proof.** We begin with the iMinDEE version of the Goldstein pruning condition, Eq. (2.1). As noted by,\(^{53}\) this condition need not be applied only to pruning rotamers of single residues; rather, it may be applied to prune conformations of sets of residues (\(^{53}\) refers to these as renormalized residues). In our case, it will be used to prune conformations of the pruning zone \( Z \). Therefore, if there exists a \( z \)-tuple \( v \) of RCs specifying a conformation of \( Z \) such that

\[
E_\Theta(u) - E_\Theta(v) + \sum_{j \in N} \min_{j_s} \left( E_\Theta(u, j_s) - E_\Theta(v, j_s) \right) > E_w + I, \tag{2.5}
\]

then \( u \) can be pruned, with \( E_\Theta(\cdot) \) denoting the intra-energy lower bound of a conformation and \( E_\Theta(\cdot, \cdot) \) denoting the pairwise-interaction-energy lower bound of a pair of conformations as usual. The intra-energy of \( u \) is the sum of the intra-energies of its constituent RCs plus the sum of the pairwise energies among these constituent RCs.
RCs:

\[ E_{\Theta}(u) = \sum_{h \in Z} \left( E_{\Theta}(h_u) + \sum_{h' \in Z, h' < h} E_{\Theta}(h_u, h'_u) \right). \tag{2.6} \]

The pairwise energy between \( Z \) in conformation \( u \) and a given residue \( j \) in RC \( j_s \) is simply the sum of pairwise energies of the constituent RCs of \( u \) with \( j_s \):

\[ E_{\Theta}(u, j_s) = \sum_{h \in Z} E_{\Theta}(h_u, j_s). \tag{2.7} \]

Similarly, if \( Z \) is in conformation \( v \), then its intra-energy is

\[ E_{\Theta}(v) = \sum_{h \in Z} \left( E_{\Theta}(h_v) + \sum_{h' \in Z, h' < h} E_{\Theta}(h_v, h'_v) \right) \tag{2.8} \]

and its pairwise energy with \( j_s \) is

\[ E_{\Theta}(v, j_s) = \sum_{h \in Z} E_{\Theta}(h_v, j_s). \tag{2.9} \]

Substituting the energies Eq. (2.6) through Eq. (2.9) into the pruning condition Eq. (2.5) yields Eq. (2.4), and the lemma follows.

Now we are ready to prove the validity of indirect pruning.

**Theorem 2.2.1.** If Algorithm 1 prunes an RC \( i_r \), then no protein conformation whose energy is within \( E_w \) of the GMEC will contain \( i_r \).

**Proof.** Suppose the algorithm prunes \( i_r \) using \( i_t \). Let \( v \) be as in step 2 of the algorithm for this pruning. This means that there exists a \( z \)-tuple \( v \) of RCs such that \( i_v = i_t \), \( h'_v \in R_{h'_v(h_v)} \) for all \( h, h' \in Z \) (i.e. \( v \) specifies a valid conformation of the pruning zone), and

\[
q_f = \sum_{h \in Z} \min_{h_a \in R_{h(h_r)}} K(h_a, h_v)
\]

\[
= \sum_{h \in Z} \min_{h_a \in R_{h(i_r)}} \left( E_{\Theta}(h_a) - E_{\Theta}(h_v) + \sum_{h' \in Z, h' < h} \left( \min_{h'_a \in R_{h'_a(h_a)}} E_{\Theta}(h_a, h'_a) \right) \right.
\]

\[
- \max_{h'_v \in R_{h'_v(h_v)}} E_{\Theta}(h_v, h'_v)) + \sum_{j \in N} \min_{j_s \in R_j}(E_{\Theta}(h_a, j_s) - E_{\Theta}(h_v, j_s)) \Bigg) > E_w + I. \tag{2.10}
\]
where $q_f$ denotes the value of $q$ when the algorithm is finished. Because $h'_v \in R_{h'}(h_v)$ for any residues $h, h' \in Z$, \( \max_{h'_v \in R_{h'}(h_v)} E_{\ominus}(h_v, h'_v) \geq E_{\ominus}(h_v, h'_v) \), which may be substituted into Eq. (2.10) to yield

\[
\sum_{h \in Z} \min_{h_a \in R_h(i_r)} \left( E_{\ominus}(h_u) - E_{\ominus}(h_v) + \sum_{h' \in Z, h' < h} \left( \min_{h'_a \in R_{h'_a}(h_u)} E_{\ominus}(h_u, h'_a) - E_{\ominus}(h_v, h'_a) \right) \right) > E_w + I.
\] (2.11)

The substitution preserves the inequality because the left-hand-side of Eq. (2.11), which contains the substitution, is greater than or equal to the left-hand-side of Eq. (2.10). Now, let $U$ be the set of conformations of the pruning zone that contain $i_r$: $U = \{u \mid M_u = Z, i_u = i_r, h'_u \in R_{h'}(h_u) \forall h, h' \in Z\}$. Applying the definition of a minimum to the $\min$ in Eq. (2.11), we know that for any $z$-tuple of RCs $u \in U$,

\[
\sum_{h \in Z} \left( E_{\ominus}(h_u) - E_{\ominus}(h_v) + \sum_{h' \in Z, h' < h} \left( \min_{h'_a \in R_{h'_a}(h_u)} E_{\ominus}(h_u, h'_a) - E_{\ominus}(h_v, h'_a) \right) \right) > E_w + I.
\] (2.12)

Since $h'_u \in R_{h'}(h_u)$ for each $h, h' \in Z$, \( \min_{h'_a \in R_{h'_a}(h_u)} E_{\ominus}(h_u, h'_a) \leq E_{\ominus}(h_u, h'_u) \), which we may substitute into Eq. (2.12) to yield

\[
\sum_{h \in Z} \left( E_{\ominus}(h_u) - E_{\ominus}(h_v) + \sum_{h' \in Z, h' < h} \left( E_{\ominus}(h_u, h'_u) - E_{\ominus}(h_v, h'_v) \right) \right) + \sum_{j \in N, j_s \in R_j} \left( E_{\ominus}(h_u, j_s) - E_{\ominus}(h_v, j_s) \right) > E_w + I,
\] (2.13)

for each $u \in U$. Finally, observe that

\[
\sum_{h \in Z} \sum_{j \in N, j_s \in R_j} \left( E_{\ominus}(h_u, j_s) - E_{\ominus}(h_v, j_s) \right) \leq \sum_{j \in N, j_s \in R_j} \sum_{h \in Z} \left( E_{\ominus}(h_u, j_s) - E_{\ominus}(h_v, j_s) \right).
\]
and therefore that the left-hand-side of Eq. (2.13) is less than or equal to that of Eq. (2.4), for a given \( u \in U \). Therefore Eq. (2.4) holds for each \( u \in U \), i.e. for each \( u \) specifying a valid conformation of \( Z \) with \( i_u = i_r \). Therefore, by Lemma 2.2.3, we can prune each conformation \( u \) of \( Z \) with \( i_u = i_r \), i.e. no such \( u \) will be in the ensemble of protein conformations within \( E_w \) of the GMEC. Thus, there is no way for \( i_r \) to be found in this ensemble, and we can prune \( i_r \) too.

\[ 2.2.8 \quad \text{Proof of tuples pruning condition} \]

**Theorem 2.2.2.** If Algorithm 2 prunes a tuple of RCs \( r \), then no protein conformation whose energy is within \( E_w \) of the GMEC will contain \( r \).

**Proof.** The proof is very similar to that of Theorem 2.2.1. Suppose the algorithm prunes \( r \) using \( t \), with \( M_r = M_t \subseteq Z \). Let \( v \) be as in the algorithm for this pruning. This means that there exists a \( z \)-tuple \( v \) of RCs such that \( i_v = i_t \) for all \( i \in M_t \), \( h'_v \in R_h(h_v) \forall h, h' \in Z \) (i.e. \( v \) specifies a valid conformation of the pruning zone), and

\[
q_f = \sum_{h \in Z - M_r} \left( \min_{h_a \in R_h(r)} K(h_a, h_v) \right) + K(r, t)
\]

\[
= \sum_{h \in Z - M_r} \min_{h_a \in R_h(r)} \left( E_\Theta(h_a) - E_\Theta(h_v) + \sum_{h'_v \in R_{h'}(h_v)} \left( \min_{h'_a \in R_{h'}(h_a)} E_\Theta(h_a, h'_a) - E_\Theta(h_v, h'_v) \right) \right) + \sum_{j \in N, j \in R_j} \left( E_\Theta(h_a, j_s) - E_\Theta(h_v, j_s) \right) + K(r, t) > E_w + I. \tag{2.14}
\]

where \( q_f \) denotes the value of \( q \) when the algorithm is finished. Now, because \( h'_v \in R_{h'}(h_v) \) for any residues \( h, h' \in Z \), \( \max_{h'_v \in R_{h'}(h_v)} E_\Theta(h_v, h'_v) \geq E_\Theta(h_v, h'_v) \), which may be...
substituted into Eq. (2.14) to yield
\[
\sum_{h \in Z-M_r} \min_{h_a \in R_h(r)} \left( E_\Theta(h_a) - E_\Theta(h_v) + \sum_{h' \in Z, h' < h} \left( \min_{h_a' \in R_{h'}(h_a)} E_\Theta(h_a, h_a') \right) \right)
\]
\[
- E_\Theta(h_v, h_v') + \sum_{j \in N, j_s \in R_j} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s) \right)
\]
\[\quad + K(r, t) > E_w + I. \tag{2.15}\]

The substitution preserves the inequality because the left-hand-side of Eq. (2.15), which contains the substitution, is greater than or equal to the left-hand-side of Eq. (2.14). Now, let \( U \) be the set of valid conformations of the pruning zone that contain \( r \), so \( U = \{ u \mid M_u = Z, h_u \in R_h(r), h_u' \in R_{h'}(h_u) \forall h, h' \in Z \} \). Applying the definition of a minimum to \( \min_{h_a \in R_h(r)} \) in Eq. (2.15), we know that for any \( z \)-tuple of RCs \( u \in U \),
\[
\sum_{h \in Z-M_r} \left( E_\Theta(h_u) - E_\Theta(h_v) + \sum_{h' \in Z, h' < h} \left( \min_{h_a' \in R_{h'}(h_u)} E_\Theta(h_a, h_a') \right) \right)
\]
\[
- E_\Theta(h_v, h_v') + \sum_{j \in N, j_s \in R_j} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s) \right) + K(r, t) > E_w + I. \tag{2.16}\]

Since \( h_u' \in R_{h'}(h_u) \) for each \( h, h' \in Z \), \( \min_{h_a' \in R_{h'}(h_u)} E_\Theta(h_a, h_a') \leq E_\Theta(h_u, h_u') \), which we may substitute into Eq. (2.16) to yield
\[
\sum_{h \in Z-M_r} \left( E_\Theta(h_u) - E_\Theta(h_v) + \sum_{h' \in Z, h' < h} (E_\Theta(h_u, h_u') - E_\Theta(h_v, h_v')) \right)
\]
\[\quad + \sum_{j \in N, j_s \in R_j} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s) \right) + K(r, t) > E_w + I, \tag{2.17}\]

for each \( u \in U \). Similarly, \( \min_{h_a' \in R_{h'}(r)} \sum_{i \in I_{h'}} \left( \chi_{h_a', i} E_\Theta(i_r, h_a') \right) \leq \sum_{i \in I_{h'}} \left( \chi_{h_a', i} E_\Theta(i_r, h_a') \right) \) for each \( u \in U \) and \( \max_{h_a' \in R_{h'}(t)} \sum_{i \in I_{h'}} \left( \chi_{h_a', i} E_\Theta(i_t, h_a') \right) \geq \sum_{i \in I_{h'}} \left( \chi_{h_a', i} E_\Theta(i_t, h_a') \right) \), where \( h' \in Z \).
$Z - M_r$, both of which we may substitute into the definition of $K(r, t)$, Eq. (2.3):

$$K(r, t) \leq \sum_{i \in M_r} \left( E_\Theta(i_r) - E_\Theta(i_t) + \sum_{h' \in M_r, h' < i} \left( E_\Theta(i_r, h'_r) - E_\Theta(i_t, h'_t) \right) \right)$$

$$+ \sum_{h' \in Z - M_r} \left( \sum_{i \in M_r} (\chi_{h' < i} E_\Theta(i_r, h'_u)) - \sum_{i \in M_r} (\chi_{h' < i} E_\Theta(i_t, h'_v)) \right)$$

$$+ \sum_{j \in N} \min_{j \in R_j} \sum_{i \in M_r} (E_\Theta(i_r, j_s) - E_\Theta(i_t, j_s))$$

(2.18)

for each $u \in U$. We may then substitute Eq. (2.18) into Eq. (2.17), yielding

$$\sum_{h \in Z} \left( E_\Theta(h_u) - E_\Theta(h_v) + \sum_{h' \in Z, h' < h} \left( E_\Theta(h_u, h'_u) - E_\Theta(h_v, h'_v) \right) \right)$$

$$+ \sum_{h \in Z - M_r} \sum_{j \in R_j} \min_{j \in R_j} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s) \right)$$

$$+ \sum_{j \in N} \min_{j \in R_j} \sum_{i \in M_r} (E_\Theta(i_r, j_s) - E_\Theta(i_t, j_s)) > E_w + I$$

(2.19)

for each $u \in U$. Finally, observe that

$$\sum_{h \in Z - M_r} \sum_{j \in R_j} \min_{j \in R_j} \left( E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s) \right)$$

$$+ \sum_{j \in N} \min_{j \in R_j} \sum_{i \in M_r} (E_\Theta(i_r, j_s) - E_\Theta(i_t, j_s)) \leq$$

$$\sum_{j \in N} \min_{j \in R_j} \sum_{h \in Z} (E_\Theta(h_u, j_s) - E_\Theta(h_v, j_s))$$

(2.20)

and therefore that the left-hand-side of Eq. (2.19) is less than or equal to that of Eq. (2.4), for a given $u \in U$. Therefore Eq. (2.4) holds for each $u \in U$, i.e. for each $u$ specifying a valid conformation of $Z$ with $i_u = i_r$ for each $i \in N$. Therefore, by Lemma 2.2.3, we can prune all conformations $u$ of $Z$ consistent with $r$, i.e. no such $u$ will be in the ensemble of protein conformations within $E_w$ of the GMEC; thus, there is no way for $r$ to be found in this ensemble, and so we can prune $r$ as well. ☐
BD, iMinDEE, and DEEPer were run using osprey\textsuperscript{41,49} on 67 test systems to compare their GMECs, thus investigating the advantages of DEEPer in sequence design. The PDB codes of the proteins used were 2ILB (3 tests), 2BGX (2 tests), 1EJG, 1FUS, 1IFC, 1LKK, 1PLC, 1POA, 1RRO, 1WHI, 2GNR, 2RHE, 2TRX, 1L6W, 1L7A, 1L7L, 1L7M, 1L8R, 1L9L, 1L9X, 2OXC, 2OXU, 2OYN, 2OZF, 2OZT, 2P02, 2P0W, 2PK8, 2PL1, 2PLT, 2PSP, 2PTH, 2RMC, 2SGA, 2WEA, 2YGS, 3CAO, 1AHO, 1C75, 1CC8, 1F94, 1FK5, 1I27, 1IQZ, 1JHG, 1M1Q, 1MJ4, 1OAI, 1OK0, 1PSR, 1R6J, 1TUK, 1U07, 1VBW, 1VFY, 1WXC, 1XMK, 1Y6X, 1ZZK, 2AIB, 2BT9, 2CC6, 2CG7, and 2CS7. Structures with hydrogens added were taken from the Richardson lab’s Top4400 database\textsuperscript{135} or provided by Pablo Gainza and Kyle Roberts (personal communication). All flexible residues were in chain A except those for 1WXC, which were in chain B. Wild-type rotamers were not used in these tests, in order to provide a fair comparison of DEEPer to previous methods. Also, the tests were run using the automatic perturbation selection mechanism (Section 2.2.2). The set of flexible residue positions and the set of allowed mutations were selected manually for each system, with a variety of secondary structures and allowed mutations allowed (including some cases with no mutations permitted). Backrub and shear parameters $\theta$ were limited to the interval $-2.5^\circ \leq \theta \leq 2.5^\circ$, and 5-11 flexible residues were chosen per system, except in a single run without continuous flexibility on a crystal structure of the \textit{E. coli} amidase AmiD (PDB code 2BGX\textsuperscript{128}) where shear and backrub parameters of 0 or $\pm 5^\circ$ were allowed and 19 flexible residues were chosen. The discrete conformational space of this run allowed visualization of the backbone conformational search space (Fig. 2.14a). Since this run was performed without continuous flexibility, rigid-rotamer DEE was performed in lieu of iMinDEE and BD, and compared to DEEPer.
Additional tests were performed without sequence changes to test the ability of DEEPer to model larger backbone changes, to recover experimentally-observed structures, and to model ensembles. When crystallographic alternates were used as a source of information on backbone conformational changes, wild-type rotamers for the starting alternate only were included in these runs; this inclusion simulates the availability of wild-type rotamers for the starting conformation when DEEPer is applied to a design problem.

2.3 Results and Discussion

In this section, we present computational tests of DEEPer. The tests show that, given a biophysical model, the additional molecular flexibility in DEEPer leads to more accurate treatment of protein conformations and sequences. These results justify the use of DEEPer in protein design calculations.

Tests of DEEPer using the automatic perturbation selection mechanism on 64 proteins consistently identified lower-energy structures than iMinDEE and BD. Lower-energy structures are more realistic assuming the correctness of the energy function in ranking conformations. (OSPReY’s energy function is based on AMBER and EEF1.) Additional tests on a high-resolution structure of the serine protease spheriscase from Lysinibacillus sphaericus (PDB code 2IXT) and on a medium-resolution structure of human ubiquitin (PDB code 1UBQ) demonstrate that DEEPer captures biophysically reasonable backbone motions and couplings of sidechain and backbone motions without introducing unrealistic conformations.

2.3.1 Comparisons to previous algorithms with less flexibility

67 sequence-design tests on 64 different proteins, as described in section 2.2.9, were run to compare the GMECs found by iMinDEE and DEEPer (Fig. 2.10, 2.11, 2.12, 2.14a). By construction, DEEPer is guaranteed to yield either the same or lower
GMEC energy than iMinDEE. Indeed, the GMECs calculated by DEEPer were lower than those calculated by iMinDEE by an average of 1.9 kcal/mol, ranging from 0 to 14.1 kcal/mol (Fig. 2.10a). 67% of these energy differences exceed the thermal energy at room temperature (0.592 kcal/mol, calculated as the universal gas constant times a room temperature of 298°C), which is a rough measure for functional significance.

GMECs were also computed using BD, a provable algorithm with more limited backbone flexibility and without continuous sidechain minimization. DEEPer yielded lower GMEC energies in every case but one. In this single case, for the α subunit of human S-adenosylmethionine synthetase 2 (PDB code 2P02), the BD GMEC was only 0.4 kcal/mol lower in energy than the iMinDEE and DEEPer GMECs, which were virtually identical to each other; subtle backbone conformational changes in BD led to the slight energetic advantage of the BD structure, outweighing some larger differences in sidechain dihedrals (which did not include any rotamer changes). On the other hand, the GMECs calculated by DEEPer were lower than those calculated by BD by an average of 6.3 kcal/mol, with the difference ranging up to 68.4 kcal/mol and exceeding the thermal energy at room temperature in 94% of runs (Fig. 2.10b).

The changes in GMEC energy from iMinDEE to DEEPer and from BD to DEEPer were very weakly correlated ($R^2=0.37$, where $R^2$ is the coefficient of determination; see Fig. 2.11a). This indicates that modeling backbone flexibility is not an effective substitute for modeling continuous sidechain flexibility or vice versa. Part of this weak correlation is explained by a few tests in which DEEPer provided a very large energetic advantage over both iMinDEE and BD, facilitated by a rotamer change or mutation that required both continuous sidechain and backbone flexibility to accommodate. For example, for porcine pancreatic spasmolytic polypeptide (PDB code 2PSP), BD and iMinDEE both had a lysine at position 54 while DEEPer had a tyrosine. A steric clash blocked BD and iMinDEE from having a tyrosine at this...
position (Fig. 2.12a). This was associated with the largest GMEC energy difference both between DEEPer and iMinDEE and between DEEPer and BD. For *Bacillus subtilis* cephalosporin D deacetylase (PDB code 1L7A), DEEPer, iMinDEE, and BD chose different rotamers for histidine 100 (Fig. 10b), although DEEPer identified the native rotamer, likely due to combined backbone and sidechain minimization at adjacent residues asparagine 101 and lysine 104. In the same run, DEEPer chose the wild-type tryptophan at nearby residue 105 instead of the alanine chosen by iMinDEE and BD; steric clashes precluded a tryptophan with either BD backbone minimization alone (Fig. 10c) or iMinDEE sidechain minimization alone. Because of these combined differences, this region was associated with one of the largest DEEPer-iMinDEE energy differences (9.8 kcal/mol), though the DEEPer-BD energy difference was below average (3.2 kcal/mol).

Even the different forms of backbone flexibility modeled by BD and DEEPer were somewhat complementary. In some tests, e.g., for the protease penicillopepsin from *Penicillium janthinellum* (PDB code 2WEA; Fig. 2.12d), the BD structure stayed very near the fixed-backbone iMinDEE structure, and DEEPer modeled more backbone motion. In other cases, e.g. for the Zβ domain of the human RNA editing enzyme ADAR1 (PDB code 1XMK; Fig. 2.12e), the reverse was true: the DEEPer and iMinDEE structures were virtually identical while BD modeled more backbone motion. However, in both cases, continuous sidechain flexibility allowed DEEPer to identify lower-energy sidechain conformations made accessible by its backbone conformational search. For example, for penicillopepsin DEEPer returned a GMEC energy 17.7 kcal/mol lower than BD, and for Zβ the DEEPer and iMinDEE GMEC energies were both 16.5 kcal/mol lower than the BD GMEC energy.
Figure 2.10: DEEPer GMEC designs compared to designs by previous algorithms with less flexibility on 67 test systems. (a) Decrease in energy from the iMinDEE GMEC to the DEEPer GMEC. (b) Decrease in energy from the BD GMEC to the DEEPer GMEC. Two systems are not shown because BD pruned all rotamers, indicating a steric clash for all rigid rotamers that was resolved by continuous minimization in iMinDEE and DEEPer. These systems are Chinese cobra phospholipase A2 (PDB code 1POA) and *Pyrococcus furiosus* hypothetical protein PF0899 (PDB code 2PK8). One system, the α subunit of human S-adenosylmethionine synthetase 2 (PDB code 2P02) had a higher DEEPer than BD GMEC energy. A dotted red line is shown in both (a) and (b) at the thermal energy at room temperature, 0.592 kcal/mol, as a rough measure of the significance of energy differences.

Notably, the perturbations’ effects on the sidechains far exceeded their effects on the backbone; the latter ranged from 0 to 0.39 Å all-atom backbone RMSD with an average of 0.06 Å. This is partly due to the “lever effect,” i.e., the greater displacement of sidechain atoms than of backbone atoms during a perturbation because the sidechain atoms are kinematically farther from the axes of the rotations involved in the perturbation (Fig. 2.13). Thus, despite the small displacements of the backbone atoms, the effects of shears and backrubs on atoms at the end of sidechains can be substantial. For example, a 2.5° backrub centered at an all-*trans* lysine residue in an ideal β sheet with no other perturbations causes a backbone RMSD for the three affected residues relative to the original structure of just 0.05 Å, and the lysine’s Cα atom moves by just 0.09 Å, but the lysine’s terminal Nζ atom moves by 0.35 Å. Because they induce only small backbone changes, shears and backrubs in the -2.5° to 2.5° parameter range are expected to be biophysically feasible in a wide range of conformations. The backbone motions in DEEPer also induced rotamer changes: one or more rotamer changes were observed in 46% of tests, and up to four rotamer
Figure 2.11: DEEPer GMEC designs compared to designs by previous algorithms with less flexibility on 67 test systems, continued. (a) Decrease in energy from the BD to the DEEPer GMEC versus decrease in energy from the iMinDEE to the DEEPer GMEC. (b) Numbers of test systems with given numbers of sidechain rotamer changes between the iMinDEE and DEEPer GMECs.

Changes per test were observed (Fig. 2.11b).

Energy differences between GMECs found by different algorithms directly measure the effectiveness of the algorithms for their desired function, which is to explore the sequence and conformational space available to a protein and find the lowest-energy sequence and conformation available. Inaccuracy in identifying low-energy conformations and sequences can come from either inaccuracy in the energy function or inaccurate identification of the lowest-energy conformation(s) and sequence(s) given the energy function. DEEPer is intended to reduce inaccuracy of the latter type, and we perform energy comparison tests to separate this task from the also important task of improving energy function accuracy. Because of its provability and its high degree of flexibility, DEEPer is uniquely useful for evaluating and improving energy functions and other modeling assumptions. Errors in the model—i.e., deviations of modeling assumptions from reality—can be identified by running DEEPer and examining deviations of the output from observed structures. Such tests would be much less effective if run using a non-provable algorithm, because errors in the
Figure 2.12: Comparisons of DEEPer GMECs (blue) to iMinDEE GMECs (red) and BD GMECs (green) for four systems. (a) Porcine pancreatic spasmolytic polypeptide (PDB code 2PSP), residues 52-58. Residue 54 is a tyrosine in the DEEPer GMEC but a lysine in the iMinDEE and BD GMECs. A steric clash (pink spikes, generated using Probe\textsuperscript{169}) persists after fixed-backbone, flexible-sidechain energy minimization of the rotamers of the DEEPer GMEC (magenta), making a tyrosine at residue 54 infeasible without backbone flexibility. (b-c) Bacillus subtilis cephalosporin D deacetylase (PDB code 1L7A), residues 100-107. The DEEPer, iMinDEE, and BD GMECs all adopt different histidine rotamers at residue 100 (b); the DEEPer rotamer is the only one in the crystal structure. Residue 105 is a tryptophan in the DEEPer GMEC but an alanine in the iMinDEE and BD GMECs (c); a steric clash persists after flexible-backbone, rigid-rotamer energy minimization of the rotamers of the DEEPer GMEC (magenta), making a tryptophan at residue 105 infeasible without continuous sidechain flexibility. (d) The protease penicillopepsin from Penicillium janthinellum (PDB code 2WEA), residues 300-305. The iMinDEE and BD GMEC backbones are very similar; the DEEPer GMEC adopts a different backbone and achieves a lower energy. (e) The Z\beta domain of the human RNA editing enzyme ADAR1 (PDB code 1XMK), residues 306-311. The iMinDEE and DEEPer GMECs are virtually identical; the BD GMEC adopts a different backbone but still has a higher energy due to a lack of sidechain flexibility.
output could be due either to errors in the model or to deviation of the results of the algorithm from the correct answer for the input model. Also, since real proteins exhibit continuous sidechain and backbone flexibility, the most accurate models must have both of these as well, and thus testing these models requires an algorithm with continuous sidechain and backbone flexibility. DEEPer’s provable search of a conformational space with both continuous sidechain and backbone flexibility is thus uniquely suited for testing modeling assumptions.

Both BD and iMinDEE provably find the lowest-energy structure in their chosen conformational spaces, but the significantly lower energies identified with DEEPer indicate that it finds additional significant conformations that involve continuous sidechain and backbone flexibility. Thus, DEEPer is a useful step toward finding better conformations and sequences given an energy function. Together with advances in energy functions, which it will hopefully facilitate, it is a useful step toward finding conformations and sequences more in line with reality.

2.3.2 Tests of larger backbone conformational changes

To further demonstrate the utility of DEEPer in modeling realistic conformations, tests were performed to show that DEEPer can model larger backbone conforma-
tional changes using partial structure switches. The allowed backbone conformations were crystallographic alternates in two segments of the structure of sphericase (2IXT [49]). In each test, the input structural information consisted of the backbones of the two crystallographic alternates as well as the sidechains only of the starting alternate (to be used as wild-type rotamers). The tests maintained the wild-type sequence (see section 2.2.9).

The first segment was loop residues 36-43, in a fairly surface-exposed region of the protein with some 3₁₀-helical character. Residues 48 and 66 were also allowed sidechain flexibility because of their close contacts with the alternates. Two tests were undertaken. The first test was run without the automatic perturbation selector, meaning the only perturbation available was the partial structure switch that changes the backbone from the starting alternate to the other alternate. The algorithm was being asked to choose a backbone and to pack sidechains onto it, using the wild-type rotamers with the starting alternate or generic rotamers with either the starting or the other alternate. It identified the GMEC as the non-starting alternate, including both its backbone and sidechain rotamers (Fig. 2.14b). 4 of these 8 rotamers differed from the starting rotamers (counting a proline flip). The second test was run with the automatic perturbation selector, so more perturbations and RCs were available. This test identified the GMEC as a modified version of the non-starting alternate, perturbed by not only the partial structure switch between alternates, but also a loop closure adjustment and some continuous backrub minimization. This GMEC was 3.4 kcal/mol lower in energy than the GMEC from the first test, thus illustrating the value of including additional backbone perturbations.

The second segment was residues 264-270, at a surface-exposed α-helical C-terminus. Two tests were again undertaken; this time both used the automatic perturbation selector, but the second also allowed up to 2.5° in either direction of continuous shear and backrub minimization. Both tests chose the other alternate’s
backbone in the GMEC. However, additional backbone perturbations, this time in the form of continuous shear and backrub minimization, reduced the energy by only 0.15 kcal/mol. This is in contrast to the second test on the first segment, where additional backbone perturbations were instrumental in finding a lower-energy conformation.

These results show that partial structure switches alone are sufficient for modeling backbone structure in some cases, but in other cases additional perturbations are important for more fully exploring conformational space. DEEPer is useful for distinguishing between these different types of cases. This capability could be very useful in multistate protein design by revealing small changes in one of the available backbone states that would make that state much more favorable for a given sequence.

2.3.3 Tests of ensemble generation

Additional runs were performed to test the ability of DEEPer to generate biophysically reasonable ensembles of low-energy conformations given fixed sequences. In residues 157-160 of 2IXT (the above-mentioned sphericase structure), the crystal structure contains alternate conformations related approximately by a shear motion. This likely indicates increased backbone dynamics compared to other parts of the structure, and indeed the DEEPer ensemble generated using the native sequence showed more diversity of backbone conformations than in tests on other systems, sampling the conformational space around and between the alternates (Fig. 2.15a).

By contrast, in residues 238 and 240-243 of sphericase modeled with a G242S mutation, the DEEPer backbone ensemble was fairly concentrated around the starting-alternate backbone (Fig. 2.15b). However, DEEPer identified one state in this ensemble significantly different from both alternates, which was generated by a large loop closure adjustment and a subsequent backrub; neither of these perturbations
alone yielded a comparably low-energy conformation. Finally, a test on residues 35-38 of ubiquitin, for which only one conformation was crystallographically observable, showed a much more concentrated ensemble (Fig. 2.15c). These results demonstrate that the combination of different types of flexibility in DEEPer allows it to effectively explore ensembles of realistic conformations, including some with substantial deviation from the starting structure.

2.3.4 Modeling entropy

As noted in the Introduction, ensembles generated by DEEPer can be used to estimate binding affinity using the $K^*$ algorithm\cite{28,48,108} which compares the partition functions of bound and unbound proteins and ligands. The partition function of a
system is defined in $K^*$’s model as $q_C = \sum_{c \in C} \exp(-\frac{E_c}{RT})$, where $C$ is all enumerated conformations (tuples of RCs at all flexible residues), $E_c$ is the minimized conformational energy for conformation $c \in C$, $R$ is the universal gas constant, and $T$ is the temperature. The association constant is then approximated as $K^* = \frac{q_{PL}}{q_P q_L}$, where $q_P$, $q_L$, and $q_{PL}$ are the partition functions of the unbound protein, the unbound ligand, and the protein:ligand complex, respectively. $K^*$ is a provably-good approximation algorithm: it guarantees that the computed binding constant will be within a fraction $\varepsilon$ of the theoretical binding constant defined by the model. The value $\varepsilon$ is specified by the user, to bound the desired accuracy of the calculation.

The conformations that contribute to $K^*$ partition functions are the result of minimization-aware DEE (iMinDEE). This means that even if two conformations in the ensemble populate the same torsional well for a given rotamer (or RC), they will in general minimize to different conformational states within that well due to the influences of their different surroundings. $K^*$ computes a large ensemble, so multiple conformational states are indeed likely to be present per torsional well. Furthermore, the same conformation (in terms of rotamers and/or RCs) may minimize differently in the bound vs. unbound states due to the presence vs. absence of the ligand. In practice, we have observed that the bound and unbound rotamer conformations predicted by $K^*$ can have obviously different means and variances, and are statistically significant as different populations. This result suggests that $K^*$ has the potential to correctly model induced motion in torsional wells using its ensembles, particularly in light of other studies showing that population shifts within torsional wells often occur upon binding. The partition functions, and quotients of the bound and unbound states, should reflect such population differences. Hence, $K^*$ can model a measure of conformational entropy and how it changes upon binding.
Nevertheless, $K^*$, like any finite sum, is a discrete approximation to true continuous entropy. Given this discrete model, $K^*$ guarantees a provably-good approximation, $\varepsilon$-close to the binding constant in the model above (see 48). Unfortunately, partition functions based on discrete ensembles are imperfect reporters of conformational entropy, since they are computed not over all possible conformational states of the protein but rather over a putatively representative low-energy subset: the discrete set of energy-minimized tuples of RCs at all flexible residues.

Some additional error may be introduced by the choice of shear and backrub parameter ranges for RCs. For example, in principle, one could choose very closely spaced shear and backrub parameters, and then the backbone conformational entropy would be weighted too heavily because of the artificially denser distribution of backbone conformational states (compared to sidechain conformational states). However, the DEEPer/$K^*$ hybrid algorithm avoids the bias toward excessive weighting of backbone conformational entropy in at least two ways. First, such errors will cancel out somewhat when the partition functions in the bound and unbound states are divided. Second, having only a single parameter interval for each shear or backrub will usually make each RC (and thus, each sufficiently low-energy tuple of RCs) correspond to its own energy well.

Finally, the DEEPer/$K^*$ approximation is analogous to the iMinDEE/$K^*$ approximation currently used in OSPREY\textsuperscript{41,49} which computes partition functions over the discrete set of tuples of continuous rotamers at all flexible residues. MinDEE/$K^*$ and iMinDEE/$K^*$ have been shown to produce improvements in predictions of binding affinity.\textsuperscript{13,28,48,108,139} Notably, $K^*$ provably computes the partition function for its specified flexibility model (DEEPer or rigid-backbone) within a user-specified margin of error $\varepsilon$. Thus, the introduction of approximate backbone conformational entropy through the combination of DEEPer with $K^*$ is also expected to increase the accuracy of predictions.
Figure 2.15: DEEPer ensembles are dependent on structural and sequence contexts. (a) The low-energy ensemble of computed models was fairly wide at residues 157-160 of sphericase (structure 2IXT) and spanned the crystallographic alternates. The GMEC was on the fringe of the ensemble. (b) Residues 238 and 240-243 also have alternates in structure 2IXT, but the low-energy ensemble from DEEPer for the G242S mutant, including the GMEC, was very tight around alternate A. One low-energy model made a significant excursion via a $> 90^\circ$ peptide flip, executed by a loop closure adjustment and backrubs. (c) Residues 35-38 in structure 1UBQ (ubiquitin) have a single well-ordered conformation. Correspondingly, the low-energy ensemble from DEEPer is very compact: the biggest departure is a single proline flip perturbation. Starting structure, black/gray; low-energy ensemble, blue; GMEC, pink. Green balls demarcate flexible-backbone regions; sidechains outside these regions are omitted for visual clarity.

2.3.5 Summary

Overall, these results illustrate the important point that sidechain motions are tightly coupled to backbone motions (Fig. 2.11b, Fig. 2.14b). In particular, rotamer switches can be enabled either by large discrete backbone motions such as partial structure switches or by small continuous backbone motions such as shears and backrubs. To model this hierarchy of conformational changes, DEEPer includes a broad repertoire of empirically-motivated and biophysically-realistic backbone move types of various magnitudes, and thus is well equipped to study functionally relevant backbone-sidechain couplings in both natural and designed proteins. This coupling is analogous to the utility of small adjustments to structures in multiple structure alignment, which have been found to produce improvements out of proportion to their size.$^{119}$
The ability of DEEPer to provably search continuous sidechain and backbone conformational space can be useful for work involving non-provable algorithms as well. Because DEEPer reliably predicts the correct GMEC or ensemble given an energy function and a set of perturbations, tests of DEEPer can be used as tests of modeling assumptions, which can allow generation of more accurate input models for other algorithms (particularly when modeling backbone flexibility). Additionally, the DEEPer GMEC energy is a lower bound on the GMEC energy that any other algorithm can find with the same biophysical model, so DEEPer can provide a benchmark for other highly flexible protein design algorithms.

2.4 Conclusions

DEEPer provides several advantages over previous algorithms. Due to its continuous, provably complete search over both sidechain and backbone degrees of freedom, it finds GMECs and partition functions that are more accurate with respect to the energy function, resulting in more trustworthy binding predictions and protein or drug designs. DEEPer’s empirically-motivated perturbations help to choose appropriate backbone conformations out of the many possibilities in a way that combines computational tractability with biophysical feasibility. Nevertheless, additional, different perturbations can be easily incorporated in the future due to the accommodating RC framework and indirect pruning method. DEEPer can also use different pairwise energy functions or constraints on the degrees of freedom (e.g., different rotamer definitions or bounds on perturbation parameters) without modification of the algorithm. The energy function and constraints can be derived from either physical or knowledge-based sources. For example, if a more accurate treatment of solvation than that provided by EEF1 is desired, the pairwise Poisson-Boltzmann solvation model of Vizcarra et al.\textsuperscript{162} could be used instead. Computational tests exploring both sequence and conformational space for 64 proteins help confirm that DEEPer
enables a biophysically realistic search of plausible backbone states and consistently yields lower-energy structures than even a continuously-flexible search of sidechain conformations.
EPIC (Energy as Polynomials in Internal Coordinates): A compact representation of continuous energy surfaces for more efficient protein design

This chapter is adapted from a paper in the Journal of Chemical Theory and Computation:


It is a collaboration with Pablo Gainza, who first posed the problem of whether the energy function could be approximated for purposes of protein design minimization with a much cheaper function. Some time later, when I was working on lower-bounding methods for A*, I found functions that served this purpose: I found I could closely approximate the Energy as Polynomials in Internal Coordinates. This is the EPIC algorithm.
Summary. In macromolecular design, conformational energies are sensitive to small changes in atom coordinates, so modeling the small, continuous motions of atoms around low-energy wells confers a substantial advantage in structural accuracy; however, modeling these motions comes at the cost of a very large number of energy function calls, which form the bottleneck in the design calculation. In this work, we remove this bottleneck by consolidating all conformational energy evaluations into the precomputation of a local polynomial expansion of the energy about the “ideal” conformation for each low-energy, “rotameric” state of each residue pair. This expansion is called Energy as Polynomials in Internal Coordinates (EPIC), where the internal coordinates can be sidechain dihedrals, backrub angles, and/or any other continuous degrees of freedom of a macromolecule, and any energy function can be used without adding any asymptotic complexity to the design. We demonstrate that EPIC efficiently represents the energy surface for both molecular-mechanics and quantum-mechanical energy functions, and apply it specifically to protein design to model both sidechain and backbone degrees of freedom.

3.1 Introduction

Computational design algorithms are an effective approach to engineer proteins and discover new drugs for many biomedically relevant challenges, such as drug resistance prediction, \textsuperscript{36} peptide-inhibitor design, \textsuperscript{139} and enzyme design. \textsuperscript{13} Protein design algorithms search through large sequence and conformational spaces for sequences that will fold to a desired structure and perform a specific function. One of the key challenges in protein design is modeling and searching the many continuous conformational degrees of freedom inherent in proteins and other molecules. Protein design algorithms must estimate optimal values for all these degrees of freedom in order to optimize the sequence of the protein, or to optimize the chemical structure of the ligand if used for drug design. Molecular dynamics simulations can be used for this
purpose if the protein sequence and ligand are known, because they can move all of the molecule’s degrees of freedom, but these simulations are computationally expensive and must be run separately for each sequence or ligand chemical structure. Hence, this direct simulation approach is unsuitable for searching large combinatorial design spaces. For example, many protein design problems require searching over trillions of sequences—far too many for individual molecular dynamics runs.

To address the combinatorially large sequence spaces inherent to protein design, dedicated protein design algorithms efficiently choose an amino-acid type and conformation for each residue in a protein that, together, minimize some energy function. Since the sidechain conformations of each amino-acid type are generally found in clusters, known as rotamers, the protein design problem has often been treated as a discrete optimization problem. In this case, the output is a set of rotamer assignments (a rotamer, including amino-acid type, is assigned to each residue). The objective function is an energy function, which maps conformations to their energies. However, because proteins are continuously flexible and have backbone as well as sidechain flexibility, some of the protein’s internal coordinates will likely have functionally significant variations from the rotamer’s “ideal” value (at the center of the cluster). Clashing ideal rotamers can often be converted to favorable conformations by relatively small adjustments in the sidechain conformations. Small adjustments in the backbone conformation away from the wild-type backbone can also be functionally significant. As a result, modeling of continuous flexibility has been shown to dramatically improve the accuracy of structural modeling in designs even using a limited set of degrees of freedom, and has led to designs that perform well experimentally. Furthermore, attempts to mimic this effect by discrete sampling at a finer resolution have been shown to either poorly approximate the continuous solutions, or to be computationally prohibitive. Modeling additional continuous degrees of freedom, with the goal of modeling all conforma-
tional variations that significantly impact protein function, is expected to increase the accuracy of designs further.

Modeling of continuous flexibility in protein design can still exploit our knowledge of rotamers, because rotamers provide an excellent prior estimate of where “energy wells” in the conformational space of the protein are likely to be. Residues’ sidechains will usually be found in the region of conformational space fairly close (e.g., within 10-20° for sidechain dihedrals) to an ideal rotamer, even with a relatively small rotamer library. As a result, if by using a “minimization-aware” search process one can find the nearest ideal rotameric conformation to the true Global Minimum-Energy Conformation (GMEC) of a protein, the GMEC itself can generally be found by local minimization initialized to that ideal rotameric conformation. Thus, protein design can fully account for continuous sidechain flexibility while still functioning as a “minimization-aware” search over discrete rotamer space. This same paradigm can be extended to continuous backbone flexibility if ideal conformations that include backbone motions— “residue conformations” or RCs—are included in the search.

Such “minimization-aware” search can take multiple forms. For example, the iMinDEE algorithm produces a gap-free, provably accurate list of rotamer assignments in order of lower bound on minimized energy. iMinDEE performs energy minimization on each of these rotamer assignments in that gap-free order until the lower bound exceeds the best minimized energy $E_b$ enumerated so far. At this point, any subsequently enumerated assignments would be guaranteed to have higher minimized energies than $E_b$, so $E_b$ is provably the global minimum energy. iMinDEE enumerates rotamer assignments efficiently using the A* search algorithm. Monte Carlo search over rotamer space can also incorporate minimization, but without any provable guarantees. A Monte Carlo search will be minimization-aware if continuous minimization is performed for sequences and conformations during (rather than after) the search and the minimized energy is used in the calculation of acceptance
probabilities for new rotamer assignments, as in\textsuperscript{165} and the final phase of.\textsuperscript{165} Nevertheless, a method without provable guarantees will likely require more conformations and sequences to be minimized to obtain the same gain in accuracy as iMinDEE, because unlike in iMinDEE, the conformations being minimized are not guaranteed to be the most promising ones. Furthermore, there is no finite number $n$ for a given protein design problem such that enumerating $n$ conformations by Monte Carlo is guaranteed to yield the GMEC.

Any minimization-aware method, however, will require a large number of subroutine calls to local minimization. Continuous energy minimization is computationally expensive, even with molecular mechanics-type energy functions that prioritize speed over accuracy. This causes the minimization of the energy function to be the bottleneck in protein design with continuous flexibility.

This bottleneck becomes more severe when more sophisticated energy functions are introduced. Computational protein design is typically performed with energy functions that prioritize speed over accuracy. For example, they typically use simplified implicit solvation models, such as EEF1.\textsuperscript{97} Vizcarra et al.\textsuperscript{162} have investigated the use of the Poisson-Boltzmann model, a much more accurate implicit solvation model, in protein design. They found it to be amenable to representation as a sum of residue-pair interactions—the form required for most protein design algorithms—but orders of magnitude more expensive than EEF1. Other methods to improve energy function accuracy are likely to face the same problem. For example, quantitatively accurate descriptions of most molecular interactions require computation of the electronic structure using quantum chemistry, but methods to do this are very computationally intensive.\textsuperscript{16} Methods to reduce the number of calls to an energy function needed in protein design could allow more accurate energy functions to be used, and thus yield more accurate results.

In protein design with only discrete flexibility, precomputation methods are typi-
cally used to reduce the number of energy function calls needed—that is, the number of conformations for which the energy must be evaluated. Before the design is started, the interaction energy of each pair of ideal, rigid rotamers at different residue positions is precomputed and stored in an *energy matrix*. Then, an ideal rotamer is chosen for each residue based on the energies in this matrix, and no further calls to the energy function are needed during the actual design calculation. The number of energy function calls required is thus quadratic in the number of residues in the system (that is, it scales as the number of pairs of residues). A precomputed energy matrix is, however, of limited use if we want to model continuous flexibility. No benefit in design is gained by performing *post-hoc* minimization on the best conformation found using ideal rotamers.\(^{40}\) This is true even if a high degree of flexibility is used for minimization, e.g., if molecular dynamics techniques are used, because the designed sequence is already determined before minimization is performed. In contrast, a minimization-aware search performs local continuous minimization for all rotamer assignments that might be optimal, in order to find the true GMEC.\(^{48}\)

Thus, an analogous energy matrix precomputation method for continuous flexibility would be very useful. It would ensure a polynomial number of energy function calls for minimization-aware protein design, in contrast to the exponential number of calls that may arise in minimization of all possibly optimal rotamer assignments (since the number of such assignments may be exponential with respect to the number of residues modeled). The search for rotamer assignments itself is unlikely to admit a polynomial-time algorithm, because it is NP-hard even to approximate.\(^{12,131}\) But a method to precompute pairwise energies for continuously flexible design would change the overall time cost from

\[(\text{a large rotamer search cost}) \times (\text{the energy function cost})\]

to

\[(\text{a large rotamer search cost}) + (\text{the energy function cost})\].

81
The rotamer search cost will necessarily be exponential in the worst case, if one wants to obtain the GMEC or an approximation to the GMEC within a fixed error threshold. But the energy function cost will be merely quadratic in the number of residues, indicating that the pairwise energy precomputation shifts the bottleneck away from the energy function calls. This brings the same improvement to minimization-aware design that energy matrix precomputation brought to non-continuously-flexible design.

We now present a pairwise energy precomputation method that admits continuous flexibility: EPIC (Energy as Polynomials in Internal Coordinates). EPIC computes a representation of the pairwise energy for each rotamer pair, not just at the rotamers’ ideal values of the internal coordinates, but for values within specified ranges around the ideal ones (Fig. 3.1). This computation is performed before the rotamer search computation is begun. This allows the rotamer search to substitute the new quickly evaluable representation for the original energy function. EPIC is implemented in the OSPREY\textsuperscript{41,48,49} open-source protein design package, which has yielded many designs that performed well experimentally—\textit{in vitro}\textsuperscript{13,36,42,54,139,143,151} and \textit{in vivo}\textsuperscript{36,54,139,143} as well as in non-human primates.\textsuperscript{143} EPIC provides a significant speedup when used with OSPREY’s default AMBER\textsuperscript{18,168} and EEF1\textsuperscript{97}-based energy function, but is also shown to be suitable for representing quantum-mechanical energies.

This paper makes the following contributions:

1. A compact, closed-form representation of energy as a function of continuous internal coordinates of a protein system.

2. A modified least-squares method to compute this representation.

3. A modified implementation of the iMinDEE\textsuperscript{40} and DEEPer\textsuperscript{68} protein design algorithms, integrated into the OSPREY\textsuperscript{13,36,41,48,49} open-source protein design...
Figure 3.1: (A) The energy of each residue is represented by EPIC as a polynomial in the internal coordinates, such as sidechain dihedrals $\chi$. Low-degree, inexpensive polynomials (blue) are tried first, and the degree is increased as needed to achieve a good fit (black) to the actual energy function (red). These polynomials are then used for design in place of the full energy function. (B) Interactions between pairs of residues are represented in terms of both residues’ internal coordinates.

package, that makes use of this representation to achieve substantial speedups. It is available online as free software.

4. Computational experiments showing that compact and accurate EPIC representations are possible both for the standard energy function in OSPREY and for energies obtained by quantum chemistry at the SCF and MP2 levels of theory.

5. Computational experiments showing that EPIC greatly speeds up minimization-aware protein design calculations, thus allowing designs to include not only more flexible residues, but also more conformational flexibility at those residues.
3.2 Methods

3.2.1 Preliminaries

EPIC, like most previous protein design algorithms, is designed for pairwise energy functions. Pairwise energy functions are sums of intra+shell and pairwise terms. Intra+shell terms are functions of the amino-acid type and conformation of one residue, and pairwise terms are functions of the amino-acid types and conformations of two residues. Each pairwise term represents the interaction between a pair of flexible residues, while each intra+shell term represents the internal energy of a residue plus its interactions with non-flexible “shell” residues (those that are frozen in a single, fixed conformation throughout the entire calculation). EPIC could be easily modified to include some higher-order terms for defined combinations of more than two residues: these terms can also be represented as polynomials in their residues’ degrees of freedom.

To find the GMEC, we must find an amino-acid type and conformation for each flexible residue such that the sum of intra+shell terms for all flexible residues, plus the sum of pairwise terms for all pairs of flexible residues, is minimized. This problem is referred to as conformational search. Conformational search can also comprise sequence search, by searching for the best conformation across many sequences’ conformational spaces. Many algorithms are available for this problem, including iMinDEE, and DEEPer, which solve it with provable accuracy. EPIC can be used in any conformational search algorithm that models continuous flexibility, because it provides polynomials that can be directly substituted for intra+shell and pairwise terms of the energy function.

The essence of EPIC is to exploit the fact that for each pairwise or intra+shell energy, the energy in the vicinity of the minimum can be described well by a relatively low-degree polynomial (usually quadratic total degree, sometimes higher; see
Fig. 3.5B). This description is computed using a modified least-squares method. We will refer to the states to which a residue may be assigned as residue conformations (RCs; cf. DEEPPer\textsuperscript{68}). In the absence of backbone flexibility, each RC will correspond to a sidechain rotamer. Within a residue conformation, the residue’s continuous energy variations can be described by a set of internal coordinates, which are subject to box constraints (i.e., bounds on each internal coordinate).

Herein, the word “polynomial” will be used in two very different senses in the description of EPIC below. First, EPIC is a polynomial representation of the energy, namely, a polynomial function with respect to the internal coordinates that is explicitly constructed by the EPIC algorithm. Second, a measure \( m \) of the computational complexity of an algorithm can be described as “polynomial”\textsuperscript{17,28} if, for input size \( n \), \( m \) grows no faster than \( n^d \) for a fixed exponent \( d \). In this case, one can construct a polynomial with respect to the size of the input that will be an upper bound on the computational cost, no matter how large the input is \((n)\). For this purpose, we will consider either the time or the number of energy function calls as \( m \)—these two measures of computational complexity are related to each other by a constant factor in current protein design algorithms with continuous flexibility, since energy function calls generally dominate the cost of these algorithms. For example, precomputation of an EPIC representation must be performed only once for every pair of RCs at different residues, and thus the number of polynomial fits (and thus the total time for precomputation of EPIC fits) does not grow faster than the square of the number of residues being modeled. On the other hand, protein design itself has been shown to be NP-hard,\textsuperscript{12,131} which means no polynomial-time algorithm is likely to exist for it. In other words, for every polynomial \( p(n_r) \) in the number \( n_r \) of residues, there are protein design problems of \( n_r \) residues that are not expected to be solvable in time \( p(n_r) \). A problem only solvable by exponential-time algorithms—those that take time scaling as \( b^n \), where \( b \) is a constant and \( n \) is the size of the input—would
Figure 3.2: The number of energy wells in a protein system scales exponentially with the number of flexible residues, leading to an exponential number of energy function calls, but EPIC can replace most of these calls with quick evaluations of low-degree polynomials. (Top) A protein may have an energy well for every combination of rotamers (rainbow) at different residues. The global minimum-energy conformation (GMEC) of a protein may be in any of these wells. We model the energy as a sum of pairwise energy terms. Each pairwise term will have wells for pairs of rotamers, but there are far fewer wells of this kind—a number quadratic in the number of residues. We can easily afford the energy function calls needed to characterize each pairwise well. (Bottom) By precomputing a polynomial representation (blue) of the energy within each well of each pairwise term (red), we enable computation of any pairwise term in any pairwise well, and thus of the full protein energy in any energy well of the protein, solely by a quick evaluation of polynomials.

typically be considered NP-hard.

Energy function calls are typically the bottleneck in protein design algorithms that model continuous flexibility. EPIC, however, ensures that the number of energy function calls in a protein design calculation is linear in the number of RC pairs, and thus polynomial in the size of the input (Fig. 3.2).
3.2.2 Basic least-squares method

Consider two interacting residues $i$ and $j$. Let us start with the “well-behaved” case where there exists a low-degree polynomial representation of a pairwise energy throughout the allowed ranges of both residues’ internal coordinates (of the form in Eq. 3.1).

We employ the notation introduced in the DEEPer algorithm. Suppose we have RCs $i_r$ and $j_s$ with pairwise energy $E(i_r, j_s, \mathbf{x})$, where $\mathbf{x}$ is the vector of internal coordinates (for example, dihedrals) affecting residues $i$ and $j$ when they have the amino-acid types corresponding to $i_r$ and $j_s$. Let $E_\Theta(i_r, j_s) = E(i_r, j_s, \mathbf{x}_0(i_r, j_s)) = \min_{\mathbf{x}} E(i_r, j_s, \mathbf{x})$, where the minimum is taken with respect to the internal coordinates over their allowed ranges for the current RCs. This definition of $E_\Theta$ is consistent with iMinDEE and DEEPer. $\mathbf{x}_0(i_r, j_s)$ is the set of internal coordinates that minimizes the pairwise energy.

We seek a multivariate polynomial $p_{i_r,j_s}(\mathbf{x})$ such that

$$E_\Theta(i_r, j_s) + p_{i_r,j_s}(\mathbf{x} - \mathbf{x}_0(i_r, j_s))$$

(3.1)

is a good approximation to $E(i_r, j_s, \mathbf{x})$. This multivariate polynomial is approximately a finite, low-degree Taylor expansion about the minimum. However, we use least-squares fits because we have found that they perform much better than Taylor expansions that are based on numerical derivatives. The fits are performed using a training set with ten times as many samples as there are parameters (polynomial coefficients) in the fit; the sampling procedure is described in Section 3.2.7. The fits are cross-validated with an independent set of samples (Section 3.2.6). The constraint $p(\mathbf{0}) = 0$ is applied, so the real energy and the polynomial will agree exactly at the minimum-energy point. This constraint is easily implemented by not including a constant term in the polynomial, and reflects the need for the highest accuracy to be attained for the lowest-energy, and thus most biophysically reasonable, conforma-
tions. As a result of this constraint, all values of the polynomial on its domain will be nonnegative. Fitting begins with a multivariate quadratic fit and then moves up to higher degrees as needed (see Section 3.2.6). Since polynomials are linear with respect to their coefficients, the fitting is a linear least-squares problem.

This method can be generalized without modification to intra-shell energies as well as to any continuous degrees of freedom, such as newly modeled backbone perturbations or rigid-body motions of ligands. In every case, the number of variables for the polynomial will be the number of continuous degrees of freedom that define the conformation of the residue or residue pair of interest. For example, in a pairwise energy computation for a rotamer of lysine and a rotamer of valine with only sidechain flexibility, the polynomial will be in five variables (the four dihedrals of lysine and one dihedral of valine). The polynomial coefficients are real numbers.

Let \( r \) be an RC assignment, represented as a tuple of RCs with one RC for each residue. Let \( i_r \) be the rotamer in \( r \) at residue \( i \). To approximate the minimized energy of an enumerated conformation \( r \), instead of minimizing the full energy

\[
E_r(x) = \sum_i E(i_r, x) + \sum_{j<i} E(i_r, j_r, x)
\]

(3.2)

with respect to the system’s continuous degrees of freedom \( x \), we simply minimize the polynomial approximation

\[
q_r(x) = \sum_i E_i(i_r) + p_{i_r}(x - x_0(i_r)) + \sum_{j<i} E_j(i_r, j_r) + p_{i_r,j_r}(x - x_0(i_r, j_r))
\]

(3.3)

with respect to \( x \). These least-squares approximations achieve high accuracy for the low-energy wells of rotamers and local backbone motions, i.e., the portions of conformational space where both the continuous degrees of freedom and the energy are relatively close to the local minimum of the pairwise energy. Higher energies may also be found close in conformational space to the local minimum, but these energies indicate strained conformations unlikely to be seen in nature. Thus, for a
“well-behaved” energy term whose energy is unstrained throughout the bounds on continuous degrees of freedom that define our current RCs, EPIC simply performs a least-squares fit of the energy, to represent it as a multivariate polynomial with respect to the continuous degrees of freedom.

Many RCs do however contain both regions with feasible energies and regions with higher energies that represent biophysically inaccessible conformations such as steric clashes. These RCs present difficulties for the basic least-squares fit, but the following algorithmic modification avoids this problem.

3.2.3 Modified least-squares method

To handle RCs with high-energy regions, we note that we do not necessarily need the polynomial to be a good approximation for the energy throughout the entire region allowed by the box constraints. We merely require that Eq. (3.2) be a good approximation for Eq. (3.3) when used with biophysically feasible, minimized values of \( \mathbf{x} \).

In particular, we can expect that the optimal, minimized structure has no clashes or other particularly large local strains. We have the advantage that while interaction energies in proteins can rise steeply towards infinity in the case of steric clashes, there is no physical phenomenon that will cause interaction energies to decrease steeply towards negative infinity. So energies are relatively well-behaved in low-energy regions. We can thus effectively partition the conformational space into relatively smooth, low-energy regions that we approximate accurately, and high-energy regions that we can rule out.

Let us denote the energy relative to the minimum as \( E'(i_r, j_s, \mathbf{x} - \mathbf{x}_0(i_r, j_s)) = E(i_r, j_s, \mathbf{x}) - E_\emptyset(i_r, j_s) \) in the pairwise case, or \( E'(i_r, \mathbf{x} - \mathbf{x}_0(i_r)) = E(i_r, \mathbf{x}) - E_\emptyset(i_r) \) in the intra+shell case. Our requirements for a “good approximation” of the energy can be defined rigorously in terms of two upper bounds \( b_1 \) and \( b_2 \) that we place on \( E' \). For each intra+shell or pairwise energy term, we estimate an upper bound
$b_1$ on $E'$ that we expect to hold for all minimized conformations that we want to output (the GMEC, or the lowest-energy $c$ conformations if we are computing a $c$-conformation ensemble). The algorithm will be able to check if $b_1$ is valid or not, so we can try again with a higher $b_1$ if needed. Additionally, we need a second, possibly looser upper bound $b_2$ on $E'$ that we are confident will be valid for all minimized conformations that we compute during our search, whether they turn out to be the GMEC or not. The value of $b_2$ must be the same for all intra-shell and pairwise terms ($b_1$ can be term-specific, though in practice a single value for $b_1$ is convenient).

If EPIC is being used with the iMinDEE algorithm for conformational search, we can provably obtain the GMEC without considering any conformations whose energies $E'$ exceed the pruning interval, an upper bound computed by iMinDEE for the difference between the lowest conformational energy lower bound (based on pairwise minimum energies) and the GMEC. Thus, when running iMinDEE, we can set $b_2$ equal to the iMinDEE pruning interval. When $b_2$ is set equal to the pruning interval, we know it is a valid upper bound on $E'$ for all minimized conformations computed during the search, and thus our GMEC calculation is provable. We can also do this when running DEEPER, which is essentially a backbone-flexible version of iMinDEE. For other algorithms we may want to set $b_2$ based on knowledge of the system being designed—setting $b_2 = 2b_1$ is likely to be an acceptable heuristic.

Our polynomial only needs to be a good fit to $E'$ for values of the internal coordinates where $E' \leq b_1$. For $E' > b_1$, we will require that the polynomial lie above $b_1$. This will ensure that when we enumerate conformations in order of minimized energy computed using polynomials, as long as the thresholds $b_1$ are chosen correctly, we will obtain non-clashing conformations before conformations with clashes, and these non-clashing conformations’ energies will be accurately represented by the polynomial fits. Furthermore, we will require that for $b_1 < E' < b_2$, the polynomial should be a lower bound on $E'$ (Fig. 3.3). This will ensure that regardless of what
thresholds $b_1$ were used, we never overestimate a conformational energy that is below the threshold $b_2$, and thus never exclude it from the enumerated list of conformations. The requirement to be a lower bound is easy to satisfy, because clashing van der Waals interactions are very steep and thus will tend to rise much more quickly than the polynomial fits. Thus, when we perform polynomial fits using thresholds, we know we will be getting a gap-free list of conformations in order of energy. If the thresholds $b_1$ were chosen to be too low, some higher-energy conformations with underestimated energy might be included as well, but these will be limited to minimized conformations containing energy terms with $E' > b_1$. This condition can be checked easily. If desired, the run can be redone with increased $b_1$ thresholds to eliminate this error. Thus, the choice of $b_1$ affects the ultimate speed of the algorithm but not its correctness.

For our experiments in this work (Section 3.3), we have set $b_1$ to 10 kcal/mol. This threshold was found to be sufficient for all experiments described in this work, and most other EPIC designs that we have tried. Physically, any pair of residues whose interaction energy is 10 kcal/mol worse than the optimal interaction for its current RC pair is likely in a highly strained conformation such as a steric clash. Thus a design requiring $b_1$ greater than 10 kcal/mol is likely to be biologically infeasible. For example, the protein is likely to unfold or undergo a large and unexpected structural change rather than suffer this local strain.

Let us use $z$ to denote a vector in the domain of our polynomial fit $p$. $p$ is considered a good representation of the energy if, for some small $\varepsilon > 0$, the following conditions are satisfied:

1. For $z$ such that $E'(i_r, j_s, z) \leq b_1$, $|p_{i_r,j_s}(z) - E'(i_r, j_s, z)| < \varepsilon$.

2. For $z$ such that $b_1 < E'(i_r, j_s, z) < b_2$, $b_1 - \varepsilon < p_{i_r,j_s}(z) < E'(i_r, j_s, z) + \varepsilon$.

3. For $z$ such that $b_2 \leq E'(i_r, j_s, z)$, $b_1 - \varepsilon < p_{i_r,j_s}(z)$.
Figure 3.3: (A) For each energy value $E'$, there is a range of “ideal” values for the EPIC fit (green). For the energies below cutoff $b_1$, which may be found in favorable conformations, this range is just the energy (the range has zero width). For higher energies, the range is defined using the cutoffs $b_1$ and $b_2$. For fitting purposes, EPIC fit values are penalized by the amount they lie outside the ideal range (the purple point represents a sample conformation for a given EPIC fit incurring the penalty indicated in red). (B) Example of curves satisfying these conditions. The EPIC fit matches the energy closely up to the cutoff $b_1$, after which it deviates from the energy, but stays in the target region shown in A, by staying below the energy. Once the energy is over $b_2$, the EPIC fit can be either above or below the true energy without leaving the target region.

These conditions are illustrated in Fig. 3.3. They can be achieved using a modified least-squares fit, using special “one-sided” penalties to enforce the inequalities in conditions 2 and 3, along with usual (two-sided) least-squares penalties to enforce condition 1. The objective function is the sum of terms from each sample in the training set. For a sample $z$ such that $E'(i_r, j_s, z) \leq b_1$, the objective function term is $(p_{i_r, j_s}(z) - E'(i_r, j_s, z))^2$ (as is typical for least squares). A term of this form is also used if the lower-bounding condition is violated, i.e., if $E' < b_2$ but $p > E'$. Otherwise, the objective function term for $z$ is $(p_{i_r, j_s}(z) - b_1)^2$ for $p_{i_r, j_s}(z) < b_1$, and 0 for $p_{i_r, j_s}(z) \geq b_1$.

If the modified least-squares method is applied to a set of samples that mostly have $E' > b_1$, then overfitting to the few points with $E' < b_1$ may occur no matter how many samples there are. As an extreme case, if all samples have $E' > b_2$, then almost any polynomial with very large values throughout its domain will give a 0
value for the objective function, but this may still provide a very poor description of the energy landscape. To avoid this situation, when a test set of $n$ samples is being drawn and $n/2$ samples with $E' > b_1$ have been drawn already, then if more samples come up with $E' > b_1$, they are redrawn to ensure that a sufficient number of samples with $E' \leq b_1$ is available (Section 3.2.7). Minimization-aware dead-end elimination pruning (both singles and pairs pruning) is performed before computation of the polynomial fits, since the pruned rotamers and pairs won’t be needed during enumeration. This pruning usually eliminates the clashing rotamers and pairs, leaving rotamers and pairs that are well suited for simple polynomial representations.

This objective function can be optimized efficiently because it is convex with respect to the polynomial coefficients (see Section 3.2.4). But we found general-purpose convex minimizers to be rather time-consuming for the higher-order fits. To address this, the algorithm described in Section 3.2.4 was developed. It exploits specific properties of the objective function to obtain a more efficient and reliable fit than a general-purpose convex minimizer would be likely to obtain.

### 3.2.4 A fast algorithm for modified least-squares fitting

The following algorithm performs a modified least-squares fit, providing a useful polynomial for energy terms that include both low-energy regions, where an accurate polynomial representation of the energy surface is required, and high-energy regions that we must exclude from our search.

Let us represent our polynomial fit $p(z)$ as $p \cdot y(z)$, where $p$ is the polynomial’s vector of coefficients, $y(z)$ is the corresponding vector of monomials built from the degree-of-freedom values $z$, and $\cdot$ is the standard inner product. For example, if $z$ consists of the two dihedrals $z_1$ and $z_2$ and we are performing a quadratic fit, then $y(z)$ will have the elements $1$, $z_1$, $z_2$, $z_1^2$, $z_2^2$, and $z_1z_2$. For each sample $s$ in our training set
of samples (see Section 3.2.7), let \( z_s \) be the vector of degree-of-freedom values, and let \( y_s = y(z_s) \) be the corresponding vector of monomials. Let \( E'_s \) be the energy for the sample, where the minimum-energy point is defined to have zero energy. Then, a modified least squares fit consists of minimizing the objective function \( f \) to obtain best-fit polynomial coefficients \( p_b \):

\[
p_b = \arg \min_p \left( \sum_{s \mid E'_s \leq b_1} (E'_s - p \cdot y_s)^2 + \sum_{s \mid p \cdot y_s \in E'_s, b_1 < E'_s < b_2} (E'_s - p \cdot y_s)^2 + \sum_{s \mid E'_s > b_1, p \cdot y_s < b_1} (b_1 - p \cdot y_s)^2 \right)
\]

(3.4)

where \( \{ s \mid E'_s \leq b_1 \} \) denotes the set of samples whose energies are less than or equal to \( b_1 \). If we define \( P_1 \) to be the set of sample points such that either

\[ E'_s \leq b_1 \]

(3.5)

or

\[ p \cdot y_s \geq E'_s, b_1 < E'_s < b_2 \]

(3.6)

and we define \( P_2 \) to be the set of sample points such that

\[ E'_s > b_1, p \cdot y_s < b_1 \]

(3.7)

then our objective function \( f \) becomes

\[
\sum_{s \in P_1} (E'_s - p \cdot y_s)^2 + \sum_{s \in P_2} (b_1 - p \cdot y_s)^2.
\]

(3.8)

Thus, if we know \( P_1 \) and \( P_2 \), minimizing the objective function is a basic least squares problem and can be solved analytically. Like basic least squares, this algorithm operates on a single “training” set of samples and provably minimizes the objective function (i.e., the error) for that training set.

We can show the objective function is convex with respect to \( p \) by noting that the contribution from each sample \( s \) is a function of the single linear combination \( u = p \cdot y_s \) of the elements of \( p \). This contribution depends on \( E'_s \), but it is always
convex (and piecewise quadratic). If $E'_s \leq b_1$, the contribution is just the parabola $(E'_s - u)^2$. If $b_1 < E'_s < b_2$, it’s the “truncated” or “flat-bottomed” parabola given by $(b_1 - u)^2$ for $u \leq b_1$, $0$ for $b_1 \leq u \leq E'_s$, and $(E'_s - u)^2$ for $u \geq E'_s$. Otherwise (if $b_2 \leq E'_s$), the contribution is the “one-sided” parabola given by $(b_1 - u)^2$ for $u < b_1$ and $0$ otherwise. Hence, the objective function is a sum of convex functions, making it convex itself. Thus, minimizing the objective function to find $\mathbf{p}$ is tractable, with any local minimum being the global minimum. As a result, we know that if for any sets of samples $P_1$ and $P_2$ we have coefficients $\mathbf{p}$ that minimize Eq. (3.8) and satisfy the conditions Eq. (3.5-3.7), then the coefficients $\mathbf{p}$ are globally optimal.

The algorithm finds $P_1$ and $P_2$ iteratively. As an initial guess, $P_1$ can be initialized to $s$ such that $E'_s \leq b_1$, and $P_2$ to be empty. (This corresponds to assuming that the one-sided restraints can all be satisfied perfectly.) This is followed by performing the basic least-squares computation of minimizing Eq. (3.8), which returns coefficients $\mathbf{p}$, and recalculating $P_1$ and $P_2$ from $\mathbf{p}$ using the conditions Eq. (3.5-3.7). This procedure is then repeated using the new $P_1$ and $P_2$ until a self-consistent solution is found. Generally, only a small minority of the samples will be moved in and out of the least-squares problem at each iteration, so the least-squares matrix can be updated quickly at each step—this is useful because forming this matrix is the bottleneck. Typically, only a few iterations are needed.

This algorithm is actually a special case of Newton’s method, because its estimate for the objective-function minimum at each iteration is the minimum of the local quadratic Taylor expansion of the objective function. This minimum can be found analytically because the local expansion is convex.

In our implementation of this algorithm, by far the bulk of its time cost is spent in forming the matrix for the first basic least-squares fit (with initial $P_1$ and $P_2$). The subsequent fits are much faster because they are only sparse updates. Thus, the modified least-squares fitting is only negligibly more expensive than the first basic
least-squares fitting.

3.2.5 Sparse atom-pair energies (SAPE)

SAPE is a method to reduce the degrees of polynomials needed by EPIC by including some non-polynomial terms in the representation of the energy.

The need for higher-order polynomial fits is driven by large values of higher derivatives. These values are contributed primarily by a small number of van der Waals (vdW) terms between pairs of atoms that are very near each other. It is possible to obtain substantial time and memory savings by evaluating these terms explicitly and fitting the rest of the energy function to a polynomial. To select atom pairs whose vdW terms are to be evaluated explicitly, a cutoff distance (3 or 4 Å; see Section 3.2.6) is chosen. Then, an atom pair’s vdW terms are evaluated explicitly if and only if the atoms can be found within that distance of each other within the bounds on internal coordinates for the given residue conformations. These terms are not polynomials in the degrees of freedom because they are inverse powers of distances between atoms, and the atom coordinates themselves are in general not polynomial functions of the degrees of freedom. For example, the expressions for atom coordinates in a sidechain in terms of the sidechain dihedral angles will include sines and cosines of those angles.

Once we decide to evaluate vdW terms for a given pair of atoms, it costs negligible extra time and memory to also calculate the electrostatic interaction between these atoms (since we already have the distance between the atoms).

3.2.6 Attaining the required accuracy

We will now describe the methods used to choose polynomial degrees for EPIC fits and ensure that fits of sufficient accuracy are obtained.

Fit accuracy is checked and controlled using cross-validation. For cross-validation
purposes, a mean-square error is computed, with absolute error used below \( E' = 1 \) kcal/mol and relative error above. This can be seen as a weighting of the error terms: the weight is 1 for \( E' \leq 1 \) and \( 1/\min(E', b_1) \) for \( E' \geq 1 \) (this levels off at \( b_1 \) to avoid excessive underweighting of the one-sided constraints). These weights, which are continuous with respect to \( E' \), are also used during the least-squares fitting.

Cross-validation is used to select the degree of the polynomial that is fit. Low-degree polynomials save time and memory both during the A* enumeration step and during the precomputation step, but may not provide a sufficiently good representation. Hence, we proceed through a sequence of increasingly expensive fits (Fig. 3.1A), and each time a fit is completed, it is cross-validated with an independently drawn set of samples. Like the training set, this cross-validation sample set has ten times as many samples as fit parameters. If the mean-square error is below a specified threshold, the fit is stored, and if it is above, we proceed to the next method. The default threshold value is set to \( 10^{-4} \). However, limited investigation suggests larger thresholds still tend to keep the errors in conformations’ minimized energies small compared to thermal energy, and thus are likely acceptable as well. It is also useful to avoid doing fits with very large number of parameters, as these have enormous time and memory costs both in the enumeration and precomputation steps. Thus, OSPREY is currently set to refuse to do fits with over 2000 parameters—this way, computations that would have prohibitive time costs may still be satisfactorily completed with a slightly higher error threshold than usual.

Some of the fits use lower-degree terms for all degrees of freedom and higher-order terms for selected degrees of freedom. These selected degrees of freedom are eigenvectors \( v_k \) of the Hessian from a modified least-squares quadratic fit (step 1 in the list of steps below). Letting \( \lambda_k \) be the eigenvalue corresponding to \( v_k \), we define

\[
D_q = \left\{ v_k \mid |\lambda_k| \geq \frac{\max_i |\lambda_i|}{q} \right\}
\]  

(3.9)
for $q > 0$. Let us define $f_n$ to be a polynomial fit of total degree $d$ (e.g., $f_2$ is a quadratic fit); $f_d(D_q)$ to be a fit to a polynomial of total degree $d$ in all degrees of freedom plus terms of total degree $d + 1$ and $d + 2$ in the degrees of freedom in $D_q$; and $s(n,c)$ to be a polynomial fit of total degree $d$ plus SAPE with a cutoff of $c$ Å. Fits were tried in the following order: $f_2$, $s(2,3)$, $f_2(D_{10})$, $f_2(D_{100})$, $f_4$, $s(4,4)$, $f_4(D_{10})$, $f_4(D_{100})$, $f_6$, and $s(6,4)$.

The Stone-Weierstrass theorem\textsuperscript{152} guarantees that a sufficiently high-degree polynomial can approximate any function on any closed and bounded portion of Cartesian space to any desired accuracy. In other words, it guarantees that any energy function can be represented by EPIC to arbitrary accuracy if we allow sufficiently high-degree polynomials. The basis of Bernstein polynomials can be used to construct such approximations with guaranteed convergence to any function.\textsuperscript{8} However, for the purpose of energy representation for protein design, modified least squares is likely to provide good approximations using much lower-degree polynomials than we would obtain using the Bernstein basis, because we do not need close approximations of the high energy in clashing regions. In these regions, we only need a reasonable lower bound that is much higher than the rotameric wells. This strategy keeps the polynomial degrees low enough to be practical.

3.2.7 Sampling to train and validate least-squares fits

Training and validation sets for EPIC fits consist of sample conformations of the residue(s) involved, specified as vectors of internal coordinates, drawn from throughout the allowed region of conformational space.

By default, samples for both training and validation sets were sampled uniformly (i.e., each degree of freedom was sampled uniformly and independently from the interval corresponding to the current rotamer or RC). Ten samples were always used in each of these training and validation sets for each parameter in a fit. However, if most
of the samples corresponded to energies above the threshold $b_1$ (see Section 3.2.3), then overfitting could result, because for such samples there are infinitely many polynomial values that yield zero error. To avoid this, we need sufficient samples from the set $B$ of conformations with energies below $b_1$; $B$ is the set of conformations where the polynomial needs to be quantitatively accurate. We ensure sufficient samples from $B$ by rejecting samples outside $B$ whenever we desire $n$ samples in total and we already have $n/2$ samples outside $B$, and thus drawing the rest of our samples uniformly from $B$ by rejection sampling. If 10,000 samples are rejected consecutively, indicating that $B$ is too small for efficient rejection sampling, then the Metropolis algorithm\textsuperscript{121} is used to sample from $B$.

We have confidence in the parameters obtained by fitting to the training samples for three reasons. First, a useful measure of the accuracy of a polynomial approximation to the energy surface is that there is a low probability that any region of the energy surface deviates significantly from the polynomial approximation (except for high-energy regions approximated by similarly high values of the polynomial). Since our cross-validation of each polynomial fit uses a large number of independent samples—ten times the number of parameters—we are left with a very low chance that our cross-validation samples will miss any such regions. Thus an insufficiently accurate polynomial surface will be detected upon cross-validation and remedied by an increase in polynomial degree. Second, errors in the minimized energies obtained using polynomial approximations are consistently low, as shown in our computational experiments (Table 3.1). Third, we expect the energy function to be relatively smooth in the vicinity of a minimum, since the gradient must be zero at the minimum, and thus we expect a polynomial of relatively low order (e.g., the Taylor series of the energy) to yield a good approximation in the vicinity of a minimum.
3.2.8  Application in proteins design algorithms

Once the polynomials are computed, they can be used in protein design algorithms wherever the energy function would ordinarily be called. The GMEC will simply be the set of rotamers for which the minimized value of Eq. (3.3) with respect to $x$ has the lowest possible value.

The simplest method to provably find the GMEC using EPIC is to use a protein design algorithm that enumerates conformations in order of a lower bound, and then instead of minimizing the full energy (Eq. 3.2), merely minimizing the polynomial-based energy (Eq. 3.3) to compute the energy for each enumerated conformation (Fig. 3.2). For example, iMinDEE/A*\textsuperscript{40} can be used for this enumeration process, and we use this algorithm in our computational experiments (Section 3.2.10).

EPIC can also be applied in free energy calculations using the $K^{*}$ algorithm,\textsuperscript{48,108} which approximates binding constants as ratios of partition functions computed from low-energy conformations enumerated by $A^{*}$. During these calculations, one can simply use the polynomials instead of the energy function to compute the partition function, given the enumerated RC assignments. This method gives a constant-time speedup, determined by the ratio of time to evaluate the energy function versus the EPIC energy.

An additional speedup is possible for branch-and-bound protein design algorithms (e.g., $A^{*}$\textsuperscript{73,99}) that use a tree structure for conformational search. These algorithms build nodes that each represent a subset of conformational space and are scored using a lower bound on the conformational energies in that space. In each node’s conformational space, some residues are restricted to a single RC; these RCs are referred to as assigned to their respective residues. At each level of the tree, an RC is assigned to one more residue. One can use the EPIC polynomials to improve the lower-bound energy for each of these nodes. At each node, we need to compute a
lower bound $L$ for the conformational energy $q_r$, which is defined in Eq. (3.3): that is, we compute $L$ such that

$$L \leq q_r(x) = \sum_i E_\Theta(i_r) + p_{i_r}(x - x_0(i_r)) + \sum_{j < i} E_\Theta(i_r, j_r) + p_{i_r,j_r}(x - x_0(i_r, j_r))$$  

(3.10)

for all RC assignments $r$ and all degree-of-freedom values $x$ that are part of the node’s conformational space. If $r$ is known (i.e., if RCs are fully assigned at all residue positions), then a tight lower bound can be computed trivially by local minimization with respect to $x$. Otherwise, we let $q_r(x) = E_\Theta(r) + E_p(r, x)$, where $E_p$ consists only of EPIC polynomials:

$$E_\Theta(r) = \sum_i E_\Theta(i_r) + \sum_{j < i} E_\Theta(i_r, j_r)$$  

(3.11)

$$E_p(r, x) = \sum_i p_{i_r}(x - x_0(i_r)) + \sum_{j < i} p_{i_r,j_r}(x - x_0(i_r, j_r))$$  

(3.12)

Now, if we compute lower bounds $L_\Theta$ and $L_p$ such that $L_\Theta \leq E_\Theta(r)$ and $L_p \leq E_p$ for all $r, x$ in our conformational space, then $L = L_\Theta + L_p$ will satisfy Eq. (3.10), giving us a valid lower bound. Computation of $L_\Theta$ has been described previously, because lower bounds of this form are computed in iMinDEE$^{40}$ and DEEPer.$^{68}$ To compute $L_p$, we use the fact that EPIC polynomials are always nonnegative; thus, for any $r$ and $x$ and any subset $S$ of the residues we are modeling,

$$\sum_{i \in S} p_{i_r}(x - x_0(i_r)) + \sum_{j \in S, j < i} p_{i_r,j_r}(x - x_0(i_r, j_r)) \leq E_p(r, x)$$  

(3.13)

If we let $S$ be the set of residues with fully assigned RCs, then there is only one possible RC $i_r$ for each residue $i \in S$, and so we can find the minimum of Eq. (3.13),

$$\min_{r, x} \left( \sum_{i \in S} p_{i_r}(x - x_0(i_r)) + \sum_{j \in S, j < i} p_{i_r,j_r}(x - x_0(i_r, j_r)) \right)$$  

(3.14)

101
exactly by local minimization with respect to $x$. Eq. (3.14) is a lower bound on $E_p(r, x)$, and thus we set $L_p$ equal to it, giving us a score for our node. We note that $L_p$ is strictly nonnegative, because Eq. (3.13) and thus Eq. (3.14) are always nonnegative.

Because this continuous minimization with respect to $x$ is more expensive and has to be performed separately at each node, it is evaluated in a lazy$^{17}$ fashion in our A* implementation. Nodes are assigned the traditional, discrete lower bound $L_\Theta$ when they are generated; this bound is fast to compute. The A* priority queue contains nodes both with and without the polynomial contribution $L_p$ included. When a new node is popped from the queue, we check if $L_p$ is present or not. If it is, we expand the node, and if it is not, we compute $L_p$ and insert the node back in the priority queue. This ensures that nodes come off the priority queue in order of their complete lower bound $L_\Theta + L_p$, as is necessary for A* to function correctly. However, it also ensures that we do not waste time computing $L_p$ for nodes whose $L_\Theta$ is high enough to preclude expansion. This method gives a combinatorial speedup, since a high polynomial contribution for a partial conformation can effectively prune an entire branch of the A* tree. In practice, though, the large constant speedup from EPIC minimization of fully assigned conformations tends to be more significant (Section 3.3.1, Table 3.2).

3.2.9 Complexity of energy evaluations

The speedup due to EPIC can be explained in terms of the asymptotic costs of EPIC polynomial evaluations compared to direct energy function calls. The cost of evaluating an EPIC polynomial scales as the number of terms in the polynomial. This cost is itself a polynomial (usually quadratic) in the number of internal coordinates of the residue pair (or single residue, in the intra+shell case) of interest. By contrast, the cost of evaluating a molecular mechanics-based energy function is generally
quadratic in the number of atoms involved, since distances between all pairs of atoms need to be considered. EPIC achieves a marked speedup because most residues have far more atoms than significantly flexible internal coordinates. For example, most protein residues have two or fewer sidechain dihedrals, but over ten atoms. The remaining internal coordinates—bond length, angles, etc.—are relatively inflexible. Polynomial evaluations are also performed entirely by addition and multiplication, which are much faster than the more complicated elementary operations (trigonometric functions, square roots, etc.) needed to evaluate molecular mechanics energy terms.

When quantum-mechanical energy functions are introduced, all the electrons must be accounted for explicitly, and even fairly approximate quantum-chemical methods have time costs that are higher-order polynomials with respect to the number of electrons. For example, any method that accounts for repulsions between all atomic orbitals (e.g., Hartree-Fock and all post-Hartree-Fock methods) must calculate repulsion integrals for all quadruples of atomic orbitals, and there are at least as many atomic orbitals as electrons. And there are far more electrons than there are atoms, and far more atoms than internal coordinates, giving EPIC an extreme performance advantage. Yet EPIC can represent the same energy surface to a high degree of accuracy, once the EPIC polynomials have been precomputed.

Whether EPIC is used or not, these types of pairwise energy evaluations must be performed for every pair of residues in a design system. In general, this means the number of pairwise energy evaluations needed is quadratic with respect to the number of residues in the system. This number can be reduced if a cutoff is applied to remove interactions between distant residues. However, this speedup applies equally for EPIC and non-EPIC calculations.
3.2.10 Computational experiments

Protein design calculations were performed in OSPREY\textsuperscript{13,36,41,48,49} with and without EPIC to investigate (a) what previously intractable systems become newly tractable with EPIC, (b) what speedups EPIC brings to conformational enumeration for previously tractable systems, and (c) what types of polynomial representations are needed for these purposes. EPIC runs were performed with SAPE and with conformational minimization for partially assigned conformations during A* search, and for comparison, runs with either one of these features omitted were also performed.

Times were compared for the A* search, including conformation enumeration and minimization, because this is the portion of the design that is not guaranteed to complete in polynomial time and thus is the bottleneck. As part of the EPIC runs, GMEC energies were also computed using the regular energy function to compare to the EPIC results, and the ratio of minimization times with and without EPIC was computed. For runs with multiple conformations very close in energy to each other (within the error range of EPIC, typically $<0.1$ kcal/mol), the time ratios were averaged.

All minimizations were performed using a cyclic coordinate descent minimizer, which is now included in OSPREY. Default OSPREY energy function settings were used where applicable: AMBER with EEF1 solvation and a distance-dependent dielectric constant of 6. Rotamers were determined using the Penultimate rotamer library.\textsuperscript{115}

Test systems were chosen to evaluate both partition function and GMEC calculations, and to include all three types of continuous degrees of freedom used in OSPREY: sidechain dihedrals, backbone perturbation (shear and backrub) parameters,\textsuperscript{68} and rigid-body rotations and translations of strands. Some of the tests are intended to be within the scope of previous methods, allowing a quantitative comparison of running times, while others are intended to show EPIC can compute previously intractable
GMECs and partition functions with provable accuracy.

For GMEC calculations (Table 3.1), the first set of systems used was taken from Gainza, Roberts, and Donald,\textsuperscript{40} and featured only sidechain dihedral flexibility. The structures for these correspond to PDB codes 2o9s, 2qsk, 2rh2, 2ril, and 3g36. The second set of systems was taken from Hallen, Keedy, and Donald,\textsuperscript{68} and included both sidechain and backbone flexibility. The structures’ PDB codes were 1aho, 1c75, 1cc8, 1f94, 1fk5, 1i27, 1iqz, 1jhg, 1l6w, 1l7a, 1l7l, 1l7m, 1l8n, 1l9l, 1l9x, 1lb3, 1m1q, and 1mwq. Three variants of the 1aho system with more residues were tried as well. Finally, a GMEC calculation was performed for the complex of the HIV surface protein gp120 with the broadly neutralizing antibody NIH45-46 (PDB code 3u7y\textsuperscript{27}).

To investigate the application of EPIC to partition function calculations (Table 3.2), we first chose systems with only sidechain dihedral flexibility from Gainza, Roberts, and Donald\textsuperscript{40} and calculated a partition function for the unliganded protein, with wild-type amino acids at all residue positions, to within 97% guaranteed accuracy. Partition function calculations such as these are the key operation in $K^\star$\textsuperscript{48,108} calculations. The structures for these correspond to PDB codes 2cs7, 2o9s, 2p5k, 2qsk, 2r2z, 2rh2, 2ril, 2wj5, 2zxy, 3a38, 3dnj, 3fgv, 3fil, 3g21, 3g36, 3hfo, and 3i2z. Furthermore, a $K^\star$ run is presented for trypsin with a small-molecule inhibitor (PDB code 3pwc); the run is tractable with EPIC but fails to finish without it. Unlike the calculations for the other, monomeric structures, the $K^\star$ run for trypsin involves calculation of three partition functions: one for the protein, one for the ligand, and one for the complex.

Each design was allowed 17 days of total runtime, after which those that had not finished were deemed to have exceeded the time limit and were terminated. $A^\star$ times with EPIC ranged from 0.7 seconds to 4 days, and the speedups due to EPIC are shown in Tables 3.1 and 3.2.

In these experiments, fitting was performed without parallelization. However,
the computation of the EPIC polynomial for each pair of RCs is an independent operation, so each can be done in parallel, meaning that parallelization to $p$ processors will give a $p$-fold speedup as long as $p$ does not approach the number of RC pairs. OSPREY currently supports computation of each residue pair in parallel, so the speedup holds as long as $p$ does not approach the number of residue pairs. In practice however, for large systems, pruning and A* take longer than the polynomial fitting, so this parallelization may not be necessary. Additionally, once the EPIC fits have been computed for a system, there may be a large number of computations that can be performed using it—calculation of partition functions for many sequences, computation of GMECs for various subsets of the sequence space, etc. These extensive reuses of the fits may be especially desirable when designing a library of sequences for experimental testing—if one performs various optimizations with different assumptions and tests top sequences from each optimization, the results will be more robust to errors in the assumptions.

To investigate the ability of EPIC to represent quantum-mechanical energy functions, EPIC calculations were also performed on the aspartame dipeptide (extracted from PDB code 1a8j\textsuperscript{30}) with the energies for EPIC samples evaluated using NWChem\textsuperscript{157} instead of using OSPREY’s usual energy function. Calculations were performed at the SCF level of theory with STO-3G and with 6-31G** basis sets, and also at the MP2 level of theory with a STO-3G basis set.\textsuperscript{16} For each rotamer of each residue, dihedrals were sampled within the allowed range for the rotamer and the total energy of the dipeptide was fit to a polynomial.

3.2.11 Applications of EPIC to other algorithms

For this study, EPIC was implemented in the context of the OSPREY protein design package OSPREY\textsuperscript{13,36,41,48,49} to run along with the algorithms (iMinDEE, DEEPer, and $K^*$) and pairwise energy functions already implemented in OSPREY. However,
EPIC would enable some other capabilities in different implementations.

First, EPIC can be applied in the context of other protein design algorithms. For example, one can apply it an iterative algorithm like FASTER\textsuperscript{26} or Monte Carlo\textsuperscript{121} that tries to find a suitably low-energy conformation by accepting or rejecting rotamer changes based on the energies of conformations with these changes. Whenever the energy is needed for a rotamer assignment, the EPIC energy for the protein can be locally minimized starting at the ideal internal coordinate values for that rotamer assignment. In this case, the matrix of EPIC polynomials substitutes directly for the matrix of pairwise rotamer energies commonly used to calculate conformational energies for these algorithms in the absence of continuous flexibility. EPIC could even be used for molecular dynamics, since most residue pairs in a molecular dynamics trajectory\textsuperscript{134} will spend most of their time in fairly relaxed conformations—the region of conformational space modeled by EPIC. In all these cases, EPIC energy evaluations would be markedly faster than regular energy function calls, particularly for expensive energy functions. Thus, EPIC would provide a substantial speedup for any algorithm whose bottleneck is energy function calls.

For docking algorithms\textsuperscript{11} that require sidechain optimization or local backbone optimization, EPIC can be used both for conformation scoring and for conformational optimization using any of the above algorithms.

3.2.12 EPIC can accommodate higher-than-pairwise energies

EPIC was implemented in this work to handle pairwise energy functions (in the sense of a sum of 1-body and 2-body energies with no terms dependent on three or more residues’ degrees of freedom), because these are currently typical for protein design and include the AMBER, CHARMM, and EEF1 energy functions we use in OSPREY. However, the true energy of proteins is not exactly pairwise decomposable, and EPIC could easily accommodate higher-order terms. EPIC simply requires that
each energy term correspond to a set \( D \) of degrees of freedom, constrained to a region in which they are relatively well-behaved (e.g., dihedrals at each residue constrained to a single rotamer); we can sample \( D \) subject to the constraints and then fit the energies as a polynomial function with domain \( D \). Thus, for any set \( R \) of more than two interacting residues, for each RC assignment to those residues, we can fit an EPIC polynomial, and thus describe the energy terms for \( R \).

In practice, the number of sets of residues that can interact significantly is quite limited, because residues typically must be physically near each other to have significant higher-than-pairwise interactions. For example, if we have a Ramachandran-based potential, its terms each depend on the \( \psi \) and \( \phi \) backbone dihedrals of a certain residue \( r \), and thus depend on the conformations of the three residues \( r - 1, r, \) and \( r + 1 \). Likewise, the conformation of a residue \( i \) can induce polarization effects in a nearby residue \( j \) that will affect the interactions of \( j \) with another residue \( k \), and this effect can be quantified using quantum chemistry, but \( i \) and \( j \) have to be physically very close to each other (\( \ll 1 \) nm) for this effect to be significant (and \( j \) and \( k \) have to be fairly close too—probably subnanometer as well, since the potential of an induced dipole falls off faster than \( 1/d^2 \) with distance \( d \)). Hence, EPIC can be used to model any realistic energy function, by accounting for all sets of residues with significant energetic interactions.

3.3 Results

Computational experiments were performed to measure what kinds of polynomial fits are necessary to accurately model different proteins with different degrees of freedom and energy functions, and what speedups EPIC brings to DEE/A* and K* calculations. The results demonstrate that EPIC brings a substantial speedup to design calculations when proteins are modeled as in previous OSPREY designs.\textsuperscript{13,36,40,41,68} They also show that EPIC efficiently represents energies calculated by quantum
chemistry, and is a potentially decisive tool for using both realistic, continuous flexibility and quantum-mechanical energy functions in protein design.

3.3.1 Application to protein designs

First, computational experiments were performed to compare GMEC search with and without EPIC, as described in Section 3.2.10. Key portions of the design calculation were timed with and without EPIC to determine the speedup for these portions (Table 3.1). On average, minimization of fully enumerated conformations was 79-fold faster using EPIC than with traditional energy function calls. Overall A* speedups due to EPIC averaged 167-fold (Fig. 3.5A). The overall A* speedup is likely greater than the minimization speedup because of the way OSPREY’s standard energy function is implemented. Each time the energy function is run on a new sequence, setup time (e.g., initialization of the energy function) is required to identify electrostatic, van der Waals, and solvation terms that will be necessary for that sequence. This setup time is eliminated by EPIC, and is not counted as part of the minimization time here, but it may be performed an exponential number of times without EPIC, since minimizations may be required for an exponential number of sequences. Runs that did not finish without EPIC are not included in these averages. 85% of the fits in these experiments were quadratic, with no SAPE needed (Figure 3.5B). GMECs from EPIC runs showed good agreement between energies from minimization of EPIC energies and energies from the actual energy function. The average energy difference was 0.04 kcal/mol, which is less than one-tenth of thermal energy at room temperature (0.592 kcal/mol, calculated as the universal gas constant times a room temperature of 298° K) and thus functionally insignificant.

Five of the 27 systems finished only with EPIC, demonstrating that EPIC allows design of larger and more diverse systems than were previously designable. For example, a redesign of the complex of HIV surface protein gp120 with the antibody
Table 3.1: GMEC calculation test cases. EPIC with SAPE solved each of these cases; in most cases, A* without EPIC was slower or did not finish within the time limit (\(^1\)). Minimization ("Min.") speedup (ratio of single-conformation minimization time without EPIC vs. time with EPIC) is reported for EPIC with and without SAPE. Similarly, A* speedup denotes the ratio of total A* time without EPIC vs. time with EPIC. Fit times (times to calculate the EPIC polynomials) and total calculation times (Tot. time) are reported for EPIC calculations with SAPE, in minutes, calculated without parallelization of fitting. "GMEC energy error" is the absolute difference between GMEC energies calculated with and without EPIC, and is reported in kcal/mol. "Fit DNF” means the time limit was exceeded during polynomial fit precomputation, and “Bad fit” means that fits for EPIC without SAPE, even at the maximum allowed polynomial degree, did not accurately represent the energy. This problem is rescued by SAPE for these cases. When neither A* without EPIC nor EPIC without SAPE was successful, the A* speedup without SAPE cannot be calculated and is listed as “n/a.” EPIC was performed with minimization of partial conformations in all cases.

<table>
<thead>
<tr>
<th>Protein name</th>
<th>PDB code</th>
<th>Mutatable residue count</th>
<th>Min. speedup with SAPE</th>
<th>Min. speedup w/o SAPE</th>
<th>A* speedup with SAPE</th>
<th>A* speedup w/o SAPE</th>
<th>Fit time (min)</th>
<th>Tot. time (min)</th>
<th>GMEC energy error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>7</td>
<td>26.58</td>
<td>13.60</td>
<td>26.88</td>
<td>48.84</td>
<td>42</td>
<td>49</td>
<td>0.01</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>9</td>
<td>23.59</td>
<td>14.53</td>
<td>105.70</td>
<td>143.52</td>
<td>435</td>
<td>456</td>
<td>0.03</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>12</td>
<td>29.66</td>
<td>Fit DNF</td>
<td>&gt;142.9</td>
<td>n/a</td>
<td>566</td>
<td>1119</td>
<td>0.04</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>14</td>
<td>26.49</td>
<td>Fit DNF</td>
<td>&gt;4.30</td>
<td>n/a</td>
<td>8079</td>
<td>19446</td>
<td>0.05</td>
</tr>
<tr>
<td>Cytochrome c553</td>
<td>1c75</td>
<td>6</td>
<td>41.03</td>
<td>24.48</td>
<td>102.23</td>
<td>73.78</td>
<td>57</td>
<td>63</td>
<td>0.01</td>
</tr>
<tr>
<td>Atx1 metallochaperone</td>
<td>1cc8</td>
<td>7</td>
<td>36.49</td>
<td>Bad fit</td>
<td>75.97</td>
<td>Bad fit</td>
<td>332</td>
<td>630</td>
<td>0.05</td>
</tr>
<tr>
<td>Bucandin</td>
<td>1f94</td>
<td>7</td>
<td>15.89</td>
<td>5.03</td>
<td>250.13</td>
<td>179.70</td>
<td>243</td>
<td>427</td>
<td>0.03</td>
</tr>
<tr>
<td>Nonspecific lipid-transfer protein</td>
<td>1fk5</td>
<td>6</td>
<td>35.25</td>
<td>144.14</td>
<td>24.17</td>
<td>24.87</td>
<td>4</td>
<td>6</td>
<td>0.01</td>
</tr>
<tr>
<td>Transcription factor IIF</td>
<td>1i27</td>
<td>7</td>
<td>27.66</td>
<td>Fit DNF</td>
<td>320.41</td>
<td>Fit DNF</td>
<td>2205</td>
<td>3793</td>
<td>0.03</td>
</tr>
</tbody>
</table>

\(^1\) A* without EPIC did not finish within the time limit, so we report a lower bound on the speedup: the ratio of the time limit (17 days) to the A* time with EPIC and SAPE. (Table continues on next page)

NIH45-46 did not finish when run without EPIC, but finished with EPIC using about a day of A* time (Fig. 3.4). This redesign allowed the mutation of 16 residues all over the gp120 surface in the interface—five in the D-loop of gp120,\(^96\) which is central to the interaction with NIH45-46, and the other 11 scattered through other parts of the interface in various types of secondary structure. Redesigns of the gp120 surface to achieve specific binding to particular antibodies has been instrumental in
<table>
<thead>
<tr>
<th>Protein name</th>
<th>PDB code</th>
<th>Mutatable residue count</th>
<th>Min. speedup with SAPE</th>
<th>Min. speedup no SAPE</th>
<th>A* speedup with SAPE</th>
<th>A* speedup no SAPE</th>
<th>Fit time (min)</th>
<th>Tot. time (min)</th>
<th>GMEC energy error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ferredoxin</td>
<td>1iqz</td>
<td>9</td>
<td>59.36</td>
<td>132.22</td>
<td>96.37</td>
<td>127.82</td>
<td>36</td>
<td>41</td>
<td>0.02</td>
</tr>
<tr>
<td>Trp repressor</td>
<td>1jhg</td>
<td>7</td>
<td>41.16</td>
<td>Bad fit</td>
<td>139.39</td>
<td>Bad fit</td>
<td>162</td>
<td>367</td>
<td>0.04</td>
</tr>
<tr>
<td>Fructose-6-phosphate aldolase</td>
<td>16w</td>
<td>6</td>
<td>194.30</td>
<td>178.60</td>
<td>152.26</td>
<td>77.73</td>
<td>197</td>
<td>264</td>
<td>0.03</td>
</tr>
<tr>
<td>Cephalosporin C deacetylase</td>
<td>17a</td>
<td>8</td>
<td>157.71</td>
<td>Fit DNF</td>
<td>&gt;3170²</td>
<td>n/a</td>
<td>3314</td>
<td>4784</td>
<td>0.04</td>
</tr>
<tr>
<td>PA-I lectin</td>
<td>17l</td>
<td>6</td>
<td>58.21</td>
<td>141.90</td>
<td>30.77</td>
<td>32.60</td>
<td>41</td>
<td>55</td>
<td>0.01</td>
</tr>
<tr>
<td>Phosphoserine phosphatase</td>
<td>17m</td>
<td>7</td>
<td>69.84</td>
<td>34.073</td>
<td>810.68</td>
<td>998.47</td>
<td>246</td>
<td>624</td>
<td>0.03</td>
</tr>
<tr>
<td>alpha-D-glucuronidase</td>
<td>18n</td>
<td>5</td>
<td>355.82</td>
<td>443.90</td>
<td>0.99</td>
<td>0.93</td>
<td>691</td>
<td>1410</td>
<td>0.01</td>
</tr>
<tr>
<td>Granulysin</td>
<td>19l</td>
<td>7</td>
<td>50.66</td>
<td>34.25</td>
<td>2.54</td>
<td>13.60</td>
<td>94</td>
<td>119</td>
<td>0.01</td>
</tr>
<tr>
<td>gamma-glutamyl hydrolase</td>
<td>19x</td>
<td>5</td>
<td>111.57</td>
<td>468.00</td>
<td>1.86</td>
<td>1.87</td>
<td>103</td>
<td>264</td>
<td>0.00</td>
</tr>
<tr>
<td>Ferritin</td>
<td>1lb3</td>
<td>5</td>
<td>103.05</td>
<td>33.31</td>
<td>51.68</td>
<td>49.33</td>
<td>148</td>
<td>174</td>
<td>0.01</td>
</tr>
<tr>
<td>Cytochrome c</td>
<td>1m1q</td>
<td>8</td>
<td>80.79</td>
<td>61.94</td>
<td>320.23</td>
<td>486.77</td>
<td>39</td>
<td>97</td>
<td>0.05</td>
</tr>
<tr>
<td>Hypothetical protein YciI</td>
<td>1mwq</td>
<td>8</td>
<td>60.05</td>
<td>48.30</td>
<td>56.10</td>
<td>16.41</td>
<td>60</td>
<td>133</td>
<td>0.00</td>
</tr>
<tr>
<td>Ponsin</td>
<td>2o9s</td>
<td>14</td>
<td>68.72</td>
<td>Bad fit</td>
<td>&gt;45.81</td>
<td>n/a</td>
<td>218</td>
<td>1732</td>
<td>0.41</td>
</tr>
<tr>
<td>Scytovirin</td>
<td>2qsk</td>
<td>10</td>
<td>64.94</td>
<td>35.45</td>
<td>296.06</td>
<td>329.99</td>
<td>95</td>
<td>188</td>
<td>0.12</td>
</tr>
<tr>
<td>Dihydrofolate reductase</td>
<td>2rh2</td>
<td>14</td>
<td>54.92</td>
<td>30.28</td>
<td>771.86</td>
<td>521.44</td>
<td>9</td>
<td>37</td>
<td>0.01</td>
</tr>
<tr>
<td>Putative monooxygenase</td>
<td>2rii</td>
<td>8</td>
<td>159.98</td>
<td>580.00</td>
<td>28.36</td>
<td>66.72</td>
<td>7</td>
<td>11</td>
<td>0.01</td>
</tr>
<tr>
<td>dpy-30-like protein</td>
<td>3g36</td>
<td>4</td>
<td>36.44</td>
<td>17.96</td>
<td>5.77</td>
<td>10.00</td>
<td>23</td>
<td>30</td>
<td>0.05</td>
</tr>
<tr>
<td>HIV gp120</td>
<td>3u7y</td>
<td>16</td>
<td>141.17</td>
<td>Bad fit</td>
<td>&gt;14.33</td>
<td>n/a</td>
<td>1773</td>
<td>7762</td>
<td>0.06</td>
</tr>
</tbody>
</table>

the development of probes to isolate these antibodies from sera. Redesign of the antibody surface of a gp120-antibody complex has also been effective in optimizing antibody affinity, which is useful for passive immunization and immunogen design. Interestingly, the redesign of the NIH45-46 complex yielded 12 top conformations within 0.06 kcal/mol of each other—two from the top sequence and ten from a double mutant. This high density of favorable conformations suggests the complex is entropically favored, a result consistent with the observed high affinity of NIH45-46 for gp120, attained through extensive affinity maturation of the antibody.

Two variations of EPIC were also tried for these systems. It was found that minimization of partial conformations during A* (Section 3.2.8) provides a speedup
Table 3.2: Partition function calculation results. EPIC was performed with SAPE in all cases. As in Table 3.1, EPIC solved all cases, and cases for which A* without EPIC did not finish are denoted by †. a Ratio of total A* time without EPIC vs. total A* time with EPIC and minimization of partial conformations. b Ratio of total A* time with EPIC but no minimization of partial conformations vs. total A* time with EPIC and minimization of partial conformations.

<table>
<thead>
<tr>
<th>Protein name</th>
<th>PDB code</th>
<th>Mutableresidue count</th>
<th>A* speedup due to EPIC&lt;sup&gt;a&lt;/sup&gt;</th>
<th>A* speedup due to EPIC&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histidine triad protein</td>
<td>2cs7</td>
<td>14</td>
<td>&gt;48.48†</td>
<td>1.52</td>
</tr>
<tr>
<td>Ponsin</td>
<td>2o9s</td>
<td>14</td>
<td>&gt;1131†</td>
<td>0.96</td>
</tr>
<tr>
<td>Transcriptional regulator AhrC</td>
<td>2p5k</td>
<td>11</td>
<td>&gt;256.0†</td>
<td>1.34</td>
</tr>
<tr>
<td>Scytovirin</td>
<td>2qsk</td>
<td>10</td>
<td>5698.04</td>
<td>1.01</td>
</tr>
<tr>
<td>Hemolysin</td>
<td>2r2z</td>
<td>12</td>
<td>&gt;217.7†</td>
<td>1.24</td>
</tr>
<tr>
<td>Dihydrofolate reductase</td>
<td>2rh2</td>
<td>14</td>
<td>&gt;50.45†</td>
<td>2.16</td>
</tr>
<tr>
<td>Putative monoxygenase</td>
<td>2ril</td>
<td>8</td>
<td>1121.34</td>
<td>1.01</td>
</tr>
<tr>
<td>alpha-cry stallin</td>
<td>2wj5</td>
<td>15</td>
<td>&gt;29.06†</td>
<td>0.80</td>
</tr>
<tr>
<td>Cytochrome c555</td>
<td>2zxy</td>
<td>14</td>
<td>&gt;8.87†</td>
<td>3.32</td>
</tr>
<tr>
<td>High-potential iron-sulfur protein</td>
<td>3a38</td>
<td>13</td>
<td>&gt;1916†</td>
<td>1.40</td>
</tr>
<tr>
<td>ClpS protease adaptor</td>
<td>3dnj</td>
<td>12</td>
<td>&gt;50.25†</td>
<td>0.93</td>
</tr>
<tr>
<td>Putative monoxygenase</td>
<td>3fgv</td>
<td>10</td>
<td>&gt;9639†</td>
<td>1.57</td>
</tr>
<tr>
<td>Protein G</td>
<td>3fil</td>
<td>14</td>
<td>&gt;1475†</td>
<td>1.31</td>
</tr>
<tr>
<td>Viral capsid</td>
<td>3g21</td>
<td>15</td>
<td>&gt;7.84†</td>
<td>1.21</td>
</tr>
<tr>
<td>dpy-30-like protein</td>
<td>3g36</td>
<td>4</td>
<td>211.87</td>
<td>0.91</td>
</tr>
<tr>
<td>Hfq protein</td>
<td>3hfo</td>
<td>10</td>
<td>&gt;194.8†</td>
<td>1.45</td>
</tr>
<tr>
<td>Cold shock protein</td>
<td>3i2z</td>
<td>14</td>
<td>&gt;872.8†</td>
<td>1.24</td>
</tr>
<tr>
<td>Trypsin</td>
<td>3pwc</td>
<td>10</td>
<td>&gt;624.2†</td>
<td>1.04</td>
</tr>
<tr>
<td>Trypsin</td>
<td>3pwc</td>
<td>11</td>
<td>&gt;131.8†</td>
<td>2.07</td>
</tr>
</tbody>
</table>

† A* without EPIC did not finish within the time limit, so we report a lower bound on the speedup: the ratio of the time limit (17 days) to the A* time with EPIC.

(2.3-fold on average), though it is not nearly as great as the speedup from faster minimization of fully assigned conformations (Fig. 3.5). Furthermore, EPIC without SAPE was often effective; however, under some circumstances it was unable to provide accurate fits (as usual, trying only the polynomial degrees described in Section 3.2.6). In systems where EPIC without SAPE was effective, it averaged insignificantly (1.1-fold) slower than EPIC with SAPE for A*. However, there were four systems that required SAPE to give accurate results (out of 27; Table 3.1), in-
Figure 3.4: Mutatable residues in the redesign of the surface of the HIV surface protein gp120 in complex with the broadly neutralizing antibody NIH45-46 (PDB code 3u7y\textsuperscript{27}). This design finished only when EPIC was used. Mutatable residues, blue backbone and pink sidechains; gp120, black backbone; NIH45-46 heavy chain, green backbone; NIH45-46 light chain, brown backbone.

Excluding the HIV gp120 complex with antibody NIH45-46 (Fig. 3.4). There were also four systems that exceeded the time limit during fitting. These limitations on EPIC without SAPE do not indicate a fundamental theoretical barrier, because in principle the Stone-Weierstrass theorem guarantees an accurate fit if the polynomial degree is sufficiently increased (see Section 3.2.6). However, they do indicate that EPIC without SAPE may sometimes require polynomial degrees that are prohibitively time-consuming for typical protein designs, and/or a higher numerical precision than the double precision efficiently supported in Java and thus used in osprey.

Experiments were also performed to compare partition function calculations with and without EPIC (Table 3.2). To obtain a provably good approximation to the partition function,\textsuperscript{48,108} many more conformations must be enumerated and minimized than for GMEC calculations. As a result, marked speedups were achieved by EPIC (Fig. 3.5C). Out of the 19 systems for which EPIC finished, only three
Figure 3.5: (A) A* times with and without EPIC. Five designs that did not finish without EPIC are shown on the right in red. (B) Proportions of each type of fit (see Section 3.2.6) required in EPIC calculations. The “quartic*” category includes both full quartic fits and quadratic fits with quartic terms added for $D_{10}$ or for $D_{100}$. Fits were all made as high-degree as needed to obtain a residual below 0.0001, as described in Section 3.2.6. Some fits have substantially lower residuals, especially quadratic fits without SAPE, since no lower fit degrees were allowed. (C) Speedups due to different EPIC methods compared to A* based on pairwise lower-bound energies; standard EPIC includes both SAPE and minimization of partial conformations. PF denotes partition function calculations; the others are GMEC calculations.
finished without EPIC (average speedup 2000-fold). The speedup in EPIC designs from minimization of partial conformations was only modest (1.4-fold).

With an A* speedup of 2-3 orders of magnitude, designs that would previously take years can be performed with EPIC in days. This will allow many designs that would otherwise be considered intractable to be completed using EPIC.

3.3.2 Quantum-mechanical energies

In addition to classical mechanics-based energy functions, we also used EPIC to fit conformational energies of the aspartame dipeptide calculated using quantum-mechanical models of electronic structure. EPIC fits for aspartame showed that quantum-mechanical energies and AMBER and EEF1 energies can be represented by polynomials of very similar degree, i.e., energy surfaces from quantum chemistry are just as polynomial-like as energy surfaces from molecular mechanics. SAPE was not found to significantly increase the accuracy of the fits, and thus were not included, though it is likely that reparameterized van der Waals and/or electrostatic terms (or other specially fit functions of the atom-pair distance) would be able to improve fit quality. This discrepancy indicates that the atom-pair energies used in SAPE are a poor approximation to the interactions between the same atom pairs predicted by quantum mechanics, and thus that energies returned by quantum-mechanical and molecular-mechanics methods are substantively different.

For Phe 2 of aspartame, the same types of polynomial fits were needed for Hartree-Fock with a STO-3G basis set, Hartree-Fock with a 6-31G** basis set, and the usual AMBER/EEF1 energy function (Fig. 3.6). These were quadratic fits for three rotamers and a quadratic fit plus quartic fits on $D_{10}$ for the fourth. MP2 with a STO-3G basis set also required quadratic fits for the first three rotamers, and required a quadratic fit plus quartic fits on $D_{100}$ for the fourth.

For Asp 1, AMBER/EEF1 required quadratic terms plus quartic terms on $D_{10}$
Figure 3.6: Intra-residue energy calculated for Phe 2 of aspartame using Hartree-Fock theory with a STO-3G basis set, and quadratic EPIC fit, as a function of the two sidechain dihedrals. The fit is very close to the energy surface, though a slight discrepancy is visible in the upper right-hand corner ($\chi_1 \approx 70^\circ$, $\chi_2 \approx 100^\circ$).

for three rotamers and on $D_{100}$ for two. Both Hartree-Fock and MP2 with a STO-3G basis set required slightly simpler fits: only quadratic for one rotamer, quadratic plus quartic terms on $D_{10}$ for three, and quadratic plus quartic terms on $D_{100}$ for one.

These results show that quantum and polarization-type effects can be represented effectively by polynomial fits, and the polynomial degrees needed are essentially the same as for AMBER/EEF1.

3.4 Conclusions

EPIC eliminates the current bottleneck in minimization-aware protein design by performing energy function calls only in a precomputation step. It thus opens several avenues for more accurate and efficient design calculations. More residues can now be mutated, and more ligands can be tested. Additional continuous conformational degrees of freedom (e.g., in the backbone) can be modeled, and minimization can be performed over a greater range for each degree of freedom when appropriate.
In this sense, EPIC helps protein design algorithms emulate the extensive continuous flexibility of molecular dynamics algorithms, while searching an exponentially large sequence space that would be intractable for molecular dynamics-based design. Furthermore, more accurate but (previously) slower energy functions can be incorporated without any asymptotic increase in computation time.

The polynomial representation of energy provided by EPIC could also allow dedicated algorithms for polynomials to be used in energy calculations. For example, since exact derivatives of polynomials are trivial to compute, EPIC is very amenable to rapid calculation of energy derivatives with respect to internal coordinates, which are used in many minimization algorithms. Note that the gradient of the EPIC polynomials may not approximate the gradient of the energy (i.e., forces) to the same degree of accuracy as the polynomials approximate the energy, because the polynomials are fit to the energies rather than the forces. Thus, differentiation of the fit polynomial may amplify noise. However, when we are numerically minimizing the polynomial approximation to the energy, we require derivatives of the polynomials themselves.

EPIC fits are tractable, accurate approximations that provide a new understanding of the energy landscape of proteins in the vicinity of ideal rotamers, and more generally in the vicinity of energy minima. This is useful because the high dimensionality of conformational space makes direct visualization difficult.

By enabling better modeling both of conformational space and of conformational energies, EPIC moves us closer to the goal of algorithms that can produce reliable predictions for our biomedically and biologically important protein and drug design problems. EPIC is thus offered to the protein design community both as an immediate speedup in designs and as an enabling technology for future improvements.
Software Availability  Our implementation of EPIC, as part of the OSPREY\textsuperscript{13,36,41,48,49} open-source protein design software package, is available for free download at http://www.cs.duke.edu/donaldlab/osprey.php.
**COMETS** (Constrained Optimization of Multistate Energies by Tree Search): A provable and efficient protein design algorithm to optimize binding affinity and specificity with respect to sequence

*One day, while thinking about how the voxel paradigm could be used systematically for drug design, I realized that we did not have an algorithm to provably handle the kinds of specificity requirements often encountered in this field. I realized most of these requirements can be phrased in terms of the linear multistate energies (LMEs) described in this chapter. So I started thinking about how $A^*$ could be adapted for constrained optimization of LMEs. The chief difficulty is that optimizing an LME involves a minimum (over conformation) within a minimum (over sequence), which would seem to require having separate $A^*$ trees for every (sequence, state) pair, if we want to maintain provable guarantees on conformational optimization for such pairs. A few days later, while running, I realized how we can build such trees as needed without being too exhaustive—we needed One Tree to Rule Them All (the sequence-based $A^*$ tree described below). I “forged” this One Tree in OSPREY, and we published*
the algorithm at the RECOMB conference and in the Journal of Computational Biology. This chapter is adapted from the latter version:


Summary. Practical protein design problems require designing sequences with a combination of affinity, stability, and specificity requirements. Multistate protein design algorithms model multiple structural or binding “states” of a protein to address these requirements. COMETS provides a new level of versatile, efficient, and provable multistate design. It provably returns the minimum with respect to sequence of any desired linear combination of the energies of multiple protein states, subject to constraints on other linear combinations. Thus, it can target nearly any combination of affinity (to one or multiple ligands), specificity, and stability (for multiple states if needed). Empirical calculations on 52 protein design problems showed COMETS is far more efficient than the previous state of the art for provable multistate design (exhaustive search over sequences). COMETS can handle a very wide range of protein flexibility and can enumerate a gap-free list of the best constraint-satisfying sequences in order of objective function value.

4.1 Introduction

Protein design requires the prediction and selection of protein sequences with desired properties, generally some combination of structure stability, binding to desired ligands, and lack of binding to undesired ligands. The gold standard for protein design is natural evolution, in which protein mutations confer fitness advantages only if several desired properties are all present: mutants must be sufficiently stable, effective at
binding or catalysis, and selective for their fitness-conferring function.\textsuperscript{36} Researchers have tried to emulate this process by directed evolution experiments.\textsuperscript{7} But methods to optimize these properties computationally\textsuperscript{28} allow enormous sequence spaces to be searched without enormous resource expenditures, and thus greatly expand the space of possible designs. Such searches require algorithms that do not analyze each candidate sequence separately: large sequence spaces are too expensive to analyze one by one. Computational protein designers have used three different strategies to achieve the desired properties with their new sequences: energy minimization of a single desired protein or complex structure (“single-state design”); heuristic minimization of some function combining multiple desired properties (“traditional multistate design methods”); and analysis of one sequence at a time in detail (“single-sequence analysis”).

Single-state design is the most developed class of dedicated protein design algorithms. It is commonly used to improve fold stability by selecting mutants that minimize the protein’s total energy,\textsuperscript{25,28,40,50,93} and to increase binding affinity by selecting mutants that minimize the energy of a complex\textsuperscript{34,48,87}. Some of these methods are provable: given a sequence space to search, a model of the protein’s conformational space, and an energy function, they are guaranteed to return the lowest-energy sequence and conformation (the global minimum-energy conformation, or GMEC). The dead-end elimination (DEE)\textsuperscript{25} and A*\textsuperscript{99} algorithms have this guarantee. In their original form, they assume a discrete conformational space, but they have been extended to include both continuous sidechain\textsuperscript{40,48} and backbone\textsuperscript{44,68} flexibility. Provable single-state methods can also enumerate either a gap-free list of the lowest-energy sequences and conformations,\textsuperscript{40,68,99} or of the sequences with the lowest-energy optimal conformations.\textsuperscript{138} Other single-state methods are not provable, most prominently Metropolis Monte Carlo-based methods,\textsuperscript{93,102} but are popular for reasons of computational speed. All these methods use some simplified model of
protein conformational flexibility. A popular but highly approximate model is to allow the conformation of each amino acid to be selected from a discrete set, referred to as *rotamers*. This model can be made substantially more accurate by allowing small, continuous conformational adjustments around the rotameric conformations, which can be incorporated while maintaining provable accuracy.40,48,68

Single-state design can be thought of as the stabilization of a desired “state” of a protein—essentially, its fold, overall conformation, and ligand-binding mode. This paradigm can be extended to include multiple states, possibly with different ligands, in order to specify multiple desired properties for the designed sequence. This strategy is known as *multistate protein design*.22 DEE has been extended to multistate design in the type-dependent DEE algorithm.173 This algorithm prunes rotamers that are guaranteed not to be part of the optimal conformation of a state of the protein. It offers a significant advantage in efficiency, but does not reduce the number of sequences that must be considered in multistate design, because it only eliminates rotamers by comparison to more favorable rotamers of the same amino-acid type. On the other hand, non-provable methods have also been developed to try to optimize objective functions based on the energies of multiple states, without considering each sequence separately. Genetic algorithms have been used to optimize differences in energy between states104 as well as other objective functions,101 and belief propagation has been used to optimize sums of energies of different states, in order to design a binding partner appropriate for multiple ligands.37–39 Type-dependent DEE can also be combined with such techniques, to reduce the conformational space that is searched heuristically.38,173 In addition, some design systems can be described fairly accurately by an energy function whose terms depend only on the amino acid types of one or a few residues;58 the CLASSY method57 derives such energy functions by least-squares fitting, and then uses them to perform efficient multistate designs that bypass conformational search entirely and use integer linear programming algorithms

122
to find the optimum of the sequence-based energy function with a provable guarantee of optimality. However, for design systems that are not well described by a cluster expansion (i.e., that exhibit significant higher-order interactions between conformational changes at different residues), previous multistate design algorithms cannot provide any guarantees about the optimality of their designed sequences without an exhaustive search over sequence space.

Methods that consider each candidate sequence explicitly are another important and highly versatile category of computational protein design methods. However, the computational costs can be very high—linear in the number of sequences, and thus exponential in the number of simultaneously mutable positions. Molecular dynamics can be applied for single-sequence analysis in protein design, using simulations over time to investigate the properties of a candidate sequence. Molecular dynamics readily models all types of protein flexibility with many different energy functions, including effects like solvent polarization or explicit solvent. It also allows the user to account for entropic contributions to binding energies. More recent algorithms account for entropy without the steep costs of simulation over time. The $K^*$ algorithm in OSPREY predicts the binding of a mutant protein sequence to a ligand by computing an ensemble of low-energy protein states to provably approximate the binding constant $K_a$ within a desired relative error for the user-specified flexibility model and energy function. Though it provides a vast speedup relative to exhaustive search over all conformations at each sequence, it does require explicit consideration of each sequence sufficient to bound the energies in its ensemble. $K^*$ in OSPREY has yielded several multistate protein designs that were successful experimentally. The calculations have involved both comparisons of the bound and unbound states of a single complex and of multiple complexes, and the OSPREY-designed proteins have performed well in vitro and in vivo as well as in non-human primates.
We now present an algorithm distinct from these three traditional strategies that combines advantages from all three: COMETS. Like other multistate methods, it optimizes an energy measure that considers multiple states: for example, it can directly optimize the binding energy (the difference in energy between the bound and unbound states), or the difference in binding energy between two different ligands. Like single-sequence analysis, it allows consideration of a wide variety of stability, affinity, and specificity requirements during sequence selection. This is facilitated by its accommodation of optimization constraints: for example, it can optimize binding to one ligand while constraining binding energy for other ligands. It provably returns the best sequence for its specified optimization problem, without performing an exhaustive search over the possible sequences. Some previous methods can do this for single-state design problems, but before COMETS it was impossible for most multistate problems. As a result, COMETS provides a vast performance improvement over the previous state-of-the-art for provable multistate design, which is exhaustive search over sequence space.

By presenting COMETS, this paper makes the following contributions:


2. An algorithm to solve problems in this framework that provably obtains the same results as exhaustive search over sequences but is combinatorially faster than this exhaustive search, as shown by empirical measurements on 52 protein design problems.

3. Support for continuous sidechain and backbone flexibility in COMETS.

4. The ability to enumerate as many constraint-satisfying sequences as desired, in a gap-free list in ascending order of the desired objective function.
Figure 4.1: Flexible and mutable residues in a design for specificity. The apoptotic regulator CED4 forms two different dimers, one to block apoptosis (left; PDB id 2a5y\textsuperscript{172}) and one to induce it (right; PDB id 3lqr\textsuperscript{133}). We want to design for specificity (to block apoptosis), so we allow mutations to some residues in the binding site (blue). To accurately model the conformational changes induced by the mutations, we also model as flexible the residues on the opposite side of each interface that interact with the mutable residues (orange, pink). Analysis of this calculation and others is described in Section 4.3.

5. An implementation of COMETS in our laboratory’s open-source OSPREY protein-design software package,\textsuperscript{13,36,48} available for download at our website\textsuperscript{49} as free software.

4.2 Methods

4.2.1 Problem formulation

Let us consider a protein design problem where we wish to consider mutating \( n \) residues. The output of our calculation will be a sequence \( s \): an ordered list of \( n \) amino acid types. We have a set \( A \) of states. Each state is a protein structure containing our \( n \) mutable residues, along with a (possibly continuous) conformation space for each sequence assignment, which we call the flexibility model for the state. We consider functions of the form

\[
f(s) = c_0 + \sum_{a \in A} c_a E_a(s) \tag{4.1}
\]
where the $c_a$ are real coefficients. We call these functions \textit{linear multistate energies} (LMEs). \textsc{comets} is an algorithm to minimize any LME $f(s)$ with respect to sequence $s$, under constraints of the form $c_i(s) < 0$, where each $c_i$ is also an LME. LMEs are suitable for representing stability, affinity, and selectivity requirements in protein design. For example, to optimize a binding energy, we set $A = \{b, u\}$ to consist of the bound state $b$ and the unbound state $u$, and optimize $f(s) = E_b - E_u$. That is, we set $c_b = 1$, $c_u = -1$ and $c_0 = 0$ for our objective function. A highly simplified, “toy” example of this setup is described in Section 4.2.5.

The choice of objective function and constraints defines the physical problem we wish to solve. We require a computational model of proteins to convert this into a computational problem. To model protein flexibility, we use the very general model of the DEEPer algorithm\textsuperscript{68} in \textsc{osprey}. The protein in each state is allowed to have any number of degrees of freedom, which can be either continuous or discrete, and which fully specify both the sequence and conformation of the protein. Each residue in each state has a set of “residue conformations” (RCs). An RC is a portion of conformational space defined by bounds on every conformational degree of freedom available to the residue. These bounds must be tight enough that once a residue conformation is assigned to every residue, the energy minimum over this limited conformational space can be found by local minimization. Thus, RCs define a partitioning of conformational space that allows local minimization to be used as a subroutine in our global search. A residue conformation is associated with a specific amino acid type. This framework is suitable for accommodating both continuous sidechain and backbone flexibility, but it reduces to the model of continuous sidechain flexibility of\textsuperscript{40,48} if only sidechain dihedrals are used as continuous degrees of freedom. If each sidechain dihedral is confined to a single value within each residue conformation, then this special case is just the commonly used rigid-rotamer approximation.\textsuperscript{25,99} In both of these special cases, each residue conformation represents a single sidechain
The model of flexibility may differ between states; in fact, different residues may be made flexible. For example, in a calculation with a bound and an unbound state of a protein, the ligand will have flexibility in the bound state, but will be absent from the unbound state (Fig. 4.1). But all states have the same set of mutable residues, and the same set of allowed amino-acid types at each mutable residue. This way, COMETS outputs a sequence applicable to all states.

To model energy, we must have an “energy function” that estimates the energy of a given sequence and conformation. Our implementation of COMETS uses a pairwise additive energy function, i.e., a sum of energy terms that each depend on the conformations of at most two residues. This property is only used in the computation of lower bounds for LMEs over subsets of the sequence space and state conformational spaces (Sections 4.2.2 and 4.2.3), so a non-pairwise energy function that admits such lower-bound computations would also be compatible with COMETS. COMETS will return optimal results for the given model of flexibility and energy function.

4.2.2 A* over sequences

COMETS uses the A* search algorithm to search sequence space. In most previous applications of A* to protein design, nodes of the tree correspond to partially defined conformations. Each partially defined conformation is specified by RC assignments for one or more residues. Thus, each node corresponds to the conformational space made up of all conformations consistent with the partial definition. A node’s score is a lower bound on all the conformational energies in this space. COMETS is similar, but nodes correspond to partially defined sequences and thus to a sequence space. A node’s score is a lower bound on the objective function for all sequences in the node’s sequence space (Fig. 4.2).

In A*, we repeatedly process the lowest-scoring node in the tree. Processing a
Figure 4.2: Expansion steps during node processing generate nodes with partially (e.g., VXXXX or AXXXX) and then fully (e.g., VFYWI) defined sequences. Once a node has a fully defined sequence, conformational trees are built for it for all states. Then conformational tree expansions lead to fully processed nodes. X, unassigned amino acid or RC; V, Val; A, Ala; F, Phe; Y, Tyr; W, Trp; I, Ile.

node means either splitting it into several nodes that partition its sequence space, or computing a higher score (i.e., tighter bound) for it (that is still a valid lower bound). Score computation may involve conformational search (Fig. 4.2), and some nodes will be processed until their sequence is fully defined and the optimal conformation for each state is fully determined. These nodes are termed fully processed, and their objective function and constraint LMEs can be evaluated exactly. When the lowest-scoring node is fully processed, we can return its sequence as optimal, because its objective function value (at optimal conformations for each state) is better than any sequence in any of the sequence spaces of the other nodes in the tree. This is because the other nodes’ scores are lower bounds on their optimal objective function values.

Types of nodes.

We will store two types of nodes in our tree (Fig. 4.2). Examples of each type of node in the toy example are given in Section 4.2.5.

The first type has a sequence that is not fully defined: not all mutable residues
have an assigned amino-acid type. At these nodes, we store information on which RCs are pruned at each residue in each state (for the assigned amino-acid types if assigned; for all amino acid types if not assigned). The pruned RCs are those that cannot be part of the optimal conformation for that state for any sequence in the sequence space of the node. We store pruned pairs of RCs as well as individual pruned RCs.

The second type of node has a fully defined sequence: an amino-acid type assigned for each mutable residue. At each such node, for each state, we store an A* tree expanding the conformational space for that sequence. These trees are identical to those used in DEEPer in osprey: their nodes each represent a subset of conformational space, defined by RC assignments to some of the residues, which restrict the values of the proteins’ degrees of freedom to the bounds associated with the assigned RCs. The score of each node is a lower bound on the energy of all conformations in its allowed conformational space.

**Node-processing operations.**

For either type of node, node processing consists of two steps: an “expansion” step and a “bounding” step (Fig. 4.3). Every time we extract a node from the priority queue, meaning it is the lowest score in the tree, we choose the appropriate processing
operation and perform it (Fig. 4.3).

Expansion step. For a node without a fully defined sequence, the expansion step splits the node \( n \) into several nodes whose sequence spaces partition the sequence space of \( n \). If the first mutable residue without an amino-acid type assigned in \( n \) is residue \( r \), then this partition can be performed by creating a node for each amino-acid type \( a \) allowed at \( r \). These child nodes will each have a sequence space identical to that of \( n \), except with the amino-acid \( a \) assigned to residue \( r \). For a node \( n \) with fully defined sequence, we split the lowest-scoring node in one of \( n \)'s conformational trees: each child node has a different RC assignment for a residue whose RC is not assigned at the parent node. This is the same type of split used in DEEPer,\(^{68}\) and essentially as in previous protein design applications of A*.

Bounding step. In the bounding step, a lower bound is computed for the objective function and for each of the constraint LMEs. If the lower bound for any of the constraint LMEs \( c_i \) is greater than 0, then we know all sequences at the node violate that constraint, and we eliminate the node. Otherwise, the node score is set to be the lower bound on the objective function. Details of the method for computing lower bounds are provided in Section 4.2.3.

For nodes without fully-defined sequences, we update the list of pruned RCs for the child node before computing bounds. Pruning is performed by type-dependent DEE\(^{173}\)—in our implementation, the various pruning algorithms available in OSPREY\(^{41,47,48}\) are used.

4.2.3 Computing lower bounds for LMEs

Previous A*-based protein design algorithms include methods to compute a lower bound on the energy of a single protein state over a sequence space.\(^{48,68,73}\) These
methods can be modified to provide a lower bound on an LME over a sequence space, with complexity as follows:

**Theorem 4.2.1.** For any sequence space $S$ defined by specifying the allowed set of amino acid types $S(i)$ at each mutable residue $i$, the lower bound on the LME Eq. (4.1) can be computed in time $O(n^2r^2s)$, where $n$ is the number of flexible or mutable residues in the system, $s$ is the number of states, and $r$ is the maximum number of RCs available at a given residue.

We use a different procedure to compute lower bounds for LMEs (linear multistate energies) depending on whether the sequence is fully defined. If it is, a simpler algorithm that often yields a tighter bound is used. If not, we have a generic algorithm, whose running time is bounded as described in Theorem 4.2.1 (proven below).

*Lower-bound algorithm for fully defined nodes*

If the sequence is fully defined, then we can bound the LME

$$c_0 + \sum_{a \in A} c_a E_a(s)$$

using upper bounds $u_a(s)$ and lower bounds $l_a(s)$ on the optimal conformational energy for our sequence $s$ in each state $a \in A$. Because we have $u_a(s) \geq E_a(s)$ and $l_a(s) \leq E_a(s)$, we have

$$c_0 + \sum_{a \in A, c_a < 0} c_a u_a(s) + \sum_{a \in A, c_a > 0} c_a l_a(s)$$

as a lower bound on Eq. (4.1). The single-state lower bounds $l_a(s)$ are computed as described in previous work on single-state design.\textsuperscript{48,68,73} These methods yield a lower bound that converges to $E_a(s)$ as the conformation trees are fully expanded. Upper bounds $u_a(s)$ are straightforward because any conformation of a given state is an upper bound on the optimal conformational energy for that state. Each time
we expand a node in the conformation tree for state $a$ and sequence $s$, we compute an upper bound $u_n$ on the minimum energy of that node’s conformation space by a quick heuristic procedure based on the FASTER algorithm.\textsuperscript{26} If $u_n$ is lower than our current estimate of $u_a(s)$, then we set $u_a(s) = u_n$. As the conformation trees are fully expanded, we must eventually encounter a node in each state’s tree whose conformational space is only the optimal conformation (or in the continuous case, only the RC assignment that yields the optimal conformation upon local minimization). At this point, the upper bound becomes tight: $u_a(s) = E_a(s)$. So we have a method to provide lower bounds for LMEs that converges to the exact value of the LME when a node is fully processed. This is what we need both for constraint enforcement and for minimization of the objective function.

Lower-bound algorithm for other nodes

If the sequence is not fully defined, then we use an alternate algorithm, which is a direct adaptation of the lower-bounding method used in single-state A* calculations.\textsuperscript{99} Nodes without the sequence fully defined cannot be fully processed, so we do not need this algorithm to converge to the exact value of the LME. However, we still prefer a reasonably tight bound in this case for efficiency purposes (to avoid having to generate too many nodes with fully defined sequences).

Let us first consider the rigid case. For the purposes of this algorithm, we will assume a pairwise energy function: an energy function that is a sum of terms dependent on at most two residues’ conformations, and can thus be expressed in the form

$$\sum_{i} E(i_r) + \sum_{j<i} E(i_r, j_r)$$  \hspace{1cm} (4.4)

where $i_r$ denotes the RC for our current conformation $r$ at residue $i$, $E(\cdot, \cdot)$ is a pairwise interaction energy between RCs, and the one-body terms $E(\cdot)$ measure the
internal energy of each RC plus its interactions with non-flexible parts of the system. Let us use $E_a$ to denote a pairwise or 1-body energy for residues in state $a$. Now, to derive a lower bound, we start with the true minimum value of the LME over our sequence space $S$, which is (using Eq. 4.1)

$$c_0 + \min_{s \in S} \sum_{a \in A} c_a E_a(s).$$

(4.5)

Letting $R_a(s)$ be the conformation space available to a sequence $s$ in state $a$, and plugging in Eq. (4.4), Eq. (4.5) can be expanded to

$$c_0 + \min_{s \in S} \sum_{a \in A} c_a \min_{r \in R_a(s)} \left( \sum_i E_a(i_r) + \sum_{j < i} E_a(i_r, j_r) \right).$$

(4.6)

To obtain a tractable bound, we relax Eq. (4.6) by moving the minima inside the sums. This operation can only decrease the value of the expression (this is a general property of interchanging summation and minimization), resulting in a valid lower bound. Let $S(i)$ be the set of amino acid types available at residue $i$, let $R_a(b, i)$ be the set of unpruned RCs available to amino acid type $b$ at residue $i$ in state $a$, and let us use the notation $i_r$ for an RC at residue $i$, following previous work.$^{25, 40, 48, 68}$ The relaxation yields

$$c_0 + \sum_i \min_{b \in S(i)} \min_{a \in A} \min_{i_r \in R_a(b, i)} \left( c_a E_a(i_r) + \sum_{j < i} \min_{b' \in S(j)} \min_{j_r \in R_a(b', j)} c_a E_a(i_r, j_r) \right).$$

(4.7)

So Eq. (4.7) is a lower bound on Eq. (4.5), i.e., for the LME we are bounding.

Each state may have flexible residues besides the mutable residues; for the purposes of Eq. (4.7), these are all considered separately. Hence, the sum over $i$ runs over all mutable residues, which are shared between all states, and also over other flexible residues, which may differ between states and are treated as different residues in the sum (with $c_a$ being nonzero only for the state to which the residue belongs).

This bound, Eq. (4.7), can also be used with continuous flexibility, with only a change in definitions for the pairwise and 1-body energies $E_a$. Each of these energies
is associated with an RC or a pair of RCs, so when we introduce continuous flexibility, \( E_a \) is no longer associated with a single conformation but with a set of conformations defined by bounds on the continuous degrees of freedom of the residues whose energy is being measured. Within these bounds there is a set of “ideal values” for all the degrees of freedom. So, for the terms \( c_a E_a \) in Eq. (4.7), we let \( E_a \) denote the pairwise or 1-body energy at the ideal values of all continuous degrees of freedom if \( c_a < 0 \), but we let \( E_a \) denote the minimum value of the 1- or 2-body energy over the allowed ranges for all continuous degrees of freedom if \( c_a > 0 \). Using these definitions, Eq. (4.6) is a valid lower bound for the LME Eq. (4.5) over our node’s sequence space, and so Eq. (4.7) is a valid lower bound, using the same reasoning as in the rigid case.

Regardless of which definition is used, the pairwise and 1-body energies \( E_a \) can be precomputed for an RC or a pair of RCs, before COMETS begins. This precomputation is described in previous work on single-state design.\(^{40,48,68}\) A tighter bound can be achieved by using a larger rotamer library for the terms with negative coefficients.

**Complexity analysis and proof of Theorem 4.2.1**

Because the algorithm we have just described can be used for any sequence tree node, it provides a constructive proof of Theorem 4.2.1.

**Proof.** The bound can be computed using Eq. (4.7). For each state \( a \) and each RC \( i_r \) at each residue \( i \), the term \( c_a E_a(i_r) \) can be computed in constant time, because \( E_a(i_r) \) is simply looked up from the precomputed single-residue energies. Then for each residue \( j \), the term \( \min_{i_s \in R_a(i_r,j)} c_a E_a(i_r,j_s) \) can be computed in \( O(r) \) time, because the minimum is over \( O(r) \) precomputed pairwise interaction energies that can be looked up and multiplied by \( c_a \) in constant time. So the \( \sum_{j<i} \) sum has \( O(n) \) terms that can each be computed in \( O(r) \) time, meaning the sum can be computed
in $O(nr)$ time. Thus, the quantity in large parentheses can be computed in $O(nr)$ time. For each residue $i$, this quantity needs to be computed $O(rs)$ times. So the cost for each term of the $\sum_i$ is $O(nr^2s)$. Therefore, the cost to evaluate the entire $\sum_i$, and thus to compute our lower bound, is $O(n^2r^2s)$.

4.2.4 Starting and finishing the calculation

Hence, to perform COMETS, we create a priority queue of A* tree nodes and initialize it with a node representing the entire sequence space we are searching. We then repeatedly extract the lowest-scoring node from the priority queue and process it with the appropriate node-processing operation.

Each operation will define either the sequence or the conformation in one of the states at a residue where it was previously not defined, so in a finite number of steps, we will achieve a node whose sequence and optimal state conformations are fully defined, i.e., a fully processed node. If our lower-bounding techniques are adequate, very few sequences will need to be fully processed, so this sequence A* tree will return the optimal sequence with great efficiency compared to exhaustive search over sequences.

Running COMETS until $n$ sequences have been returned will yield the $n$ sequences that have the lowest objective function values among all sequences satisfying the constraints.

4.2.5 Toy example

Let us introduce a highly simplified, “toy” example to explain the algorithm. Say we are designing a peptide inhibitor for a protein, “targetin,” involved in some disease. We want our peptide to not bind a related protein, “offtargetin,” because this binding would cause toxicity. However, we do have a peptide (say, a natural product) that binds both targetin and offtargetin, and we have crystal structures of
this peptide with both targetin (structure T) and offtargetin (structure O). Thus, we set up a multistate design as follows. There will be four states: peptide bound to targetin (structure T), unbound peptide in targetin-binding conformation (peptide from structure T), peptide bound to offtargetin (structure O), and unbound peptide in offtargetin-binding conformation (peptide from structure O). These states will be called T-bound, T-unbound, O-bound, and O-unbound respectively. We will optimize the binding energy for the peptide-targetin interaction, which is the difference between the T-bound and T-unbound state energies. We will constrain binding energy for the peptide-offtargetin interaction to be worse than a threshold $E_u$: say, 10 kcal/mol worse than the wild-type binding energy. We will also constrain the unbound state to be stable, by demanding that the average of the T-unbound and O-unbound state energies be at most 10 kcal/mol worse than wild type. All of these constraints, and the objective function, are simple LMEs.

For simplicity, our toy peptide will only have one mutable residue position (residue 1), to which we may assign either the amino acid type alanine (1 rotamer) or valine (3 rotamers). We will also model only one residue as flexible in each of our proteins: Phe 75 of targetin (4 rotamers) and Leu 75 of offtargetin (5 rotamers). Fig. 4.4 illustrates the conformational search in COMETS on this problem subject to these modeling assumptions. We also present empirical results for much larger, realistic designs with similar objective function and constraint setups in Section 3 of the main text.

**Types of nodes.** In this toy example, we create only one node with a partially defined sequence. The sequence space for it is $\{A,V\}$, since these are the two sequences available to the peptide. Two nodes with fully defined sequence are created: one for each of the two sequences A and V. For the V node, the conformational tree for each state will have the same structure as a single-state A* tree for a single-sequence
conformational search in which residue 1 is a valine (Fig. 4.4).

Expansion of nodes. The expansion step for the starting node (sequence space \{A,V\}) splits the sequence space to create a node for each amino-acid type at residue 1; thus the two nodes will have sequences spaces \{A\} and \{V\}. For the node \{V\}, because the sequence space is fully defined, the expansion step expands nodes in the conformational trees, in the same way that nodes are expanded in single-state A* calculations\(^{48,68,99}\) (Fig. 4.4).
4.2.6 Computational experiments

Protein design calculations were performed in order to measure the efficiency of COMETS and its ability to design proteins with properties undesignable by single-state methods (Section 4.3). In this section, we provide details of these test cases.

Systems of four types were used: designs for specificity on a protein that can form two or more different complexes; optimization of the binding energy for a single complex; stabilization of a single protein robust to choice of force field; and stabilization of the reduced form of angiotensinogen relative to the oxidized form or vice versa.

Systems of the first type involve a protein that can form two different complexes whose structures are known. This type can be viewed as a realistic version of the toy example above (Section 4.2.5). Proteins were redesigned in four different ways: either to prefer one complex over the other or to stabilize both, and with either rigid or continuous flexibility modeling. For designs to favor one complex, the objective function was the binding energy for that complex (the energy of the bound state minus that of the unbound state), and for designs to favor both complexes, the objective function was the sum of the complexes’ binding energies. In each case, constraints were placed on all binding energies and on unbound state energies. Desired binding energies were constrained to be better than the wild-type binding energy for the same complex, while undesired binding energies were constrained to be worse than wild-type. To ensure protein stability, the average of the unbound state energies was constrained to be no more than 10 kcal/mol worse than the wild-type average. The pairs of complex structures for these designs were obtained from random entries in the INstruct database. The pairs’ PDB ids were 2a40/2a41, 2a5y/3lqr (the angiotensinogen run in Fig. 1), 2gzd/2gzh, 2ngr/1grn, 3egd/3egx, 3efo/3eg9, 3k75/3lqc, 3ktp/3ktr, 3nlf/3n1q, and 3n1g/3n1m. These complexes were drawn from both protein-protein
and protein-peptide interactions. In addition to these two-complex systems, a set of larger designs was performed on a set of ten complexes of bovine trypsin crystallized with ten different variants of bovine pancreatic trypsin inhibitor (BPTI). Designs were performed either to maintain binding by the wild-type BPTI while blocking binding to the nine mutant variants, or to enhance binding to all variants. Protein design to identify enzyme variants resistant to certain inhibitors has been used in the study of bacterial resistance to antibiotics, and has applications to the selection of inhibitors more robust to resistance.

The second type of system was optimization of the binding energy for a single complex, with constraints handled as in the specificity calculations. Treating affinity calculation as a multistate design allows accounting for variations in the unbound state energy, both to ensure the stability of the unbound state and to compute its effect on binding affinity. Structures for these designs were also drawn from the INstruct database: 1b6c, 1nez, 1stf, 1vyh, 2b4s, 2h0d, and 2nqa.

The third type of system was stabilization of a single protein, but in a way meant to produce stabilizing mutations that are robust to the choice of force field. This strategy may be useful for obtaining mutations that are more likely to work in practice. For these calculations, the protein’s energy was optimized using the AMBER force field, but the energies using both the AMBER and CHARMM force fields were constrained to improve relative to the wild type. The proteins used here were drawn from the test set of protein energy optimizations in Gainza, Roberts, and Donald. The structures’ PDB ids were 2o9s, 2qsk, 2ril, 3a38, and 3g36.

Finally, the fourth type of system was the stabilization of the reduced form of angiotensinogen relative to the oxidized form (Fig. 4.7) or vice versa. This was treated as an unconstrained optimization of the difference between the states, which have been crystallized (PDB ids 2wxy and 2wxx respectively). This type of setup arises in experimental studies of protein function, when one wishes to design a mutant.
to “freeze” a protein in one of two or more conformational states in order to study the functional role of the chosen state. For example, the N600K mutation in the motor protein Ncd\textsuperscript{150} induces Ncd to adopt a conformational state typically only induced by ATP binding; studies of this mutant helped to establish the temporal and mechanistic relationship between the nucleotide binding and force generation of this motor.\textsuperscript{71}

4.3 Results

Protein design calculations were performed in order to measure the efficiency of COMETS and its ability to design proteins with properties undes ignable by single-state methods. Systems of four types were used: designs for specificity on a protein that can form two or more different complexes; optimization of the binding energy for a single complex; stabilization of a single protein robust to choice of force field; and stabilization of the reduced form of angiotensinogen relative to the oxidized form or vice versa. Details of these test cases are in Section 4.2.6.

4.3.1 Measurement of efficiency

COMETS was run on 52 protein design test cases to measure its efficiency advantages across a range of different objective functions and constraints. The test cases used 44 protein structures, and 25 modeled flexibility using rigid rotamers while the other 27 used continuous flexibility.

Exhaustive search, the only other provable algorithm for multistate design, must calculate the GMEC for each sequence in each state. For a $s$-state design space with $N$ sequences, this means that $N$ sequences must be considered explicitly and $sN$ state GMECs must be calculated—a formidable proposition, since $N$ grows exponentially with the number of mutable residues and each state GMEC calculation is NP-hard.\textsuperscript{131} To measure the ability of COMETS to avoid these calculations, the number $g$ of state
Figure 4.5: Number $g$ of state GMECs calculated in COMETS runs with (A) rigid or (B) continuous flexibility, compared to the number $sN$ of state GMECs in the entire design space ($sN$ is the number of sequences in the design space times the number of states). Results are shown both for calculation of the best sequence and for enumeration of the best five, when possible under the design constraints. Exhaustive search would have to calculate all state GMECs (green curve).

GMECs calculated by each run of COMETS was measured and compared to $sN$. Also, COMETS provably need not even consider each sequence explicitly, even briefly. To determine if this reduced consideration of sequences provides a significant advantage in efficiency, the number $m$ of sequence tree nodes created in each COMETS run was measured and compared to $N$. Hence, $m$ is the number of partial sequences explicitly considered in a COMETS run.

Many provable algorithms, including A*\textsuperscript{99} and integer linear programming,\textsuperscript{89} and non-provable methods like Monte Carlo\textsuperscript{93} can minimize an LME using (a) an exhaustive search over sequences without (b) also exhaustively searching over conformations. So even without COMETS there is no need for an exhaustive search over conformational space. However, all previous provable methods for typical (non-sequence-based) energy functions must still compute the GMEC of each state for every sequence when performing multistate design, because they are intended to calculate the minimum of an energy function (with respect to sequence and conformation).
Figure 4.6: Speedup due to reduced explicit consideration of sequences in COMETS, compared to exhaustive search (green line), for designs with rigid rotamers. $m$: number of sequence tree nodes created in COMETS. $N$: number of sequences in the design space. Magnifying this speedup, COMETS handles sequences that it considers explicitly very efficiently (Fig. 4.5).

In contrast, COMETS calculates the *constrained minimum* (over all sequences) of a linear combination of minima (over all conformations) of energy functions. Hence, in this paper, we measure the ability of COMETS to *avoid* computing GMECs for most of the sequences, and sometimes even to avoid any explicit consideration of most of the remaining sequences. These are the main novel abilities of COMETS.

*Reduction in number of state GMECs calculated.*

COMETS calculates only a very small portion of state GMECs (Fig. 4.5)—often only the state GMECs for the sequences being returned as optimal. To calculate the best sequence in rigid designs, the average run needed to calculate only 0.05% of the state GMECs in the design space. This portion increased to 0.1% for enumeration of the best five sequences. For continuous designs,40,68 2% of the state GMECs were calculated for runs finding only the best sequence, and 4% were calculated for runs enumerating the best five sequences.
Reduction in number of sequences considered explicitly.

Reduced explicit consideration of sequences was found to provide a significant combinatorial speedup in COMETS runs without continuous flexibility. For calculation of the best sequence in these rigid designs, the median $m/N$ was 0.02, and many runs with larger design spaces generated significantly fewer sequence tree nodes relative to the design space size (Fig. 4.6)—the largest sequence space to return a constraint-satisfying sequence had 47 million sequences with $m/N = 2 \times 10^{-6}$ (i.e., a $5 \times 10^5$-fold speedup). The median increased to 0.03 for enumeration of the best five sequences. For continuous designs, the median $m/N$ values were 0.63 for the best sequence and 0.69 for the best five.

Provably finding unsatisfiable constraints.

The statistics above exclude runs for which no sequences can satisfy the constraints. COMETS can provably verify when no satisfying sequences exist, and did so for 8 of the 27 continuous runs and 5 of the 25 rigid runs.

4.3.2 Differences in sequences returned by multistate designs and single-state proxies

Single-state design is often used as a proxy or a “first step” in multistate design. To test whether this approximation yields sequences similar to the optimal ones from multistate design, sequence divergences were calculated between optimal sequences from multistate design and optimal sequences from corresponding proxy single-state designs.

Our results indicate that single-state approaches are likely to yield sequences far from the optimal one. For specificity design problems favoring a complex P:A over a complex P:B, mutable-residue sequence divergence between the single-state optimal sequence for complex P:A and the multistate optimal sequence was 33% (averaged
over 13 designs). Similarly, for multispecificity designs (optimizing the sum of binding energies for complexes P:A and P:B), the best sequence averaged 36% sequence divergence from the single-state optimum for complex P:A (10 designs). These divergences are nearly as high as the 39% (8 design pairs) average sequence divergence between comparable specificity and multispecificity designs—that is, between a protein optimally designed to bind A while not binding B, and a protein optimally designed to bind both A and B. So the difference is quite functionally significant.

Further details on the test cases are provided in Section 4.2.6 and Tables 4.1 and 4.2.

These results show that explicit, provable multistate design provides significant advantages in the calculation of optimal sequences for a wide range of problems, and that COMETS provides an efficient way to perform such designs. The number of sequences and of state GMECs considered could likely be reduced substantially further using improved energy bounds. Thus, COMETS liberates provable multistate protein design from the efficiency barrier imposed by exhaustive search.
Figure 4.7: For the design run optimizing the difference in energy between the reduced and oxidized states of angiotensinogen (PDB ids 2wxy and 2wxx, respectively), the single mutation Y12I (blue to green) was found to fit well into the reduced state, but to cause steric clashes (pink) in the oxidized state. The selection of this single mutant as optimal required explicit multistate design to destabilize the undesired oxidized state, while maintaining the stability of the reduced state: optimization of either the reduced or oxidized state alone yielded aromatic residues at position 12.

Table 4.1: Protein design test cases with continuous flexibility, as described in Section 3 of the main text. Type is “aff” for designs for affinity, “stab” for designs for stability robust to force field choice, “spec” for designs to be specific to one complex, “multi” for designs to be multispecific to more than one, and “red” or “ox” for designs to favor either a reduced or oxidized state (each type is described further in Section 4.2.6). As described in Section 3 of the main text, \( s \) is the number of states, \( N \) is the number of sequences in the designed states, \( m \) is the number of sequence tree nodes created, and \( g \) is the number of state GMECs computed. Subscripts 1 and 5 denote calculation of the best sequence or enumeration of the best 5 sequences respectively. \( N_e \) denotes the number of sequences enumerated (all runs were set to enumerate the five best sequences, but \( N_e < 5 \) if less than five sequences in the design space satisfy the constraints). Table continues on next page. *BPTI set: 3btd, 3bte, 3btf, 3btg, 3bth, 3btk, 3btm, 3btq, 3btt, 3btw

<table>
<thead>
<tr>
<th>Protein redesigned</th>
<th>Mutability residues</th>
<th>PDB id(s)</th>
<th>Type</th>
<th>( s )</th>
<th>( N )</th>
<th>( N_e )</th>
<th>( m_1 )</th>
<th>( m_5 )</th>
<th>( g_1 )</th>
<th>( g_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta-2-microglobulin</td>
<td>52, 54, 56, 57, 63</td>
<td>inez</td>
<td>aff</td>
<td>2</td>
<td>5488</td>
<td>5</td>
<td>2386</td>
<td>2386</td>
<td>84</td>
<td>110</td>
</tr>
<tr>
<td>Papain</td>
<td>18, 19, 21, 159, 177, 181</td>
<td>1stf</td>
<td>aff</td>
<td>2</td>
<td>9408</td>
<td>5</td>
<td>5784</td>
<td>5784</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Trypsin</td>
<td>189, 190, 192, 195, 213</td>
<td>BPTI set*</td>
<td>multi</td>
<td>20</td>
<td>896</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

145
<table>
<thead>
<tr>
<th>Protein</th>
<th>Positions</th>
<th>BPTI set*</th>
<th>Species Code</th>
<th>Stringency</th>
<th>Molecules/Proteins</th>
<th>Score</th>
<th>Molecules/Proteins</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trypsin</td>
<td>189, 190, 192, 195, 214</td>
<td></td>
<td>spec</td>
<td>20</td>
<td>896</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scytovirin</td>
<td>1, 6, 10, 13, 28, 43, 48, 58, 61, 76</td>
<td>2qsk</td>
<td>stab</td>
<td>2</td>
<td>896</td>
<td>1</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>Putative monooxygenase</td>
<td>5, 13, 21, 55, 57, 59, 61, 70</td>
<td>2ril</td>
<td>stab</td>
<td>2</td>
<td>9604</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-potential iron-sulfur protein</td>
<td>6, 7, 14, 19, 23, 26, 41, 60, 64, 69, 73, 74, 78</td>
<td>3a38</td>
<td>stab</td>
<td>2</td>
<td>1882384</td>
<td>5</td>
<td>25708</td>
<td>40141</td>
</tr>
<tr>
<td>dpy-30-like protein</td>
<td>64, 68, 87, 91</td>
<td>3g36</td>
<td>stab</td>
<td>2</td>
<td>196</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actin</td>
<td>24, 25, 26, 28, 345, 349</td>
<td>2a40/2a41</td>
<td>multi</td>
<td>4</td>
<td>392</td>
<td>1</td>
<td>248</td>
<td>97</td>
</tr>
<tr>
<td>CED-4</td>
<td>1, 5, 227, 229, 259, 265, 279, 282</td>
<td>2a5y/3lqr</td>
<td>multi</td>
<td>4</td>
<td>288</td>
<td>5</td>
<td>221</td>
<td>221</td>
</tr>
<tr>
<td>CED-5</td>
<td>1, 5, 227, 229, 259, 265, 279, 283</td>
<td>2a5y/3lqr</td>
<td>spec</td>
<td>4</td>
<td>288</td>
<td>1</td>
<td>217</td>
<td>78</td>
</tr>
<tr>
<td>Rab-11A</td>
<td>44, 46, 47, 48, 50</td>
<td>2gzd/2gzh</td>
<td>multi</td>
<td>4</td>
<td>2744</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rab-11A</td>
<td>44, 46, 47, 48, 51</td>
<td>2gzd/2gzh</td>
<td>spec</td>
<td>4</td>
<td>2744</td>
<td>5</td>
<td>601</td>
<td>601</td>
</tr>
<tr>
<td>DNA polymerase beta</td>
<td>291, 309, 311, 322, 324</td>
<td>3k75/3lqc</td>
<td>multi</td>
<td>4</td>
<td>2744</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA polymerase beta</td>
<td>291, 309, 311, 322, 325</td>
<td>3k75/3lqc</td>
<td>spec</td>
<td>4</td>
<td>2744</td>
<td>3</td>
<td>1388</td>
<td>556</td>
</tr>
<tr>
<td>Protein</td>
<td>Accession Numbers</td>
<td>Structure Type</td>
<td>Ref</td>
<td>Multiplier</td>
<td>ID</td>
<td>Multiplier</td>
<td>ID</td>
<td>Multiplier</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-----</td>
<td>------------</td>
<td>----</td>
<td>------------</td>
<td>----</td>
<td>------------</td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 584</td>
<td>3KTP/3KTR</td>
<td></td>
<td>multi</td>
<td>4</td>
<td>448</td>
<td>1</td>
<td>352</td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 585</td>
<td>3KTP/3KTR</td>
<td></td>
<td>spec</td>
<td>4</td>
<td>448</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CDO</td>
<td>867, 872, 874, 901, 918</td>
<td>3N1F/3N1Q</td>
<td></td>
<td>multi</td>
<td>4</td>
<td>112</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CDO</td>
<td>867, 872, 874, 901, 919</td>
<td>3N1F/3N1Q</td>
<td></td>
<td>spec</td>
<td>4</td>
<td>112</td>
<td>1</td>
<td>112</td>
</tr>
<tr>
<td>Brother of CDO</td>
<td>753, 756, 758, 760, 789, 804</td>
<td>3N1G/3N1M</td>
<td></td>
<td>multi</td>
<td>4</td>
<td>224</td>
<td>5</td>
<td>205</td>
</tr>
<tr>
<td>Brother of CDO</td>
<td>753, 756, 758, 760, 789, 805</td>
<td>3N1G/3N1M</td>
<td></td>
<td>spec</td>
<td>4</td>
<td>224</td>
<td>1</td>
<td>224</td>
</tr>
<tr>
<td>Type I TGFbeta receptor</td>
<td>199, 203, 267, 268, 269</td>
<td>1B6C</td>
<td></td>
<td>aff</td>
<td>2</td>
<td>64</td>
<td>5</td>
<td>64</td>
</tr>
<tr>
<td>Leupeptin inhibitor</td>
<td>356, 357</td>
<td>2NQA</td>
<td></td>
<td>aff</td>
<td>2</td>
<td>361</td>
<td>5</td>
<td>91</td>
</tr>
<tr>
<td>Ponsin</td>
<td>824, 826, 828, 834, 840, 842, 848, 849, 850, 859, 861, 863, 872, 877</td>
<td>2O9S</td>
<td></td>
<td>stab</td>
<td>2</td>
<td>4374</td>
<td>1</td>
<td>665</td>
</tr>
<tr>
<td>Angiotensinogen</td>
<td>12, 15, 136, 138, 140</td>
<td>2WXY/2WXX</td>
<td></td>
<td>red</td>
<td>2</td>
<td>224</td>
<td>5</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

| 147 |
Table 4.2: Protein design test cases without continuous flexibility. Columns as in Table 4.1. Table continues on next page.

<table>
<thead>
<tr>
<th>Protein redesigned</th>
<th>Mutatable residues</th>
<th>PDB id(s)</th>
<th>Type</th>
<th>s</th>
<th>N</th>
<th>N_e</th>
<th>m_1</th>
<th>m_5</th>
<th>g_1</th>
<th>g_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angiotensinogen</td>
<td>12, 15, 136, 138, 141</td>
<td>2wxx/2wxy</td>
<td>ox</td>
<td>2</td>
<td>224</td>
<td>5</td>
<td>203</td>
<td>207</td>
<td>10</td>
<td>22</td>
</tr>
<tr>
<td>Sec24d</td>
<td>833, 834, 835, 836, 1025</td>
<td>3efo/3eg9</td>
<td>spec</td>
<td>4</td>
<td>4116</td>
<td>2</td>
<td>1830</td>
<td>565</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Leupeptin inhibitor      | 356, 357           | 2nqa      | aff  | 2 | 361| 5   | 37  | 73  | 2   | 10  |
| Trypsin                  | 189, 190, 192, 195, 213 | BPTI set* | spec | 20| 2476099| 5  | 14707| 14707| 20  | 100 |
| Scytovirin               | 1, 6, 10, 13, 28, 43, 48, 58, 61, 76 | 2qsk      | stab | 2 | 4.7x10^7| 1  | 109 | 2   |
| Scytovirin               | 1, 6, 10, 13, 28, 43, 48, 58, 61, 76 | 2ril      | stab | 2 | 1.7x10^10| 0  |
| dpy-30-like protein      | 64, 68, 87, 91     | 3g36      | stab | 2 | 6859| 2   | 73  | 2   |
| CED-4                    | 1, 227, 229, 259, 265, 279, 282 | 2a5y/3lqr | multi| 4 | 87808| 5  | 39  | 151 | 4   | 20  |
| CED-4                    | 1, 227, 229, 259, 265, 279, 282 | 2a5y/3lqr | spec | 4 | 87808| 5  | 30  | 87  | 4   | 20  |
| Rab-11A                  | 44, 46, 47, 48, 50  | 2gzd/2gzh | multi| 4 | 2476099| 5  | 42193| 45883| 4   | 20  |

148
<table>
<thead>
<tr>
<th>Protein</th>
<th>GenBank Accession Numbers</th>
<th>Type</th>
<th>Accession Number</th>
<th>PDB Accession Number</th>
<th>Chain</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rab-11A</td>
<td>44, 47, 48, 50</td>
<td>2gzd/2gzh</td>
<td>spec</td>
<td>2476099</td>
<td>5</td>
<td>59077</td>
</tr>
<tr>
<td>Cdc42</td>
<td>62, 63, 64</td>
<td>2ngr/1grn</td>
<td>multi</td>
<td>6859</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cdc43</td>
<td>62, 63, 65</td>
<td>2ngr/1grn</td>
<td>spec</td>
<td>6859</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sec24a</td>
<td>430, 435, 496, 748, 750, 752, 808</td>
<td>3egd/3egx</td>
<td>multi</td>
<td>5488</td>
<td>5</td>
<td>1480</td>
</tr>
<tr>
<td>Sec24a</td>
<td>430, 435, 496, 748, 750, 752, 809</td>
<td>3egd/3egx</td>
<td>spec</td>
<td>5488</td>
<td>5</td>
<td>1480</td>
</tr>
<tr>
<td>DNA polymerase beta</td>
<td>291, 309, 311, 322, 324</td>
<td>3k75/3lqc</td>
<td>multi</td>
<td>2744</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DNA polymerase beta</td>
<td>291, 309, 311, 322, 324</td>
<td>3k75/3lqc</td>
<td>spec</td>
<td>2744</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 584</td>
<td>3ktp/3ktr</td>
<td>multi</td>
<td>2476099</td>
<td>5</td>
<td>4249</td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 584</td>
<td>3ktp/3ktr</td>
<td>spec</td>
<td>2476099</td>
<td>5</td>
<td>1315</td>
</tr>
<tr>
<td>CDO</td>
<td>867, 872, 874, 901, 918</td>
<td>3n1f/3n1q</td>
<td>multi</td>
<td>2476099</td>
<td>5</td>
<td>50851</td>
</tr>
<tr>
<td>CDO</td>
<td>867, 872, 874, 901, 918</td>
<td>3n1f/3n1q</td>
<td>spec</td>
<td>2476099</td>
<td>5</td>
<td>60661</td>
</tr>
<tr>
<td>PAF-acetylhydrolase</td>
<td>194, 212, 235, 236, 238, 254, 316</td>
<td>1vyh</td>
<td>aff</td>
<td>1792</td>
<td>5</td>
<td>1275</td>
</tr>
<tr>
<td>Protein tyrosine phosphatase 1B</td>
<td>7, 11, 12, 268, 269, 272</td>
<td>2b4s</td>
<td>aff</td>
<td>2</td>
<td>2744</td>
<td>5</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------</td>
<td>------</td>
<td>-----</td>
<td>---</td>
<td>------</td>
<td>---</td>
</tr>
<tr>
<td>RING2</td>
<td>22, 29, 32, 36</td>
<td>2h0d</td>
<td>aff</td>
<td>2</td>
<td>2744</td>
<td>5</td>
</tr>
<tr>
<td>Angiotensinogen</td>
<td>12, 15, 136, 138, 141</td>
<td>2wxx/2wxy</td>
<td>ox</td>
<td>2</td>
<td>224</td>
<td>5</td>
</tr>
<tr>
<td>Sec24d</td>
<td>833, 834, 835, 836, 1025</td>
<td>3efo/3eg9</td>
<td>multi</td>
<td>4</td>
<td>4116</td>
<td>5</td>
</tr>
<tr>
<td>Sec24d</td>
<td>833, 834, 835, 836, 1025</td>
<td>3efo/3eg9</td>
<td>spec</td>
<td>4</td>
<td>4116</td>
<td>4</td>
</tr>
</tbody>
</table>

4.4 Conclusions

COMETS fills an important lacuna in protein design. A designer can now optimize any linear combination of optimal state energies, using constraints to ensure the desired combination of stability, affinity, and specificity. This can all be done with provable guarantees of optimality, both for the output sequence and for the state conformational energies of each candidate sequence. A wide range of conformational flexibility, both continuous and discrete, can be accommodated. Thus, COMETS offers a wide range of advantages to the molecular design community.
Fast gap-free enumeration of conformations and sequences for protein design

I once got the idea to try to speed up A* by expanding residues in a dynamic order. But when I talked to Pablo about this, it turned out he had had the same idea, but had produced much better empirical results supporting it than I had. Kyle Roberts also had some interesting ideas about speeding up A*, so we all wrote a paper about it, and this chapter is adapted from parts of that paper:


Kyle and Pablo did most of this, and it’s mostly their ideas. Before talking to Pablo, I had however tried out a couple of residue-ordering strategies that Pablo had not considered, including the harmonic mean-based strategy discussed in the paper, so that strategy is my main contribution to this work. As such, this chapter will cover the portions of the work that focus on dynamic ordering of residues in A*.
5.1 Methods

5.1.1 Background: The A* algorithm for the protein design problem

In CSPD, a protein conformation can be represented as a vector of $n$ rotamers $\mathbf{a} = (a_1, a_2, ..., a_n)$, where $n$ is the number of residue positions allowed to mutate during the design search. The total energy for the conformation $\mathbf{a}$ is defined as

$$E(\mathbf{a}) = E_{\text{templ}} + \sum_{i=0}^{n} E(a_i) + \sum_{i=0}^{n} \sum_{j=i+1}^{n} E(a_i, a_j), \quad (5.1)$$

where $E_{\text{templ}}$ is the template energy (i.e., the energy of the backbone atoms and side chain residues that are not allowed to move or mutate), $E(a_i)$ is the internal energy of rotamer $a_i$ plus the energy of $a_i$ with the template, and $E(a_i, a_j)$ is the pairwise energy between rotamers $a_i$ and $a_j$. In its simplest form, the goal of CSPD is to find the rotamer vector with the lowest energy, known as the GMEC: $\mathbf{g} = \arg\min_\mathbf{a} E(\mathbf{a})$.

To reduce the size of the protein conformational search space, DEE can be used to prune rotamers that are guaranteed to not be part of the GMEC. After DEE pruning, many low-energy protein conformations remain unpruned and must be searched to find the lowest-energy structures.

To enumerate protein conformations that remain after DEE pruning, a branch-and-bound algorithm based on the A* algorithm can be used. A* searches protein conformations by representing the design problem as a tree search and traverses only the branches of the tree that might lead to the lowest energy structure. Each level of the tree represents a residue position in the protein that is being designed (Fig. 5.1). Each internal node in the tree represents a partial rotamer assignment, where the number of assigned rotamers is equal to the node’s depth in the tree. Therefore, every leaf node of the tree is a complete rotamer assignment. Formally, each node $x$ at depth $m$ in the tree contains a partial rotamer assignment $\mathbf{p} = (p_1, p_2, \cdots, p_m)$.
where \( p_m \) is the assigned rotamer at the \( m^{th} \) residue position. The remaining residue positions \( U = \{m + 1, \cdots, n\} \) have not been assigned a rotamer yet. Every node \( x \) is scored with an \( f \)-score, which is the sum of the partially assigned conformation’s energy \( g(x) \) and a bound on the remaining possible rotamer assignments for that node \( h(x) \):

\[
f(x) = g(x) + h(x).
\]  

(5.2)

Since \( g(x) \) scores the unique partially assigned conformation at node \( x \), the energy of this partial conformation can be computed exactly:

\[
g(x) = \sum_{i=1}^{m} \left( E(p_i) + \sum_{j=i+1}^{m} E(p_i, p_j) \right)
\]  

(5.3)

In contrast, \( h(x) \) must estimate the minimum energy of all remaining rotamer assignments for node \( x \). Any function that provides a lower-energy bound for the protein conformations can be used by A* to enumerate low-energy conformations. In the following canonical bound is presented:

\[
h(x) = \sum_{j=m+1}^{n} \min_{q_j \in Q_j} \left( E(q_j) + \sum_{i=1}^{m} E(p_i, q_j) + \sum_{k=j+1}^{n} \min_{q_k \in Q_k} E(q_j, q_k) \right).
\]  

(5.4)

We refer to the \( f \)-score (Eq. 5.2) that incorporates the above canonical bound (Eq. 5.4) as the traditional \( f \)-score. In Eq. (5.4), \( Q_j \) refers to the set of unpruned rotamers that are allowed at residue position \( j \).

The A* algorithm proceeds by iteratively finding the A* node with the lowest \( f \)-score and expanding the node by creating child nodes that assign specific rotamers to the next unassigned residue position. To expand a node at depth \( m \) in the A* tree, a child node is created for each rotamer \( r \) at residue position \( m + 1 \) with the partial rotamer assignment \( (p_1, \cdots, p_m, r) \). The A* algorithm progressively expands nodes
until the lowest \( f \)-score node is a leaf node. This leaf node is guaranteed to be the GMEC because the lower bounds for all remaining conformations are higher than the leaf node’s energy. If desired, A* can continue to enumerate conformations in order of lowest energy, which provides a gap-free, in-order list of low-energy conformations. We refer to the A* algorithm described in this section and presented in\(^{99}\) as Trad-A*.

In Table 5.1 we summarize the A* method terminology introduced in this section.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Residue (Variable) Orderings</strong></td>
<td></td>
</tr>
<tr>
<td>Trad-A* (or Sequential)</td>
<td>Residue positions in the A* search are ordered by their location in the protein’s amino acid sequence.</td>
</tr>
<tr>
<td>StaticMinDom</td>
<td>Residue positions are expanded in order of increasing variable domain size (i.e., the number of available rotamers per residue position).</td>
</tr>
<tr>
<td>StaticMaxDom</td>
<td>Opposite of StaticMinDom. Residue positions are expanded in order of decreasing variable domain size.</td>
</tr>
<tr>
<td>StaticDomCmed</td>
<td>Residue positions are expanded based on the ratio of the variable’s domain size divided by the sum of the median pairwise energies to every other residue position.(^{154})</td>
</tr>
<tr>
<td>StaticHMean</td>
<td>Residue positions are ordered based on the harmonic mean of all the position’s energetic interactions (Eq. 5.5).</td>
</tr>
<tr>
<td>DynMin</td>
<td>The residue position to be expanded is chosen dynamically such that it has the largest minimum ( f )-score.</td>
</tr>
<tr>
<td>DynHMean</td>
<td>The residue position to be expanded is chosen dynamically such that it maximizes the harmonic mean of its ( f )-scores.</td>
</tr>
</tbody>
</table>

### 5.1.2 Ordered A* trees

The A* algorithm was originally developed for motion planning in robotics as a faster alternative to Dijkstra’s algorithm.\(^{73}\) While protein design shares some similarities with this motion planning technique, there is a distinct difference between the two problems. In motion planning, each edge in the tree corresponds to a planned motion, and the output of the algorithm is the complete path from tree root to leaf node, which corresponds to the planned robot motion. However, in protein design, only
the final leaf node (i.e., the full rotamer assignment) is required as output, so the
path that was taken to get to the leaf is discarded. In other words, the order in
which the rotamers are assigned to residue positions in the A* tree does not matter
for correctness and the path in the protein design A* tree does not have any inherent
meaning, in stark contrast to motion planning.

While the residue ordering within the A* tree does not affect correctness (final
output conformation and sequence), the complexity (A* runtime) can be drastically
affected by the order. Intuitively, if a subtree in the A* tree does not contain the
GMEC, it is beneficial to prune it and prevent exploration of this subtree. If the
algorithm encountered paths with high energy bounds early in the search, then A*
could avoid expanding an exponential number of paths. Encountering paths with
high low-energy bounds early can be achieved by ordering the tree such that nodes
with high bounds are expanded closest to the root of the A* tree and all its subtrees.
Consider the example in Figure 5.2. Residues 3, 4, and 5 form a clique, such that
the choice of a rotamer at one residue significantly affects the choice at the other
two residues. If the traditional sequential residue ordering is used for the A* search,
all of the nodes at a depth of 1, 2, 3, 4, and 5 must be created (Fig. 5.2C) in order
to guarantee that all other paths do not lead to the GMEC. However, when the
ordering is switched (Fig. 5.2D-E), only one or two levels must be explored in the
subtrees that do not lead to the GMEC (Fig. 5.2D-E). This is well-known in the field
of constraint satisfaction problems (CSPs), where depth-first search combined with
backtracking is used to find a variable assignment that satisfies all of the problem
constraints. Much work has been done developing variable ordering heuristics and
evaluating their performance on various CSPs (e.g.\textsuperscript{2,6,51}).

The guiding principle behind a good residue ordering is to order the residues such
that the number of expanded nodes in the A* tree is small. This can be done by
ordering paths so that they fail (i.e., have large bounds) as fast as possible. In the
protein design A* search, this can be done by choosing to expand residue positions that will increase the lower bound on the conformations within the tree as much as possible. Based on this idea, we introduce four static (predetermined) residue orderings to replace the traditional sequential ordering. We also present two new dynamic ordering methods that choose which residue position to expand next at each node based on the possible increase in energy bound (i.e., Eq. 5.2).

Stat**ic A** ordering.

The traditional A* enumeration in protein design uses a sequential static ordering of residues in the A* tree. Specifically, depth $m$ in the tree corresponds to the $m^{th}$ mutable residue in the protein design problem. However, there is no reason why this would be the optimal ordering of residue positions within the tree. We have implemented four alternative variable orderings, StaticMinDom, StaticMaxDom, StaticDomCmed, and StaticHMean to determine how these ordering methods affect the speed and efficiency of the protein design A* search. The **StaticMinDom** ordering expands residue positions in order of increasing variable domain size (i.e., number of available rotamers per residue position). By expanding variables with a small domain first this greedily minimizes the total size of the A* tree. The **StaticMaxDom** ordering is the opposite of StaticMinDom and expands residue positions in order of decreasing variable domain size. By expanding residue positions with many rotamers early in the search, the total number of conformations that the $h$-score must bound for a specific node is reduced, which could lead to a more direct convergence to the GMEC. The **StaticDomCmed** ordering method is defined in and chooses the residue position to expand based on the ratio of the variable’s domain size divided by the sum of the median pairwise energies to every other residue position. By using the median variable costs, this ordering tries to take into account the lower bound increase that will occur when a specific variable is chosen and find
the variable that will increase the lower bound the most. Similar to StaticDomCmed, the \textbf{StaticHMean} ordering scores every position based on the harmonic mean of all the position’s energetic interactions:

\begin{equation}
\text{SHM}(i) = \sum_{j \neq i} \left( \frac{|Q_i \times Q_j| - 1}{\sum_{(q_i, q_j) \in (Q_i \times Q_j - \{\arg \min_{r_i, r_j} E(r_i, r_j)\})} \left( E(q_i, q_j) - \min_{r_i, r_j} E(r_i, r_j) \right)^{-1}} \right).
\end{equation}

StaticHMean first normalizes every pairwise interaction between residues $i$ and $j$ by subtracting the minimum pair energy for any pair of rotamers at the two positions ($\min_{r_i, r_j} E(r_i, r_j)$). Then, StaticHMean computes the harmonic average for each residue pair using the normalized pairwise energies. All the harmonic terms involving a specific residue are summed and used as the residue’s score. Residues are then ordered in the A* tree by decreasing order of this score.

\textit{Dynamic A* ordering.}

Dynamic A* reordering allows the A* algorithm to choose which residue position to expand next at an A* node based on which residue position will move the search closest to the GMEC energy. This removes the correspondence between the depth in the A* tree and the residue position that exists in traditional A* algorithms. A dynamic A* node contains a set of rotamers $P$ that have been assigned and a set of unassigned residue positions $U$. There is no requirement that the rotamers in $P$ have sequential residue positions, in contrast to the definition of the partial rotamer assignment $p$ in Section 5.1.1. Hence, the residue positions in $U$ need not be sequential either. Therefore, dynamic A* adds a step to the traditional A* search
where the next residue position to be expanded is chosen from $U$. Different strategies can be used to choose the next residue position for a given $A^*$ node. We tested two ways to choose which residue to expand, DynMin and DynHMean. **DynMin** chooses the next variable based on the maximum of the variable’s minimum $f$-score. **DynHMean** chooses the residue with the maximum harmonic mean $f$-score (with respect to the parent’s $f$-score).

In mathematical terms, the next chosen residue position by DynMin is the position $i$ such that:

$$i = \arg \max_{i \in U} \left( \min_{q_i \in Q_i} f(P \cup \{q_i\}) \right). \quad (5.6)$$

The next residue $i$ for the DynHMean dynamic ordering is chosen as:

$$i = \arg \max_{i \in U} \left( \frac{|Q_i|}{\sum_{q_i \in Q_i} \left( f(P \cup \{q_i\}) - f(P) \right)} \right). \quad (5.7)$$

$A^*$ enumeration with dynamic ordering proceeds by directly choosing the next residue position to expand at each node instead of expanding residue positions in a predetermined order. The traditional $A^*$ algorithm in Section 5.1.1 can be updated to reflect this change:

$$f(x) = g(x) + h(x) \quad (5.8)$$

$$g(x) = \sum_{p_i \in P} \left( E(p_i) + \sum_{p_j \in P, j > i} E(p_i, p_j) \right) \quad (5.9)$$
\[
    h(x) = \sum_{j \in U} \min_{q_j} \left( E(q_j) + \sum_{p \in P} E(p_i, q_j) + \sum_{k \in U, k > j} \min_{q_k} E(q_j, q_k) \right). \quad (5.10)
\]

### 5.1.3 Benchmarking methods

**Benchmarking test set.** The protein systems from\(^{40}\) were used as a test set to evaluate the proposed algorithmic improvements. Briefly, crystal structures of protein chains with a maximum resolution of 1.3 Å and less than 100 residues in length were chosen using the PISCES server.\(^{166}\) Proteins in the test set were chosen such that they had less than 10% sequence identity with all other proteins in the test set. The test set consists of the proteins with the following PDB ids: 1AHO, 1CC8, 1F94, 1FK5, 1G6X, 1I27, 1IQZ, 1JHG, 1JNI, 1L9L, 1LNI, 1M1Q, 1MJ4, 1MWQ, 1OAI, 1OK0, 1PSR, 1R6J, 1T8K, 1TUK, 1U07, 1U2H, 1UCR, 1UCS, 1USM, 1V6P, 1VBW, 1VFY, 1WXC, 1X6I, 1XMK, 1Y6X, 1ZZK, 2AIB, 2B97, 2BT9, 2BWF, 2CC6, 2CG7, 2COV, 2CS7, 2D8D, 2DSX, 2FCW, 2FHZ, 2FMA, 2GOM, 2HBA, 2HIN, 2HLR, 2HS1, 2IC6, 2J8B, 2O9S, 2P5K, 2QCP, 2QSK, 2R2Z, 2RH2, 2RIL, 2WJ5, 2ZXY, 3A38, 3D3B, 3DNJ, 3FGV, 3FIL, 3G21, 3G36, 3HFO, 3I2Z, 3JTZ, 3LAG.

**Side-chain placements.** Side-chain placement runs selected all residues with < 100% relative side-chain solvent accessible surface area (SASA) and searched over all wild-type amino acid rotamers at each chosen residue position. SASA values were determined with the program NACCESS.\(^{79}\) The number of flexible residues for each system ranges from 45 to 97 with an average of 71 flexible residues.

**Protein core designs.** Protein core design runs selected core residues with < 30% relative SASA to mutate during the design search. Each mutable residue was allowed to take on its wild-type identity and mutate to the 5-7 most likely amino acid type substitutions based on the BLOSUM62 matrix.\(^{75}\) SASA values were determined with
the program NACCESS. The number of mutable residues for each system ranges from 11 to 42 with an average of 28 mutable positions per design system.

**Protein surface designs.** The protein surface designs were similar to the protein core designs, except that all residues with > 50% relative SASA were chosen to mutate during the design search. The number of mutable residues for each design system ranged from 17 to 48 residues, with an average of 28 mutable positions per design system.

**OSPREY parameters.** The protein design runs used the Richardsons’ Penultimate Rotamer Library, while the side-chain placement runs used the Penultimate Rotamer Library doped with the crystal structure side-chain conformations. The objective energy function consisted of the following terms: the AMBER van der Waals and Coulombic potential, EEF1 solvation, a hydrogen bond potential, an entropic factor and reference energies. The following energy function weights were used: distDepDiel=true, dielectConst=4.0, solvation=0.40, vdwMult=0.95, hbond=3.0, and entropy=5.0. To ensure that a sufficient number of rotamers were present for the enumeration step, only Goldstein DEE pruning was used during the pruning stage. Each design was run on a single Intel(R) Xeon(R) CPU E5-2695 v2 2.40GHz processor with 4 GB of RAM (except the continuous designs were allowed 10 GB of RAM). A design was considered to have failed if it ran out of memory or did not complete within one day of computation. For example, all the difficult side-chain placement and protein core design runs conducted with Trad-A* failed because they ran out of memory. For a given protein system, the same energy matrix and DEE pruning results are used for each enumeration technique. Therefore, to understand the gains from our new enumeration methods, the runtimes reported in this manuscript reflect only the enumeration time, i.e., the time A* took to solve the problem.
5.2 Results

We tested all the A* enhancements described in the Methods Section (Table 5.1) on a test set of 73 protein design systems. To assess the benefits of these methods compared to Trad-A*, we focus our analysis on difficult problems: those problems that take over two minutes to run using Trad-A*. Hence, all problems that were solved by Trad-A* in under two minutes were removed from further consideration.

We performed two types of designs to test these algorithms. To test exclusively the performance of the ordering methods, we performed side-chain placement designs and further analyzed the 39 difficult design systems. Side-chain placement is a variation of CSPD where no mutations are allowed, so the search is performed on rotamers of the same amino acid type. Then, all combinations of the new methods were tested on protein designs of the cores of the protein test set, 29 difficult designs in total.

5.2.1 Variable ordering

We evaluated the static and dynamic variable orderings using side-chain placements of the test protein systems. All residue orderings were tested using the Trad-A* f-score. Four static variable orderings were tested in addition to the standard sequential residue ordering used in Trad-A*. Because the sequential Trad-A* ordering does not use any information about the variable to choose the ordering, this ordering can be considered a random or arbitrary ordering. On the other hand, each new residue ordering is based upon a fail-first principle that tries to order the residue positions in a favorable manner.

Out of the 73 systems tested, 39 were classified as difficult. The sequential Trad-A* algorithm solved 10 of these problems; StaticMinDom solved 9 problems; StaticDomCmed solved 10 problems while StaticMaxDom and StaticHMean solved 13 and 26 systems, respectively. In addition, of the 10 problems solved by sequential
ordering, the StaticMinDom, StaticMaxDom, StaticDomCmed, and StaticHMean orderings were each faster than sequential ordering for 8, 6, 9, and all 10 systems, respectively. For the 10 systems that completed with sequential ordering, StaticHMean and StaticMaxDom required the least number of A* nodes (median number of expanded nodes: 3900 and 81000, respectively), while Trad-\(A^*\), StaticMinDom, and StaticDomCmed required many more node expansions (362000, 109000, and 129000 nodes, respectively). Overall, these results show that the StaticMaxDom and StaticHMean methods that specifically focus on quickly improving the A* \(f\)-score lower bound outperform the other orderings. Our newly proposed dynamic orderings improve upon the StaticMaxDom and StaticHMean orders by specifically analyzing the \(f\)-scores of future nodes to find more favorable residue orderings than can be found through static methods.

The dynamic variable orderings, DynMin and DynHMean were able to solve 30 and 31 of the difficult side-chain placement problems, improving upon the static variable orderings. DynMin and DynHMean performed faster than the sequential ordering for all test systems. DynMin and DynHMean expand fewer nodes than the static variable methods in 25 and 29 cases respectively, achieving up to a 2700-fold reduction in the number of expanded nodes (Fig. 5.3) within the 10 problems that were solved by all methods. The dynamic ordering methods require more computation per node to find the efficient paths through the A* tree; there is an average 43-fold increase in time needed to expand an A* node (51 expanded nodes/second versus 2200 nodes/second). However, the reduction in the number of nodes that must be expanded far outweighs the additional time needed to determine which variable to expand next.
Figure 5.1: **Tree representation of protein conformation space.** (A) A toy example of three serine residues (shown in orange, blue and grey) belonging to the antibody VRC07 (PDB id: 4OLZ\textsuperscript{143}), partially shown in white cartoon. (B) A 2D representation of (A), and for the purposes of this toy example, we allow each residue to mutate to only two rotamers (shown here as a star and a circle). (C) Protein design algorithms compute pairwise interactions between rotamers based on an input energy function, and these are shown here in matrices between residue pairs. For simplicity, all internal rotamer energies are zero, and the pairwise energies not shown have a zero value. (D) The protein conformation and sequence space can be represented as a tree. In a tree representation, each level represents a residue, each inner node (each of the nodes between the root of the tree, r, and the leaves of the tree) represents a partially assigned conformation, and each child assigns a rotamer choice for the next residue. Each leaf represents a fully assigned conformation. A naïve approach to solve the protein design problem would explore this tree completely. The optimal path is shown in red. (E) Branch-and-bound algorithms such as A* explore a small part of the tree by computing energy lower bounds (called f-scores and shown next to each node) on the possible conformations allowed at each inner node. A* expands nodes in order of their f-score and guarantees that the optimal solution is found (shown in red).
Figure 5.2: Toy example that demonstrates the effect of residue position reordering on an A* search. (A) Toy design problem where seven residue positions are each allowed to mutate to two rotamers (represented by a star and a circle). Each residue position is colored by a unique color: orange (position 0), cyan (position 1), grey (position 2), purple (position 3), maroon (position 4), green (position 5), and black (position 6). (B) Diagram showing the pairwise energies between all rotamers in this toy example. For simplicity, assume that all intra-rotamer energies are zero and can be ignored, and that the interactions between pairs that are not joined by an edge are zero. (C-E) The A* algorithm explores only part of the full conformation tree to compute the optimal conformation. A* iteratively expands the node with the lowest $f$-score (shown by the dotted red path for the nodes in the optimal conformation path) until a leaf is reached. Each expansion results in the creation of new nodes representing the children of the expanded node. To compute the optimal conformation efficiently, it is desirable to expand the fewest number of nodes. The number of nodes expanded can be dramatically reduced by changing the ordering of the tree. (C) The traditional A* algorithm for protein design (Trad-A*) sorts residues in the arbitrary sequential order given by the protein sequence. The bounds on the energies for each inner node in the optimal conformation are shown in red, and the path that leads to the optimal conformation is marked in a thick, red, dashed line. In this toy example, Trad-A* expands 33 nodes, and creates 67 nodes (the 33 expanded nodes plus their children). (D-E) Large speedups in A* can be achieved by a rational ordering of nodes. The energies of each node in the optimal conformation are shown. (D) In a static reordering, residue levels are reordered once before A* runs. In this toy example, A* with static reordering must only expand 13 nodes and create 25 nodes to compute the optimal conformation. (E) In a dynamic reordering, the next level is chosen independently for each path “on the fly” (i.e., as the A* algorithm expands nodes). In this dynamic reordering example, at depth $m = 2$ the solution path expands position 3 (purple) while the alternative path expands position 4 (maroon). A* with dynamic ordering must expand only 9 nodes, and create a total of 17 nodes, to compute the optimal conformation. (F) The optimal conformation for this example is shown.
Figure 5.3: The number of expanded A* tree nodes is greatly reduced by improved variable ordering methods. **Top:** The total number of conformations A* had to search through for 31 difficult side-chain placement problems. The size of the conformation space shown is the number of conformations remaining after dead-end elimination pruning. **Bottom:** The number of A* nodes expanded by three different A* orderings for the 31 side-chain placement problems. Data is shown for the sequential residue ordering used in Trad-A* (red circles), the StaticHMean static variable ordering (purple pentagons), and DynHMean dynamic variable ordering (green squares). The x-axis is labeled by the PDB id used for each side-chain placement problem. All of the runs used the Trad-A* $f$-score. Trad-A* failed to solve 21 problems (right of the red vertical line) and StaticHMean failed to solve 5 of the problems. For visual clarity the x-axis is ordered first by the number of nodes expanded by Trad-A*, second by StaticHMean, and finally by DynHMean.
In the summer of 2014, I did an internship at the NIH’s Vaccine Research Center (VRC) in Maryland. I was working in the computer room next to crystallographer Young Do Kwon when our PI, Peter Kwong, came in. I heard them discussing the new structure that Young had just solved. It turned out it was the first unliganded, atomic-resolution structure of the HIV-1 Env trimer, which is the main surface protein of HIV! I got the wonderful opportunity to participate in the analysis of this structure, which was published in Nature Structural and Molecular Biology, with many other collaborators at VRC and elsewhere:

Briefly, I worked with Ivelin Georgiev of the VRC to develop a score called the Antigen Structural Compatibility Score, which uses the structures of Env-based immunogen candidates (such as the unliganded trimer) and of (Env fragment : antibody) complexes to predict the suitability of the candidates for use in vaccines. I will describe this method and relevant other portions of the work here, adapted from the paper.

Summary of paper. As the sole viral antigen on the HIV-1–virion surface, trimeric Env is a focus of vaccine efforts. Here we present the structure of the ligand-free HIV-1–Env trimer, fix its conformation and determine its receptor interactions. Epitope analyses revealed trimeric ligand-free Env to be structurally compatible with broadly neutralizing antibodies but not poorly neutralizing ones. We coupled these compatibility considerations with binding antigenicity to engineer conformationally fixed Envs, including a 201C 433C (DS) variant specifically recognized by broadly neutralizing antibodies. DS-Env retained nanomolar affinity for the CD4 receptor, with which it formed an asymmetric intermediate: a closed trimer bound by a single
CD4 without the typical antigenic hallmarks of CD4 induction. Antigenicity-guided structural design can thus be used both to delineate mechanism and to fix conformation, with DS-Env trimers in virus-like-particle and soluble formats providing a new generation of vaccine antigens.

6.1 Methods

6.1.1 Computation of antibody-epitope r.m.s. deviation, volume overlap and epitope presence.

HIV-1–specific antibody–antigen complex structures were compiled from the PDB, and antibodies were defined as broadly or poorly/non neutralizing, according to published or in-house neutralization data of diverse viral strains. Antibodies that were deemed to have insufficient evidence for being classified as broadly or poorly/non neutralizing were excluded from the analysis. A single antibody representative was included in the analysis in cases in which multiple antibody clonal relatives were found. The epitope residues for each antibody were defined on the basis of the respective antibody–antigen complex crystal structures, with an antigen residue being defined as an epitope residue if any of its heavy atoms were within 5.5 Å of any antibody heavy atom. To compute the r.m.s. deviation between the epitope residues in the antibody–antigen complex structure and the same residues in the ligand-free trimer structure, first the epitope residues from the complex structure were aligned to the ligand-free trimer structure with the align function in PyMOL, and then the Cα r.m.s. deviation of the epitope residues was calculated. To remove outlier residues, the top and bottom 10% of the Cα deviations were removed from the r.m.s. deviation calculation. To calculate the volume overlap between a given antibody and the ligand-free trimer structure, the alignment from above was used to compute the overlap volume between the antibody from the complex structure and the ligand-free trimer structure with the phase_volCalc utility from Schrödinger.
An antibody epitope was considered to be present in the ligand-free trimer structure if at least 70% of the epitope residues as defined by the antibody-antigen complex structure were also present in the ligand-free trimer structure. For mapping the per-residue r.m.s. deviation computation onto the ligand-free trimer structure, residues part of any antibody epitope (including epitopes with less than 70% total residues present) were included in the analysis; if a given residue was part of more than one antibody epitope, the highest r.m.s. deviation value for that residue among all epitopes was used. Antibody-volume-overlap values were mapped onto the ligand-free trimer structure for all residues part of the epitope for the given antibody; if a residue was part of more than one antibody epitope, then the lowest volume overlap for that residue among all epitopes was used. Correlations of structural properties with neutralization and/or binding data were computed with the Spearman correlation coefficient with two-tailed P values.

6.1.2 Structural compatibility analysis.

For a given antibody, the antigenic structural compatibility (ASC) score with the HIV-1–Env ligand-free prefusion trimer structure was computed on the basis of comparison to a structure of the antibody bound to an Env-derived antigen (for example, gp120 core or V3 peptide). ASC scores were computed on a 0–1 scale with the following variables: (i) The fraction $f$ of epitope residues (as defined by the structure of the antibody complex) exposed to solvent in the ligand-free trimer structure was computed. A residue was considered to be accessible to solvent if its solvent-accessible surface area (SASA) was at least half its SASA in the respective antibody complex structure, and $f$ was set to 0 if $<70\%$ of epitope residues were present in the antigen. (ii) A resolution estimate $r$ was used, such that Ca r.m.s. deviations $d$ below $r = 2$ were not penalized in the scores. (iii) The volume-overlap values were used to define a volume-overlap factor $v$ that is equal to 1 for overlap below 200 Å$^3$ and is equal to 0
for overlap over 1000 Å², and decays linearly in between. Intuitively, the ligand-free trimer structure is expected to be structurally compatible with an antibody if $f$ and $v$ are high and if the r.m.s. deviation $d$ is low, because such conditions would indicate similarity between the ligand-free trimer structure and the Env conformation in the antibody complex. Thus, the ASC score for each antibody with the ligand-free trimer was defined by the formula: $fv \exp(-0.5 \max(0, d - r))$.

6.2 Results

6.2.1 Structural compatibility and appropriate target conformation

In addition to facilitating virus entry, HIV-1 Env functions to evade the humoral immune response, a function in which glycan shielding and conformational change are critical. In the refined ligand-free Env structure, we observed electron density corresponding to single protein-proximal N-acetyl glucosamine residues at all sites of N-linked glycosylation, except at residues 197, 262 and 332 (with Env numbering following standard HXB2 convention), for which we observed additional monosaccharide residues, or at residue 137, which was mostly disordered. Overall, despite differences in glycosylation and lattice packing, the structure of the ligand-free trimer assumed a closed conformation, which was highly similar to that of antibody-bound trimers, especially the PGT122-35O22–bound trimer, with which it had an r.m.s. deviation in Cα positions of less than 1 Å, substantially lower than observed with monomeric gp120 (Fig. 6.1a).

To determine the appropriateness of the ligand-free closed trimer as a vaccine template, i.e., structural specificity for broadly neutralizing antibodies and incompatibility with non-neutralizing or poorly neutralizing antibodies, we first sought to categorize antibodies according to their functional efficacy (Fig. 6.1b). We defined broadly neutralizing antibodies as those with greater than 35% breadth on a diverse panel of 170 isolates and defined ineffective antibodies as those with less than
15% breadth. (For an isolate to be considered sensitive in this breadth analysis, we used a cutoff for antibody half-maximal inhibitory concentration (IC50) of <50 µg/ml). Antibodies b12 (Ref. 10), 35O22 (Ref. 78), and PGT135 (Ref. 163) were close to the cutoff for the broadly neutralizing category. Some antibodies showed clade-specific breadth, for example, V3-directed 447-52D, which neutralizes over 20% of clade B isolates. However, we nonetheless classified 447-52D as ineffective because its overall breadth was only 12%. We then analyzed the ligand-free closed structure for its structural compatibility with antibody epitopes—most determined structurally in the context of antibody-bound subunit or antibody-bound peptide—on the basis of two measures: antibody-volume overlap and epitope r.m.s. deviation (Fig. 6.1c). Antibody-volume overlap, which involves the superposition of epitopes in the ligand-free trimer and the antibody-bound context, was strongly anticorrelated with neutralization breadth (P = 0.0007; Fig. 6.1d). Epitope r.m.s. deviation, which compares epitope structural differences in the ligand-free trimer and the antibody-bound context, varied with breadth but did not achieve statistical significance. An antigenic structural-compatibility score (ASC), which combined both overlap and r.m.s. deviation, did achieve significance (P = 0.0031; Fig. 6.1d).

The ligand-free closed structure was compatible with the epitopes for all broadly neutralizing antibodies, except those of the membrane-proximal external region, which recognize epitopes C terminal to residue 664, and those of antibodies b12 (Ref. 179) and CH103 (Ref. 107), with CH103 exceeding a 2-Å threshold of epitope similarity and b12 exceeding a volume threshold of 500 Å³ (Fig. 6.1d). In light of the poor correlation of epitope r.m.s. deviation with neutralization breadth (Fig. 6.1d), the r.m.s. deviation threshold was somewhat arbitrary. Nonetheless, the specific incompatibility of these moderately effective CD4-binding-site antibodies suggests that movement of residues of the CD4-binding site could occur relative to the ligand-free closed trimer; indeed, induced trimer movements have been observed for b12 (which
binds poorly to the BG505 SOSIP.664 trimer\textsuperscript{145}) by EM10 and HDX29. By contrast, none of the epitopes for non-neutralizing or poorly neutralizing antibodies were structurally compatible with the ligand-free closed structure (Fig. 6.1d). These results indicate that the ligand-free closed trimer is structurally specific for neutralizing antibodies and thus is in an appropriate target conformation for immunogen design.

6.2.2 Appropriate target antigenicity for an Env-vaccine antigen

Structural specificity, as measured by epitope compatibility, is only one of the requirements of an appropriate vaccine template, and antigenic specificity, as measured by antibody binding, is also crucial.

The BG505 SOSIP.664 has previously been shown to be antigenically specific for broadly neutralizing antibodies, though binding to weakly neutralizing antibodies such as those directed to the V3 loop has been reported.\textsuperscript{145}

Our structural-compatibility analysis indicated that V3 antibodies are incompatible with the ligand-free closed state (Fig. 6.1c), suggesting that the binding of BG505 SOSIP.664 to V3 antibodies might not be intrinsic to the closed conformation of the SOSIP.664 construct but may instead be an artifact of alternative folding. Indeed, we found that negative selection\textsuperscript{59} by weakly neutralizing V3-directed antibodies substantially reduced V3 antibody binding to BG505 SOSIP.664; when we tested recognition on a panel of V3-directed antibodies, negative selection by 447-52D and by a V3-antibody cocktail reduced recognition by V3-directed antibodies to a level similar to that observed for noncognate antibody binding. We also tested CD4-negative selection but did not observe a substantial non-CD4-binding subportion of Env trimers. Together, these results indicated that an appropriate target antigenicity for an Env-vaccine antigen would involve no recognition by ineffective antibodies (including those directed at the V3 region), while maintaining recogni-
Figure 6.1: Ligand-free HIV-1 Env trimer is structurally compatible with epitopes of broadly neutralizing, but not ineffective, antibodies. (a) Superposition of ligand-free and antibody-bound HIV-1–Env structures. Left, ligand-free gp120 core monomer shown in ribbon representation, with regions of less (or greater) than 2-Å r.m.s. deviation upon antibody binding shown in green (or magenta) and representative antibody-bound structures in gray. Middle and right, ligand-free and antibody-bound HIV-1–Env trimers. At right, antibodies PGT122 and 35O22 are shown in gray semitransparent surface, and the rear protomer has been removed for clarity. (b) Breadth-potency plot of broadly neutralizing (green) and ineffective (magenta) antibodies on a diverse 170 HIV-1–isolate panel. (c) Structural compatibility of ligand-free trimer by antibody epitope. The ligand-free Env structure is displayed as Cα ribbon, with antibody-epitope residues colored green (structurally compatible) or magenta (incompatible) or gray for nonepitope regions. r.m.s. deviation (solid fill) and volume overlap (striped fill) with the indicated antibody–Env complexes are shown in bar graph, with two linear scales split at r.m.s. deviation and antibody-antigen–volume overlap cutoffs of 2 and 500 Å³, respectively; bars below the respective cutoffs are colored green and magenta otherwise. Antibody labels are colored green if the epitope is structurally compatible, magenta if incompatible and gray if not present in the structure. (d) Ligand-free-trimer structural compatibility versus antibody breadth. Volume overlap (left), r.m.s. deviation (middle) and antigenic structural compatibility (ASC) score (right), plotted versus antibody breadth on a diverse 170 HIV-1–isolate panel. P values for Spearman correlations provided (n = 14 antibodies).
tion of broadly neutralizing antibodies (except those with moderate neutralization, directed at the CD4-binding site).
LUTE (Local Unpruned Tuple Expansion): Accurate continuously flexible protein design with general energy functions and rigid-rotamer-like efficiency

While EPIC reduces the number of energy function calls in continuously flexible protein design from exponential to polynomial in the problem size, it doesn’t do much to reduce the number of minimizations we must perform. To address this, I thought about alternate energy surface representations that would admit faster lower-bound calculations, and a discrete representation seemed like the best option. This became LUTE, which we wrote up as a RECOMB paper:


I think LUTE has tremendous potential in the sense that it separates basically any
kind of advanced biophysical modeling—continuous flexibility, continuous entropy, very accurate energy functions, etc.—from the NP-hard optimization problem that is asymptotically the bottleneck in protein design. So with LUTE, all these expensive calculations can be run only a polynomial number of times to create a very accurate LUTE energy matrix, which then is fed to the reasonably efficient algorithms we have for discrete optimization. This can include very powerful new discrete optimization algorithms, like Jonathan Jou’s BWM* algorithm, a connection he and I analyze in this chapter (which is based on the RECOMB paper). I don’t think my current implementation of LUTE really does justice to the algorithm, because it doesn’t handle free energy, quantum-chemical energy estimation, or explicit water. But I hope to work on these issues in the future, and maybe other people will as well.

Summary. Most protein design algorithms search over discrete conformations and an energy function that is residue-pairwise, i.e., a sum of terms that depend on the sequence and conformation of at most two residues. Although modeling of continuous flexibility and of non-residue-pairwise energies significantly increases the accuracy of protein design, previous methods to model these phenomena add a significant asymptotic cost to design calculations. We now remove this cost by modeling continuous flexibility and non-residue-pairwise energies in a form suitable for direct input to highly efficient, discrete combinatorial optimization algorithms like DEE/A* or Branch-Width Minimization. Our novel algorithm performs a local unpruned tuple expansion (LUTE), which can efficiently represent both continuous flexibility and general, possibly non-pairwise energy functions to an arbitrary level of accuracy using a discrete energy matrix. We show using 47 design calculation test cases that LUTE provides a dramatic speedup in both single-state and multistate continuously flexible designs.
7.1 Introduction

Protein design algorithms compute protein sequences that will perform a desired function. They generally do this by minimizing the energy of a desired binding or structural state (or some combination thereof) with respect to sequence. Given a model of the conformational space of a protein and its energy function (which maps conformations to their energies), this is a well-defined computational problem.

Previously, this minimization problem has been most efficient to solve if two restrictions are imposed on the model. First, the conformational space of the protein is modeled as discrete. Specifically, each residue takes on conformations from a discrete set (typically, experimentally observed sidechain conformations known as rotamers). Hence, we optimize with respect to the amino-acid type and rotamer of each residue. Second, the energy function is assumed to be residue-pairwise, i.e., it is assumed to be a sum of terms that each depend on the amino-acid types and conformations of at most two residues.

A large body of efficient algorithms has been developed for this restricted case of the protein design problem, many of which offer provable accuracy. In particular, the dead-end elimination (DEE) algorithm removes rotamers that provably cannot be part of the global minimum-energy conformation (GMEC). The A* algorithm from artificial intelligence finds the optimal conformation using these unpruned rotamers. This DEE/A* framework has been generalized to model free energies for each sequence instead of simply GMECs (the K* algorithm). It has also been generalized to optimize combinations of stability and specificity by minimizing, with respect to sequence, a linear combination of the conformationally optimized energies of several bound and unbound states of a protein, instead of just the energy of a single state (the COMETS algorithm). Several methods in addition to DEE/A* have also been used to address the protein design problem. Some of these, such
as Metropolis Monte Carlo and simulated annealing,\textsuperscript{93,102} lack provable guarantees of accuracy, and thus may miss the optimal conformation significantly. Other algorithms with provable accuracy are also available, largely building on techniques from integer linear programming\textsuperscript{89,140} and weighted constraint satisfaction.\textsuperscript{140,154,155} Notably, treewidth- and branch-width-based algorithms, such as TreePack\textsuperscript{170} and BWM*\textsuperscript{83} solve this problem with provable accuracy in polynomial time for systems whose residue interaction graph has treewidth or branch-width bounded by a constant.\textsuperscript{83}

However, proteins are actually continuously flexible, and continuous flexibility both in the sidechains\textsuperscript{40} and backbone\textsuperscript{68} has been shown to result in significantly lower energies and biologically better sequences.\textsuperscript{40,68} Although a residue sidechain will usually be found in the \textit{vicinity} of the modal conformation for a rotamer, its dihedral angles will often differ from this mode by 10° or more.\textsuperscript{81} These continuous adjustments are often critical for determining what conformations are sterically feasible.\textsuperscript{40} Thus, incorporation of continuous flexibility modeling substantially increases the accuracy of designs. The minDEE and iMinDEE methods\textsuperscript{40,48} do this for continuous sidechain flexibility, and DEEPer\textsuperscript{68} for simultaneous continuous sidechain and backbone flexibility. These methods replace the traditional discrete rotamers used in DEE/A* with voxels in the conformation space of each residue, called \textit{residue conformations (RCs)}. An RC is defined as an amino acid type together with bounds on each of the conformational degrees of freedom of the residue (e.g., sidechain dihedrals).\textsuperscript{68} The modal conformation for a rotamer is usually found at the center of this voxel. In this model, the conformation space of a entire protein is a union of voxels, each of which is constructed as the cross-product of single-residue voxels. Thus, each voxel in the conformation space of the entire protein is represented by a list of RCs, one for each residue being modeled as flexible. RCs are constructed to be small enough that we can use local minimization to find the optimal energy.
within the voxel. This applies to both the single-residue and entire-protein voxels.

However, the global minimum energy in this model could not previously be computed directly by DEE/A*. Instead, DEE/A* was used to enumerate RC lists (protein conformational voxels) in order of a lower bound on minimized energy.\(^{40,48,68}\) Subsequently, the optimal energy for each RC list with a sufficiently low-energy lower bound was computed by minimization. The lower bound was computed from minimized pairwise interaction energies.\(^{48}\) This minimization was accelerated significantly by precomputing polynomials to approximate the energy landscape, using the EPIC algorithm.\(^{64}\) However, minimization was still the bottleneck in continuously flexible designs and prevented them from approaching the efficiency of designs with discrete flexibility. **In essence, these previous methods modeled continuously flexibility by modifying DEE/A* and making it do much more work. In contrast, LUTE achieves much greater efficiency by representing continuous flexibility in a form suitable for direct input into DEE/A*.**

We must also address the question of the energy function. The energy landscape of a real protein is not residue-pairwise, or otherwise exactly described solely as the sum of local terms. There is, however, ample evidence that protein interactions are local in a more general sense\(^{33,64,162,175}\)—i.e., that the cross-derivative of the energy with respect to conformational degrees of freedom of two residues will tend to zero fairly quickly as the distance between the residues increases. These properties are also observed for more realistic energy functions that return an energy for the entire protein, rather than breaking the energy into terms as molecular mechanics does. For example, the Poisson-Boltzmann model for implicit solvation\(^{148}\) and quantum-chemical models return an energy for the entire system on which they are run. Thus, a viable approach to modeling protein energies more realistically is to infer local terms from full-protein energies. Vizcarra et al.\(^{162}\) apply this approach to the Poisson-Boltzmann model, calculating pairwise energies from differences in
full-protein conformational states and achieving a pairwise energy matrix that quite accurately matches the Poisson-Boltzmann energies of full conformations. However, their method can only accommodate rotamer pairs, does not support continuous flexibility, and can only be used when substituting a single rotamer into a conformation is possible while maintaining the conformation of the other residues. This is impossible when residues share conformational degrees of freedom, which is typically needed for backbone flexibility,44,68 and may also cause problems in the case of steric clashes. Also, DEE/A* has been generalized to accommodate higher-than-pairwise energy terms if these terms are computed explicitly for particular tuples, e.g., triples of residues.116 However, most energy functions modeling higher-than-pairwise effects, including Poisson-Boltzmann, return a single energy for the entire system, rather than a sum of explicit local terms as required by algorithms such as those in Ref. 116.

Hence, today’s protein and drug designers are faced with a choice. They can neglect continuous flexibility and energy terms that aren’t explicitly local (e.g., explicitly pairwise), thus incurring significant error. Or they can pay a massive overhead to incorporate them—by enumerating many conformations (for continuous flexibility) or searching exhaustively (for non-pairwise energy functions). We now offer a way around this dilemma. We construct an energy function that is an explicit, discrete sum of local energy terms, which are associated with tuples of RCs. This function maps RC lists, which represent voxels in the conformation space of a protein, to energies. But it will approximate, to arbitrary accuracy, the minimized voxel energy, which can be computed with any energy function: no need for residue-pairwiseness or any other local representation. Computing this approximation is a machine learning problem, and we attack it with a least-squares method. Our approach has some resemblance to cluster expansion methods, which have previously been used in quantum mechanics15 and to represent optimized energies for protein
sequences.\textsuperscript{57,58} However, as discussed in Ref. 64, approximations of energy surfaces can be much more compact if unrealistically high-energy regions of conformational space are excluded from the approximation (and from the subsequent conformational search). Thus, unlike cluster expansion methods, we exclude \textit{pruned} tuples of RCs, making our derived energy function a \textit{local unpruned tuple expansion}, or LUTE. Because conformational and sequence search using the LUTE energy function is a discrete optimization problem of the type solved by DEE/A*, BWM*, and other very efficient algorithms, it allows designs to run quickly using these algorithms, while still approximating continuous flexibility and highly realistic energy functions to a high level of accuracy.

We have implemented LUTE in the \textsc{osprey}\textsuperscript{41,48,49} open-source protein design package, which has yielded many designs that performed well experimentally—\textit{in vitro}\textsuperscript{13,36,42,54,139,143,151} and \textit{in vivo}\textsuperscript{36,54,139,143} as well as in non-human primates.\textsuperscript{143} \textsc{osprey} contains a wide array of flexibility modeling options and provably accurate design algorithms,\textsuperscript{41,49} allowing LUTE to be used for many types of designs.

By presenting LUTE, this paper makes the following contributions:

1. A method to represent continuous flexibility and general energy functions to arbitrary accuracy in a local unpruned tuple expansion (LUTE) that can be used directly as input to discrete combinatorial search algorithms like DEE/A*.

2. A free implementation of LUTE in our laboratory’s open-source \textsc{osprey} protein-design software package,\textsuperscript{13,36,48,49} available for download\textsuperscript{49} upon publication as free software,\textsuperscript{49} supporting representation of both continuous sidechain and backbone flexibility and of molecular-mechanics and Poisson-Boltzmann energy functions.

3. Integration of LUTE with the DEE/A*, \textsuperscript{99} iMinDEE,\textsuperscript{40} BWM*, \textsuperscript{83} and \textsc{comets}\textsuperscript{61} algorithms for sequence and conformational search.

181
4. Bounds on the time and space complexity of protein design calculations that model continuous flexibility and/or use energy functions with non-local terms. The time and space complexity are exponential merely in the branch-width $w$ of the residue interaction graph, and thus the designs can be done in polynomial time for systems whose branch-width is bounded by a constant.

5. Experimental results for 47 computational design calculations on 36 protein structures using LUTE, which demonstrate its accuracy and efficiency in single-state designs, multistate designs and for both $n$-body Poisson-Boltzmann and pairwise energy functions.

7.2 Methods

The basic strategy of LUTE is to create a discrete, quick-to-evaluate energy matrix that tells us everything we need to know for design purposes about the continuous energy landscape of a protein. Section 7.2.1 describes this energy matrix and why it works. Section 7.2.2 explains how it is calculated. Section 7.2.3 describes what terms need to be included in the energy matrix. Each term corresponds to a tuple of RCs at different positions. Section 7.2.4 describes how LUTE is used in search and how it is compatible with a variety of different algorithms, allowing it to facilitate many types of protein design calculations. Section 7.2.5 provides further details of our implementation of LUTE combined with the iMinDEE algorithm for sequence and conformational search. Finally, Section 7.2.6 presents novel pruning algorithms that we have developed to solve some of the larger continuously flexible designs that LUTE has made possible.

7.2.1 The discrete expansion

Our goals in protein design (both GMEC and binding/partition function calculations) can be posed in terms of a discrete function $E(r)$ that maps an ordered
list \( \mathbf{r} \) of RCs to an energy. The list \( \mathbf{r} \) contains exactly one RC per residue and thus represents a voxel \( V(\mathbf{r}) \) in conformation space, where a vector \( \mathbf{x} \) of sequence and conformational degrees of freedom satisfies \( \mathbf{x} \in V(\mathbf{r}) \) if the degree-of-freedom bounds defined by each RC in \( \mathbf{r} \) are respected by every degree of freedom in \( \mathbf{x} \). The conformational degrees of freedom in \( \mathbf{x} \) will generally be continuous internal coordinates, e.g., sidechain dihedrals. We let \( E'(\mathbf{x}) \) denote the energy of the protein system, as a function of all its degrees of freedom.

For calculation of the GMEC energy \( E_g \), we wish to minimize \( E'(\mathbf{x}) \) with respect to \( \mathbf{x} \). Letting \( R \) be the set of all possible voxels, the domain over which we minimize is a finite union of voxels \( \bigcup_{\mathbf{r} \in R} V(\mathbf{r}) \):

\[
E_g = \min_{\mathbf{x} \in \bigcup_{\mathbf{r} \in R} V(\mathbf{r})} E'(\mathbf{x}) = \min_{\mathbf{r} \in R} \min_{\mathbf{x} \in V(\mathbf{r})} E'(\mathbf{x}),
\]

which can be expressed in the form \( \min_{\mathbf{r} \in R} E(\mathbf{r}) \) where

\[
E(\mathbf{r}) = \min_{\mathbf{x} \in V(\mathbf{r})} E'(\mathbf{x}).
\]

Similarly, partition function calculations seek to calculate the partition function

\[
q = \int_{\bigcup_{\mathbf{r} \in R} V(\mathbf{r})} \exp \left( -\frac{E'(\mathbf{x})}{RT} \right) d\mathbf{x} = \sum_{\mathbf{r} \in R} \int_{V(\mathbf{r})} \exp \left( -\frac{E'(\mathbf{x})}{RT} \right) d\mathbf{x}
\]

where \( R \) is the gas constant and \( T \) is the temperature. Letting

\[
E(\mathbf{r}) = -RT \ln \left( \int_{V(\mathbf{r})} \exp \left( -\frac{E'(\mathbf{x})}{RT} \right) d\mathbf{x} \right)
\]

we have a formulation of \( q \) in terms of the discrete free energy function \( E(\mathbf{r}) \):
\[ q = \sum_{r \in R} \exp \left( -\frac{E(r)}{RT} \right) \] (7.5)

Alternately, if we use the definition Eq. (7.2) to define \( E(r) \), then Eq. (7.5) gives us the approximation used in Refs. 108 and 48 for the partition function.

**Figure 7.1:** LUTE makes continuously flexible design efficient by representing continuous flexibility using local, discrete energy terms. (A) Protein design with discrete flexibility searches over a discrete (albeit large) conformational space (“Conf”), looking for low-energy (“\( E \)”) conformations. Highly efficient algorithms like DEE/A* are available for this problem. (B) Protein design with continuous flexibility must search over a large space of voxels (blue) in a continuous conformational space, but we are usually interested only in the minimum-energy point of each voxel. We thus want a way to search combinatorially over these minimum-energy points. (C) The minimized energy of a voxel in protein conformational space depends on all rotamers in the voxel (arrow 1). But we can expand this minimized energy as a sum of local contributions from low-order tuples (e.g., pairs) of residues (arrows 2, 3). (Minimized conformations shown in red, ideal rotamers in blue). (D) This expansion, known as LUTE, gives us a discrete combinatorial search problem of the same form as protein design with discrete flexibility (arrows 4, 5). But this new discrete problem searches over the minimum-energy points (red) of voxels in continuous conformational space (blue). We can solve this problem very efficiently. Figure shows Leu 29, Leu 51, Phe 55, and Lys 59 of the Atx1 metallochaperone (PDB id 1CC8\textsuperscript{142}).

Because \( r \) is a discrete variable, the energy \( E(r) \) can be decomposed as a sum
of energies associated with tuples of RCs (Fig. 7.1). If all the RCs in a tuple are in the list \( r \), then that tuple’s energy will contribute to \( E(r) \). Most higher-order tuples of RCs consist of residues too far apart to have higher-order interactions, and thus do not contribute significantly to the energy (see Section 7.1 and Ref. 64). We can reduce the number of tuples needed substantially further if we only try to represent favorable, non-clashing conformations. By eliminating high-energy conformations, this restriction of conformational space greatly reduces the range of energy values over which \( E(r) \) must be accurate. To achieve this, we prune tuples that cannot be part of favorable conformations, and consider only conformations whose tuples are all unpruned. Our expansion is much more efficient to compute after provably unfavorable tuples are pruned. Hence, we are able to represent the energy \( E(r) \) as a local unpruned tuple expansion, or LUTE.

Let us consider a conformational space with continuous and discrete degrees of freedom, consisting of RCs, and a mapping \( E(r) \) that we can readily calculate. For example, in a typical continuously flexible design, \( E(r) \) is defined by Eq. (7.2), which we assume can be calculated by local minimization. Suppose we have a set \( T \) of tuples of RCs at different residue positions. \( T \) can contain pairs but also may contain triples, etc. We then define our local unpruned tuple expansion as a mapping \( m : T \rightarrow \mathbb{R} \cup \{ \perp \} \). \( m \) defines a real coefficient for each tuple \( t \in T \), except for pruned tuples, for which \( m(t) = \perp \). Let \( T_r \) denote the set of tuples in \( T \) that consist only of RCs in \( r \). For example, if \( T \) is the set of all possible RC pairs, then \( T_r \) will consists of all pairs of RCs in the list of RCs \( r \). Then LUTE predicts \( r \) to be a pruned conformation if \( m(t) = \perp \) for any \( t \in T_r \), and otherwise it predicts \( E(r) = \sum_{t \in T_r} m(t) \). We refer to the data structure representing the mapping \( m \) as the \textit{LUTE energy matrix}. We call it an energy matrix because it takes a form similar to that of traditional pairwise energy matrices,\textsuperscript{25,40,68,99} although it contains
significantly different numerical values when computed for the same design system.

The limiting behavior of LUTE is favorable. As we expand the set \( T \), we must eventually approach perfect accuracy, because if \( T \) is the set of all tuples of RCs at different positions, then \( m \) can represent \( E(r) \) for each full RC list \( r \) explicitly.

If we assume locality of \( E(r) \) (see Section 7.1), we can expect inaccuracies to diminish fairly quickly with increasing size of \( T \), because the component of \( E(r) \) modeling the interactions of a residue \( i \) will depend only on the RCs assigned to residues fairly close in space to \( i \). As a result, we expect a relatively compact LUTE expansion for any practical protein design problem. In practice, expansions in pairs and triples have worked well (see Section 7.3).

### 7.2.2 Calculation of expansion by least squares

The energy (as represented by LUTE) of a conformation \( r \) is a linear function of all the unpruned tuple energies \( m(t) \), given by \( \sum_{t \in T_r} m(t) = \sum_{t \in T} m(t) I_{t \in T_r} \), where the indicator function \( I_c \) is 1 if \( c \) is true and 0 otherwise. Thus, once we have decided which tuples to include in \( T \), we can use least squares to obtain the energy \( m(t) \) for each tuple \( t \in T \).

We draw two sample sets for each fit—one for training and one for cross-validation. For each \( r \) in either of the sample sets, we evaluate the ground-truth \( E(r) \) as described in Eq. (7.2) using local minimization. Each sample set is chosen such that each unpruned tuple appears in at least 10 samples. Furthermore, no sample is allowed to contain any pruned tuples. To achieve this, we iterate through all the tuples. If the tuple appears in less than 10 samples, then we draw enough samples to cover the difference. To draw a sample containing a tuple \( t \), we first assign the RCs in \( t \) to their respective residues. Then we iterate through each remaining position in random order, picking a random RC that does not introduce any pruned tuples into the sample. Because pairs and triples can be pruned, it is possible for this sampling
process to reach a point where no RCs are available for a given position. Thus, if the first 5 random sampling attempts are unsuccessful, then we switch to depth-first search (with randomized ordering of RCs) to search for a sample. This way, if there is a sample containing $t$, we are guaranteed to find it, and if not, then $t$ can be pruned.

The least-squares matrix for large problems is too big to allow efficient direct solution. Hence, we use conjugate gradient\textsuperscript{124} on the least-squares normal equations, which is accurate and very efficient, especially since the least-squares matrix is sparse.

7.2.3 Selection of tuples to include

For many designs, a pairwise expansion suffices—i.e., $T$ is the set of RC pairs $(i_r,j_s)$ where $i > j$. The accuracy of this pairwise expansion can be enhanced by first performing triples pruning—in this case, if a triple $t$ is pruned, then we include $t$ in $T$ and set $m(t) = \bot$, but all $t' \in T$ such that $m(t') \neq \bot$ are pairs. However, sometimes higher accuracy is needed; we thus select additional, unpruned triples to add to $T$. To exploit locality, the triples are chosen at residue positions that have strong pairwise interactions. First, the interaction between each residue pair is quantified by maximizing the absolute value of the minimized pairwise interaction energy over all RC pairs at the residue pair. Then, for each residue position $i$, the 2 residues $j$ and $k$ interacting most strongly with $i$ are identified (a higher number than 2 can be used if indicated by the least-squares residual). The residue pairs $(i,j)$ and $(i,k)$ are heuristically deemed “strongly interacting” pairs, and any residue triple containing at least two strongly interacting pairs is deemed a strongly interacting residue triple. Finally, all RC triples at strongly interacting residue triples are added to $T$. Though LUTE can support higher-order tuples than triples, we have not found this to be necessary so far.
Most algorithms for protein design with discrete rotamers take a matrix of pairwise energies as input. By simply substituting a LUTE energy matrix for this pairwise energy matrix, we can convert any of these algorithms into an equally efficient design algorithm that searches a continuous search space instead of a discrete one, and/or that optimizes a non-pairwise energy function instead of a pairwise one. The LUTE energy matrix is computed once, before the search, which takes only polynomial time in the number of residues. For example, we need quadratic time to compute a LUTE matrix for which \( T \) is all pairs of RCs.

If we know which RCs to prune, then the computation by least-squares of the LUTE matrix does not depend on the type of search problem we are solving (it can be single-state or multistate, and GMEC-based or ensemble-based). However, the most efficient pruning algorithm for RCs in a continuous search space, iMinDEE, requires as input a pruning interval in addition to a pairwise lower-bound energy matrix. The pruning interval is an upper bound on the gap between the optimal conformation and the lowest lower bound on a conformational energy in our search space, computed based on lower bounds on pairwise energies. The pruning is more efficient if this upper bound is relatively tight, and there are a few ways to obtain a valid but relatively tight bound. We will now discuss how this can be done for either single-state or multistate protein designs.

For single-state designs that aim to minimize an energy function with respect to sequence and (continuous) conformation, the method described in Ref. 40 to obtain a pruning interval is appropriate.

For multistate designs, two algorithms that are highly compatible with LUTE are \( K^* \) and COMETS, and each of these algorithms offers an effective way to compute a pruning interval. \( K^* \), a partition function-based binding optimization al-
gorithm, provides methods to check if too many RCs have been pruned so that some can be unpruned if needed. This unpruning can be done by enlarging the pruning interval. COMETS optimizes, with respect to sequence, linear combinations of conformationally optimized energies of several protein states (bound states with different ligands, unbound states, etc.) Pruning is performed separately for each state, using type-dependent dead-end elimination. Generally, for purposes of ensuring stability of each state, we will apply a constraint in COMETS calculations demanding that the optimized conformation energy for our optimized sequence be below some threshold. This threshold is our upper bound on the optimized energy for each state. Likewise, for each state, we can use standard protein design algorithms like DEE/A* to compute a lower bound based on pairwise energies for the optimal conformation over all sequences. Putting these bounds together, we obtain a pruning interval for each state.

Using these techniques, we can obtain a valid but relatively tight pruning interval and thus an appropriate LUTE energy matrix for many kinds of single-state and multistate designs (e.g., see Figs. 7.4-7.7).

7.2.5 iMinDEE with LUTE

iMinDEE calculates the GMEC for a protein system, and can also enumerate an ensemble of the lowest-energy conformations of a protein in gap-free ascending order of minimized conformational energy. We have implemented iMinDEE to run along with LUTE as part of the OSPREY open-source protein design software package. As in previous versions of iMinDEE, we estimate a pruning interval, prune based on that interval, calculate the GMEC, and then repeat with a larger interval if needed (at most one repeat is needed; see Ref. 40 for a proof). Thus, the protocol consists of the following steps:

1. Precalculate the matrix of energy lower bounds.
2. Prune using iMinDEE.\textsuperscript{40}

3. Compute an EPIC matrix.\textsuperscript{64}

4. Prune using EPIC (see Section 7.2.6).

5. Perform LUTE fitting (see Section 7.2.2).

6. Prune based on the LUTE matrix, using rigid DEE,\textsuperscript{25} since the LUTE matrix gives an accurate discrete expansion of the energy and thus admits pruning without the iMinDEE pruning interval.

7. Calculate the lower pairwise lower-bound. This is just like enumerating the first conformation from A* in iMinDEE. Though this step is in general NP-hard, very effective algorithms are available for it, as we have shown in Ref. 140.

8. Compute the GMEC using rigid A*.\textsuperscript{99}

9. If needed, repeat the process (from step 2 onward) using a higher iMinDEE pruning interval; see Ref. 40.

So if we are calculating the GMEC, then the only NP-hard steps here are the pairwise lower-bound calculation and the GMEC calculation on the LUTE matrix, each of which are of similar complexity to enumerating the first conformation from iMinDEE, and comparable to computing a rigid GMEC. Thus, for large designs, the entire LUTE-based minimized GMEC (minGMEC) calculation is of similar complexity to enumerating the first conformation in traditional iMinDEE, or simply to performing a rigid GMEC calculation.

7.2.6 Pruning enhancements

In order to complete some of the larger designs in Section 7.3, we found it necessary to enhance our pruning capabilities somewhat compared to previous work. Pruning
consisted of Goldstein singles and pairs, and triples at selected, highly interacting triples of residues (using the triple selection criteria in Section 7.2.3). In addition to this triple selection, we introduced two novel pruning enhancements, which build on iMinDEE: a “competitor pruning” technique that speeds up pruning substantially with minimal loss of pruning power, and a “continuous pruning” technique that uses bounds on minimized energies of partial conformations to prune RCs and tuples of RCs that would otherwise require a much tighter iMinDEE pruning interval to prune.

**Competitor pruning**

DEE and related pruning algorithms can prune an RC or tuple of RCs by comparing its energies to those of a “competitor”, which is another RC at the same residue (or tuple at the same tuple of residues). Previous implementations of these algorithms either allowed any unpruned RC or tuple to act as a competitor, or they only allowed a few “magic bullets,” which can reduce pruning power significantly. For some of the larger designs, pruning was the bottleneck, so it was important to have a method to speed up pruning without sacrificing much pruning power. We found that using a reduced set of competitors—the set of RCs that are unpruned using the iMinDEE pruning condition with iMinDEE interval 0—results in such a speedup. To use this set, we begin pruning with a “competitor pruning” procedure, in which we perform pruning at iMinDEE interval 0. However, this pruning does not actually remove RCs and tuples from consideration. Rather, it removes them from the set of competitors that will be used in the the actual pruning step (Fig. 7.2).

Competitor pruning is fast because pruning with a 0 interval allows a large amount of single-RC pruning, so we spend much less time iterating through pairs. The set of competitors chosen this way is also typically much smaller than the full set of unpruned RC tuples. Hence, the actual pruning step is sped up significantly by using competitor pruning first. We have observed pruning power to be reduced
very little when using competitor pruning, and the following analysis suggest why this is the case.

Suppose we are performing Goldstein pruning with a fixed set of witnesses. Then any RC tuple $t_1$ that can be pruned at all can be pruned using a competitor $t_2$ such that $t_2$ cannot be pruned with iMinDEE interval 0. We can show this by taking note of two facts. (1) Pruning with iMinDEE interval 0 is transitive with a fixed set of witnesses. That is, if tuple $t_1$ can be pruned using competitor $t_2$, and $t_2$ can be pruned using competitor $t_3$, then $t_1$ can be pruned using competitor $t_3$. (2) If an RC tuple $t_1$ can be pruned using competitor $t_2$ at an iMinDEE interval $I > 0$, then $t_1$ can be pruned using $t_2$ at iMinDEE interval 0. Now, suppose that $t_1$ can be pruned using $t_2$ at an iMinDEE interval $I$. There are two cases: $t_2$ can be pruned at iMinDEE interval 0, or it cannot. If it can be pruned using a tuple $t_3$ at pruning interval 0, then $t_1$ can be pruned using $t_3$ at iMinDEE interval $I$ (using notes 1 and 2). Recursively applying the same argument to $t_3$, we can be certain that $t_1$ can be pruned using a competitor that cannot be pruned with iMinDEE interval 0. In practice, a pruning cycle removes witnesses at each iteration, so the condition about a fixed set of witnesses does not strictly hold. But we have still observed pruning power to be reduced very little when using competitor pruning.
Figure 7.2: Competitor pruning reduces the set of RCs we need to consider as competitors during iMinDEE pruning. In iMinDEE pruning, we take a “candidate” RC (or RC pair or triple) \( i_r \) and compare it to another RC (or pair or triple) \( i_t \) by evaluating the pruning checksum

\[
E_{\Theta}(i_r) - E_{\Theta}(i_t) + \sum_{j \neq i} \min_s E_{\Theta}(i_r, j_s) - E_{\Theta}(i_t, j_s) > I
\]

Do pruning

Without competitor pruning

All possible RCs

Candidates

Competitors

\( E_{\Theta}(i_r) - E_{\Theta}(i_t) + \sum_{j \neq i} \min_s E_{\Theta}(i_r, j_s) - E_{\Theta}(i_t, j_s) > I \)

Do pruning

With competitor pruning

All possible RCs

Competitor RCs

Candidates

Competitors

\( E_{\Theta}(i_r) - E_{\Theta}(i_t) + \sum_{j \neq i} \min_s E_{\Theta}(i_r, j_s) - E_{\Theta}(i_t, j_s) > I \)

Do pruning

Continuous pruning

Another issue that can reduce pruning efficiency in iMinDEE is the fact that the lower bound on a candidate RC tuple \( r \) may not be very tight. We can tighten it by including the minimized partial conformational energy for the partial conformation consisting of the RCs in \( r \), as in the A* step for EPIC. Suppose we have a
lower bound for the energy of a partial conformation \( p \) based on pairwise energies. As discussed in Ref. 64, this bound remains valid if we add to it the minimized sum \( c \) of EPIC terms for all intra+shell and pairwise polynomials that represent RCs or pairs in \( p \). iMinDEE is also based on this lower bound (the partial conformation here is the tuple we are trying to prune), so we can add \( c \) to the iMinDEE checksum (the left-hand side of the pruning condition) and still have a valid pruning condition with the same pruning interval \( I \). \( c \) is always nonnegative, so continuous pruning is always at least as good as regular iMinDEE. \( c = 0 \) if the partial conformation is just a single RC, but may be quite large if the partial conformation contains RCs that do not pack together well, which is exactly when we want to prune. Effectively, continuous pruning allows us to group together terms in our lower bound on the conformational energy, so that the bound is tighter (Fig. 7.3).
Figure 7.3: Continuous pruning increases pruning power by using a tighter lower bound on conformational energy. iMinDEE pruning is based on lower bounds on conformational energy, which are constructed by adding up lower bounds on pairwise interaction energies (red arrows) and one-body energies (circles). Without continuous pruning (left), each flexible residue and each flexible residue pair has its lower-bound energy computed separately. But with continuous pruning (right), if we add the continuous contribution $c$ for a pair of RCs, then we effectively replace the three lower-bound terms for that pair (the two one-body energies and the pairwise interaction energy) with a single lower bound on the energy of the pair. This tighter bound can significantly increase pruning power, because if a residue pair has an unavoidable clash, the lower bound on the pair’s internal energy is likely to show this even if each of the three constituent energies is individually capable of continuously minimizing to a better value.

7.3 Results

We present here complexity results and computational experiments regarding the performance of LUTE. In Section 7.3.1, we show that the combination of LUTE with the BWM* search algorithm is guaranteed to solve continuously flexible protein designs in polynomial time given a residue interaction graph with branch-width bounded by a constant. In Sections 7.3.2 and 7.3.3, we present 30 single-state and 17 multistate protein design calculations using LUTE. We measure the gains in efficiency provided by LUTE and its ability to accurately and efficiently perform calculations that, due to their large amount of continuous flexibility (Section 7.3.2) or non-pairwise energy function (Section 7.3.3), are inaccessible to previous algorithms. These results include designs with both continuous sidechain and backbone flexibility. Sidechain
dihedrals were allowed $9^\circ$ of continuous motion in either direction relative to the modal value for each sidechain rotamer,\textsuperscript{81} while backbone flexibility (when present) was modeled as in Ref. 68.

### 7.3.1 Polynomial-time protein design with continuous flexibility

**Figure 7.4:** LUTE markedly reduces the cost of continuously flexible conformational search. Ratios (without LUTE:with LUTE) of the number of nodes in the A* tree before enumeration of the GMEC (or of the last conformation if several conformations closely spaced in energy were calculated; see Ref. 64), versus number of flexible residues. A 20-residue sidechain placement with node ratio $2 \times 10^5$ is not shown because it would break the scale.

Protein design in the general case is NP-hard.\textsuperscript{12,131} In practice, however, many designs exhibit special properties that make them more tractable. For example, the residue interaction graph—the graph whose edges encode nonnegligible interactions between pairs of residues—of practical designs often has low branch-width. It has been previously shown that protein design with discrete rotamers can be performed in asymptotic time exponential only in the branch-width\textsuperscript{83} $w$. Furthermore, these branch-widths can be small irrespective of the number of mutable residues.\textsuperscript{83} Thus, for many protein designs with discrete rotamers the corresponding GMEC can be found in polynomial time. If one substitutes the LUTE matrix for the discrete pairwise energy matrix in this complexity result, then design with continuous flexibility
and a constant-bounded branch-width can be solved in polynomial time as well. We
will make this rigorous using Theorem 7.3.1. In this theorem, a LUTE energy func-
tion is a function $E(r) = \sum_{t \in T_r} m(t)$, where $m : T \rightarrow \mathbb{R} \cup \{\perp\}$ maps RC tuples to
real coefficients (for this purpose, the coefficient $\perp$ of a pruned tuple is effectively
$\infty$). Let $n$ be the number of mutable residues, $q$ be the maximum number of allowed
RCs at any mutable residue position, and $\beta_t$ and $\beta_s$ be the time and space costs
(respectively) to compute the branch-decomposition. This theorem establishes the
complexity both of GMEC calculations and of enumeration of subsequent conforma-
tions in gap-free ascending order of energy. The latter is essential for calculation
of partition functions, which can be used to account for entropy in predictions of
binding.\textsuperscript{48,83,108} Let us now derive and prove this theorem.

\textit{High-level intuition.} The theoretical guarantees of BWM*\textsuperscript{83} are unchanged across
different discrete formulations of the protein design problem. Given a sparse energy
function, the corresponding sparse graph and branch-decomposition can be com-
puted. BWM* computes the GMEC of the corresponding energy function in time
exponential merely in $w$, the branch-width of the branch-decomposition. The time
cost is polynomial in the number of residues in the system, $n$, and the maximum
number of RCs per residue, $q$.

To realistically describe a protein design system, a LUTE energy function need
only have nonzero coefficients for tuples whose interactions are nonnegligible, and
thus we can compute a sparse residue interaction graph for it (Lemma 7.3.1). As such,
when a LUTE energy matrix whose residue interaction graph has branch-width $w$
is provided as input to the protein design problem, BWM* computes the minimized
GMEC in time exponential in $w$, enumerates a gap-free list of conformations in
$O(n \log q)$ additional time, and computes a gap-free ensemble that is guaranteed to
contain the minGMEC.

**Lemmas and theorem.** For an \( n \)-residue protein design with at most \( q \) rotamers at each position, let \( E \) be a LUTE energy function whose tuples have arity at most \( k \). We can then define the residue interaction graph of the LUTE energy function as follows:

**Definition 1.** Let \( E(r) = \sum_{r \in T_r} m(r) \) be a LUTE energy function, where \( m : T \rightarrow \mathbb{R} \cup \{\bot\} \) maps RC tuples to real coefficients. The sparse residue interaction graph of \( E(r) \) is defined to be a hypergraph \( G' = (V, T) \), where \( V \) is the set of all mutable residues.

**Lemma 7.3.1.** For a given LUTE energy function \( E(r) \) with finite tuple arity \( k \), \( G' \) can be computed in \( \mathcal{O}(n^k) \) time.

*Proof.* There are at most \( \binom{n}{k} \) tuples, corresponding to at most \( \binom{n}{k} = \mathcal{O}(n^k) \) edges in \( G' \).

**Lemma 7.3.2** (Given in Ref. 83). Given a branch-decomposition with branch-width \( w \), BWM* computes the GMEC of the sparse energy function in \( \mathcal{O}(nw^2q^{3w}) \) time, and enumerates each additional conformation in order of increasing sparse energy in \( \mathcal{O}(n \log q) \) time.

**Theorem 7.3.1.** For a LUTE energy function whose residue interaction graph has branch-width \( w \), the GMEC can be computed in \( \mathcal{O}(nw^2q^{3w} + \beta) \) time and \( \mathcal{O}(nwq^{3w}) \) space, and each additional conformation can be enumerated in order of LUTE energy in \( \mathcal{O}(n \log q) \) time and \( \mathcal{O}(n) \) space.

*Proof.* This follows from Lemmas 7.3.1 and 7.3.2.
LUTE single-state designs were run on 23 protein design systems from Ref. 64 with 4-16 mutable residues, as well as five larger systems (17-40 mutable residues), to measure the efficiency of LUTE and to observe the behavior of LUTE on the larger systems. Many of these larger systems are intractable by previous methods (except post-hoc minimization methods that do not account for continuous flexibility during search). The results show that the discrete DEE/A* search with LUTE is dramatically more efficient even compared to EPIC, which offers previously state-of-the-art efficiency for continuously flexible design (Fig. 7.4). They also demonstrate that LUTE can handle very large continuously flexible designs—including a 40-residue sidechain placement, which covers a large fraction of the residues in the Atx1 metallochaperone (Fig. 7.6, left), and a 20-residue design on the same structure with 5 amino-acid types allowed at every position. Furthermore, the LUTE energy matrix consistently represented the true energy landscape very closely (Fig. 7.5). Optimal sequences and conformations with LUTE differed significantly from the same designs run without continuous flexibility: the same top conformation was returned in only 2 of the 28 single-state designs. On average, 31% of the RCs in the optimal conformations differed from each other. This is consistent with previous work showing that protein design calculations with and without continuous flexibility differ significantly in their results.40,68

For many systems, LUTE achieved a fit with residual under 0.01 (kcal/mol)² with only a pairwise expansion. In cases when the pairwise expansion’s residual was higher, an expansion in sparse triples was performed instead. In all but one case, the triples expansion’s residual was less than thermal energy at room temperature (0.59 kcal/mol, i.e., 0.35 (kcal/mol)²), and thus deemed insignificant.

The one outlier case was a 14-residue design on ponsin (PDB id: 2O9S). It exhib-
Figure 7.5: **LUTE accurately represents continuously minimized energies.** Residuals for LUTE ((kcal/mol)$^2$) on the cross-validation data set, measuring the difference between the EPIC energy and a pairwise expansion (blue) or one with sparse triples (red; computed only if pairwise residual exceeded 0.01). *x* axis: number of flexible residues. Inset: All the same data plotted on a linear scale.

...ited significant local minimization errors, which caused even the matrix of pairwise lower-bound energies (computed before LUTE precomputation begins) to have errors of at least $\sim$10 kcal/mol. These errors indicate the failure of either our local minimizer or our assumption that local minimization suffices within RCs. As a result of these errors, the LUTE residual even with triples was 1.9 kcal/mol for this system, seven times worst than the next worst residual (the 40-residue Atx1 design). Our software now detects this problem and warns the user before the LUTE computation begins.

17 multistate protein designs were also performed, using a combination of LUTE with our COMETS$^{61}$ multistate protein design algorithm (see Section 7.2.4). The systems from these designs were taken from Ref. 61; details are provided in Section 7.4. The same designs were run with and without continuous flexibility, with LUTE used in the continuous case. As discussed in Ref. 61, COMETS provably returns the same results as exhaustive search over sequences, but it provides a speedup compared to that exhaustive search by (a) considering only a portion of the sequences in the search space explicitly, and (b) only performing a full conformational optimization for...
Figure 7.6: LUTE enables very large provably accurate protein designs with continuous flexibility (left) and with Poisson-Boltzmann energy functions (right). Left: previously, protein designs with continuous flexibility only finished when performed with significantly fewer flexible residues, compared to designs with discrete rotamers. Even 20-residue designs were often intractable. But LUTE solved a sidechain placement problem with continuous flexibility in which 40 residues (purple) were made flexible in the Atx1 metallochaperone (PDB id 1CC8\textsuperscript{142}). Right: previous designs using the Poisson-Boltzmann energy function could not optimize this function directly, but only used Poisson-Boltzmann energies to rerank top hits from optimization of a simpler, pairwise energy function. But LUTE can optimize the Poisson-Boltzmann energy function directly—e.g., in a sidechain placement of 20 residues (purple) of Atx1.

a small portion of the sequences in (a). However, previously\textsuperscript{61} (a) was only significant in designs without continuous flexibility, and (b) was much more pronounced without continuous flexibility. LUTE brings continuously flexible COMETS designs up to speed with discrete designs on the same system (Fig. 7.7).

7.3.3 Designs that provably optimize Poisson-Boltzmann energies

We also ran LUTE conformational optimization calculations on two proteins using the Poisson-Boltzmann energy function, which is non-pairwise. This energy was evaluated using Delphi\textsuperscript{123,141} in place of the pairwise EEF1\textsuperscript{97} solvation energy that is used by default in OSPREY. Interestingly, triple energies did not provide significant benefit here, but LUTE was found to describe the Poisson-Boltzmann energy landscape with a high degree of accuracy. Previous work has shown that an accurate pairwise representation can be obtained for Poisson-Boltzmann energies of discrete,
**Figure 7.7:** LUTE brings rigid-rotamer-like efficiency to provably accurate multistate designs with continuous flexibility. Top: The number of sequences considered explicitly in continuously flexible designs using LUTE with COMETS (blue) is similar to the number considered in discrete COMETS designs on the same system (red) and far less than the total number of sequences in the search (green). Bottom: In the same designs, the number of full conformational optimizations needed is also similar for the continuous (LUTE, blue) and discrete (red) designs. Usually, only the optimal sequence (or top 5 if enumerating 5 sequences) required full conformational optimization, while exhaustive search must fully conformationally optimize every sequence in every state in the multistate design (green). Design test cases taken from Ref. 61. * denotes enumeration of the top 5 sequences. Missing discrete calculations mean that discrete search was unable to find a non-clashing conformation for the wild-type protein, or that 5 sequences satisfying the design constraints were not available in the discrete search space.

Rigid rotamers, but our LUTE results show that a very accurate representation of continuously minimized Poisson-Boltzmann energies is possible as well. With continuous flexibility, a 6-residue sidechain placement on the unliganded TIR1/IAA7 complex (PDB code 2P1Q) with continuous flexibility achieved a total residual of $6 \times 10^{-4}$ and took about 4 days. Furthermore, a 20-residue sidechain placement without continuous flexibility on the bacterial metallochaperone protein Atx1 (PDB code 1CC8; Fig. 7.6, right) was solved in 2.5 hours, with total residual 0.04 (kcal/mol).

Unlike previous protein design calculations that use Poisson-Boltzmann energies, our new calculations provably return the minimum of the (LUTE-approximated) Poisson-Boltzmann energy over the entire conformational space, rather than simply over a set of top hits from an initial search that used a cheaper energy function.
7.4 Details of protein design runs
Table 7.1: **Single-state protein design test cases.** Each design was run with continuous sidechain flexibility, and some also included continuous backbone flexibility. Designs were performed with and without LUTE, using EPIC\(^{64}\) in both cases. Designs not finishing without LUTE (after three weeks of computation time) are marked DNF. \(n_L\) and \(n_n\) denote the number of nodes in the A* tree after enumeration of the GMEC (or of the last conformation if several conformations closely spaced in energy were calculated; see Ref. 64), with and without LUTE respectively. The residual of the pairwise LUTE least-squares fit is denoted \(r_p\). The final least-squares fit residual is denoted \(r_f\). \(r_f = r_p\) for designs with \(r_p < 0.01\), but otherwise we computed a LUTE matrix with sparse triples and report the residual from the triples fit as \(r_f\). These residuals are reported for the second iteration of iMinDEE, if two iterations were required (iMinDEE always finishes after at most two iterations).\(^{40}\) The residual from the first round of iMinDEE is typically much lower because of the greater amount of pruning in the first iteration. Table continues on next page.

<table>
<thead>
<tr>
<th>Protein name</th>
<th>PDB code</th>
<th>Residue count</th>
<th>Backbone flexibility</th>
<th>(n_L)</th>
<th>(n_n)</th>
<th>(r_p)</th>
<th>(r_f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>7</td>
<td>Y</td>
<td>42</td>
<td>751</td>
<td>0.0008</td>
<td>0.0008</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>9</td>
<td>Y</td>
<td>49</td>
<td>402</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>12</td>
<td>Y</td>
<td>46182</td>
<td>2149464</td>
<td>1.03</td>
<td>0.18</td>
</tr>
<tr>
<td>Cytochrome c553</td>
<td>1c75</td>
<td>6</td>
<td>Y</td>
<td>3</td>
<td>893</td>
<td>0.0004</td>
<td>0.0004</td>
</tr>
<tr>
<td>Atx1 metallochaperone</td>
<td>1cc8</td>
<td>7</td>
<td>Y</td>
<td>141</td>
<td>10816</td>
<td>0.05</td>
<td>0.006</td>
</tr>
<tr>
<td>Bucandin</td>
<td>1f94</td>
<td>7</td>
<td>Y</td>
<td>0</td>
<td>167</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>Nonspecific lipid-transfer protein</td>
<td>1fk5</td>
<td>6</td>
<td>Y</td>
<td>0</td>
<td>32</td>
<td>(3\times10^{-5})</td>
<td>(3\times10^{-5})</td>
</tr>
<tr>
<td>Transcription factor II F</td>
<td>1i27</td>
<td>7</td>
<td>Y</td>
<td>137</td>
<td>28148</td>
<td>0.07</td>
<td>0.02</td>
</tr>
<tr>
<td>Ferredoxin</td>
<td>1iqz</td>
<td>9</td>
<td>Y</td>
<td>21</td>
<td>1550</td>
<td>0.007</td>
<td>0.007</td>
</tr>
<tr>
<td>Trp repressor</td>
<td>1jhg</td>
<td>7</td>
<td>Y</td>
<td>221</td>
<td>26079</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td>Fructose-6-phosphate aldolase</td>
<td>1l6w</td>
<td>6</td>
<td>Y</td>
<td>0</td>
<td>132</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>PA-I lectin</td>
<td>1l7l</td>
<td>6</td>
<td>Y</td>
<td>0</td>
<td>35</td>
<td>(5\times10^{-10})</td>
<td>(5\times10^{-10})</td>
</tr>
<tr>
<td>Phosphoserine phosphatase</td>
<td>1l7m</td>
<td>7</td>
<td>Y</td>
<td>0</td>
<td>445</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>alpha-D-glucuronidase</td>
<td>1l8n</td>
<td>5</td>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>(7\times10^{-29})</td>
<td>(7\times10^{-22})</td>
</tr>
<tr>
<td>Granulysin</td>
<td>1l9l</td>
<td>7</td>
<td>Y</td>
<td>0</td>
<td>29</td>
<td>0.0006</td>
<td>0.0006</td>
</tr>
<tr>
<td>Ferritin</td>
<td>1lb3</td>
<td>5</td>
<td>Y</td>
<td>0</td>
<td>155</td>
<td>0.0006</td>
<td>0.0006</td>
</tr>
<tr>
<td>Cytochrome c</td>
<td>1m1q</td>
<td>8</td>
<td>Y</td>
<td>0</td>
<td>1197</td>
<td>0.0004</td>
<td>0.0004</td>
</tr>
<tr>
<td>Hypothetical protein YciI</td>
<td>1mwq</td>
<td>8</td>
<td>Y</td>
<td>0</td>
<td>274</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>Ponsin</td>
<td>2o9s</td>
<td>14</td>
<td>N</td>
<td>205873</td>
<td>162750</td>
<td>4.52</td>
<td>1.93</td>
</tr>
<tr>
<td>Scytovirin</td>
<td>2qsk</td>
<td>10</td>
<td>N</td>
<td>0</td>
<td>10</td>
<td>(1\times10^{-11})</td>
<td>(1\times10^{-11})</td>
</tr>
<tr>
<td>Protein</td>
<td>ID</td>
<td>Length</td>
<td>Numb.</td>
<td>EC50</td>
<td>IC50</td>
<td>p-value 1</td>
<td>p-value 2</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Putative monooxygenase</td>
<td>2ril</td>
<td>8</td>
<td>N</td>
<td>71</td>
<td>8125</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>dpy-30-like protein</td>
<td>3g36</td>
<td>4</td>
<td>N</td>
<td>0</td>
<td>35</td>
<td>0.0008</td>
<td>0.0008</td>
</tr>
<tr>
<td>HIV gp120</td>
<td>3u7y</td>
<td>16</td>
<td>N</td>
<td>125</td>
<td>48935</td>
<td>3×10⁻⁵</td>
<td>3×10⁻⁵</td>
</tr>
<tr>
<td>Atx1 metallochaperone</td>
<td>1cc8</td>
<td>20</td>
<td>N</td>
<td>35670</td>
<td>DNF</td>
<td>0.10</td>
<td>0.003</td>
</tr>
<tr>
<td>Scorpion toxin</td>
<td>1aho</td>
<td>17</td>
<td>N</td>
<td>198857</td>
<td>DNF</td>
<td>0.80</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Note: EC50 and IC50 values are in nanomolar (nM) units, and p-values are reported as p < 0.05.
Table 7.2: **Multistate protein design test cases.** These designs systems were taken from Ref. 61 and were run using the COMETS multistate design algorithm. Each design was run both with continuous flexibility, using LUTE, and without continuous flexibility. Type is “aff” for designs for affinity, “stab” for designs for stability robust to force field choice, and “multi” for designs to be multispecific to two different complexes (the types of designs, and the COMETS algorithm in general, are described further in Ref. 61). In each design, $N$ is the number of sequences in the search space and $k$ is the number of sequences enumerated (we enumerate either the top sequence or the top 5). $m_L$ and $m_d$ are the numbers of sequences explicitly considered using LUTE and in discrete search respectively. Similarly, $g_L$ and $g_d$ are the numbers of full conformational optimizations performed using LUTE and in discrete search respectively. “WC” indicates that the wild-type protein was found to have a clash unavoidable by discrete conformational search, and “NF” indicates that the discrete search provably cannot find five sequences that satisfy the design constraints. Table continues on next page.

<table>
<thead>
<tr>
<th>Protein re-designed</th>
<th>Mutableresidues</th>
<th>PDB id(s)</th>
<th>Type</th>
<th>$N$</th>
<th>$k$</th>
<th>$m_L$</th>
<th>$m_d$</th>
<th>$g_L$</th>
<th>$g_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CED-4</td>
<td>1, 5, 227, 229, 259, 265, 279, 282</td>
<td>2a5y/3lqr</td>
<td>multi</td>
<td>288</td>
<td>1</td>
<td>28</td>
<td>17</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Rab-11A</td>
<td>44, 46, 47, 48, 50</td>
<td>2gzd/2gzh</td>
<td>multi</td>
<td>2744</td>
<td>1</td>
<td>269</td>
<td>106</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 584</td>
<td>3ktp/3ktr</td>
<td>multi</td>
<td>448</td>
<td>1</td>
<td>51</td>
<td>35</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CDO</td>
<td>867, 872, 874, 874, 901, 918</td>
<td>3n1f/3n1q</td>
<td>multi</td>
<td>112</td>
<td>1</td>
<td>10</td>
<td>WC</td>
<td>4</td>
<td>WC</td>
</tr>
<tr>
<td>Brother of CDO</td>
<td>753, 755, 756, 758, 760, 764, 789, 804</td>
<td>3n1g/3n1m</td>
<td>multi</td>
<td>224</td>
<td>1</td>
<td>35</td>
<td>WC</td>
<td>4</td>
<td>WC</td>
</tr>
<tr>
<td>Rab-11A</td>
<td>44, 46, 47, 48, 50</td>
<td>2gzd/2gzh</td>
<td>multi</td>
<td>2744</td>
<td>5</td>
<td>277</td>
<td>NF</td>
<td>20</td>
<td>NF</td>
</tr>
<tr>
<td>Polyadenylate-binding protein 1</td>
<td>564, 571, 580, 582, 584</td>
<td>3ktp/3ktr</td>
<td>multi</td>
<td>448</td>
<td>5</td>
<td>71</td>
<td>83</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Beta-2-microglobulin</td>
<td>52, 54, 56, 57, 63</td>
<td>1nez</td>
<td>aff</td>
<td>5488</td>
<td>1</td>
<td>321</td>
<td>WC</td>
<td>2</td>
<td>WC</td>
</tr>
<tr>
<td>Papain</td>
<td>18, 19, 21, 159, 177, 181</td>
<td>1stf</td>
<td>aff</td>
<td>9408</td>
<td>1</td>
<td>350</td>
<td>1349</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

206
7.5 Conclusions

The protein design problem enjoys a wide array of powerful algorithms for conformational and sequence search. These algorithms take a discrete energy matrix and perform sequence optimizations, both in the single-state and multistate cases. At the same time, previous work in bioinformatics and quantum chemistry has made great progress toward quantitatively accurate modeling of the flexibility and energy landscapes of biomolecular systems. Uniting these fields to perform designs with highly realistic modeling would result in great biomedical impact, both in protein and drug design. However, because state-of-the-art flexibility and energy modeling methods do not produce a discrete matrix, there is a gap between these fields. LUTE offers a strategy to bridge this gap. By representing continuous flexibility and general energy functions in a discrete matrix, it greatly increases the realism of the modeling that discrete combinatorial optimization algorithms like DEE/A* can directly accommo-
date. We thus believe that LUTE can serve as a foundation for greatly improved biomolecular design protocols.
Developing algorithms for protein design has been a uniquely exciting research project, and I would like to continue to work on this as an independent investigator. As such, the following is adapted from a research statement that I have used to apply for faculty jobs. In this section, I would like to summarize what I and others have achieved in the past and what I see as the next steps in this project.

I want to continue developing algorithms that systematically identify drug candidates with a desired function. Most drugs act by binding to and blocking the activity of a target molecule that is involved in disease—for example, bacterial or viral molecules, or human proteins that are involved in cancer. Once a target is identified, drug design can be formulated as a well-defined computational problem, consisting of several steps that require algorithms ranging from optimization to geometry to machine learning. I have produced the first provably accurate algorithms for some of the critical steps as part of my graduate work. I plan to attack the remaining unsolved steps as an independent investigator. If these efforts are successful, they will produce software capable of generating many novel, viable drug candidates for a
wide range of diseases. To this end, I propose the following three research directions. Each direction takes a class of open problems in drug design, and aims to produce efficient and highly accurate algorithms for it—preferably provably accurate algorithms.

8.1 Algorithms for Improved Flexibility Modeling

To design molecules, we must model their flexibility—the ability of their atoms to move around in space. In particular, accurate design requires a suitable model of a molecule’s continuous internal coordinates. Previous methods to model flexibility for design purposes either require exhaustive search over drug candidates (e.g., by molecular dynamics simulation, or model only a limited set of degrees of freedom. My PhD thesis research focused on modeling the most important degrees of freedom needed to model local motions in proteins. I developed DEEPer, the first provably accurate algorithm to simultaneously account for continuous degrees of freedom in both the backbone and the sidechains of a protein, which can be seen as the two main chemical components of the protein. Modeling of protein flexibility is very important, because most drug targets and an increasing number of new drugs are proteins. But there is currently no systematic way to search over the entire range of protein backbone flexibility found in nature, and search methods are even more lacking for non-protein drugs.

I will address this problem using novel internal coordinates and established knowledge about protein structure. For example, the full range of flexibility of a protein loop—a type of flexible segment of a protein that is often essential for binding and catalysis—is described by empirically validated constraints on dihedral angles between atoms in the protein backbone. I have developed a set of internal coordinates that admits efficient local search over all biophysically feasible directions of motion for a protein backbone, which can be constrained based on the dihedrals.
Building on DEEPer and this internal coordinate system, I will develop a provably accurate algorithm to design protein loops, accounting for the full range of flexibility allowed by the known dihedral constraints. I will also generalize this approach from proteins to a larger class of molecules that could be suitable as drugs. I will draw on ideas from kinematics, robotics, and numerical analysis in this work.

Additionally, molecules in nature are found in a continuous distribution over many geometries (weighted by energy), but previous combinatorial drug design algorithms have modeled them as being in a single geometry or a discrete distribution. My new LUTE algorithm makes it relatively straightforward to model a continuous distribution, so I will implement this model and develop algorithms to use it efficiently.

My goal for this research direction is to resolve a dilemma in current methods—one is forced to choose between exhaustively searching over candidates or using combinatorial search methods that severely restrict the set of degrees of freedom that are modeled. The exhaustive search approach can simulate the motion of the system over time with respect to all degrees of freedom, but this simulation is relatively computationally expensive even for a single candidate, and thus intractable for large sets of candidates, which will often grow exponentially with the size of the drug being designed. For example, many recent drugs are peptides (very small proteins), consisting of a chain of amino acids. We can choose the chemical structure of each amino acid, and there are 20 natural options for each, leading to a set of \( 20^{10} \) drug candidates even for a small peptide of 10 amino acids. This is far too many to consider explicitly even briefly, much less to simulate in detail. Combinatorial drug design methods can avoid this exhaustive search, but the neglected degrees of freedom can be of great functional importance. For example, the human body produces proteins called antibodies whose sequence it can adjust to target different types of
pathogens (much as we wish to adjust the chemical structure of drugs). The backbone structures of these different antibodies usually fall into a set of discrete classes, but the backbones in each class vary continuously and significantly with respect to all degrees of freedom allowed by the backbone dihedral constraints. If we wish to design drugs as effectively as the human body designs antibodies, we must learn to model this level of continuous backbone flexibility, as well as a comparable level of flexibility for non-protein molecules.

8.2 Energy Function Algorithms

The energy of a chemical system determines its behavior. The system will favorably adopt geometries that have low energies, and thus accurate evaluation of the energy given the coordinates of the atoms is essential to predicting the behavior of the system. This coordinates-to-energy mapping is known as the energy function. Accurate energy functions are very computationally expensive. The physics that determines the energy of a molecule is very well understood, but since previous design methods require a huge number of energy computations, many approximations are currently made to defray this expense. In particular, the energy is almost always assumed to be a sum of terms depending on the coordinates of only a few atoms (this approximation strategy is known as molecular mechanics). Furthermore, previous protein design algorithms have not modeled the water molecules that surround proteins in living cells, but rather have used implicit solvent models that only roughly approximate the effects of the water molecules’ presence. I believe that the long-term improvement of drug design requires fundamental changes to how energy is calculated in order to make it closely approximate reality. I propose two approaches to do this. First, I will greatly reduce the number of energy function calls needed in design calculations; my thesis research has already made significant progress in this direction. Second, I will develop more efficient methods to evaluate energy functions.
My graduate research developed two machine learning algorithms—EPIC\textsuperscript{64} and LUTE\textsuperscript{67}—that reduce the number of energy function calls needed in design calculations from exponential to polynomial in the size of the input. They do this by learning a compact representation of the energy surface for a given problem. I will extend these methods to accommodate highly accurate energy functions that use quantum chemistry and explicit modeling of water molecules—effects usually excluded from design calculations because of their computational cost. I will also use ideas from LUTE\textsuperscript{67} and other methods to speed up these energy functions themselves. My goal here is to make them efficient enough to use routinely in design calculations, in place of the much more approximate energy functions that are currently used.

Energy estimation methods that admit efficient search always express the energy as a sum of local terms of some kind, but LUTE requires uniquely weak assumption about these terms. By least-squares fitting for a particular protein system, it breaks down the energy into terms dependent on the geometries of one or a few residues (small portions of the protein, with about 10-20 atoms each). These terms are far less local than terms depending on the positions of a few atoms, as in molecular mechanics or even in modified versions of molecular mechanics that account for electronic polarization.\textsuperscript{85} We can expect greater accuracy when we apply only these weaker locality assumptions. Although molecular mechanics can often predict the interaction energy of a single pair of residues within a few kcal/mol, accurately and consistently predicting drug efficacy requires a margin of error of a few kcal/mol for the full binding energy of a drug to its target, which requires a significantly smaller relative margin of error than molecular mechanics can provide.\textsuperscript{120} Building on LUTE, I will develop methods to enable this higher level of accuracy.
8.3 Optimizing Chemical Composition Subject to Pharmacokinetic Constraints

Not every molecule that binds a target protein is effective as a drug. It must bind that protein specifically, and it must be able to enter the right part of the body and stay there long enough to be effective (that is, it must have good pharmacokinetics). These issues, especially the pharmacokinetics, have received little attention in previous computational protein and drug design methods, which have focused on achieving binding and largely left these other issues to be tested experimentally. But including specificity and pharmacokinetic constraints in drug design would greatly increase the chance of a drug candidate turning into a successful drug. Thus, I will work on incorporating such constraints into the optimization process used in drug design.

In my graduate work, I addressed the problem of specificity by developing COMETS, a constrained optimization algorithm for the chemical composition of a molecule. For example, it can optimize the binding of a drug to one protein target, while constraining it not to bind another. COMETS is the first provably accurate, general algorithm to address such constraints. I will add pharmacokinetic constraints to this framework. These constraints will ensure that a drug can be absorbed fast enough into the cells where it is needed, without being broken down or excreted out of the body too quickly. Chemical heuristics (e.g., molecule size constraints) are currently used heuristically in drug design to estimate pharmacokinetic properties, but I will adapt the COMETS framework to incorporate these constraints systematically into optimization. I will also explore more rigorous modeling of some of these processes. Drug design algorithms that systematically optimize properties beyond binding would be a powerful and novel addition to the drug design toolkit.
8.4 Summary

Drug design algorithms in general, and protein design algorithms in particular, have received a fair amount of attention from researchers. But most of this attention has focused on algorithms with no guarantees of accuracy, and indeed their predictions have often failed in experimental tests. Furthermore, most of the provably accurate algorithms have focused on only one step in the design calculation—the combinatorial optimization aspect,\(^\text{62,125,140}\) in which we look for the lowest-energy geometry and chemical composition of a molecule given a discrete representation of its flexibility. But highly accurate design requires that each step of the design be accurate. Furthermore the other steps—including flexibility modeling\(^\text{68}\) and energy function evaluation\(^\text{64,67}\)—offer fascinating algorithmic problems. So, building on the algorithms I have developed in graduate school, I will develop algorithms that address these important but previously under-researched problems. These algorithms will facilitate a holistic and comprehensive approach to drug design.

As I have done in my graduate research,\(^\text{62,64,67,68}\) I will validate each new algorithm with “unit tests” that evaluate their efficiency and accuracy for performing their specified step, rather than by tests of the entire design protocol (which will likely succeed or fail based on the performance of other components of the protocol). My goal is to develop methods that can be applied without modification to a wide variety of proteins and drug design problems. As a later stage in validation, I will also seek out experimental collaborators who are working on drug design problems, and apply my methods to these systems.

Drug design currently relies heavily on experimental screening methods, in which a set of known molecules is tested experimentally. This approach has worked very well for many problems, but many targets still appear intractable to it. For example,
we have not found drugs to target many of the most important proteins involved in cancer, and as a result many cancers remain highly lethal. Even bacterial infections, which have been among the greatest success stories of pharmacology so far, are showing more and more antibiotic resistance, so we require innovations in drug design to ensure that they do not reemerge as a major cause of death. However, the set of molecules that could be viable drugs is far greater than the set of molecules made by humans so far, and with a systematic computational method, we can search this far larger set of candidates. When such a method is fully developed, we can comprehensively address antibiotic resistance and a host of diseases that are difficult to treat using current methods.
Bibliography


Mark A. Hallen and Bruce R. Donald. comets (Constrained Optimization of Multistate Energies by Tree Search): A provable and efficient protein design


[125] Yuchao Pan, Yuxi Dong, Jingtian Zhou, Mark Hallen, Bruce R. Donald, Jianyang Zeng, and Wei Xu. cOSPREY: A cloud-based distributed algorithm for large-scale computational protein design. 2015. Submitted manuscript.


231


[178] Aiwu Zhou, Robin W. Carrell, Michael P. Murphy, Zhenquan Wei, Yahui Yan, Peter L. D. Stanley, Penelope E. Stein, Fiona Broughton Pipkin, and Randy J.


Biography

Mark Andrew Hallen was born on June 27, 1988 in Ithaca, New York. He earned a BS from Duke University in 2009, with a double major in math and chemistry. He started his research career in high school at North Carolina State University, where he worked in Hans Hallen’s lab on preparation of hydrophilic self-assembled monolayers.\textsuperscript{65} During his undergraduate studies, he worked in Sharyn Endow’s lab at Duke, which studies molecular motors and the cytoskeleton.\textsuperscript{31, 63, 66, 70, 71, 106, 111, 183} He specialized in analysis of fluorescence microscopy data, including developing novel methods, some of which were developed in collaboration with Anita Layton.\textsuperscript{69} He continued at Duke for his PhD, performing research with Lingchong You,\textsuperscript{60} David Beratan, and Michael Fitzgerald\textsuperscript{171} before settling into Bruce Donald’s lab and specializing in protein design for this thesis.\textsuperscript{61, 62, 64, 67, 68, 95, 140} He is completing this thesis in 2016 in the computer science department, with a certificate in Structural Biology and Biophysics. Mark has received the Goldwater and National Merit scholarships as well as the NDSEG, James B. Duke, PhRMA Informatics and Dolores Zohrab Liebmann fellowships.