PLUG (Pruning of Local Unrealistic Geometries) removes restrictions on biophysical modeling for protein design

Mark A. Hallen*

July 12, 2018

Running title: PLUG protein design algorithm

*Toyota Technological Institute at Chicago, 6045 S. Kenwood Ave., Chicago, IL 60637. Phone: 773 834 2500. Fax: 773 357 6970. Email: mhallen@ttic.edu

Acknowledgments I would like to thank Bruce Donald, Jonathan Jou, Marcel Frenkel, and Jeff Martin for helpful discussions, and the TTIC endowment for funding.

Abstract

Protein design algorithms must search an enormous conformational space to identify favorable conformations. As a result, those that perform this search with guarantees of accuracy generally start with a conformational pruning step, such as dead-end elimination (DEE). However, the mathematical assumptions of DEE-based pruning algorithms have up to now severely restricted the biophysical model that can feasibly be used in protein design. To lift these restrictions, I propose to prune local unrealistic geometries (PLUG) using a linear programming-based method. PLUG’s biophysical model consists only of well-known lower bounds on interatomic distances. PLUG is intended as pre-processing for energy-based protein design calculations, whose biophysical model need not support DEE pruning. Based on 96 test cases, PLUG is at least as effective at pruning as DEE for larger protein designs—the type that most require pruning. When combined with the LUTE protein design algorithm, PLUG greatly facilitates designs that account for continuous entropy, large multistate designs with continuous flexibility, and designs with extensive continuous backbone flexibility and advanced non-pairwise energy functions. Many of these designs are tractable only with PLUG, either for empirical reasons (LUTE’s machine learning step achieves an accurate fit only after PLUG pruning), or for theoretical reasons (many energy functions are fundamentally incompatible with DEE).

Keywords: Algorithms, molecular conformation, linear programming, drug design, entropy, biophysics ■

INTRODUCTION

Computational protein design is the identification of protein sequences that optimize a desired function, typically involving binding^{6,9,31,37}. Specifically, given the 3-D structure of a protein, protein design algorithms can identify mutations in the sequence of the protein that will minimize its energy—either the energy of the protein itself^{5,6,10,15,27}, or a derived quantity such as a binding energy of the protein to a ligand^{7,13,25,30}. Predictions of sequences’ energies are based on a *biophysical model*. This model includes an energy function mapping molecular geometry to energy, as well as a set of conformational degrees of freedom that are considered flexible, and bounds on how much each degree of freedom can move.

Efficient protein design algorithms must search an enormous space of sequences and conformations, which grows exponentially as mutable residue positions are added to the design. Approximations are thus commonly made to speed up the search, at a significant cost in accuracy. One common assumption is that each residue can only access a limited, discrete set of conformations known as *ideal rotamers*^{5,23}, which correspond to observed modal values in the space of sidechain dihedral angles. Another is that the energy function is a sum of terms depending on at most two residues’ conformations—that is, it is *pairwise*. The most efficient protein design algorithms rely on these assumptions to calculate an *energy matrix* that stores precomputed energetic interactions between pairs of rotamers at different residues. This precomputation dramatically speeds up protein design calculations. An energy matrix can be used to efficiently perform simulated annealing^{4,27,48}, which can yield an answer quickly with no guarantees of accuracy and with error that empirically increases for larger designs⁴⁴, or as input to a search algorithm with provable guarantees of accuracy. Many algorithms with such guarantees are available, including DEE/A*^{5,10,13,17,22,29,36}, integer linear programming^{26,40}, branch-²⁴ and tree⁵⁰ decomposition-based methods, and weighted constraint satisfaction algorithms^{40,46,47}.

Although simulated annealing and DEE/A* have both been extended to account for continuous flexibility, this comes at a significant computational cost. Simulated annealing becomes very inefficient when the energy matrix is not available. Hence continuous flexibility, if accounted for at all, is used only in a small number of rounds of sampling at the end of the

calculation⁴⁸, which may prevent the identification of favorable conformations significantly different from the best conformation found in discrete search.

DEE/A* has been extended to account for continuous flexibility while maintaining guarantees of accuracy^{10,13,20,21,38,39}, by using lower bounds on pairwise interaction energy that are calculated by minimizing pairwise interaction energies for particular rotamer pairs in isolation. In particular, while DEE determines if a rotamer can be pruned by computing a difference between energies based on a discrete energy matrix and checking if it is greater than zero⁵, the iMinDEE algorithm computes the same energy difference using the pairwise energy matrix and checks if it is greater than an energy buffer called the *pruning interval*¹⁰. iMinDEE is proven to be accurate if the pruning interval is large enough. It must be at least the difference between the globally optimal energy computed using the pairwise minimized energy matrix, and the true globally optimal energy (calculated using minimization of fully defined conformations rather than just pairs). Thus the pruning interval is in some sense an upper bound on how much the pairwise minimized energy can deviate from the true minimized energy, for purposes of pruning.

The machine learning-based LUTE algorithm²¹ is currently the most efficient way to perform protein design with continuous flexibility. It learns an energy matrix that closely approximates the continuously minimized energies. LUTE can be applied to multiple states of a protein system—for example bound and unbound states, possibly with multiple binding partners—to perform *multistate designs* that optimize for properties like binding affinity and specificity, using the COMETS algorithm¹⁸. However, the presence of high-energy conformations in a conformational space can cause the number of parameters required for a low-residual LUTE fit to be prohibitively large. Thus, a pruning method like iMinDEE must be performed before the fitting of the LUTE matrix. This limits the applicability of LUTE, and indeed of continuously flexible protein design in general, to designs that are either suitable for running iMinDEE pruning (requiring pairwise energy functions and certain other conditions) or have small enough sequence and conformational spaces to allow efficient and accurate LUTE fits with only very obvious steric clashes being pruned²¹. Thus, a more general pruning method would help for such designs.

PLUG provides this more general pruning method. Rather than relying on an energy

matrix or indeed any biophysical assumptions about energy, it merely generates geometric constraints representing steric clashes (pairs of atoms coming unrealistically close together). The conditions that cause steric clashes have actually been characterized more precisely than protein energetics in general, and are routinely used in visualizing protein structures and detecting problems in crystallographic structure determination, e.g. using the Probe software⁴⁹ and the MolProbity server². PLUG eliminates clashing portions of conformational space from consideration in designs, enabling more efficient and accurate application of LUTE and other protein design algorithms.

PLUG is implemented in the OSPREY^{11,13,14,35} open-source protein design package. OSPREY has yielded many designs that performed well experimentally—*in vitro*^{1,8,12,16,38,43,45} and *in vivo*^{8,16,38,43} as well as in non-human primates⁴³—and contains a wide array of flexibility modeling options and provably accurate design algorithms^{11,14}. These features will allow PLUG to be used for many different types of designs.

By presenting PLUG, this paper makes the following contributions:

1. A method for pruning portions of protein (or other polymer) conformational space based on geometry and a model of steric clashes.
2. A method to apply this pruning method in protein designs using the LUTE algorithm, greatly facilitating its use with extensive backbone flexibility, non-pairwise energy functions, and continuous sidechain entropy and in multistate designs. Many of these designs would not be practical without PLUG, as shown empirically here.
3. Empirical results on 96 design test cases using 36 protein structures, demonstrating that PLUG’s pruning power is comparable to the previous state of the art, while it allows the use of a wider range of biophysical models in protein design.
4. An implementation of PLUG in the open-source OSPREY^{11,13,14,35} protein design software package, available at www.github.com/markhallen369/OSPREY_refactor

MATERIALS AND METHODS

Background: Voxel representation of conformational space

Consider a polymer (for example, a protein) consisting of a chain of *residues*. Chemically, the polymer is defined by its *sequence*, which defines each residue as a particular chemical species (a particular amino-acid type, in the case of a protein). Designing the polymer requires searching over a sequence space, represented as a set of possible chemical species for each residue. For example, a simple sequence space for a protein might consist of all sequences in which a protein has either valine or alanine at residue position 1, either glutamate or aspartate at residue 2, and the wild-type amino-acid type at all other residues. The objective function for the search over sequence space can vary from design to design. One may wish to simply minimize the energy of the protein (to optimize stability), or optimize binding to a particular partner, or optimize specific binding to one partner but not another. Each of these types of designs can be approached using biophysical modeling, which requires a representation of the polymer’s conformational space: the set of 3-D geometries that each sequence can exhibit¹⁸. The conformational flexibility of the polymer will be modeled using n internal coordinates $\mathbf{x} = \{x_i | i \in \{1, \dots, n\}\}$. These internal coordinates can be sidechain dihedral angles or backbone perturbations²², or even backbone internal coordinates generated using the CATS algorithm¹⁹ that allow the protein backbone to move locally in all locally feasible directions. The conformational space of the system is then defined as the union of *voxels*^{10,13,22}. Each voxel v is defined by a sequence and the inequality constraints

$$a_i(v) \leq x_i \leq b_i(v), \quad (1)$$

for $i \in \{1, \dots, n\}$, where $a_i(v)$ and $b_i(v)$ are voxel-specific bounds defined per the modeling assumptions. If $a_i(v) < b_i(v)$, the coordinate x_i has *continuous flexibility* in v .

The role of PLUG will be to identify voxels with continuous flexibility that are not biophysically feasible, and to exclude them from the conformational space. This simplifies the problem and enables the subsequent use of algorithms (e.g., LUTE²¹) that can search over a very large space of voxels as long as those voxels are “well-behaved” energetically. For example, when a conformation contains non-bonded atoms that are very close to each

other, the energy is extremely sensitive to small conformational changes, leading to an ill-conditioned search problem that is difficult for LUTE to solve. But conformations with such atoms are not biophysically feasible (they are referred to as *sterically clashing*), and PLUG can remove them.

It is useful to consider not only voxels in the conformation space of the entire protein, but also voxels in the conformational space $C(P)$ of a subset P of the protein’s residues. Such a voxel is defined only by amino-acid type assignments to the residues in P , along with bounds on a subset of the internal coordinates: for example, the conformation of the residues in P will not depend on the value of the sidechain dihedrals of a residue $r \notin P$.

If $|P| = 1$ then the voxel is called a *residue conformation* or RC ²². The conformational space of each residue is small enough to enumerate explicitly (often <100 RCs). For example, Ref. 32 tabulates the values of the sidechain dihedral angles for each amino-acid type in a large number of experimentally determined protein structures, and the vast majority of them fall in a small number of voxels in sidechain dihedral space. For purposes of this paper, each cluster in sidechain dihedral space (i.e., each *rotamer*) identified in the Penultimate rotamer library (Ref. 32) is treated as a voxel, and each dihedral is allowed to deviate by up to 9° from the modal value in the cluster.

For $|P| > 1$, $C(P)$ consists of all possible combinations $v = v_1 \cap v_2 \cap \dots$ where v_1, v_2 , etc. are RC assignments for each of the residues in P . The voxels v in $C(P)$ are intersections of RCs because they consist of the set of conformations bounded by the inequalities defining each of their constituent RCs $v_1, v_2, \dots$. Thus the number of voxels in $C(P)$ scales exponentially with respect to $|P|$. As a result, efficient combinatorial search algorithms are necessary to search the enormous sequence and conformational space of the entire polymer, even if conformational flexibility is not modeled for residues that are far from any mutable residues. PLUG is a preprocessing step for this combinatorial search.

Linear model of steric clashes

Computational prediction of favorable conformations in proteins is still very difficult³³. However, crystallographic data as well as high-level quantum chemistry calculations have established limits on how close nonbonded pairs of atoms can be found in a biophysical system

(except as minuscule contributors to a conformational ensemble). This work will use the lower bounds from Ref. 49 on the distance between two nonbonded atoms of particular element types. These lower bounds include exceptions for hydrogen bonds and for atoms within a few bonds of each other, which are also used here. If continuous flexibility is neglected, it is easy to remove a voxel v whose conformations all violate these lower bounds, because v consists of a single conformation and inter-atom distances can be computed directly. In the presence of continuous flexibility, it becomes a more difficult problem. Let $\mathbf{x} = \{x_i | i \in \{1, \dots, m\}\}$ be the set of internal coordinates bounded in v . Thus we can say $\mathbf{x} \in v$ if the vector $\mathbf{x}$ of internal coordinate values respects the box constraints that define v . Let $\{a_j\}$ be the set of atoms whose conformation is described by v ; for example if $|v|$ is an RC (single-residue voxel) then this set will only include atoms in that residue. Let $d_{jk}(\mathbf{x})$ denote the distance between atoms a_j and a_k , as a function of the continuous internal coordinates $\mathbf{x}$. Finally, let b_{jk} denote the lower bound based on Ref. 49 on that distance; for example if a_j is a carbon and a_k is a hydrogen not within a few bonds of a_j , then b_{jk} will be the minimum sterically feasible distance between a nonbonded carbon and hydrogen. PLUG aims to determine if there exists a conformation in v without steric clashes: that is, $\mathbf{x} \in v$ such that

$$d_{jk}(\mathbf{x}) \geq b_{jk} \quad \forall (j, k) \quad (2)$$

In general this problem is NP-hard even if all the atomic coordinates vary linearly with respect to all the internal coordinates. However, within the constraints of a voxel, we can use an approximation that makes the problem tractable: we assume that interatomic *distances* vary linearly with respect to continuous internal coordinates. For example, within the voxel representing a single rotamer, a 1° increase in the value of a particular sidechain dihedral has roughly the same effect on interatomic distances no matter what the other dihedrals' values are. The error in this approximation is similar to the uncertainty in the lower bound on interatomic distances (a few tenths of an angstrom: lower bounds on interatomic distances can reasonably be taken as 0.25 to 0.4 angstroms of overlap between van der Waals spheres⁴⁹; see Fig. 1). Moreover, the approximation is often empirically very accurate (see Fig. 2). Areas of conformational space erroneously excluded by the linearized constraint are likely to be

missed by an energy minimizer anyway. This is because a convex obstacle like a clashing atom is likely to block the minimizer from exploring conformational space on the other side (see Fig. 2, middle bottom).

Linearized inequalities representing clashes are computed by starting at the center of a voxel (where the minimizer also starts), identifying a point $\mathbf{x}$ where $d_{jk}(\mathbf{x}) = b_{jk}$ by Newton's method, and then taking the numerical derivative of $d_{jk}(\mathbf{x})$ with respect to x (Fig. 3), yielding a constraint

$$\mathbf{u}_{jk} \cdot \mathbf{x} \geq w_{jk} \quad (3)$$

that linearly approximates the true constraint $d_{jk}(\mathbf{x}) \geq b_{jk}$. (The quantities $\mathbf{u}_{jk}$ and w_{jk} are chosen to provide this linear approximation).

Once these linear constraints are computed, the problem of satisfying Eq. (2) becomes a linear programming problem: finding $\mathbf{x}$ such that

$$\begin{aligned} a_i(v) &\leq x_i \leq b_i(v) \quad \forall i \\ \mathbf{u}_{jk} \cdot \mathbf{x} &\geq w_{jk} \quad \forall (j, k) \end{aligned} \quad (4)$$

where the first set of constraints just defines the voxel (see Eq. 1). Thus, sterically clashing voxels can be identified by linear programming: these are simply voxels for which the system of inequalities in Eq. (4) is unsatisfiable.

Matrix precomputations

Since the introduction of DEE⁵ and energy matrices, protein design algorithms have relied on precomputing matrices that tabulate functions of a single RC and/or of a low-order tuple of RCs. PLUG is no exception. First of all, a design with PLUG must begin with the computation of a PLUG matrix: a list of linear constraints representing all pairs of atoms in the system that may clash sterically and give rise to a linear constraint (Eq. 3). Since a pair of atoms can involve at most two different residues, the PLUG matrix can be organized as lists of constraints for each RC and each pair of RCs in the system. Linearized clash constraints for pairs of atoms in different continuously flexible residues are recorded in RC-pair entries of the PLUG matrix; the rest of the constraints (between two atoms in the same residue, or

between residues with and without continuous flexibility) are recorded in single-RC entries. Then when PLUG pruning is performed for any (> 2)-residue voxel, the constraints will be looked up for individual RCs and pairs of RCs within the higher-order tuple of RCs defining the higher-order voxel. In other words, if L is the list of RCs describing a voxel, then the set of linear constraints to be used for trying to prune the voxel is

$$\left(\bigcup_{r \in L} C(r) \right) \cup \left(\bigcup_{r \in L, s \in L} C(r, s) \right) \quad (5)$$

where $C(r)$ and $C(r, s)$ are lists of constraints for RC r and RC pair (r, s) respectively, both of which are looked up in the PLUG matrix. Precomputation of the PLUG matrix is the only part of PLUG that requires explicit modeling of molecular geometries, to identify the coefficients for the linear constraints. Next, pruning of RCs and pairs and triples of RCs can be performed. Like DEE, this results in a pruning matrix, representing what RCs and tuples of RCs are pruned.

Complexity

Pruning using PLUG is asymptotically more efficient than pruning using DEE. If there are n residues with at most r RCs each, there are $O(n^2 r^2)$ pairs and $O(n^3 r^3)$ triples of RCs to prune, and thus PLUG pruning takes $O(n^3 r^3)$ time if triples are pruned and $O(n^2 r^2)$ if not. By contrast, DEE pruning requires comparing a candidate tuple of RCs for pruning to a *competitor* tuple of the same order. So DEE takes $O(n^3 r^6)$ time with triples: it iterates over $O(n^3)$ triples of residue positions, $O(r^3)$ candidate triples of RCs given the positions, and $O(r^3)$ competitor triples of RCs given the positions and the candidate triple. Similarly DEE pairs pruning takes $O(n^2 r^4)$ time. The precomputation of the PLUG matrix itself takes $O(n^2 r^2)$ time, just like computing a pairwise energy matrix, because that is how many pairs of RCs are present in the system. So for both PLUG and DEE it is the pruning of triples (or of pairs if triples are not pruned) that is asymptotically the bottleneck.

Protocols for using PLUG in protein design

Once the PLUG and pruning matrices have been precomputed, PLUG can be used in a few ways in the protein design computation. First, it can be used as a pre-pruning step just

like DEE, removing RCs or tuples of RCs from consideration during combinatorial search. Second, PLUG can be used to prune many-residue voxels later in the design computation. The latter is particularly useful when computing a LUTE matrix²¹, to ensure that the training set for the fitting of LUTE coefficients does not contain inevitably clashing voxels. In this case PLUG pruning is used as part of the pruning check for partial conformations during drawing of sample voxels for the training and validation sets (see Ref. 21). In other words, the LUTE matrix is trained to predict the continuously minimized energies only of non-clashing voxels. This significantly reduces LUTE residuals in the presence of backbone flexibility modeled with CATS¹⁹ (see Fig. 4). The reason this higher-order pruning is helpful is because a pair of residues may be able to escape a clash when no other residues are around, but the motion that escapes that clash may cause a clash for a different residue pair, once the conformational space of a larger number of residues is taken into account.

RESULTS AND DISCUSSION

Pruning power of PLUG

The numbers of RCs pruned by PLUG and iMinDEE were compared at several values of the pruning interval for 96 test cases (Table 1), drawn from Refs. 19 and 20. PLUG pruning is comparable to iMinDEE at a pruning interval of about 5 kcal/mol (Fig. 5), which corresponds to a relatively small design. PLUG is more powerful than iMinDEE at higher pruning intervals, since PLUG pruning is independent of that interval. Up to a pruning interval of ~15 kcal/mol, iMinDEE and PLUG together prune significantly (~10-20%) more than PLUG alone. However, designs with many flexible positions (see Refs. 10, 21, etc.) or with continuous backbone flexibility (see Ref. 19) will frequently exceed even a pruning interval of 15 kcal/mol. Moreover, since smaller designs can be accomplished with little pruning, these results indicate PLUG is a suitable replacement for iMinDEE for protein design in general, and can bring the power of iMinDEE to designs in which a pairwise energy matrix and/or pruning interval is not available.

For 19 of the 96 design test cases tried here (Table 2), PLUG pruned all RCs—it found

that steric clashes are unavoidable. Although it is possible to find a minimum-energy sequence and conformation for each of these test cases, this conformation is expected to be biophysically unrealistic (e.g., see Fig. 6). In this sense PLUG improves the realism of protein design. It ensures that the minimum-energy voxel contains a non-clashing conformation. Thus, if PLUG is used in a design, either the predicted minimized-energy conformation will be free of steric clashes, or it will be obtainable by energy minimization starting at a conformation free of steric clashes (and thus any steric clashes will be purely due to the energy function judging other interactions as more important than the steric clash).

PLUG enables the highly efficient LUTE algorithm to incorporate extensive backbone flexibility

The CATS algorithm¹⁹ greatly increased the level of continuous backbone flexibility available in combinatorial protein design, but at a significant computational cost. Precomputation of a LUTE matrix²¹, which faithfully represents continuously minimized energies despite having the form of a discrete energy matrix, can reduce this cost. Unfortunately, conformational spaces with CATS flexibility have many steric clashes, resulting in high LUTE residuals and impeding the use of LUTE to speed up CATS. Fortunately, PLUG pruning appears to resolve this problem (Fig. 4; Table 3). For each of 13 designs drawn from Ref. 19, a pairwise LUTE matrix matched the CATS-minimized energies of PLUG-allowed voxels to within less than the thermal energy at room temperature (average residual 0.06 (kcal/mol)²; maximum residual 0.22). This quality of fit indicates that LUTE can perform the design with negligible error. By contrast, it was impossible to fit a matrix of similar quality to a conformational space that included PLUG-pruned voxels. Using DEE pruning only, only four of the 13 residuals were below the thermal energy at room temperature, and another four were over 100 (kcal/mol)² (i.e., completely unusable). Thus, PLUG is essential for consistent accuracy when performing LUTE computations with CATS backbone flexibility. As a result, PLUG brings the benefits of LUTE to designs with extensive backbone flexibility: efficiency (and thus the ability to handle larger designs), and compatibility with general energy functions.

PLUG is necessary for designs where DEE-based pruning is impossible

LUTE can be used to learn a pairwise energy matrix from an energy function that is not pairwise—for example one including Poisson-Boltzmann solvation energies or using the free energy rather than the minimized energy of a voxel²¹. This allows such energy functions to be used in combinatorial protein design calculations, which is otherwise impossible. However, previous applications of LUTE to such energy functions²¹ was limited by the fact that only pruning of the most obvious steric clashes was available. With PLUG, it becomes possible to prune with efficiency similar to DEE for any energy function. Using PLUG as a pre-pruning step, pairwise LUTE matrices were precomputed for six of the CATS designs in Fig. 4 using the Poisson-Boltzmann equation, as implemented in Delphi^{34,41}, to calculate the solvation energy for minimized conformations. This energy function is not pairwise, but yielded residuals that were all below the thermal energy at room temperature (Table 4). The average was 0.17 (kcal/mol)², with the highest residual, 0.35 (kcal/mol)², being 0.03 (kcal/mol)² lower than for an equivalent computation with OSPREY’s default, pairwise EEF1²⁸ solvation energy function (no DEE pruning, and with CATS backbone flexibility). Without PLUG pruning, LUTE residuals for the Poisson-Boltzmann energy function were unusable (all over 100 (kcal/mol)²).

A pairwise LUTE matrix was also computed for each of these systems with a rigid backbone but with continuous sidechain entropy. Residuals were all below 0.005 (kcal/mol)² (Table 4). This calculation defines the energy of a voxel differently than usual. Instead of minimizing the energy E' with respect to the values $\mathbf{x}$ of the continuous degrees of freedom within the voxel, the voxel energy is defined as the free energy

$$-RT \ln \left(\int_v \exp \left(-\frac{E'(\mathbf{x})}{RT} \right) d\mathbf{x} \right) \quad (6)$$

where R is the gas constant, T is the temperature, and v is the voxel. The integral was computed by Monte Carlo integration. Together with the K^* free energy algorithm^{13,30}, which approximately discrete partition functions efficiently, this method can be used to compute free energies of binding that include both continuous and discrete entropy.

Also, the procedure for provable computation of a pruning interval¹⁰, and thus for

iMinDEE, is not applicable to multistate designs unless suitable energy constraints are applied manually by the user. iMinDEE for multistate designs is also limited by the fact that it can only use competitors of the same amino-acid type as a candidate RC for pruning (i.e., only type-dependent DEE⁵¹ can be performed). PLUG suffers no such reduction in power or limitation on provability, and thus will greatly facilitate multistate designs. In this case, the strength of PLUG is that its pruning threshold is based on well-known geometric criteria, not on an energy threshold that must somehow be tuned or known *a priori* to achieve meaningful results.

CONCLUSIONS

The LUTE algorithm²¹ allows efficient incorporation of many types of advanced modeling into protein design. However, it requires a pre-pruning step to remove “ill-behaved” conformations. Previously, this pruning was performed with the DEE algorithm, which imposed significant restrictions on the biophysical model. But now these restrictions are gone: PLUG prunes with similar power to DEE, but with no biophysical assumptions except for well-established knowledge of protein geometry and a validated linearization approximation. The full power of LUTE is thus unleashed. This facilitates great improvements in several areas of biophysical modeling in protein design: energy functions, multistate design, continuous entropy, and backbone flexibility.

References

- [1] Chen, C.-Y., Georgiev, I., Anderson, A. C., and Donald, B. R. (2009). Computational structure-based redesign of enzyme activity. *Proceedings of the National Academy of Sciences of the USA*, 106(10):3764–3769.
- [2] Chen, V. B., Arendall, III, W. B., Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S., and Richardson, D. C. (2010). MolProbity: all-atom structure validation for macromolecular crystallography. *Acta Crystallographica D*, 66:12–21.
- [3] Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W., and Kollman, P. A. (1995). A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117:5179–5197.
- [4] Das, R. and Baker, D. (2008). Macromolecular modeling with Rosetta. *Annual Review of Biochemistry*, 77:363–382.
- [5] Desmet, J., de Maeyer, M., Hazes, B., and Lasters, I. (1992). The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356:539–542.
- [6] Donald, B. R. (2011). *Algorithms in Structural Molecular Biology*. MIT Press, Cambridge, MA.
- [7] Floudas, C. A., Klepeis, J. L., and Pardalos, P. M. (1999). Global optimization approaches in protein folding and peptide docking. In *Mathematical Support for Molecular Biology*, volume 47 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 141–172. American Mathematical Society.
- [8] Frey, K. M., Georgiev, I., Donald, B. R., and Anderson, A. C. (2010). Predicting resistance mutations using protein design algorithms. *Proceedings of the National Academy of Sciences of the USA*, 107(31):13707–13712.

- [9] Gainza, P., Nisonoff, H. M., and Donald, B. R. (2016). Algorithms for protein design. *Current Opinion in Structural Biology*, 39:16–26.
- [10] Gainza, P., Roberts, K., and Donald, B. R. (2012). Protein design using continuous rotamers. *PLoS Computational Biology*, 8(1):e1002335.
- [11] Gainza, P., Roberts, K. E., Georgiev, I., Lilien, R. H., Keedy, D. A., Chen, C.-Y., Reza, F., Anderson, A. C., Richardson, D. C., Richardson, J. S., and Donald, B. R. (2013). OSPREY: Protein design with ensembles, flexibility, and provable algorithms. *Methods in Enzymology*, 523:87–107.
- [12] Georgiev, I., Acharya, P., Schmidt, S., Li, Y., Wycuff, D., Ofek, G., Doria-Rose, N., Luongo, T., Yang, Y., Zhou, T., Donald, B. R., Mascola, J., and Kwong, P. (2012). Design of epitope-specific probes for sera analysis and antibody isolation. *Retrovirology*, 9(Suppl. 2):P50.
- [13] Georgiev, I., Lilien, R. H., and Donald, B. R. (2008). The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. *Journal of Computational Chemistry*, 29(10):1527–1542.
- [14] Georgiev, I., Roberts, K. E., Gainza, P., Hallen, M. A., and Donald, B. R. (2009). OSPREY (Open Source Protein Redesign for You) user manual. Available online: www.cs.duke.edu/donaldlab/software.php Updated, 2015. 94 pages.
- [15] Georgiev, I. S., Rudicell, R. S., Saunders, K. O., Shi, W., Kirys, T., McKee, K., O’Dell, S., Chuang, G.-Y., Yang, Z.-Y., Ofek, G., Connors, M., Mascola, J. R., Nabel, G. J., and Kwong, P. D. (2014). Antibodies VRC01 and 10E8 neutralize HIV-1 with high breadth and potency even with Ig-framework regions substantially reverted to germline. *The Journal of Immunology*, 192(3):1100–1106.
- [16] Gorczynski, M. J., Grembecka, J., Zhou, Y., Kong, Y., Roudaia, L., Douvas, M. G., Newman, M., Bielnicka, I., Baber, G., Corpora, T., Shi, J., Sridharan, M., Lilien, R.,

- Donald, B. R., Speck, N. A., Brown, M. L., and Bushweller, J. H. (2007). Allosteric inhibition of the protein-protein interaction between the leukemia-associated proteins Runx1 and CBF β . *Chemistry and Biology*, 14:1186–1197.
- [17] Gordon, D. B., Hom, G. K., Mayo, S. L., and Pierce, N. A. (2003). Exact rotamer optimization for protein design. *Journal of Computational Chemistry*, 24:232–243.
- [18] Hallen, M. A. and Donald, B. R. (2016). 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. *Journal of Computational Biology*, 23(5):311–321.
- [19] Hallen, M. A. and Donald, B. R. (2017). CATS (Coordinates of Atoms by Taylor Series): protein design with backbone flexibility in all locally feasible directions. *Bioinformatics*, 33(14):i5–i12.
- [20] Hallen, M. A., Gainza, P., and Donald, B. R. (2015). A compact representation of continuous energy surfaces for more efficient protein design. *Journal of Chemical Theory and Computation*, 11(5):2292–2306.
- [21] Hallen, M. A., Jou, J. D., and Donald, B. R. (2017). LUTE (Local Unpruned Tuple Expansion): Accurate continuously flexible protein design with general energy functions and rigid-rotamer-like efficiency. *Journal of Computational Biology*, 24(6):536–546.
- [22] Hallen, M. A., Keedy, D. A., and Donald, B. R. (2013). Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility. *Proteins: Structure, Function and Bioinformatics*, 81(1):18–39.
- [23] Janin, J., Wodak, S., Levitt, M., and Maigret, B. (1978). Conformation of amino acid side-chains in proteins. *Journal of Molecular Biology*, 125(3):357–386.
- [24] Jou, J. D., Jain, S., Georgiev, I., and Donald, B. R. (2016). BWM*: A novel, provable, ensemble-based dynamic programming algorithm for sparse approximations of computational protein design. *Journal of Computational Biology*, 23(6):413–424.

- [25] Karanicolas, J. and Kuhlman, B. (2009). Computational design of affinity and specificity at protein-protein interfaces. *Current Opinion in Structural Biology*, 19(4):458–463.
- [26] Kingsford, C. L., Chazelle, B., and Singh, M. (2005). Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039.
- [27] Kuhlman, B. and Baker, D. (2000). Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences of the USA*, 97(19):10383–10388.
- [28] Lazaridis, T. and Karplus, M. (1999). Effective energy function for proteins in solution. *Proteins: Structure, Function, and Bioinformatics*, 35(2):133–152.
- [29] Leach, A. R. and Lemon, A. P. (1998). Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *Proteins: Structure, Function, and Bioinformatics*, 33(2):227–239.
- [30] Lilien, R. H., Stevens, B. W., Anderson, A. C., and Donald, B. R. (2005). A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase A phenylalanine adenylation enzyme. *Journal of Computational Biology*, 12(6):740–761.
- [31] Lippow, S. M. and Tidor, B. (2007). Progress in computational protein design. *Current Opinion in Biotechnology*, 18(4):305–311.
- [32] Lovell, S. C., Word, M. J., Richardson, J. S., and Richardson, D. C. (2000). The penultimate rotamer library. *Proteins: Structure, Function, and Genetics*, 40:389–408.
- [33] Merz., Jr., K. M. (2010). Limits of free energy computation for protein-ligand interactions. *Journal of Chemical Theory and Computation*, 6(5):1769–1776.
- [34] Nicholls, A. and Honig, B. (1991). A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *Journal of Computational Chemistry*, 12(4):435–445.

- [35] Ojewole, A., Lowegard, A., Gainza, P., Reeve, S. M., Georgiev, I., Anderson, A. C., and Donald, B. R. (2017). OSPREY predicts resistance mutations using positive and negative computational protein design. In *Computational Protein Design*, volume 1529 of *Methods in Molecular Biology*. Humana Press, New York. In press.
- [36] Pierce, N. A., Spriet, J. A., Desmet, J., and Mayo, S. L. (2000). Conformational splitting: A more powerful criterion for dead-end elimination. *Journal of Computational Chemistry*, 21(11):999–1009.
- [37] Regan, L. (1999). Protein redesign. *Current Opinion in Structural Biology*, 9(4):494–499.
- [38] Roberts, K. E., Cushing, P. R., Boisguerin, P., Madden, D. R., and Donald, B. R. (2012). Computational design of a PDZ domain peptide inhibitor that rescues CFTR activity. *PLoS Computational Biology*, 8(4):e1002477.
- [39] Roberts, K. E. and Donald, B. R. (2015). Improved energy bound accuracy enhances the efficiency of continuous protein design. *Proteins: Structure, Function, and Bioinformatics*, 83(6):1151–1164.
- [40] Roberts, K. E., Gainza, P., Hallen, M. A., and Donald, B. R. (2015). Fast gap-free enumeration of conformations and sequences for protein design. *Proteins: Structure, Function, and Bioinformatics*, 83(10):1859–1877.
- [41] Rochia, W., Sridharan, S., Nicholls, A., Alexov, E., Chiabrera, A., and Honig, B. (2002). Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: Applications to the molecular systems and geometric objects. *Journal of Computational Chemistry*, 23(1):128–137.
- [42] Rosenzweig, A. C., Huffman, D. L., Hou, M. Y., Wernimont, A. K., Pufahl, R. A., and O’Halloran, T. V. (1999). Crystal structure of the Atx1 metallochaperone protein at 1.02 Å resolution. *Structure*, 7(6):605–617.
- [43] Rudicell, R. S., Kwon, Y. D., Ko, S.-Y., Pegu, A., Louder, M. K., Georgiev, I. S., Wu, X., Zhu, J., Boyington, J. C., Chen, X., Shi, W., Yang, Z.-Y., Doria-Rose, N. A., McKee,

- K., O'Dell, S., Schmidt, S. D., Chuang, G.-Y., Druz, A., Soto, C., Yang, Y., Zhang, B., Zhou, T., Todd, J.-P., Lloyd, K. E., Eudailey, J., Roberts, K. E., Donald, B. R., Bailer, R. T., Ledgerwood, J., Program, N. C. S., Mullikin, J. C., Shapiro, L., Koup, R. A., Graham, B. S., Nason, M. C., Connors, M., Haynes, B. F., Rao, S. S., Roederer, M., Kwong, P. D., Mascola, J. R., and Nabel, G. J. (2014). Enhanced potency of a broadly neutralizing HIV-1 antibody *in vitro* improves protection against lentiviral infection *in vivo*. *Journal of Virology*, 88(21):12669–12682.
- [44] Simoncini, D., Allouche, D., de Givry, S., Delmas, C., Barbe, S., and Schiex, T. (2015). Guaranteed discrete energy optimization on large protein design problems. *Journal of Chemical Theory and Computation*, 11(12):5980–5989.
- [45] Stevens, B. W., Lilien, R. H., Georgiev, I., Donald, B. R., and Anderson, A. C. (2006). Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme’s mechanism and selectivity. *Biochemistry*, 45(51):15495–15504.
- [46] Traoré, S., Allouche, D., André, I., de Givry, S., Katsirelos, G., Schiex, T., and Barbe, S. (2013). A new framework for computational protein design through cost function network optimization. *Bioinformatics*, 29(17):2129–2136.
- [47] Traoré, S., Roberts, K. E., Allouche, D., Donald, B. R., André, I., Schiex, T., and Barbe, S. (2016). Fast search algorithms for computational protein design. *Journal of Computational Chemistry*, 37(12):1048–1058.
- [48] Wang, C., Schueler-Furman, O., and Baker, D. (2005). Improved side-chain modeling for protein-protein docking. *Protein Science*, 14:1328–1339.
- [49] Word, J. M., Lovell, S. C., LaBean, T. H., Taylor, H. C., Zalis, M. E., Presley, B. K., Richardson, J. S., and Richardson, D. C. (1999). Visualizing and quantifying molecular goodness-of-fit: small-probe contact dots with explicit hydrogen atoms. *Journal of Molecular Biology*, 285(4):1711–1733.
- [50] Xu, J. and Berger, B. (2006). Fast and accurate algorithms for protein side-chain packing. *Journal of the ACM*, 53(4):533–557.

- [51] Yanover, C., Fromer, M., and Shifman, J. M. (2007). Dead-end elimination for multi-state protein design. *Journal of Computational Chemistry*, 28(13):2122–2129.

Table 1: Protein design test cases for examining the pruning power of PLUG and iMinDEE¹⁰. Each design has N_m mutable residues and a total of N_r RCs across all residues, of which D_I remain unpruned after pruning with iMinDEE using a pruning interval of I kcal/mol and P_i remain unpruned after pruning with both PLUG and iMinDEE using a pruning interval of I kcal/mol. In every case, P_{10^6} is equal to the number of RCs that PLUG pruning without iMinDEE leaves unpruned, because iMinDEE with a pruning interval of 1000000 prunes only RCs that PLUG alone (which does not use a pruning interval) can prune. “Flex” denotes the level of backbone flexibility for the test case (C=CATS; D=DEEPER; N=None). Test cases where PLUG pruned all conformations are omitted from this table, and instead listed in Table 2. Table continues on next 4 pages.

Protein name	PDB id	N_m	Flex	N_r	D_0	P_0	D_5	P_5	D_{10}	P_{10}	D_{15}	P_{15}	D_{20}	P_{20}	D_{10^6}	P_{10^6}
Atx1 metal-lochaperone	1CC8	7	C	127	7	7	95	86	112	92	120	96	120	96	126	96
Bucandin	1F94	7	C	110	7	27	57	42	76	46	83	58	90	59	99	59
Transcription factor IIF	1I27	7	C	110	7	7	75	49	89	58	93	63	94	63	97	63
Ferredoxin	1IQZ	9	C	108	9	9	48	47	86	52	100	52	102	52	106	52
Fructose-6-phosphate aldolase	1L6W	6	C	54	6	6	30	22	41	37	50	37	51	41	52	41
PA-I lectin	1L7L	6	C	72	6	6	39	28	53	33	59	37	59	37	69	37
alpha-D-glucuronidase	1L8N	5	C	81	5	5	7	6	14	8	34	27	47	29	64	29
Granulysin	1L9L	7	C	28	7	7	18	13	20	13	22	14	22	14	25	14
gamma-glutamyl hydrolase	1L9X	5	C	9	5	5	9	9	9	9	9	9	9	9	9	9
Ferritin	1LB3	5	C	99	5	5	32	20	38	25	41	25	45	25	63	25
Cytochrome c	1M1Q	8	C	126	8	10	86	80	101	82	106	86	110	86	116	86
Transcriptional regulator AhrC	2P5K	6	C	94	6	6	75	64	81	69	85	70	85	70	88	70
Scytovirin	2QSK	5	C	82	5	5	69	60	74	66	77	68	78	68	80	68

Protein name	PDB id	N_m	Flex	N_r	D_0	P_0	D_5	P_5	D_{10}	P_{10}	D_{15}	P_{15}	D_{20}	P_{20}	D_{10^6}	P_{10^6}
Putative monooxygenase	2RIL	7	C	110	7	7	72	58	83	62	88	65	88	65	93	65
alpha-crystallin	2WJ5	6	C	87	6	6	63	52	75	56	75	56	77	56	79	56
Cytochrome c555	2ZXY	7	C	84	7	7	50	46	63	52	73	52	75	52	80	52
High-potential iron-sulfur protein	3A38	6	C	118	6	6	41	35	60	37	72	37	75	37	86	37
Scorpion toxin	1AHO	7	D	176	7	7	58	54	89	78	98	80	99	80	140	80
Atx1 metal-lochaperone	1CC8	7	D	127	7	7	84	72	99	78	110	82	111	82	120	82
Bucandin	1F94	7	D	110	7	16	34	43	55	52	69	55	71	56	82	56
Nonspecific lipid-transfer protein	1FK5	6	D	86	6	6	36	30	53	39	54	39	54	39	72	39
Transcription factor IIF	1I27	7	D	110	7	7	55	38	69	48	72	50	72	50	80	50
Ferredoxin	1IQZ	9	D	108	9	9	53	48	68	56	69	56	76	56	98	56
Fructose-6-phosphate aldolase	1L6W	6	D	54	6	6	36	29	41	34	46	38	47	38	50	38
PA-I lectin	1L7L	6	D	72	6	6	28	19	38	26	43	28	45	28	49	28
Phosphoserine phosphatase	1L7M	7	D	112	7	7	36	19	48	20	53	20	58	20	71	20
alpha-D-glucuronidase	1L8N	5	D	81	5	5	7	6	7	6	27	26	31	28	55	28
gamma-glutamyl hydrolase	1L9X	5	D	9	5	5	9	9	9	9	9	9	9	9	9	9
Ferritin	1LB3	5	D	99	5	5	28	18	37	24	38	24	40	24	54	24
Cytochrome c	1M1Q	8	D	126	8	8	72	57	86	77	92	81	95	81	100	81
Sulfite oxidase	1MJ4	7	D	329	7	7	62	57	97	69	108	69	128	69	214	69
Histidine triad protein	2CS7	8	D	132	8	8	73	64	95	67	99	67	100	69	104	69

Protein name	PDB id	N_m	Flex	N_r	D_0	P_0	D_5	P_5	D_{10}	P_{10}	D_{15}	P_{15}	D_{20}	P_{20}	D_{10^6}	P_{10^6}
Ponsin	2O9S	7	D	124	7	7	50	42	78	54	87	59	90	59	103	59
Transcriptional regulator AhrC	2P5K	6	D	94	6	6	51	45	78	60	84	61	84	62	87	62
Scytovirin	2QSK	5	D	82	5	5	57	55	67	62	69	62	74	62	79	62
Hemolysin	2R2Z	9	D	124	9	9	52	47	67	58	72	58	77	58	95	58
Dihydrofolate reductase	2RH2	6	D	91	6	6	51	46	54	46	61	50	63	50	70	50
Putative monooxygenase	2RIL	7	D	110	7	7	56	48	63	50	68	53	74	53	81	53
alpha-crystallin	2WJ5	6	D	87	6	6	41	39	58	48	59	48	59	48	70	48
Cytochrome c555	2ZXY	7	D	84	7	7	39	39	47	42	50	42	51	42	72	42
High-potential iron-sulfur protein	3A38	6	D	118	6	6	28	21	36	22	44	22	46	22	60	22
Scorpion toxin	1AHO	7	N	176	7	7	54	53	85	76	91	79	92	79	126	79
Atx1 metal-lochaperone	1CC8	7	N	127	7	7	78	68	92	76	100	80	109	80	119	80
Bucandin	1F94	7	N	110	7	16	33	41	49	49	66	52	69	53	81	53
Nonspecific lipid-transfer protein	1FK5	6	N	86	6	6	28	28	47	30	52	30	52	30	62	30
Transcription factor IIF	1I27	7	N	110	7	7	47	23	63	29	68	31	68	31	77	31
Ferredoxin	1IQZ	9	N	108	9	9	44	35	64	41	67	41	67	41	96	41
Fructose-6-phosphate aldolase	1L6W	6	N	54	6	6	34	30	38	34	42	38	42	38	49	38
PA-I lectin	1L7L	6	N	72	6	6	26	18	36	23	40	25	42	25	48	25
Phosphoserine phosphatase	1L7M	7	N	112	7	7	31	18	47	19	49	19	50	19	69	19
gamma-glutamyl hydrolase	1L9X	5	N	9	5	5	9	8	9	8	9	8	9	8	9	8

Protein name	PDB id	N_m	Flex	N_r	D_0	P_0	D_5	P_5	D_{10}	P_{10}	D_{15}	P_{15}	D_{20}	P_{20}	D_{10^6}	P_{10^6}
Ferritin	1LB3	5	N	99	5	5	27	19	36	24	36	24	36	24	48	24
Cytochrome c	1M1Q	8	N	126	8	8	57	42	75	53	93	56	95	56	99	56
Sulfite oxidase	1MJ4	7	N	329	7	7	61	52	93	62	105	62	106	62	207	62
Histidine triad protein	2CS7	8	N	132	8	8	66	60	93	60	96	60	98	63	102	63
Transcriptional regulator AhrC	2P5K	6	N	94	6	6	48	43	72	52	80	55	82	55	87	55
Scytovirin	2QSK	5	N	82	5	5	58	56	70	61	74	64	74	64	78	64
Dihydrofolate reductase	2RH2	6	N	91	6	6	49	41	53	41	56	45	58	45	68	45
alpha-crystallin	2WJ5	6	N	87	6	6	49	42	57	46	58	46	59	46	64	46
Cytochrome c555	2ZXY	7	N	84	7	7	38	34	45	36	47	36	49	36	66	36
ygfY	1X6I	14	N	847	176	110	314	156	369	168	382	168	395	168	470	168
Histidine triad protein	2CS7	14	N	882	365	37	440	162	483	228	508	254	523	258	613	258
Ponsin	2O9S	14	N	714	130	22	254	121	315	139	346	141	359	141	424	141
Transcriptional regulator AhrC	2P5K	11	N	710	29	11	227	124	266	144	284	149	341	157	423	160
Scytovirin	2QSK	10	N	845	10	10	28	27	36	29	56	42	79	45	239	68
Hemolysin	2R2Z	12	N	625	360	105	383	142	388	145	392	145	392	145	411	145
Dihydrofolate reductase	2RH2	14	N	887	399	139	519	186	553	227	567	236	574	238	603	238
alpha-crystallin	2WJ5	15	N	823	428	134	491	205	499	215	512	215	518	215	610	215
Cytochrome c555	2ZXY	14	N	807	92	14	201	119	260	140	297	144	312	145	450	145
High-potential iron-sulfur protein	3A38	13	N	846	170	13	244	108	270	128	333	140	356	156	521	198
ClpS protease adaptor	3DNJ	12	N	693	132	65	275	124	309	134	325	137	347	137	463	137
Putative monooxygenase	3FGV	10	N	710	262	10	276	47	299	111	314	135	329	149	411	153

Protein name	PDB id	N_m	Flex	N_r	D_0	P_0	D_5	P_5	D_{10}	P_{10}	D_{15}	P_{15}	D_{20}	P_{20}	D_{10^6}	P_{10^6}
Protein G	3FIL	14	N	790	437	181	489	242	536	265	553	271	560	271	627	271
Viral capsid	3G21	15	N	971	364	103	450	213	483	225	546	230	551	230	634	230
dpy-30-like protein	3G36	4	N	310	4	4	24	12	48	17	61	28	77	42	189	71
Hfq protein	3HFO	10	N	608	253	24	303	125	319	139	325	140	333	140	368	140
Cold shock protein	3I2Z	14	N	749	411	86	431	158	437	168	441	172	445	173	457	173

Table 2: Protein design test cases in which PLUG pruned all conformations. Abbreviations as in Table 1.

Protein name	PDB id	N_m	Flex	N_r
Nonspecific lipid-transfer protein	1FK5	6	C	86
Cephalosporin C deacetylase	1L7A	8	C	108
Cephalosporin C deacetylase	1L7A	8	D	108
Cephalosporin C deacetylase	1L7A	8	N	108
Phosphoserine phosphatase	1L7M	7	C	112
alpha-D-glucuronidase	1L8N	5	N	81
Dachshund	1L8R	8	C	84
Dachshund	1L8R	8	D	84
Dachshund	1L8R	8	N	84
Granulysin	1L9L	7	D	28
Granulysin	1L9L	7	N	28
Ponsin	2O9S	7	C	124
Ponsin	2O9S	8	N	124
Hemolysin	2R2Z	9	N	124
Dihydrofolate reductase	2RH2	6	C	91
Putative monooxygenase	2RIL	7	N	110
High-potential iron-sulfur protein	3A38	6	N	118
Putative monooxygenase	2RIL	8	N	532
HPI integrase	3JTZ	14	N	962

Table 3: Test cases with CATS backbone flexibility for learning a pairwise LUTE energy matrix. The residuals are the mean-square errors of LUTE’s prediction of continuously minimized voxel energies (in kcal/mol). iMinDEE is used in all test cases, while PLUG is used only when indicated.

Protein name	PDB id	N_m	Residual w/ PLUG	Residual w/o PLUG
Cytochrome c553	1C75	6	0.04	0.19
Atx1 metallochaperone	1CC8	7	0.15	141.95
Bucandin	1F94	7	0.22	78175.22
Fructose-6-phosphate aldolase	1L6W	6	0.01	0.21
PA-I lectin	1L7L	6	0.07	4.86
alpha-D-glucuronidase	1L8N	5	8.78×10^{-14}	5.98×10^{-11}
Granulysin	1L9L	7	1.23×10^{-7}	0.03
gamma-glutamyl hydrolase	1L9X	5	6.05×10^{-7}	2.59×10^{-6}
Ferritin	1LB3	5	1.14×10^{-6}	1.21
Transcriptional regulator AhrC	2P5K	6	0.08	12.99
Scytovirin	2QSK	5	0.0034	0.02
alpha-crystallin	2WJ5	6	0.0012	1680.14
Cytochrome c555	2ZXY	7	0.19	2979.09

Table 4: Test cases learning a pairwise LUTE energy matrix using an energy model that does not support DEE pruning—only PLUG is possible here. The residuals are the mean-square errors of LUTE’s prediction of voxel energies (in kcal/mol). For the PB test cases, the voxel energies are the sum of the AMBER energy and a Poisson-Boltzmann implicit solvation term, minimized over voxels using CATS backbone flexibility. For the Integ test cases, the voxel energies are free energies computed by integrating the Boltzmann distribution corresponding to OSPREY’s default AMBER/EEF1 energy function over a voxel with no backbone flexibility.

Protein name	PDB id	N_m	PB residual	Integ residual
Scorpion toxin	1AHO	7	0.13	5.85×10^{-4}
Cytochrome c553	1C75	6	0.09	6.53×10^{-4}
Atx1 metallochaperone	1CC8	7	0.26	0.0031
Bucandin	1F94	7	0.13	0.0011
Transcription factor IIF	1I27	7	0.35	0.0042
Ferredoxin	1IQZ	9	0.04	9.16×10^{-4}

Figure 1: **Example of a linearized steric clash, with error estimation.** Suppose the position of atom 1 is determined by two orthogonal, continuous degrees of freedom x_1 and x_2 . Atom 1 must avoid a clash with atom 2, whose center (in the plane of x_1 and x_2) is at C. The orange arc, centered at C, denotes the boundary of clashing conformational space (by a strict definition⁴⁹), at which atom 1's van der Waal sphere overlaps atom 2's by 0.4 Å. The black dotted line is tangent to the orange arc at point A, and is a linear approximation to the orange arc and thus to the clash. The error in this linear approximation within the voxel ranges from 0 (at point A) to the distance between points B and B', at the edge of the voxel. The radius of the orange arc is typically ~ 1 Å, while point B will generally be $\lesssim 0.5$ Å from A even with continuous backbone flexibility (see Ref. 19), meaning there are $\lesssim 0.5$ radians of arc between A and B. We can thus estimate the error in the linear approximation as $1/\cos(0.5)-1 = 0.14$ Å (occurring when B, B' and C are collinear with A and B 0.5 radians apart on an orange arc of radius 1 Å), meaning B' lies within the purple arc, representing a less strict definition of a clash (0.25 Å of steric overlap⁴⁹). Thus the linear approximation itself is a biophysically valid definition of a clash, when used within the bounds of the voxel.

Figure 2: **Steric clashes can be approximated as linear constraints.** Each graph shows the region of conformational space where a particular atom pair is clashing (red), shown in the plane of the first two continuous degrees of freedom. The length and width of the graph are set to the voxel size. These are the first six atom pairs in the PLUG matrix for the optimal voxel of a design on PDB id 1CC8⁴² from Ref. 20.

Figure 3: **Computation of a linearized clash.** Newton's method starting at the center (green X) of the voxel (dark blue rectangle) leads to a point (blue X) on the boundary (orange) of the sterically clashing region, where a tangent plane (the linearized clash constraint; black) can be computed by numerical differentiation.

Figure 4: **PLUG allows accurate energy fitting using LUTE²¹ for designs with CATS¹⁹ backbone flexibility.** Residuals for LUTE fitting ((kcal/mol)²) after pruning with both PLUG and iMinDEE (blue) vs. only with iMinDEE (red); test cases marked by PDB id. Top: logarithmic scale. Bottom: linear scale with horizontal gridline spacing set to the thermal energy at room temperature; thus test cases with residuals below the first gridline (including all PLUG cases) have negligible error, while residuals overshooting this graph likely indicate an unusable fit.

Figure 5: **The pruning power of PLUG is similar to that of DEE.** Fraction of RCs unpruned after pruning by iMinDEE (red) or iMinDEE and PLUG together (blue), as a function of the pruning interval used for iMinDEE. Pruning was performed using the AMBER energy function³, which is pairwise and thus admits iMinDEE pruning (PLUG pruning is not dependent on the energy function). The green line shows the pruning power of PLUG alone, which is independent of the pruning interval (this is equivalent to the rightmost blue dot, because with an infinite pruning interval, iMinDEE only prunes the most obvious steric clashes). Values are the average over 17 design test cases with CATS¹⁹ backbone flexibility (left), 24 cases with DEEPer²² backbone flexibility (middle), or 36 cases without backbone flexibility (right). Test cases where PLUG pruned all RCs are not included.

Figure 6: **For a design test case deemed impossible by PLUG, the global minimum-energy conformation clashes sterically.** PLUG pruned all RCs for the test case from Ref. 19 that redesigned a nonspecific lipid-transfer protein from corn (PDB id 1FK5). When the global minimum-energy conformation was computed without PLUG¹⁹, a biophysically unrealistic steric clash (pink) is observed.

Figure 1

Figure 2

Figure 3

Figure 4

Figure 5

Figure 6