Supplementary Information: *BBK** (Branch and Bound over K^*): A Provable and Efficient Ensemble-Based Protein Design Algorithm to Optimize Stability and Binding Affinity over Large Sequence Spaces

Adegoke A. Ojewole[1,2,†], Jonathan D. Jou[2,†], Vance G. Fowler, Jr.[3] and Bruce R. Donald[2,4,5,*]

[1]Program in Computational Biology and Bioinformatics, Duke University, Durham, NC 27708
[2]Department of Computer Science, Duke University, Durham, NC 27708
[3]Department of Infectious Diseases, Duke University Medical Center, Durham, NC 27710
[4]Department of Biochemistry, Duke University Medical Center, Durham, NC 27710
[5]Department of Chemistry, Duke University, Durham NC 27708
[†]These authors contributed equally to the work.
*Corresponding author, brd+recomb17@cs.duke.edu

The following is supplementary information for the following paper:

A.A. Ojewole, J.D. Jou, V.G. Fowler, and B.R. Donald. *BBK**: A Provable and Efficient Ensemble-based Protein Design Algorithm to Optimize Stability and Binding Affinity over Large Sequence Spaces. Journal of Computational Biology. (2017).

Every section of the paper has been provided with a corresponding and extended supplementary information section to provide full details and substantiate the claims of the paper.

1 Summary of Supplementary Information Material

Protein design algorithms that compute binding affinity search for sequences with an energetically favorable free energy of binding. Recent work shows that the biological accuracy of such protein designs is improved by the following design principles: *ensemble-based design* and *continuous conformational flexibility*. Unlike many algorithms, which compute only the Global Minimum Energy Conformation (GMEC), ensemble-based algorithms capture entropic contributions to binding by approximating K_a as the ratio of bound to unbound state partition functions over molecular ensembles. And, compared to conventional protein designs, which fix the backbone and limit side-chain flexibility to discrete, rigid rotational isomers (i.e. rotamers), design with backbone flexibility and continuous side-chain rotamers better accounts for conformational changes induced by amino acid and rotamer substitutions during design. A third design principle, *provable guarantees of accuracy*, ensures that an algorithm computes the best sequences defined by the *input model* (i.e. input structures, energy function, and allowed protein flexibility). Although modeling ensembles and continuous flexibility consequentially improves the predictive power of protein designs, previous provable methods model these phenomena at significant cost to both empirical and asymptotic runtime. Furthermore, previous provable methods are *single-sequence* algorithms, which require (a) separate analysis of each sequence's conformation space and (b) exhaustive enumeration of all possible sequences. Exhaustive enumeration, the current state of the art, is very costly: linear in the number of sequences and thus exponential in the number of mutable residues, making the approach computationally intractable for large sequence spaces. To address these computational challenges, we introduce a new protein design algorithm,

BBK*, that retains all aforementioned design principles yet provably and efficiently computes the tightest binding sequences without exhaustively searching all possible sequences. A key innovation of *BBK** is the *multi-sequence* (MS) bound: rather than compute binding affinity separately for each possible sequence, as single-sequence methods do, *BBK** efficiently computes a single provable upper bound to approximate K_a for *a combinatorial number of sequences*. *BBK** uses MS bounds to prune a combinatorial number of sequences during the search, entirely avoiding single-sequence computation for all pruned sequences. This combinatorial pruning produces a significant empirical runtime speedup: *BBK** runs in time *sub-linear* in the number of sequences. To our knowledge, *BBK** is the first provable ensemble-based algorithm to do so. Computational experiments on 204 protein design problems showed that *BBK** finds the tightest binding sequences while approximating K_a for up to 10^5-fold fewer sequences than the previous state-of-the-art algorithms, which require exhaustive enumeration of sequences. Furthermore, in 51 protein-ligand design problems, we showed that *BBK** provably approximates K_a up to 1982-fold faster than the previous state-of-the-art iMinDEE/A^*/K^* algorithm. Therefore, *BBK** not only accelerates protein designs that are possible with previous provable algorithms, it also efficiently performs designs that are too large for previous methods.

2 Supplementary Introduction

Protein design is the prediction of protein sequences with desired biochemical functions, which often involve binding to a target ligand. Computational protein design casts the functional design problem into a structural optimization problem whose goal is to find amino acid sequences that fold into a specified three-dimensional structure. Protein design algorithms search a space defined by a biophysical *input model*, which defines the sequence and structural search space (i.e. the input structure, allowed amino acid mutations, and allowed protein flexibility), the optimization objective (e.g. design for binding affinity), and the energy function [2]. Protein design algorithms [7, 11] have successfully predicted protein sequences that fold and bind desired targets *in vitro* and *in vivo*. For example, these algorithms have been used, with experimental validation, to predict drug resistance [9, 38, 43] and design enzymes [4, 15, 34, 53], new drugs [21], inhibitors of protein-protein interactions [17, 44], epitope-specific antibody probes [14], and neutralizing antibodies [19, 47].

Computational methods can potentially search a large number of sequences to predict the proteins that most tightly bind a target ligand in less time and with fewer resources than *in vitro* methods such as phage display [3,40]. However, four computational challenges have prevented protein design algorithms from realizing this potential. First, for each binding interface, an exponentially large number of conformations in each binding partner's ensemble must be pruned or considered to accurately predict binding affinity [7,17,20,34]. Second, for each sequence, finding the lowest energy conformations that most influence binding affinity is NP-hard [28,41,42,59,60], making algorithms that guarantee optimality expensive for larger designs. Third, mutating a protein sequence (as many algorithms do) induces conformational changes in the protein structure. Since such conformational changes occur over many continuous degrees of freedom, algorithms that model continuous flexibility must search over a large, continuous conformation space. Fourth, the number of protein sequences (i.e. the *sequence space*) grows exponentially with the number of simultaneously mutable residues in the design. Therefore, previous algorithms either focus on accurately modeling smaller designs or attempt larger designs by making simplifications that (a) ignore the ensemble nature of proteins, (b) disregard continuous conformational flexibility, or (c) return heuristic solutions with no guarantees. A discussion of these simplifications and their ramifications for protein design follows.

Global Minimum Energy Conformation (GMEC)-based algorithms [6, 12, 22, 45, 57] assume that the lowest energy conformation accurately predicts binding affinity. However, GMEC-based design ignores the reality that proteins at physiological conditions exist as thermodynamic ensembles, whose very nature governs binding [20]. Therefore, GMEC-based designs cannot accurately model entropic change due to binding [8] and can disproportionately favor sequences with energetically favorable GMECs over sequences

with tight binding affinity [4,17,34,44,48]. In fact, recent studies show that ensemble-based binding predictions more closely correlate with *in vitro* binding measurements than do GMEC-based predictions [44, 51].

Although protein flexibility plays a key role in binding, many protein design algorithms [31,52,54,55,57] rely on a simplified, discrete model of side-chain flexibility. Widely used due to computational convenience, the discrete model abstracts frequently observed low-energy regions of χ-angle space, known as *rotamers*, to single points. However, discrete rotamers model a small subset of protein energetics, which are sensitive to small atomic movements not permitted by the discrete model. To overcome this limitation, researchers have developed provable algorithms [12, 17, 23, 24, 44, 45] that incorporate *continuous rotamers* [12, 17], which model continuous flexibility in the low-energy torsional regions around discrete rotamers. The DEEPer algorithm [25] generalizes continuous rotamers by modeling coupled continuous backbone and side-chain flexibility. Despite numerous recent algorithmic improvements [23, 24, 45], protein design with continuous rotamers incurs some computational overhead. Nevertheless, continuous rotamers more accurately represent side-chain conformation space, clearly finding conformations with lower energies than those found using discrete rotamers [11, 12]. Furthermore, continuous rotamers improved the biological accuracy of designs [44], finding sequences more similar to native sequences than those found with discrete rotamers [12]. These results suggest that continuous side-chain flexibility is necessary for more accurate design predictions.

Another important aspect of design algorithms is the quality of the computed results. Whereas GMEC-based design is NP-hard [28, 42, 59, 60], computation of thermodynamic ensembles and associated partition functions is #P-hard [37, 56, 57]. Provable protein design algorithms either return the optimal sequences or conformations [6, 12, 25, 30, 45, 52, 54, 55, 57] with respect to the input model, or return provably good approximate solutions [17, 23, 24, 34, 44]. Non-provable algorithms such as Metropolis Monte Carlo methods [29, 31, 32] instead use stochastic methods to rapidly sample the space described by the input model. These algorithms are popular for their speed, ease of implementation, and amenability to complex input models (i.e. backbone flexibility [25, 39] and dense rotamer libraries [49, 50]) but return solutions without any guarantees. Indeed, a recent large-scale study [52] showed that a popular non-provable simulated annealing algorithm [31] computed the best sequence for fewer than 30% of the smallest design problems and consistently failed to compute the best sequence in all large design problems.

Thus, the predictive power of protein design algorithms is improved when the following design principles are incorporated: ensemble-based design, a realistic model of structural flexibility, and provable optimality of the computed sequences with respect to the input model. Although use of each design principle increases the *per-sequence* cost of protein design, designs typically involve searching over a large number of sequences. *Single-sequence* algorithms, which explicitly evaluate each possible sequence, are powerful and versatile. Molecular dynamics [33, 61], for instance, is frequently applied to design for binding affinity. The approach utilizes two design principles (i.e. ensemble-based design and continuous flexibility) in our desiderata. However, one can benefit from all three design principles and avoid the large computational cost of simulation. The K^* algorithm [17, 34, 44] in OSPREY [12, 13, 15–17, 22–25, 27, 34, 44, 45] uses a combination of dead-end elimination pruning [12, 17] and A^* [26, 30, 46] gap-free conformation enumeration to provably and efficiently approximate K_a as the ratio of ε-approximate partition functions between the bound and unbound states of a protein-ligand complex. Although K^* is considerably more efficient than exhaustive conformation enumeration for all possible sequences, K^* and all previous provable ensemble-based algorithms that model continuous side-chain flexibility [23, 24] are single-sequence algorithms. The empirical and asymptotic runtime complexity of single-sequence algorithms is linear in the number of possible sequences, and therefore exponential in the number of mutable residues. As the number of mutable residues increases, the space of possible sequences increases exponentially, and designs with many mutable residues rapidly become intractable when using single-sequence algorithms. The COMETS algorithm [22] in OSPREY is the only provable multi-state design algorithm that computes the optimal sequences without exhaustively searching the design sequence space. Although it supports continuous conformational flexibility, its binding predictions are GMEC-based rather than ensemble-based.

To efficiently search large sequence spaces while retaining all the benefits of provable guarantees, ensemble-based design, and continuous side-chain flexibility, we present a new, provable algorithm: Branch and Bound over K^* (*BBK**). The key innovation of *BBK** is the *multi-sequence* (MS) bound: rather than approximate binding affinity for each possible sequence, *BBK** efficiently computes provable upper and lower bounds on the binding affinities of *partial sequences*, which are shared by a combinatorial number of full sequences. Single-sequence methods are unable to compare partial sequences. Instead, they must compute and compare single-sequence bounds for multiple combinatorially large sets of sequences: one large set for each partial sequence. In contrast, *BBK** compares the partial sequences directly by computing an MS bound for each partial sequence, and pruning provably worse partial sequences (and therefore combinatorially large sets of full sequences) so they do not have to be searched or enumerated. In doing so, it rapidly prunes a vast portion of the sequence space, and entirely avoids single-sequence computation for all pruned sequences. This combinatorial pruning results in a significant empirical runtime speedup: *BBK** runs in time *sub-linear* in the number of sequences. To our knowledge, *BBK** is the first provable, ensemble-based algorithm to do so. *BBK** not only avoids explicitly computing all possible sequences, but also provably enumerates sequences in a gap-free decreasing order of binding affinity. In contrast, all previous provable ensemble-based algorithms – including the state-of-the-art iMinDEE/A^*/K^* algorithm [12, 17] – are single-sequence algorithms whose running time is linear in the number of sequences in the design space. Moreover, *BBK** performs each binding affinity calculation much more rapidly. Therefore, *BBK** provides a vast performance improvement over the previous state-of-the-art, by not only accelerating protein designs that were possible with previous provable algorithms, but also efficiently handling large designs that previous algorithms could not compute in a reasonable amount of time.

By presenting *BBK**, our paper makes the following contributions:

1. A novel, ensemble-based algorithm that provably computes the same results as the previous state of the art (exhaustive search over sequences) but is combinatorially faster, returns a gap-free list of sequences in decreasing order of binding affinity and runs in time sub-linear in the number of sequences.
2. Proofs of correctness for multi-sequence bounds, a key innovation in *BBK**.
3. A new two-pass bound that more efficiently computes a provable partition function ε-approximation.
4. 255 protein designs showing that *BBK** approximates binding affinity for complete sequences up to 1982-fold faster than the best previous algorithm and that *BBK** computes the best binding sequences in a large sequence space up to 10^5-fold more efficiently than exhaustive search.
5. Support for both continuous side-chain and backbone flexibility, demonstrating the ability of *BBK** to handle multiple modes of protein flexibility in addition to large conformation and sequence spaces.
6. An implementation of *BBK** in our laboratory's open-source OSPREY [12, 13, 15–17, 22–25, 27, 34, 44, 45] protein design software package, available for download upon publication as free software.

3 Computing the Partition Function

To successfully design for improved binding affinity K_a, design algorithms must consider the energy of more than just the GMEC. In particular, all algorithms that design for improved K_a optimize the ratio of partition function Z for the bound and unbound states of the protein and ligand (Eq. 2). Protein design can be cast as an optimization problem. For an n-residue protein design problem with at most a amino acids per mutable residue, let P, L, and PL denote the unbound protein, unbound ligand, and bound protein-ligand complex, respectively. For each sequence s, let $\mathbf{Q}(s)$ be the set of discrete rigid rotamer conformations defined by the allowed amino acids for each mutable residue of s. For a rigid rotamer conformation c, let E_X be a pairwise energy function with respect to input structure X, which may be one of P, L, or PL. In particular, we will consider the case of design with *continuous rotamers* [12, 17]. We define $E_X(c)$ to be the energy of c for structure X after energy-minimizing the side-chains of mutable residues.

3.1 K^*

We define the Boltzmann-weighted partition function $Z_{_X}(s)$ as:

$$Z_{_X}(s) = \sum_{c \in \mathbf{Q}(s)} \exp(-E_{_X}(c)/RT). \tag{1}$$

We define the K^* score, a partition function ratio that approximates binding affinity K_a, as:

$$K^*(s) = \frac{Z_{_{PL}}(s)}{Z_{_P}(s)Z_{_L}(s)}. \tag{2}$$

As stated in Sec. 2, the K^* approximation and, by extension, the full partition function, are #P-hard to compute [37, 56, 57]. Therefore, researchers have not only developed heuristic algorithms that rapidly compute loose partition function bounds, but also developed efficient, provable algorithms that compute ε-approximations to the partition function. Probabilistic algorithms bound the partition function either provably [58] or non-provably [10]. An efficient ε-approximation to $Z_{_X}(s)$ is computed in [17, 34, 44]. However, these methods are designed to compute partition functions for single sequences. For an n-residue design with at most t possible amino acids at each residue and q rotamers per amino acid, provable single-sequence methods must compute or bound the partition functions of all t^n sequences, each with q^n conformations. Thus, previous single-sequence algorithms for protein design for binding affinity are exponential in n ($\mathcal{O}(t^n)$) when computing the sequence with the best predicted binding affinity.

Therefore, to provably find the best binding sequences, new, efficient provable algorithms are needed to search over an exponentially large sequence space, for which each sequence represents an exponentially large conformation space. ***BBK**** addresses this need. ***BBK**** compares *partial sequences* (for which some mutable residues have not been assigned an amino acid identity) without computing the partition functions for all full sequences (which assign an amino acid to each mutable residue). ***BBK**** computes bounds on the free energies of partial sequences, and avoids enumerating conformations from sequences with poor binding affinity, by pruning sequences during search. As we will describe in Sec. 4, pruning these sequences circumvents prohibitive computational costs required to compute many single-sequence K^* scores.

4 A^* Search Over Sequences, with Multi-sequence (MS) bounds

It may at first seem counter-intuitive to compute the sequence with optimal binding affinity, along with its predicted K^* score, without explicitly computing the K^* scores of all possible sequences. Indeed, all previous ensemble-based provable methods, as well as many heuristic methods, are *single-sequence* methods: they must individually evaluate and compare each sequence to provably return the optimal sequence. In contrast, ***BBK**** bounds the K^* ratios of a combinatorial number of sequences efficiently and can prune these sequences without computing any single-sequence bounds. The key to this improvement is the observation that a partial sequence s' with poor predicted binding affinity adversely affects the K^* score of the combinatorial set of sequences that contain s'. That is, the best possible K^* score consistent with s' limits the K^* ratio of all sequences consistent with s'. Henceforth, we will refer to a bound on the binding affinity for a sequence as a *bound on the sequence*. To compute a bound on all sequences consistent with s', ***BBK**** computes the partition function for an ensemble that contains conformations from multiple sequences. Fig. 1 illustrates the difference between single-sequence and multi-sequence ensembles. The K^* ratio of a multi-sequence ensemble is a provable upper bound on the best possible K^* ratio of all sequences that contain s'. We show this *multi-sequence* bound (MS bound) is not only cheaper to compute, but it also allows ***BBK**** to compare a combinatorial number of sequences without computing any single-sequence bounds. By bounding every possible sequence consistent with a partial sequence, ***BBK**** can provably eliminate those sequences, and prune a combinatorial number sequences without performing any single-sequence computation. Fig. 2 illustrates the combinatorial speedup provided by MS bound pruning, whereby pruning the

partial sequence obviates computation of all single sequences containing the partial sequence. MS bounds are not limited to pruning based on binding affinity, however. They are also useful for pruning combinations of sequences whose *favorable* K^* scores are due to energetically *unfavorable* unbound states that cannot fold into ligand-binding poses. Single-sequence methods compute binding affinity for each of these sequences. In contrast, *BBK** prunes these partial sequences using MS bounds, thereby avoiding many costly single-sequence computations. Details are provided in Appendix A.7.

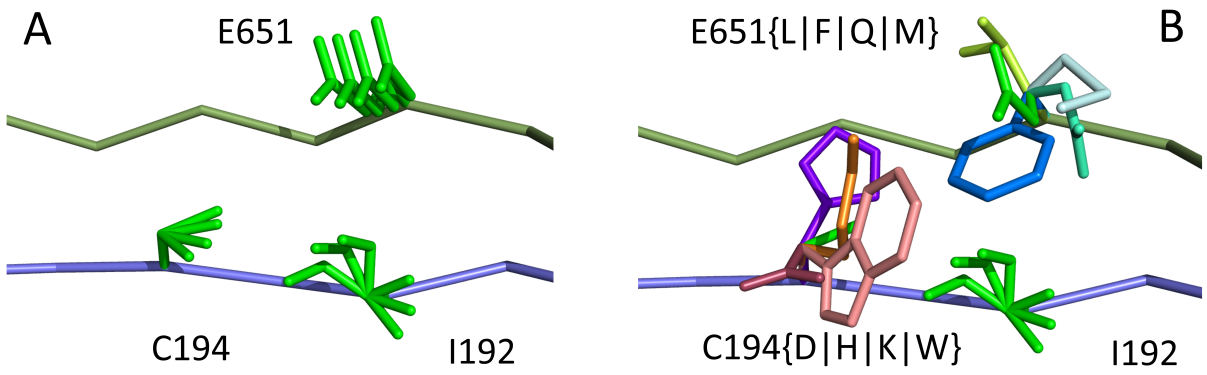

Figure 1: **A toy protein design problem in which conformational ensembles (A) and optimal mutations (B) must be computed at 3 residues.** Residues of the fibronectin F1 module (Fn, blue ribbon), and of a fragment of *S. aureus* fibronectin binding protein A (FNBPA-5, green ribbon) are shown (PDB Id: 2RL0). Side-chain conformations, labeled with amino acid identity, are also shown per residue. (A) Previous provable methods require a *fully defined sequence* to compute a *single-sequence (SS)* ε-approximation bound on binding affinity (i.e. a K^* score, Eq. 2). (B) **A key innovation in this paper is the *multi-sequence (MS) bound* for binding affinity in protein design.** An MS bound is a provable bound on the binding affinity of a *partial sequence*. Unassigned residues, whose amino acid identities are not defined by the partial sequence, adopt side-chain conformations from multiple amino acids, shown as the blue, purple, pink, and light blue ensemble. Thus, an MS bound is a provable upper bound on the binding affinity of all sequences containing that partial sequence, and is obtained without computing any SS bounds. (The full analysis of the Fn:FNBPA-5 design problem is described in Sec. 5.3.)

The improvement of *BBK** over single-sequence methods can be measured using *cost per sequence*. We show the improvement is threefold: *BBK** (a) reduces the cost to compute a bound on a combinatorial number of sequences, (b) eliminates all computational costs once a sequence is pruned, and (c) when it must compute a bound for a single sequence, computes a bound that is in many cases cheaper than the bounds computed by previous single-sequence algorithms. To guarantee that the first sequence returned is optimal, an algorithm must either compute or bound the partition function for all possible sequences. Previous provable algorithms compute a provable *single-sequence bound* of the partition function, called an ε-approximation (SS-ε bound), for each sequence [17, 34, 44]. These SS-ε bounds are guaranteed to be within a user-specified ε of the K^* score for a sequence. *BBK** also provably returns the optimal sequences, but does so without enumerating all possible sequences. Instead of SS-ε bounds, *BBK** computes an MS bound, which is an upper bound on the best possible K^* score of multiple sequences that share a common partial sequence.

We will now compare the cost of bounding sequences with single-sequence algorithms to the cost with *BBK**. Consider an n-residue protein design: we are given an initial partial sequence s', which fixes amino acid identity (but not the rotamer) for a residues, and u residues do not have a fixed amino acid identity ($a + u = n$). If the design problem allows at most t amino acids per unassigned (u) residue and at most q rotamers for any amino acid, there are t^u sequences containing s', and q^a partial conformations defined by s'.

A complete sequence would still have q^n conformations, and computing the energy of a conformation takes $\mathcal{O}(n^2)$ time using a pairwise energy function. Thus, a single-sequence algorithm would spend $\mathcal{O}(t^u q^n n^2)$ worst-case time individually computing the K^* scores of all t^u sequences. In contrast, the cost of an MS bound is $\mathcal{O}(q^a(a^2+q^2t^2un))$, which includes $\mathcal{O}(q^a a^2)$ time to compute the pairwise energy of the a assigned residues of all q^a partial conformations, and $\mathcal{O}(q^{a+2}t^2un)$ time to compute a bound on the energy of each partial conformation. By reducing two exponentials from t^u to t^2, and from q^n to q^{a+2}, ***BBK**** computes an MS bound in time sublinear in the number (t^u) of sequences. The cost to compute a single, provable MS bound (that holds for all t^u sequences) is therefore significantly smaller than the cost to compute t^u single-sequence bounds. Furthermore, these MS bounds are used to prune *partial sequences* containing *combinations of mutations*: for a pruned partial sequence s', all t^u sequences containing s' are provably eliminated from search without any additional computation. That is, ***BBK**** provably, *combinatorially* prunes the search space. Since this pruning occurs during search, rather than as a preprocessing step, the pruning performed by ***BBK**** is distinct from previous pruning techniques such as dead-end elimination [5,6]. ***BBK**** pruning is used after dead-end elimination pruning to further improve empirical performance. Finally, MS bounds are in many cases inexpensive to compute when compared to the $\mathcal{O}(q^n n^2)$ complexity of computing an SS-ε bound for a single-sequence. Since there are q^a partial conformation energy bounds to compute, the cost of an MS bound increases exponentially as a increases. Obviously, when $a \ll n$, $q^{a+2} \ll q^n$. This is very advantageous for A^* search, because a is initially very small: when ***BBK**** begins search, $a = 1$, and increases one at a time. Furthermore, $a + u = n$, and a never exceeds n. Thus in many cases $a \ll n$, and MS bound costs of $\mathcal{O}(q^{a+2}t^2un)$ are significantly smaller than the SS-ε costs of $\mathcal{O}(q^n n^2)$ for a single sequence. Use of MS bounds enables ***BBK**** to efficiently bound and prune sequences that would otherwise require $\mathcal{O}(q^n n^2)$ time *each* to evaluate.

The algorithmic advances that make MS bounds possible are new bounds on partial and full sequences. We denote the design states unbound protein, unbound ligand, and bound complex as P, L, and PL respectively. The following definitions of these new bounds are sufficient for the theorems provided in the main paper – the precise definitions involve some subtleties, which are deferred to Appendix A. Given a sequence s and a state $X \in \{P, L, PL\}$, the function $L_{_X}(s)$ is a provable lower bound of the partition function for s in state X, and $U_{_X}(s)$ is a provable upper bound on the partition function for s in state X. For a partial sequence s', $L_{_X}(s')$ and $U_{_X}(s')$ are, respectively, partition function lower and upper bounds for the combinatorial number of sequences containing s'. These lower- and upper-bounding functions are combined into an upper-bounding function $K_a^+(s')$ on the partition function ratio of s'.

Definition 1. *Let s be a sequence. $K_a^+(s)$ is defined as follows:*

$$K_a^+(s) = \frac{U_{_{PL}}(s)}{L_{_P}(s)L_{_L}(s)}. \tag{3}$$

The following theorem establishes the relationship between the partition function ratio of a partial sequence and the partition function ratio of any sequence containing the partial sequence:

Theorem 1. *Let s be a partial or complete sequence. For any partial sequence $s' \subset s$, $K_a^+(s')$ bounds $K_a^+(s)$ from above:*

$$K_a^+(s') \geq K_a^+(s) \geq \frac{Z_{_{PL}}(s)}{Z_{_P}(s)Z_{_L}(s)} = K^*(s). \tag{4}$$

A proof of this theorem is provided in Appendix A.3. Theorem 1 shows that the bounds used by ***BBK**** are *admissible*. That is, they never underestimate the K^* ratio of any partial sequence. Thus, ***BBK**** uses $K_a^+(s')$ as the optimistic bounding function for A^* search. Previously, A^* search has been used to provably enumerate conformations within some energy window E_w of the GMEC [30] and to provably approximate

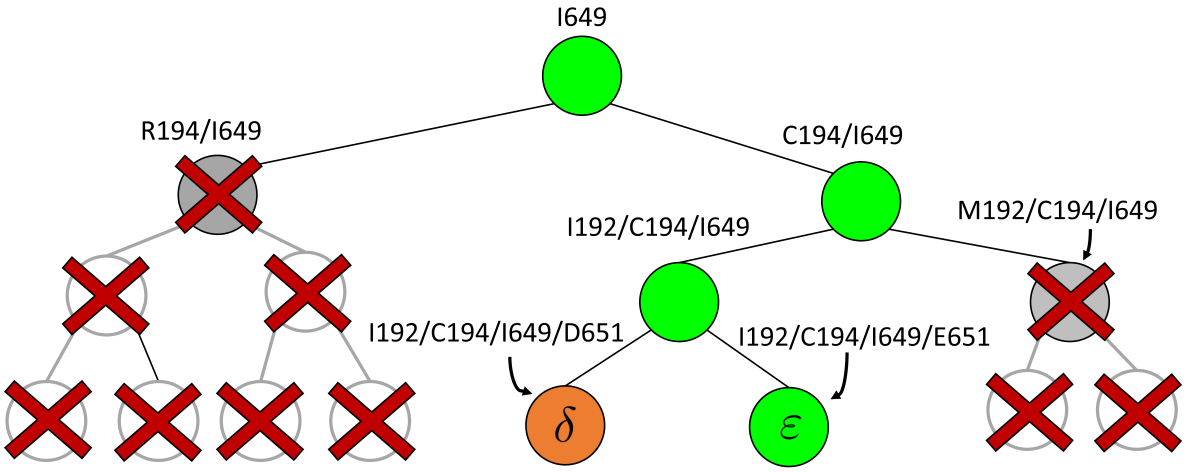

Figure 2: ***BBK**** **pruning efficiently explores the sequence space.** An example design of residues 192 and 194 of the fourth fibronectin F1 module, and residues 649 and 651 of a fragment *S. aureus* fibronectin binding protein a-5 is shown (Fig. 1, PDB Id: 2RL0). As *BBK** searches the sequence space (tree above) its multi-sequence bounds provably prunes sub-trees from the sequence space. All sequences containing R194/I649 are pruned (red crosses) after computing exactly one multi-sequence bound: the bound on the partial sequence R194/I649, which is an upper bound for all sequences containing R194/I649. Sequences containing M192/C194/I649 are pruned (red crosses) after computing only the multi-sequence bound for the partial sequence M192/C194/I649. All pruned sequences and partial sequences, shown as empty gray circles, have no additional computation performed. Even though single-sequence bounds are initiated for both I192/C194/I649/E651 and I192/C194/I649/D651, the latter is pruned after computing a mere δ-approximation bound (orange leaf node), which is cheaper, and not as tight as a ε-approximate bound. A provable ε-approximation bound (green leaf node) is computed for only the optimal sequence, I192/C194/I649/E651. In contrast, single-sequence methods compute separate ε-approximate bounds (which are expensive) for all 8 possible sequences, shown as leaf nodes in the tree.

the partition function of single sequences [7, 17, 34, 44]. Since Eq. (4) defines an admissible bound over sequences, all of the provable guarantees of A^* apply to *BBK**. With these guarantees, *BBK** provably searches over *sequences* rather than conformations, and is guaranteed to return a gap-free list of sequences in order of decreasing binding affinity.

4.1 Algorithm Overview

*BBK** bounds all possible sequences either with the MS bounds described in Sec. 4, or by computing a single-sequence bound as described in [15, 34, 44]. In brief, to bound a single sequence, *BBK** computes a gap-free list of conformations whose cumulative conformational energies (Eq. 1) are used to bound the K^* ratio. The algorithm reports an error bound δ such that the computed bound is guaranteed to be no more than a $(1+\delta)$ factor greater than the true K^* ratio. We will refer to these single-sequence, δ-approximate bounds [17, 18] as *SS-δ bounds*. As the gap-free list used for an SS-δ bound grows in size, the computed single-sequence bound becomes tighter (δ decreases). Eventually, $\delta \leq \varepsilon$, and the single-sequence bound becomes an SS-ε bound. We will refer to an SS-δ bound constructed this way as a *running bound*, which *BBK** incrementally tightens as it enumerates additional conformations [17]. In effect, *BBK** repeatedly performs one of two operations: (a) construct and compute t new MS bounds by extending a partial sequence s' with all allowed mutations at one residue, or (b) incrementally tighten the running bound of an existing SS-δ for some sequence.

To begin, *BBK** creates a max heap whose node values correspond to either full or partial sequences and

whose node keys are an upper bound on all K^* ratios (Eq. 2) in the sequence space represented by the node. The heap is initialized with a node whose value is an empty sequence (i.e., no mutable residues have fixed amino acid identities), which represents the entire sequence space. *BBK** then repeatedly removes the max node x of the queue, and performs one the following operations:

1. *Branch*. If x contains a partial sequence s', then s' is expanded. Expansion creates t new child nodes by selecting an unassigned residue r in s', and creating a new child node for each allowed amino acid a at r. Each child node contains s' plus the additional mutation of a assigned at r. These nodes are bounded with MS bounds or SS-δ bounds, and reinserted into the heap.
2. *Update*. If x contains a complete sequence s, whose bound is an SS-δ bound, then *BBK** enumerates 8 additional conformations, tightening the SS-δ bound. x is reinserted into the heap, with the updated SS-δ bound as its key. This is important for pruning, we will describe later.
3. *Return*. If x contains a complete sequence, whose bound is an SS-ε bound, then the sequence in this node has the best K^* ratio of all unenumerated sequences, and is returned.

BBK* terminates when it has enumerated the top k sequences (by SS-ε bound), where k is a user-specified number. Notably, because SS-δ bounds are incrementally tightened, the algorithm computes very few SS-ε bounds. Instead, many sequences are pruned after computing an SS-δ bound, which often requires significantly fewer conformations to compute than the tighter SS-ε bound. For example, in Fig. 2, single-sequence computation is initiated for two sequences, but a provable ε-approximation is computed for only the optimal sequence. The other sequence is pruned from search after computing a SS-δ bound. A detailed description of the algorithm is provided in Appendix A.8.

5 Computational Experiments

We implemented *BBK** in our laboratory's open source OSPREY [13] protein design package and compared our algorithm to the previous state-of-the-art single-sequence iMinDEE/A^*/K^* algorithm [17,34,44]. We computed the five best binding sequences using both *BBK** and iMinDEE/A^*/K^* for 204 different protein design problems from 51 different protein-ligand complexes. For each protein-ligand interface, we created four design problems spanning the wild-type sequence and all sets of single, double, triple, and quadruple mutants, respectively. In each design problem, we modeled either 8 or 9 residues at the protein-ligand interface as mutable and flexible. Each mutable residue was allowed to assume its wild-type identity or mutate to 13-19 other amino acids. The size of the resulting design problems ranged from 10 to 2.6×10^6 sequences and 10^5 to 10^{11} conformations (over all sequences). In all cases, we modeled continuous side-chain flexibility using continuous rotamers [12, 45]. As in [12, 17], rotamers from the Penultimate Rotamer Library [35] were allowed to minimize to any conformation within 9° of their modal χ-angles. For all design problems, we performed minimized dead-end elimination pruning (minDEE) [17], followed by either iMinDEE/A^*/K^* or *BBK**. The initial pruning window [12] was 0.1 kcal/mol, and the SS-ε bound accuracy was 0.683 (details are provided in Appendix A.9). Each design either provably returned the optimal sequences or was terminated after 30 days. A detailed description of the 51 protein-ligand systems in our experiments, the 204 protein design problems based on these systems, and our experimental protocol is provided in Appendix C.1.

5.1 Performance Comparison

As the size of the sequence space increases, so does the efficiency of *BBK** over iMinDEE/A^*/K^* (Fig. 3), demonstrating that the complexity of *BBK** is in practice sub-linear in the number of sequences. Because iMinDEE/A^*/K^* computes individual sequence binding energies as SS-ε bounds, we first measured the efficiency of *BBK** using the number of SS-ε bounds computed. Since K^* scores in iMinDEE/A^*/K^* are based on minimized conformation energies [12, 17, 34], whose computation represents a major computational bottleneck, we also measured efficiency using the number of conformation energy minimizations performed. Last, we compared the running times of *BBK** to those of iMinDEE/A^*/K^*.

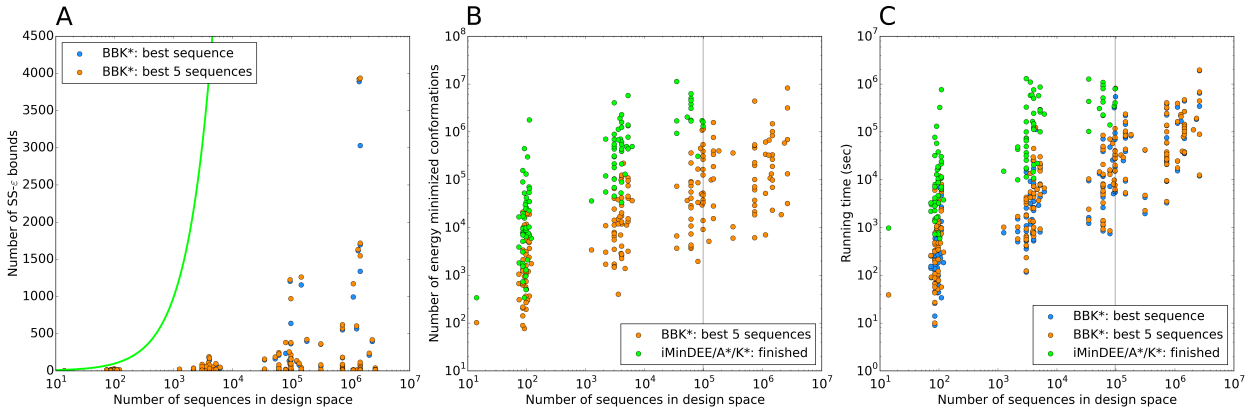

Figure 3: ***BBK**** **is up to five orders of magnitude more efficient than iMinDEE/A^*/K^*.** *BBK** completed all 204 designs within a 30 day limit, while iMinDEE/A^*/K^* completed only 107. (A) The number of SS-ε bounds performed vs. the number of sequences in the design space. Results are shown for computing only the the best sequence (blue) and computing the best five sequences (orange). Single-sequence algorithms, including the best previous algorithm iMinDEE/A^*/K^*, must compute binding affinity for all possible sequences (green curve). *BBK** required up to 6×10^5-fold fewer SS-ε bounds to find the best sequences. (B) The number of energy-minimized conformations by *BBK** and iMinDEE/A^*/K^* vs. the number of sequences in the design space. iMinDEE/A^*/K^* completed only 107 of 204 designs (left of the vertical line) before the 30-day timeout. For these designs, *BBK** was up to 1700-fold more efficient. (C) *BBK** and iMinDEE/A^*/K^* running times vs. the number of sequences in the design space. For the 107 designs completed by iMinDEE/A^*/K^* within 30 days (left of the vertical line), *BBK** was up to 800-fold more efficient than iMinDEE/A^*/K^*.

We divide the design problem sizes into three categories: the smallest problems have between 10 and 10^2 sequences; medium-sized problems contain between 10^2 and 10^4 sequences; and the largest problems contain between 10^4 and 10^7 sequences. After 30 days, *BBK** completed all 204 designs, but iMinDEE/A^*/K^* completed only 107 of 204 designs: all 39 of the smallest designs, 54 of 63 medium-sized designs, and only 14 of the 111 largest designs. We now discuss results for the 107 designs completed by iMinDEE/A^*/K^*. Because iMinDEE/A^*/K^* computes individual sequence binding energies as SS-ε bounds, we first measured the efficiency of *BBK** using the number of SS-ε bounds computed (Fig. 3(A)). For small, medium, and large designs, respectively, *BBK** was on average 17-fold, 162-fold, and 2568-fold more efficient than iMinDEE/A^*/K^*. Next, we measured efficiency using the number of conformation energy minimizations performed (Fig. 3(B)). Here, *BBK** minimized, on average, 10-fold, 43-fold, and 113-fold fewer conformations for small, medium, and large-sized designs, respectively, compared to iMinDEE/A^*/K^*. Last, we compared empirical running times for both methods (Fig. 3(C)). On average, *BBK** was 36-fold, 67-fold, and 97-fold faster than iMinDEE/A^*/K^* for small, medium, and large-sized designs, respectively.

Based on the data from the 107 designs that iMinDEE/A^*/K^* was able to complete within 30 days, we conclude that *BBK** provides a combinatorial speedup over iMinDEE/A^*/K^*. Crucially, *BBK** is not only more efficient, but also retains the provability guarantees and biophysical modeling improvements (i.e. ensemble-based design, continuous flexibility) employed by *single-sequence* iMinDEE/A^*/K^*. In one large design (2.6×10^6 sequences), involving a camelid single-domain V_HH antibody fragment in complex with RNASE A (PDB Id: 2P49), *BBK** pruned more than 99.9% of sequences to provably find the best 5 sequences. Therefore, *BBK** can design over similarly sized sequence spaces to high throughput experimental screening methods such as phage display [3, 40].

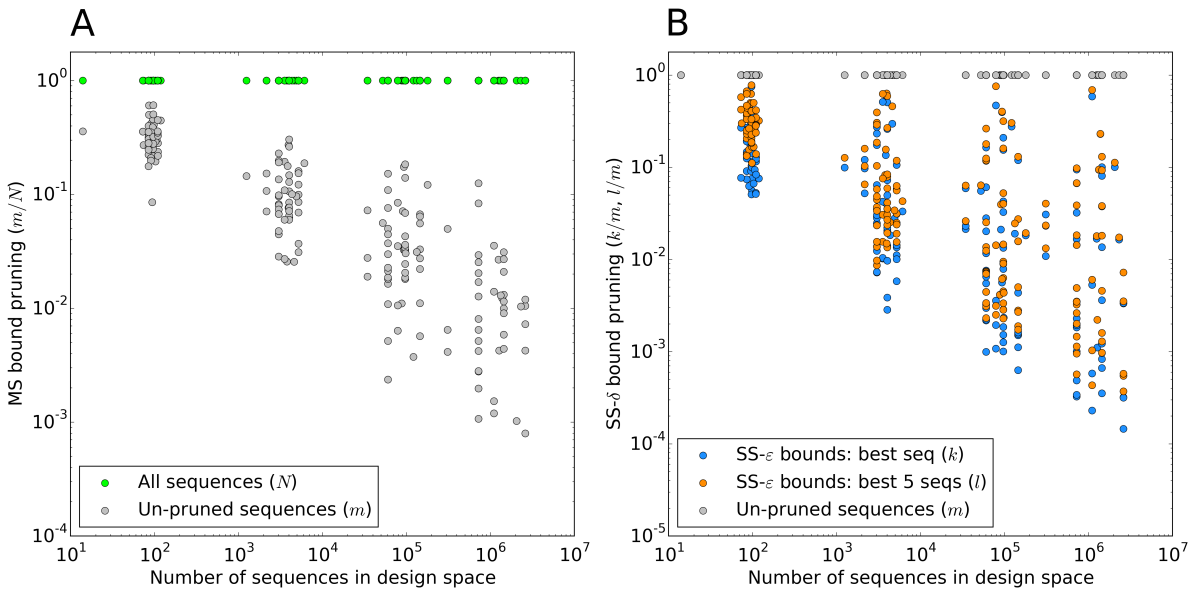

Figure 4: ***BBK**** **used MS and SS-δ bounds to prune up to 99.99999% of the sequence space.** (A) Sequence space reduction due to MS pruning. The fraction of un-pruned sequences (gray) normalized to the total number of sequences (green). *BBK** used MS bounds to provably prune up to 99.9% of the sequence space. *BBK** does not compute SS-ε bounds for pruned sequences. (B) The fraction of *BBK** SS-ε bounds (blue for the best sequence and orange for the best 5 sequences) normalized to the number of sequences not pruned by MS bound pruning (gray). To compute the best sequences, *BBK** performed SS-ε bounds for as few as 0.01% of the un-pruned sequences. The remaining 99.99% of these sequences were pruned at mere SS-δ accuracy.

5.2 Sequence Space Pruning

*BBK** owes its efficiency to two complementary modes of sequence pruning: MS bound pruning and SS-δ bound pruning. As described in Sec. 4, an MS bound on partial sequence s' is a provable upper bound on the best possible K^* ratio for all sequences that contain s'. By pruning s', *BBK** avoids computing SS-ε bounds for the combinatorial number of sequences s containing s', thereby avoiding most of the computational bottleneck faced by all single sequence methods. Fig. 4(A) illustrates the efficiency gains in *BBK** due to MS bound pruning. In small, medium, and large design problems, respectively, *BBK** pruned up to 90%, 99% and 99.9% of the sequence design space using MS bound pruning. These data show that the amount of MS pruning increased significantly with the size of the design space.

Unlike an MS bound, an SS-δ bound is a provable upper bound on an SS-ε bound (Sec. 4.1) for a *single* full sequence, s. By pruning s at SS-δ bound accuracy, *BBK** stops processing conformations for s before reaching the more precise ε-approximation required for an SS-ε bound. (*BBK** computes SS bounds using our novel two-pass algorithm, which speeds up SS bound computation up to 1982-fold relative to our previous best algorithm, iMinDEE/A^*/K^*. Further details and empirical measurements of the method are provided in Appendix A.10 and B.1, respectively,.) Fig. 4(B) illustrates the efficiency gains in *BBK** due to SS-δ bound pruning. In small, medium, and large design problems, respectively, SS-δ pruning eliminates up to 98%, 99.9% and 99.99% of the sequences not pruned by MS pruning. These data show that the amount of SS-δ pruning increased with the size of the design problem.

Importantly, MS bound pruning and SS-δ bound pruning have multiplicative synergy, producing a combined pruning effect of up to 99.99999% of the original sequence space while provably finding the five best binding sequences. In one example, we re-designed the protein-protein interface (PPI) of a camelid affinity-matured single-domain V_HH antibody fragment (PDB Id: 2P4A). The sequence space, 2.6×10^6 sequences, consisted of all quadruple mutants in the 9-residue PPI. *BBK** pruned all but 2078 sequences

using MS pruning and then pruned 2071 sequences from these remaining 2078 sequences using SS-δ bound pruning. These data show how *BBK** prunes a combinatorial number of sequences from the design space, producing dramatic efficiency gains over single-sequence methods.

5.3 Design with Coupled Continuous Side-Chain and Backbone Flexibility

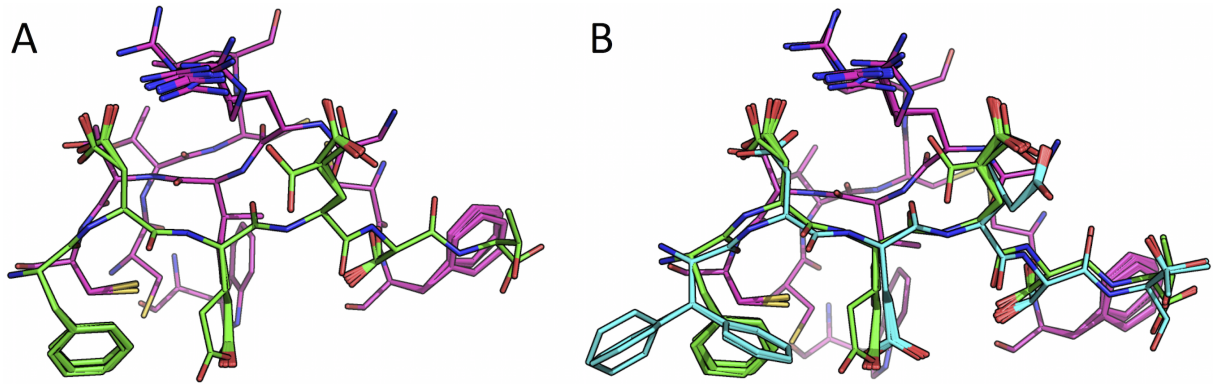

Figure 5: ***BBK**** **efficiently handles coupled continuous side-chain and local backbone flexibility.** Selected residues from ensembles, computed by *BBK**, of human fibronectin F1 modules 4-5 (magenta) in complex with a fragment of *S. aureus* fibronectin binding protein A 5 (FNBPA-5, PDB Id: 2RL0, [36]). The design space consisted of the wild-type sequence and either 15 or 25 single amino-acid mutants. (A) Ensemble of the wild-type sequence based on the original crystal structure. The design used a fixed FNBPA-5 backbone (green) and continuous side-chain flexibility. (B) Ensemble of the wild-type sequence using two backbones: the original FNBPA-5 backbone (green) and a second backbone (PDB Id: 2RKY, cyan) with RMSD 1.3 Å from the original (found using the MASTER program [62]). The sequence rankings (by K^* score, Eq. 2) from the fixed and flexible backbone models had Spearman correlation coefficients of ρ=0.53 and ρ=0.82 in the 15 and 25 mutant designs, respectively. This shows that the flexible backbone model favors binding in very different sequences than the fixed backbone model does.

Protein design with continuous rotamers is more accurate than design with rigid rotamers [12]. However, design with an ensemble of backbones and coupled continuous side-chain and local backbone flexibility, which is enabled by the DEEPer framework [25], is even more realistic. This is because the combination of side-chain and backbone flexibility better accounts for possible conformational changes induced by amino acid substitutions. To determine whether design with a fixed backbone and continuous rotamers (i.e. the fixed backbone model) predicts tight binding in the same sequences as does a model with both local backbone flexibility *and* continuous rotamers (i.e. the flexible backbone model), we used *BBK** to redesign the Human Fibronectin F1:*Staphylococcus aureus* FNBPA-5 interface [36] (PDB Id: 2RL0) for binding affinity. Then, we compared the sequence rankings produced by *BBK** using each input model. In our designs, we searched over the wild-type sequence and either 15 or 25 single amino acid mutants at the PPI. In our four design problems, 10 residues at the interface were modeled as flexible and allowed to mutate to between 2-5 other amino acids, yielding conformation spaces ranging from 10^4-10^9 conformations. We used the MASTER program [62] to find an alternate FNBPA backbone (PDB Id: 2RKY, [1]) with an RMSD of 1.3 Å from the wild-type backbone. This backbone allowed for more interactions between F156 and T654, between S189 and E652, and between R191 and D650 at the PPI. Coupled continuous side-chain and local backbone flexibility were modeled using DEEPer [25]. With the flexible backbone model, *BBK** searched over both backbones. Details of our experimental protocol are provided in Appendix C.3.

In the first experiment, we re-designed the Fibronectin F1:FNBPA-5 interface for binding affinity over the wild-type sequence and 15 single amino-acid polymorphisms. Our results showed that using the flexible backbone model versus the fixed backbone model increased the size of the design conformation space by

1417-fold but only increased the running time by 4-fold in *BBK**. By comparison, iMinDEE/A^*/K^* required 48-fold more time than *BBK** to complete the flexible backbone design. Our results also showed that the *BBK** sequence rankings between the two input models had a Spearman correlation coefficient of ρ=0.53. Thus, the flexible backbone model favors binding in different sequences than the fixed backbone model does. For instance, the FNBPA-5 D650E mutant is predicted to bind less tightly than the wild-type in the fixed backbone model (Fig. 5(A)) but not in the flexible model (Fig. 5(B)). In our second experiment, the sequence design space consisted of the wild-type sequence and 25 single amino-acid polymorphisms. The *BBK** sequence rankings produced by the two input models had a Spearman correlation coefficient of ρ=0.82 (additional details are provided in Appendix B.2). Relative to the fixed backbone model, the flexible backbone model increased the size of the design conformation space by 8447-fold but only increased the running time by only 1.7-fold in *BBK**. iMinDEE/A^*/K^* required 89-fold more time than *BBK** to complete the design using the flexible backbone model.

These results show that *BBK** makes designs with backbone flexibility feasible. Compared to the previous state-of-the-art iMinDEE/A^*/K^*, *BBK** is not only able to provably compute the tightest-binding sequences for designs that were previously too large, but can also efficiently handle designs with additional degrees of backbone flexibility. It is important to note that these experiments are only possible with provable algorithms. Without the provable guarantees of *BBK**, it would be difficult and perhaps unsound to compare the results of computational protein design with and without coupled continuous side-chain and backbone flexibility, since difference induced by the fixed backbone and rotamer model cannot be deconvolved from differences stemming from undersampling or inadequate stochastic computation. Thus, *BBK** provides provable methods to analyze the difference in predicted sequences between different models of side-chain and backbone flexibility.

6 Conclusion

BBK* fills an important *lacuna* in protein design: we presented a novel algorithm that can search efficiently not over the energies of single-conformations, but over the binding affinity of sequences. *BBK** is, to our knowledge, the first provable, ensemble-based algorithm to search in order of binding affinity and run in time *sub-linear* in the number of sequences. Previously, protein designers either employed heuristic algorithms to compute locally optimal sequences, or computed provably accurate approximations of binding affinity for each sequence individually. *BBK** not only computes the globally optimal sequences, it does so while combinatorially pruning the search space. Our experiments show that *BBK** can search over sequence spaces of up to 2.6×10^6 sequences, a capacity comparable to high-throughput experimental screening methods such as phage display. Thus, *BBK** liberates binding affinity-based protein design from the efficiency barrier imposed by exhaustive search. Ensemble-based design for affinity over large sequence spaces was previously possible only with heuristic algorithms (with no guarantees), or using high-throughput wet-bench experiments. Ensemble-based design for affinity over large sequence spaces was previously possible only with heuristic algorithms (with no guarantees), or using high-throughput wet-bench experiments. *BBK** enables computational protein design by providing new K_a algorithms, with provable guarantees, for these large-scale protein designs.

Acknowledgments

We would like to thank Drs. Mark Hallen and Pablo Gainza for helpful discussions and for providing useful protein-ligand binding problems; Dr. Jeffrey Martin for assisting with software optimizations; Jack Holland, Gevorg Grigoryan, Hunter Nisonoff, Anna Lowegard and all members of the Donald lab for helpful discussions; and the NSF (GRFP DGF 1106401 to AAO) and NIH (R01-GM78031 and R01-GM118543 to BRD, R01-HL119648 to VGF) for funding.

References

[1] Richard J Bingham, Enrique Rudiño-Piñera, Nicola A G Meenan, Ulrich Schwarz-Linek, Johan P Turkenburg, Magnus Höök, Elspeth F Garman, and Jennifer R Potts. Crystal structures of fibronectin-binding sites from Staphylococcus aureus FnBPA in complex with fibronectin domains. *Proc Natl Acad Sci U S A*, 105(34):12254–8, Aug 2008.

[2] F Edward Boas and Pehr B Harbury. Potential energy functions for protein design. *Curr Opin Struct Biol*, 17(2):199–204, Apr 2007.

[3] Sara Carmen and Lutz Jermutus. Concepts in antibody phage display. *Brief Funct Genomic Proteomic*, 1(2):189–203, Jul 2002.

[4] Cheng-Yu Chen, Ivelin Georgiev, Amy C Anderson, and Bruce R Donald. Computational structure-based redesign of enzyme activity. *Proc Natl Acad Sci U S A*, 106(10):3764–9, Mar 2009.

[5] B I Dahiyat and S L Mayo. Protein design automation. *Protein Sci*, 5(5):895–903, May 1996.

[6] J Desmet, M De Maeyer, B Hazes, and I Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356(6369):539–42, Apr 1992.

[7] Bruce R Donald. *Algorithms in Structural Molecular Biology*. MIT Press, Cambridge, MA, 2011.

[8] Sarel J Fleishman, Sagar D Khare, Nobuyasu Koga, and David Baker. Restricted sidechain plasticity in the structures of native proteins and complexes. *Protein Sci*, 20(4):753–7, Apr 2011.

[9] Kathleen M Frey, Ivelin Georgiev, Bruce R Donald, and Amy C Anderson. Predicting resistance mutations using protein design algorithms. *Proc Natl Acad Sci U S A*, 107(31):13707–12, Aug 2010.

[10] Menachem Fromer and Chen Yanover. A computational framework to empower probabilistic protein design. *Bioinformatics*, 24(13):i214–22, Jul 2008.

[11] Pablo Gainza, Hunter M Nisonoff, and Bruce R Donald. Algorithms for protein design. *Curr Opin Struct Biol*, 39:16–26, 2016.

[12] Pablo Gainza, Kyle E Roberts, and Bruce R Donald. Protein design using continuous rotamers. *PLoS Comput Biol*, 8(1):e1002335, Jan 2012.

[13] Pablo Gainza, Kyle E Roberts, Ivelin Georgiev, Ryan H Lilien, Daniel A Keedy, Cheng-Yu Chen, Faisal Reza, Amy C Anderson, David C Richardson, Jane S Richardson, and Bruce R Donald. OSPREY: protein design with ensembles, flexibility, and provable algorithms. *Methods Enzymol*, 523:87–107, 2013) (Program, user manual, and source code are available at www.cs.duke.edu/donaldlab/software.php .

[14] I. Georgiev, S. Schmidt, Y. Li, D. Wycuff, G. Ofek, N. Doria-Rose, T. Luongo, Y. Yang, T. Zhou, B. R. Donald, J. Mascola, and P. Kwong. Design of epitope-specific probes for sera analysis and antibody isolation. *Retrovirology*, 9, 2012.

[15] Ivelin Georgiev and Bruce R Donald. Dead-end elimination with backbone flexibility. *Bioinformatics*, 23(13):i185–94, Jul 2007.

[16] Ivelin Georgiev, Ryan H Lilien, and Bruce R Donald. Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. *Bioinformatics*, 22(14):e174–83, Jul 2006.

[17] Ivelin Georgiev, Ryan H Lilien, and Bruce R Donald. 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. *J Comput Chem*, 29(10):1527–42, Jul 2008.

[18] Ivelin S Georgiev. *Novel Algorithms for Computational Protein Design, with Applications to Enzyme Redesign and Small-Molecule Inhibitor Design*. PhD thesis, Duke University. http://hdl.handle.net/10161/1113, 2009.

[19] Ivelin S Georgiev, Rebecca S Rudicell, Kevin O Saunders, Wei Shi, Tatsiana Kirys, Krisha McKee, Sijy O'Dell, Gwo-Yu Chuang, Zhi-Yong Yang, Gilad Ofek, Mark Connors, John R Mascola, Gary J Nabel, and Peter D Kwong. Antibodies VRC01 and 10E8 neutralize HIV-1 with high breadth and potency even with IG-framework regions substantially reverted to germline. *J Immunol*, 192(3):1100–6, Feb 2014.

[20] M K Gilson, J A Given, B L Bush, and J A McCammon. The statistical-thermodynamic basis for computation of binding affinities: a critical review. *Biophys J*, 72(3):1047–69, Mar 1997.

[21] Michael J Gorczynski, Jolanta Grembecka, Yunpeng Zhou, Yali Kong, Liya Roudaia, Michael G Douvas, Miki Newman, Izabela Bielnicka, Gwen Baber, Takeshi Corpora, Jianxia Shi, Mohini Sridharan, Ryan Lilien, Bruce R Donald, Nancy A Speck, Milton L Brown, and John H Bushweller. Allosteric inhibition of the protein-protein interaction between the leukemia-associated proteins Runx1 and CBF-beta. *Chem Biol*, 14(10):1186–97, Oct 2007.

[22] Mark A Hallen and Bruce R Donald. 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. *J Comput Biol*, 23(5):311–321, 2016.

[23] Mark A Hallen, Pablo Gainza, and Bruce R Donald. Compact representation of continuous energy surfaces for more efficient protein design. *J Chem Theory Comput*, 11(5):2292–306, May 2015.

[24] Mark A Hallen, Jonathan D Jou, and Bruce R Donald. LUTE (Local unpruned tuple expansion): Accurate continuously flexible protein design with general energy functions and rigid Rotamer-Like efficiency. *J Comput Biol*, Epub ahead of print, 2016.

[25] 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*, 81(1):18–39, Jan 2013.

[26] P.E. Hart, Nilsson N.J., and B Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans on SSC*, 4:100–114, 1968.

[27] Jonathan D Jou, Swati Jain, Ivelin S Georgiev, and Bruce R Donald. BWM*: a novel, provable, ensemble-based dynamic programming algorithm for sparse approximations of computational protein design. *J Comput Biol*, 23(6):413–424, 2016.

[28] Carleton L. Kingsford, Bernard Chazelle, and Mona Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039, 2005. PMID: 15546935.

[29] B Kuhlman and D Baker. Native protein sequences are close to optimal for their structures. *Proc Natl Acad Sci U S A*, 97(19):10383–8, Sep 2000.

[30] A R Leach and A P Lemon. Exploring the conformational space of protein side chains using dead-end elimination and the a* algorithm. *Proteins*, 33(2):227–39, Nov 1998.

[31] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, Ian W Davis, Seth Cooper, Adrien Treuille, Daniel J Mandell, Florian Richter, Yih-En Andrew Ban, Sarel J Fleishman, Jacob E Corn, David E Kim, Sergey Lyskov, Monica Berrondo, Stuart Mentzer, Zoran Popović, James J Havranek, John Karanicolas, Rhiju Das, Jens Meiler, Tanja Kortemme, Jeffrey J Gray, Brian Kuhlman, David Baker, and Philip Bradley. Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. *Methods Enzymol*, 487:545–74, 2011.

[32] C Lee and M Levitt. Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. *Nature*, 352(6334):448–51, Aug 1991.

[33] J. Leech, J. F. Prins, and J. Hermans. Smd: Visual steering of molecular dynamics for protein design. *Computational Science and Engineering*, 3(4):38–45, 1996.

[34] Ryan H Lilien, Brian W Stevens, Amy C Anderson, and Bruce R Donald. 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. *J Comput Biol*, 12(6):740–61, 2005.

[35] S C Lovell, J M Word, J S Richardson, and D C Richardson. The penultimate rotamer library. *Proteins*, 40(3):389–408, Aug 2000.

[36] Steven K Lower, Supaporn Lamlertthon, Nadia N Casillas-Ituarte, Roberto D Lins, Ruchirej Yongsunthon, Eric S Taylor, Alex C DiBartola, Catherine Edmonson, Lauren M McIntyre, L Barth Reller, Yok-Ai Que, Robert Ros, Brian H Lower, and Vance G Fowler, Jr. Polymorphisms in fibronectin binding protein A of Staphylococcus aureus are associated with infection of cardiovascular devices. *Proc Natl Acad Sci U S A*, 108(45):18372–7, Nov 2011.

[37] Hunter Nisonoff. Efficient partition function estimation in computational protein design: Probabalistic guarantees and characterization of a novel algorithm. *B.S. Thesis. Department of Mathematics, Duke University. http://hdl.handle.net/10161/9746*, May 2015.

[38] Adegoke Ojewole, Anna Lowegard, Pablo Gainza, Stephanie M Reeve, Ivelin Georgiev, Amy C Anderson, and Bruce R Donald. OSPREY predicts resistance mutations using positive and negative computational protein design. *Methods Mol. Biol.*, 1529:291–306, 2017. PMID: 27914058.

[39] Noah Ollikainen, René M de Jong, and Tanja Kortemme. Coupling protein side-chain and backbone flexibility improves the re-design of protein-ligand specificity. *PLoS Comput Biol*, 11(9):e1004335, 2015.

[40] Gábor Pál, Jean-Louis K Kouadio, Dean R Artis, Anthony A Kossiakoff, and Sachdev S Sidhu. Comprehensive and quantitative mapping of energy landscapes for protein-protein interactions by rapid combinatorial scanning. *J Biol Chem*, 281(31):22378–85, Aug 2006.

[41] Jian Peng, Raghavendra Hosur, Bonnie Berger, and Jinbo Xu. iTreePack: protein complex Side-Chain packing by dual decomposition. *arXiv:1504.05467 [q-bio.BM]*, 2015.

[42] Niles A Pierce and Erik Winfree. Protein design is NP-hard. *Protein Eng*, 15(10):779–82, Oct 2002.

[43] Stephanie M Reeve, Pablo Gainza, Kathleen M Frey, Ivelin Georgiev, Bruce R Donald, and Amy C Anderson. Protein design algorithms predict viable resistance to an experimental antifolate. *Proc Natl Acad Sci U S A*, 112(3):749–54, Jan 2015.

[44] Kyle E Roberts, Patrick R Cushing, Prisca Boisguerin, Dean R Madden, and Bruce R Donald. Computational design of a PDZ domain peptide inhibitor that rescues CFTR activity. *PLoS Comput Biol*, 8(4):e1002477, 2012.

[45] Kyle E Roberts and Bruce R Donald. Improved energy bound accuracy enhances the efficiency of continuous protein design. *Proteins*, 83(6):1151–64, Jun 2015.

[46] Kyle E Roberts, Pablo Gainza, Mark A Hallen, and Bruce R Donald. Fast gap-free enumeration of conformations and sequences for protein design. *Proteins*, 83(10):1859–77, Oct 2015.

[47] Rebecca S Rudicell, Young Do Kwon, Sung-Youl Ko, Amarendra Pegu, Mark K Louder, Ivelin S Georgiev, Xueling Wu, Jiang Zhu, Jeffrey C Boyington, Xuejun Chen, Wei Shi, Zhi-Yong Yang, Nicole A Doria-Rose, Krisha McKee, Sijy O'Dell, Stephen D Schmidt, Gwo-Yu Chuang, Aliaksandr Druz, Cinque Soto, Yongping Yang, Baoshan Zhang, Tongqing Zhou, John-Paul Todd, Krissey E Lloyd, Joshua Eudailey, Kyle E Roberts, Bruce R Donald, Robert T Bailer, Julie Ledgerwood, NISC Comparative Sequencing Program, James C Mullikin, Lawrence Shapiro, Richard A Koup, Barney S Graham, Martha C Nason, Mark Connors, Barton F Haynes, Srinivas S Rao, Mario Roederer, Peter D Kwong, John R Mascola, and Gary J Nabel. Enhanced potency of a broadly neutralizing HIV-1 antibody in vitro improves protection against lentiviral infection in vivo. *J Virol*, 88(21):12669–82, Nov 2014.

[48] Daniele Sciretti, Pierpaolo Bruscolini, Alessandro Pelizzola, Marco Pretti, and Alfonso Jaramillo. Computational protein design with side-chain conformational entropy. *Proteins*, 74(1):176–91, Jan 2009.

[49] Maxim V Shapovalov and Roland L Dunbrack, Jr. A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*, 19(6):844–58, Jun 2011.

[50] Reshma P Shetty, Paul I W De Bakker, Mark A DePristo, and Tom L Blundell. Advantages of fine-grained side chain conformer libraries. *Protein Eng*, 16(12):963–9, Dec 2003.

[51] Nathaniel W Silver, Bracken M King, Madhavi N L Nalam, Hong Cao, Akbar Ali, G S Kiran Kumar Reddy, Tariq M Rana, Celia A Schiffer, and Bruce Tidor. Efficient computation of small-molecule configurational binding entropy and free energy changes by ensemble enumeration. *J Chem Theory Comput*, 9(11):5098–5115, Nov 2013.

[52] David Simoncini, David Allouche, Simon de Givry, Céline Delmas, Sophie Barbe, and Thomas Schiex. Guaranteed discrete energy optimization on large protein design problems. *J Chem Theory Comput*, 11(12):5980–9, Dec 2015.

[53] Brian W Stevens, Ryan H Lilien, Ivelin Georgiev, Bruce R Donald, and Amy C Anderson. Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme's mechanism and selectivity. *Biochemistry*, 45(51):15495–504, Dec 2006.

[54] Seydou Traoré, David Allouche, Isabelle André, Simon de Givry, George Katsirelos, Thomas Schiex, and Sophie Barbe. A new framework for computational protein design through cost function network optimization. *Bioinformatics*, 29(17):2129–36, Sep 2013.

[55] Seydou Traoré, Kyle E Roberts, David Allouche, Bruce R Donald, Isabelle André, Thomas Schiex, and Sophie Barbe. Fast search algorithms for computational protein design. *J Comput Chem*, 37(12):1048–58, May 2016.

[56] L. G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[57] Clément Viricel, David Simoncini, Thomas Schiex, and Sophie Barbe. Guaranteed weighted counting for affinity computation: beyond determinism and structure. *The 22nd International Conference on Principles and Practice of Constraint Programming*, September 2016.

[58] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. A new class of upper bounds on the log partition function. *CoRR*, abs/1301.0610, 2013.

[59] J Xu. Rapid protein side-chain packing via tree decomposition. *9th Annual International Conference, RECOMB*, 3500:423–439, 2005.

[60] J Xu and B Berger. Fast and accurate algorithms for protein side-chain packing. *The Journal of the ACM*, 53(4):533–557, 2006.

[61] Fang Zheng, Wenchao Yang, Mei-Chuan Ko, Junjun Liu, Hoon Cho, Daquan Gao, Min Tong, Hsin-Hsiung Tai, James H Woods, and Chang-Guo Zhan. Most efficient cocaine hydrolase designed by virtual screening of transition states. *J Am Chem Soc*, 130(36):12148–55, Sep 2008.

[62] Jianfu Zhou and Gevorg Grigoryan. Rapid search for tertiary fragments reveals protein sequence-structure relationships. *Protein Sci*, 24(4):508–24, Apr 2015.

Appendix

The following is an appendix, which provides additional information to substantiate the claims made in the Sections 1 through 6. In Appendix A, a detailed description of the *BBK** algorithm, described in Section 4, along with proofs of correctness, space complexity, and time complexity are given. Appendix B provides additional results to support the results discussed in Section 5, and Appendix C provides details of the computational experiments performed in Section 5.

A Supplementary Methods

In this section, we provide the proofs necessary to show admissibility of the K_* lower bound. We begin with definitions for K^*, a discrete approximation of the binding affinity K_a, then define its multiplicative inverse K_*, which we wish to approximate. We then give upper bounds on the denominator and lower bounds on the numerator for the terms in K_*, and show that these bounds can be combined to calculate a provable lower bound for K_*. Finally, we show that these bounds on K_* are guaranteed to be an ε-approximation of K_*, for which ε can be arbitrarily reduced to any user-specified value with additional computation.

A.1 Definitions

A.1.1 Protein design problem

In this subsection, we define the protein design problem. To define the upper and lower bounds computed on *partial sequences* during A^* search, we define assignments to sequences and conformations more precisely. Additionally, we define two rigid, quadratic time pairwise energy functions that provide an upper and lower bound on a given conformation.

For an n-residue protein design problem with at most t amino acids and q rotamers per amino acid, let the set of mutable residues be R. Let P, L, and PL denote the unbound protein, unbound ligand, and bound protein-ligand complex, respectively. For each mutable residue $r \in R$, let A_r be the set of allowed amino acids at r, and Q_r be the set of allowed rotamers at r.

A.1.2 Sequences and Conformations

Let s be a set of candidate mutations, or a sequence assignment. That is, let $s = \bigcup_{i=1}^{d} \{(r_i, a_i)\}$ be a set of d 2-tuples, each consisting of one mutable residue $r_i \in R$ and one amino acid $a_i \in A_{r_i}$. For convenience, we will also denote the amino acid assignment at mutable residue $j \in R$ as s_j, such that for a tuple $(j, a) \in s$, $s_j = a$. Additionally, we define $\mathbf{A}(s)$ to be the set of mutable residues contained in s, while $\mathbf{U}(s) = R - \mathbf{A}(s)$ is the set of mutable residues not in s, i.e. the unassigned residues of s. Let us now consider a partial sequence $s' = \bigcup_{i=1}^{d'} \{(r_i, a_i)\}$ of d' 2-tuples. We will refer to s as being *consistent* with a partial sequence s' to mean that for all $r \in \mathbf{A}(s') \cap \mathbf{A}(s), s'_r = s_r$. When $d' < d$ we will refer to this consistency as $s \supset s'$.

Similarly, we define a conformation $c = \bigcup_{i=1}^{d} (r_i, q_i)$ to be a set of d 2-tuples, each composed of a mutable residue $r_i \in R$ and a rotamer $q_i \in Q_i$, where Q_i is the set of rotamers allowed at residue r_i. For convenience, we will denote the rotamer assignment of c at mutable residue $j \in R$ as c_j. We define $\mathbf{A}(c)$ and $\mathbf{U}(c)$ analogously for conformations as we did for sequences. Let us now consider a partial conformation $c' = \bigcup_{i=1}^{d'} \{(r_i, q_i)\}$ of d' 2-tuples. We will refer to c as being *consistent* with a partial conformation c' to mean that for all $r \in \mathbf{A}(s') \cap \mathbf{A}(s), c'_r = c_r$. When $d' < d$ we can simply state this consistency as $c \supset c'$.

For any amino acid assignment s_r at residue r, Let $\mathbf{Q}(s_r)$ be the set of allowed rotamers defined by s_r. We further extend this definition as a function over sequences. $\mathbf{Q}(s)$ takes as input a partial or full sequence

s and returns the set of partial or full conformations consistent with s:

Definition 2.

$$\boldsymbol{Q}(s) = \prod_{r \in \boldsymbol{A}(s)} \boldsymbol{Q}(s_r). \tag{5}$$

A.1.3 Energy functions

For a given rigid rotamer conformation c let E_X be a pairwise energy function, such that $E_X(c)$ is the corresponding post-minimization energy of that conformation with respect to the protein state $X \in \{P, L, PL\}$. We also define $E_X^{\odot}(c)$, the energy of c before rotamer minimization, and $E_X^{\ominus}(c)$, the lower bound of energy of $E_X(c)$ calculated with precomputed, optimistic *pairwise-minimized* energies instead of minimizing all mutable residues simultaneously. If $|c| < n$, we define the three energy functions to be the one-body and two-body energy function terms consisting only of the residues in $\mathbf{A}(c)$. By definition, $E_X^{\ominus}(c) \leq E_X(c) \leq E_X^{\odot}(c)$ for any conformation c.

A.1.4 K^*

We can now define the Boltzmann-weighted partition function $Z_X(s)$:

$$Z_X(s) = \sum_{c \in \mathbf{Q}(s)} \exp(-E_X(c)/RT). \tag{6}$$

The K^* ratio, which approximates binding affinity K_a, can then defined to be:

$$K^*(s) = \frac{Z_{PL}(s)}{Z_P(s) Z_L(s)}. \tag{7}$$

We refer the reader to [17, 34, 44] for a more rigorous explanation of K^*.

Similarly, we define K_* to be the multiplicative inverse of K^*:

$$K_*(s) = \frac{Z_P(s) Z_L(s)}{Z_{PL}(s)}. \tag{8}$$

A.2 Bounding the Partition Function

In the following subsection, we define two new functions: $L_X(s)$ and $U_X(s)$, which provide lower and upper bounds on $Z_X(s)$, respectively. We then show the relationship between these bounds and $Z_X(s)$, and show that for any sequence s they bound $Z_X(s)$ from below and above, respectively, not only for s but for any partial sequence $s' \subset s$. For convenience, we will assume the energy functions used below are consistent with the state X defined by U_X and L_X, and abbreviate $E_X(c)$ to $E(c)$. We do the same for $E_X^{\odot}(c)$ and $E_X^{\ominus}(c)$.

A.2.1 Lower bounds: $L_X(s)$

In this subsection, we define the function $L_X(s)$ for a sequence s. We begin with partial conformation c' and define an energy upper bound that also holds for the energy of any conformation c such that $c \supset c'$. That is, given c', we seek an upper bound on all conformations consistent with c'. We define two functions $g(c')$ and $h^{\oplus}(c')$ whose sum $g(c') + h^{\oplus}(c') \geq E(c)$ for all c consistent with c'. $g(c')$ and $h^{\oplus}(c')$ are defined below.

$g(c')$ is the sum of unary and pair-wise rotamer energies over assigned residues $\mathbf{A}(c')$, defined as:

Definition 3.

$$g(c') = \sum_{i \in \boldsymbol{A}(c')} \left(E(c'_i) + \sum_{\substack{j \in \boldsymbol{A}(c') \\ j > i}} E(c'_i, c'_j) \right). \tag{9}$$

To define $h^{\oplus}(c')$, we first define the function $h^{\oplus}(c', i)$, which is a function of c' and a mutable residue $i \in R$. We first define q_i to be the candidate rotamer q at residue i. Additionally, let $Q_i = \bigcup_{a \in A_i} \mathbf{Q}(a)$ be the set of all allowed rotamers at i. Since i has not been assigned an amino acid identity, Q_i may be the union of rotamer sets from multiple amino acids, and a rotamer $q \in Q_i$ may be from any of multiple allowed amino acids. We define Q_k similarly.

Definition 4.

$$h^{\oplus}(c', i) = \max_{q \in Q_i} \left(E^{\odot}(q_i) + \sum_{j \in \mathbf{A}(c')} E^{\odot}(q_i, c'_j) + \sum_{\substack{k \in \boldsymbol{U}(c') \\ k<i}} \max_{q' \in Q_k} E^{\odot}(q_i, q'_k) \right) \tag{10}$$

$h^{\oplus}(c')$ is then defined to be a sum of $h^{\oplus}(c', i)$ over the unassigned residues $\mathbf{U}(c')$:

Definition 5.

$$h^{\oplus}(c') = \sum_{i \in \boldsymbol{U}(c')} h^{\oplus}(c', i) \tag{11}$$

By definition, $g(c) + h^{\oplus}(c) \geq E(c)$. This is most easily seen from the fact that $E^{\odot}(q_i) \geq E(q_i)$. We now show that for a partial conformation c', any superset $c \supset c'$ conformation must have an upper bound less than our equal to than the upper bound of c'.

Proposition 1. *For all conformations c and all partial conformations $c' \subset c$, $g(c') + h^{\oplus}(c') \geq g(c) + h^{\oplus}(c)$*

Proof. Suppose $g(c') + h^{\oplus}(c') < g(c) + h^{\oplus}(c)$. Since $\mathbf{A}(c') \subset \mathbf{A}(c)$, there exists at least one residue $i \in \mathbf{A}(c) - \mathbf{A}(c')$ with assigned rotamer c_i where the following must be true:

$$\begin{aligned} h^{\oplus}(c', i) &= \max_{q \in Q_i} \left(E^{\odot}(q_i) + \sum_{j \in \mathbf{A}(c')} E^{\odot}(q_i, c'_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} \max_{q' \in Q_k} E^{\odot}(q_i, q'_k) \right) \\ &< E(c_i) + \sum_{j \in \mathbf{A}(c')} E(c_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} E(c_i, c_k) \\ &\leq E^{\oplus}(c_i) + \sum_{j \in \mathbf{A}(c')} E^{\odot}(c_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} E^{\odot}(c_i, c_k). \end{aligned}$$

In this case, $h^{\oplus}(c', i)$ is not the maximum as defined, or the inequality must not be true. Since $h^{\oplus}(c', i)$ can be updated to take this larger value without any constraint, the inequality must not be true, and $g(c') + h^{\oplus}(c') \geq g(c) + h^{\oplus}(c)$. □

We now define $\tau(c')$, which is a lower bound on the number of full conformations consistent with c' for any one sequence. For the following definitions of and τ, we define A_i to be the set of allowed amino acids at residue i.

Definition 6.

$$\tau(c') = \prod_{i \in \boldsymbol{U}(c')} \min_{a \in A_i} |\boldsymbol{Q}(a)| \tag{12}$$

By definition, if $|c| = n$ then $\tau(c) = 1$, and for any partial sequence s' such that $\mathbf{A}(s') = \mathbf{U}(c)$, $\tau(c) \leq |\mathbf{Q}(s')|$. Finally, we combine $g(c')$, $h^{\oplus}(c')$, and $\tau(c')$ to compute a lower bound on the partition function contribution of c':

Definition 7.

$$\ell(c') = \exp\left(\Big(-g(c') - h^{\oplus}(c')\Big)/RT\right)\tau(c'). \tag{13}$$

Let us consider a partial conformation c' and a sequence s. We will refer to s as being consistent with c' to mean that there exists a nonempty set $C \subseteq \mathbf{Q}(s)$ such that for all $c \in C$, $c' \subset c$. By definition, $\ell(c')$ is a lower bound on the contribution of c' to the K^* score (Eq. 7) of any sequence s consistent with c'. That is, for any partial conformation c' and partial sequence s' such that $\mathbf{A}(s') = \mathbf{U}(c')$, $\ell(c') \leq \exp\left(\Big(-g(c') - h^{\oplus}(c')\Big)/RT\right)|\mathbf{Q}(s')|$. This follows directly from Proposition 1 and the definition of $\tau(c)$. With these terms defined, we can now combine them into the partition function lower bound for a sequence s:

Definition 8.

$$L_X(s) = \sum_{c \in \boldsymbol{Q}(s)} \ell(c). \tag{14}$$

Next, we will show that $L_X(s)$ is a lower bound of $Z_X(s)$ for all partial and complete sequences s. We begin by considering the bound $\ell(c')$ of an arbitrary partial conformation c'. Since we have already shown that for any conformation c and partial conformation $c' \subset c$, $\tau(c) \leq \tau(c')$ and $g(c)+h^{\oplus}(c) \leq g(c')+h^{\oplus}(c')$, it follows that the partial conformation $c' \subset c$ satisfies $\ell(c') \leq \ell(c)$. We now show that for any sequence s and any subsequence $s' \subset s$, $L_X(s') \leq L_X(s)$. We do so by considering an arbitrary partial conformation $c' \in \mathbf{Q}(s')$. The contribution of $\ell(c')$ bounds the sum $\sum_{\substack{c \supset c' \\ c \in \mathbf{Q}(s)}} \ell(c)$ for all conformations $c \in \mathbf{Q}(s)$ consistent with c'.

Lemma 2. *Let s' be a partial sequence, and $c' \in \boldsymbol{Q}(s')$ be a partial conformation. For any sequence $s \supset s'$, let $\boldsymbol{Q}_{c'}(s) = \prod_{i \in \boldsymbol{A}(c')} \{c'_i\} \times \prod_{j \in \boldsymbol{A}(s) - \boldsymbol{A}(c')} \boldsymbol{Q}(s_j)$ be the set of all conformations in $\boldsymbol{Q}(s)$ consistent with c'. Then for all s', $s \supset s'$, the following inequality holds:*

$$\ell(c') \leq \sum_{c \in \boldsymbol{Q}_{c'}(s)} \exp(-E(c)/RT) \tag{15}$$

Proof. We begin by noting that from Proposition 1, $g(c') + h^{\oplus}(c') \geq E(c)$ for all $c \in \mathbf{Q}(s)$ and therefore $\exp((-g(c') - h^{\oplus}(c')/RT) \leq \exp(-E(c)/RT)$ for all $c' \in \mathbf{Q}(s')$. It remains only to be shown that $\tau(c') \leq |\mathbf{Q}(s - s')|\tau(c)$, which follows from the definition of $\tau(c')$. □

We will give a bound on $L_X(s)$, namely that for all s', $s \supset s'$, $L_X(s') \leq L_X(s)$, in Section 2.3. We first define the upper bounding function, $U_X(s)$.

A.2.2 Upper bounds: $U_X(s)$

In this subsection we define the function $U_X(s)$ for a sequence s. Similarly to the Section 2.1, we first define a lower bound on the energy of any conformation c in terms of a partial conformation $c' \subset c$. For all $c' \subset c$, the following bound holds:

$$g(c') + h^{\ominus}(c') \leq E(c) \tag{16}$$

Where $g(c')$ is as defined in Equation 5, and the lower bound $h^{\ominus}(c')$ is defined as follows:

Definition 9.

$$h^{\ominus}(c') = \sum_{i \in \boldsymbol{U}(c')} h^{\ominus}(c', i). \tag{17}$$

Definition 10.

$$h^{\ominus}(c', i) = \min_{q \in Q_i} \left(E^{\ominus}(q_i) + \sum_{j \in \boldsymbol{A}(c')} E^{\ominus}(q_i, c'_j) + \sum_{\substack{k \in \boldsymbol{U}(c') \\ k<i}} \min_{q' \in Q_k} E^{\ominus}(q_i, q'_k) \right) \tag{18}$$

By definition, $g(c) + h^{\ominus}(c) \leq E(c)$. We now show that for a conformation c, any partial conformation c' must have a lower bound equal to or lower than the lower bound of c.

Proposition 3. *For any conformations c and* $c' \subset c, g(c') + h^{\ominus}(c') \leq g(c) + h^{\ominus}(c)$.

Proof. Suppose $g(c')+h^{\ominus}(c') > g(c)+h^{\ominus}(c)$. Since $\mathbf{A}(c') \subset \mathbf{A}(c)$, there exists at least one $i \in \mathbf{A}(c)-\mathbf{A}(c')$ where the following must be true:

$$\begin{aligned} h^{\ominus}(c', i) = & \min_{q \in Q_i} \left(E^{\ominus}(q_i) + \sum_{j \in \mathbf{A}(c')} E^{\ominus}(q_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} \min_{q' \in Q_k} E^{\ominus}(q_i, q'_k) \right) \\ > & E(c_i) + \sum_{j \in \mathbf{A}(c')} E(c_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} E(c_i, c_k) \\ \geq & E^{\ominus}(c_i) + \sum_{j \in \mathbf{A}(c')} E^{\ominus}(c_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} E^{\ominus}(c_i, c_k) \end{aligned}$$

In this case, $h^{\ominus}(c', i) \neq \min_{q \in \mathbf{Q}_i} \left(E^{\ominus}(q_i) + \sum_{j \in \mathbf{A}(c')} E^{\ominus}(q_i, c_j) + \sum_{\substack{k \in \mathbf{U}(c') \\ k<i}} \min_{q' \in Q_k} E^{\ominus}(q_i, c_k) \right)$, or $g(c') + h^{\ominus}(c') \not> g(c) + h^{\ominus}(c)$. Since $h^{\ominus}(c', i)$ can be updated to take this lower value without any constraint, the inequality must not be true, and $g(c') + h^{\ominus}(c') \leq g(c) + h^{\ominus}(c)$. □

We now define $\lambda(c')$, which is an upper bound on the number of full conformations consistent with c for any one sequence.

Definition 11.

$$\lambda(c) = \prod_{i \in \boldsymbol{U}(c')} \max_{a \in A_i} |\boldsymbol{Q}(a)|. \tag{19}$$

We then combine the definitions to compute an upper bound on the partition function contribution of any partial conformation c':

Definition 12.

$$u(c') = \exp\left(\Big(- g(c') - h^{\ominus}(c') \Big) / RT \right) \lambda(c'). \tag{20}$$

Next, we will show that $U_X(s)$ is a upper bound of $Z_X(s)$ for all partial and complete sequences s. We begin by considering the bound $u(c')$ of an arbitrary partial conformation c'. Since we have already shown that for any conformation c, the partial conformation $c' \subset c$ satisfies $u(c') \geq u(c)$, we now show that for any sequence s and any subsequence $s' \subset s$, $U_X(s') \geq U_X(s)$. We do so by considering an arbitrary partial conformation $c' \in \mathbf{Q}(s')$. The contribution of $u(c')$ bounds $\sum_{\substack{c \supset c' \\ c \in \mathbf{Q}(s)}} u(c)$ for all conformations $c \in \mathbf{Q}(s)$ consistent with c'.

Lemma 4. *Let s' be a partial sequence, and $c' \in \boldsymbol{Q}(s')$ be a partial conformation. Let $\boldsymbol{Q}_{c'}(s) = \prod_{i \in \boldsymbol{A}(c')} \{c'_i\} \times \prod_{j \in \boldsymbol{A}(s) - \boldsymbol{A}(c')} \boldsymbol{Q}(s_j)$ be the set of all conformations in $\boldsymbol{Q}(s)$ consistent with c'. For all s such that $s' \subset s$, the following inequality holds:*

$$u(c') \geq \sum_{c \in \boldsymbol{Q}_{c'}(s)} \exp(-E(c)/RT) \tag{21}$$

Proof. We begin by noting that from Proposition 3, $g(c') + h^{\ominus}(c') \leq E(c)$ for all $c \in \mathbf{Q}(s)$ and therefore $\exp((-g(c') - h^{\ominus}(c')/RT) \geq \exp(-E(c)/RT)$. It remains only to be shown that $\lambda(c') \geq |\mathbf{Q}(s - s')|\lambda(c)$, which follows from the definition of $\lambda(c')$. □

With these terms defined, we can define an upper bound on the partition function for a partial sequence s':

Definition 13.

$$U_X(s') = \sum_{c' \in \boldsymbol{Q}(s')} u(c') \tag{22}$$

We can now give bounds on $L_X(s)$ and $U_X(s)$, to show that for any $s' \subset s$, $L_X(s')$ and $U_X(s')$ are admissible heuristic functions for s.

A.2.3 $L_X(s') \leq L_X(s) \leq Z_X(s)$

By definition, $L_X(s) \leq Z_X(s)$. We now show that for all s', $s \supset s'$, $L_X(s') \leq L_X(s)$.

Lemma 5. *For any partial sequence s' and consistent sequence $s \supset s'$,*

$$L_X(s') \leq L_X(s) \tag{23}$$

Proof. Since there exists a partial conformation $c' \in \mathbf{Q}(s')$ for all $c \in \mathbf{Q}(s)$, the results from Lemma 2 extend to the summation over $\mathbf{Q}(s')$ and $\mathbf{Q}(s)$.

$$\sum_{c' \in \mathbf{Q}(s')} \ell(c') \leq \sum_{c \in \mathbf{Q}(s)} \ell(c) \tag{24}$$

By simple substitution of Definition 7, we derive the target relationship:

$$L_X(s') \leq L_X(s). \tag{25}$$

□

That is, our lower-bounding function L_X, for any complete sequences s, where $|s| = n$, and any partial sequences s', $L_X(s)$ is guaranteed to be a lower bound of $Z_X(s)$.

A.2.4 $U_X(s') \geq U_X(s) \geq Z_X(s)$

By definition, $U_X(s) \geq Z_X(s)$. We now show that for all s', $s \supset s'$, $U_X(s') \geq U_X(s)$.

Lemma 6. *For any partial sequence s' and consistent sequence $s \supset s'$,*

$$U_X(s') \geq U_X(s). \tag{26}$$

Proof. By definition, $U_X(s) \geq U_X(s)$. Since there exists a partial conformation $c' \in \mathbf{Q}(s')$ for all $c \in \mathbf{Q}(s)$, the results from Lemma 2 extend to the summation over $\mathbf{Q}(s')$ and $\mathbf{Q}(s)$.

$$\sum_{c' \in \mathbf{Q}(s')} u(c') \geq \sum_{c \in \mathbf{Q}(s)} u(c) \tag{27}$$

By simple substitution of Definition 12, we derive the desired relationship:

$$U_X(s') \geq U_X(s). \tag{28}$$

□

A.3 Admissibility

We now define the $K_a^+(s)$, which never underestimates the best possible K^* ratio attainable given a partial sequence. Without loss of generality, we define $K_a^+(s)$ to be an upper bound on K^*, which is a provable approximation of K_a. Since $\mathrm{K_a} = \frac{1}{\mathrm{K_d}}$, $K_a^+(s)$ is a provable upper bound on $\mathrm{K}^*(s)$, and $\frac{1}{K_a^+(s)}$ is a provable lower bound on $\mathrm{K}_*(s)$.

Definition 1.

$$K_a^+(s') = \frac{U_{PL}(s')}{L_P(s')L_L(s')}. \tag{29}$$

Theorem 1. *$K_a^+(s)$ is admissible. That is, for all $s, s' \subset s$,*

$$K_a^+(s') \geq K_a^+(s) \geq \frac{Z_{PL}(s)}{Z_P(s)Z_L(s)} = K^*(s). \tag{30}$$

Proof. This follows directly from Lemma 5 and Lemma 6. □

A.4 ε-approximations of the Partition Function

In this subsection, we give the bounds on the error ε for our partition function approximation for each term, and then show that our ε-approximation of the full K_* score can compute an arbitrary ε-approximation for any user-specified ε. For convenience, we assume all equations operate on a full sequence assignment s, and abbreviate $Z_X(s)$ to Z_X and $U_{PL}(s)$ to U_{PL} for the remainder of the proofs.

We first define three epsilon approximations: $(1-\varepsilon_1)Z_P$,$(1-\varepsilon_2)Z_L$, and $(1+\varepsilon_3)Z_{PL}$. We will use the ε-approximations as defined by [34, 44] for $(1-\varepsilon_1)Z_P$ and $(1-\varepsilon_2)Z_L$.

Proposition 1. *Let Z_P^* and Z_L^* be the partition function score approximation for so far for Z_P and Z_L, respectively. There exist $0 \leq \varepsilon_1 \leq 1$ and $0 \leq \varepsilon_2 \leq 1$ such that $(1-\varepsilon_1)Z_P \leq Z_P^* \leq Z_P$ and $(1-\varepsilon_2)Z_L \leq Z_L^* \leq Z_L$*

These results follow directly from the formulation in [34]. We now define ε_3 for use as an upper bound on Z_{PL}.

Proposition 2. *Let Z^*_{PL} be the partition function approximation score of computed so far for Z_{PL}. There exists $0 \leq \varepsilon_3 \leq 1$ such that $Z_{PL} \leq (1+\varepsilon_3)Z^*_{PL}$*

Proof. By definition, $Z^*_{PL} \leq Z_{PL}$. Thus, for any $0 \leq \varepsilon_3 \leq 1$, $(1+\varepsilon_3)Z^*_{PL} \leq (1+\varepsilon_3)Z_{PL}$. We can expand U_{PL} into the sum of three terms: $U_{PL} = Z^*_{PL} + q'_{PL} + p^*_{PL}$, where Z^*_{PL} is the current approximation of Z_{PL}, q'_{PL} is a bound on the cumulative contribution of unenumerated conformations and p^*_{PL} is a bound on the cumulative contribution of conformations pruned by DEE. By defining $\varepsilon_3 = \frac{q'_{PL}+p^*_{PL}}{Z^*_{PL}}$, it follows that

$$U_{PL} = Z^*_{PL} + q'_{PL} + p^*_{PL} = (1+\varepsilon_3)Z^*_{PL}. \tag{31}$$

□

Theorem 3.

$$Z_{PL} \leq U_{PL} \leq (1+\varepsilon_3)Z_{PL}. \tag{32}$$

Proof. By definition, $Z_{PL} \leq Z^*_{PL} + q'_{PL} + p^*_{PL}$. Since $Z^*_{PL} \leq Z_{PL}$, for any arbitrary ε_3 such that $0 \leq \varepsilon_3 \leq 1$, $(1+\varepsilon_3)Z^*_{PL} \leq (1+\varepsilon_3)Z_{PL}$. $U_{PL} = Z^*_{PL} + q'_{PL} + p^*_{PL} = (1+\varepsilon_3)Z^*_{PL}$ from Proposition 2, and therefore the inequality holds. □

We now show that K^+_a is a provable ε-approximation of the K^* ratio.

Lemma 4. *For provable ε-approximations $(1+\varepsilon_3)Z_{PL}$, $(1-\varepsilon_1)Z_P$, and $(1-\varepsilon_2)Z_L$ where $0 \leq \varepsilon_1 \leq 1$, $0 \leq \varepsilon_2 \leq 1$, and $0 \leq \varepsilon_3 \leq 1$, there exists $0 \leq \varepsilon_4 \leq 1$ such that*

$$\frac{(1-\varepsilon_1)Z_P(1-\varepsilon_2)Z_L}{(1+\varepsilon_3)Z_{PL}} = (1-\varepsilon_4)\frac{Z_P Z_L}{Z_{PL}}. \tag{33}$$

Proof. Defining ε_4 to be $1 - \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{(1+\varepsilon_3)}$, we first observe that $0 \leq \varepsilon_4 \leq 1$. We then obtain the following result:

$$\frac{(1-\varepsilon_1)Z_P(1-\varepsilon_2)Z_L}{(1+\varepsilon_3)Z_{PL}} = \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{(1+\varepsilon_3)}\frac{Z_P Z_L}{Z_{PL}} \tag{34}$$

$$= \left(1 - \left(1 - \frac{(1-\varepsilon_1)(1-\varepsilon_2)}{(1+\varepsilon_3)}\right)\right)\frac{Z_P Z_L}{Z_{PL}} \tag{35}$$

$$= (1-\varepsilon_4)\frac{Z_P Z_L}{Z_{PL}}. \tag{36}$$

□

A.5 Space and Time Complexity

The following subsection gives proofs of the space and time complexity of *BBK**. In particular, we show the space and time complexity of computing a multi-sequence bound (MS bound). We also show the worst-case space and time complexity for computing the optimal sequence (i.e., the sequence with the best K^* ratio).

Let s' be a partial sequence. We will define the number of assigned residues $a = |\mathbf{A}(s')|$, as well as the number of unassigned residues $u = |\mathbf{U}(s')|$. For a protein design problem with n mutable residues, at most t amino acids at any mutable residue, and at most q rotamers for any amino acid, the following theorem bounds the time to compute an MS bound.

Theorem 5. *BBK* computes an MS bound in $\mathcal{O}(q^a(a^2 + q^2t^2un))$ time.*

Proof. The number of partial conformations consistent with s' ($|\mathbf{Q}(s')|$) is at most q^a. For each of these q^a partial conformations, the time to compute the a^2 pair-wise energy terms between them is $\mathcal{O}(a^2)$. The time to compute the bounds $\ell(c')$ and $u(c')$ for each partial conformation is $\mathcal{O}(q^2t^2un)$, requiring search over every possible combination of amino acid (t) and rotamer (q) for each unassigned rotamer. □

The following theorem bounds the space to compute an MS bound.

Theorem 6. *BBK* computes an MS bound in $\mathcal{O}(1)$ space.*

Proof. An MS is a single scalar value, which is computed as a sum. Only the sum need be stored during computation. □

Notably, the worst-case time complexity to compute an MS bound is linear in u and merely polynomial in t. iMinDEE/A^*/K^* must compute t^u provable ε-approximations, taking $\mathcal{O}(t^uq^nn^2)$ worst-case time.

Theorem 7. *BBK* computes the top k sequences, where k is a user-specified number, in $\mathcal{O}(t^nq^nn^2)$ time and $\mathcal{O}(t^nq^n)$ space.*

Proof. In the worst case, *BBK** computes an MS bound for all t^{n-2} partial sequences in $\mathcal{O}(t^nq^nn^2)$ time. It can compute the MS bound for a partial sequence s' which has only one unassigned residue in $\mathcal{O}(tq^nn^2)$ time. Therefore, it computes t^{n-1} MS bounds in $\mathcal{O}(t^nq^nn^2)$ time. As shown in Theorem 6, MS bounds are computed in $\mathcal{O}(1)$ space. Finally, in the worst case *BBK** computes a provable ε-approximation for all t^n sequences, taking $\mathcal{O}(t^nq^nn^2)$ time and $\mathcal{O}(t^nq^n)$ space. After computing t^n ε-approximations it is guaranteed to return the optimal sequence. □

These deterministic $\mathcal{O}(t^nq^nn^2)$ worst-case bounds guarantee optimality after a finite amount of time and space. In contrast, heuristic methods have no guarantees.

A.6 Ensuring K_a^+ can be used as a provable A^* bounding function

We now give some necessary, technical results for an A* algorithm when using K$_*$ as its bounding function. Without loss of generality, we show that a provable ε-approximation $\frac{1}{K_a^+(s)}$ of K$_*(s)$ from Lemma 4 is a monotonically increasing lower bound on K$_*(s)$.

A.6.1 Stability of K_a^+

Lemma 8 (below) states that for any arbitrary $0 \leq \varepsilon_1 \leq 1$, $0 \leq \varepsilon_2 \leq 1$, and $0 \leq \varepsilon_3 \leq 1$, the computed K$_*$ approximation is guaranteed to be a lower bound. That is, it is possible to arbitrarily improve the ε-approximation of any of the three terms to reduce ε_4 (Eq.33). This capability is an improvement over previous K^* approximation algorithms [17, 34, 44], which efficiently compute a provable approximation of the K^* ratio for single sequences. A different idea is required for MS bounds. In particular, previous methods computed the following provable approximation:

$$\frac{(1-\varepsilon_1)Z_{P}(1-\varepsilon_2)Z_{L}}{(1-\varepsilon_3)Z_{PL}}. \tag{37}$$

When two sequences have similar ε-approximate K$_*$ scores computed as in Eq. (37), determining which of the two is better cannot be done by arbitrarily reducing one of ε_1, ε_2, or ε_3. For some values of ε_3, (e.g. $(1-\varepsilon_3) = (1-\varepsilon_1)(1-\varepsilon_2)$), the computed K$_*$ approximation is no longer an upper *or* lower bound and hence cannot be used to compare K$_*$ scores of two different sequences. Thus, the new ε-approximation given in Eq. (33) and shown to be a lower bound in Lemma 8 is needed when performing A* search over sequences by K$_*$ score.

Lemma 8.

$$\frac{(1-\varepsilon_1)Z_{_P}(1-\varepsilon_2)Z_{_L}}{(1+\varepsilon_3)Z_{_{PL}}} \leq \frac{Z_{_{PL}}}{Z_{_P}Z_{_L}}. \tag{38}$$

Furthermore, our computed K_* bound is a lower bound on the K_* approximation defined in Eq. (37).

Corollary 9.

$$\frac{(1-\varepsilon_1)Z_{_P}(1-\varepsilon_2)Z_{_L}}{(1+\varepsilon_3)Z_{_{PL}}} \leq \frac{(1-\varepsilon_1)Z_{_P}(1-\varepsilon_2)Z_{_L}}{(1-\varepsilon_3)Z_{_{PL}}}. \tag{39}$$

A.7 Enforcing Ensemble-based Sequence Stability Constraints during MS bound Computation

When designing for tight binding affinity, the design objectives are often to (a) compute the best binding sequences whose (b) unbound states are energetically favorable. The constraint on the unbound state is intended to exclude sequences whose *favorable* K^* scores are due to energetically *unfavorable* unbound states that cannot fold into ligand-binding poses. MS bounds are also useful for pruning these sequences.

For any given sequence, s, a high K^* ratio

$$K^*(s) = \frac{Z_{_{PL}}(s)}{Z_{_P}(s)Z_{_L}(s)}$$

denotes tight predicted affinity for the ligand. Thus, the constraint, (b), requires that unbound state partition function, $Z_{_P}(s)$, be greater than some ensemble-based threshold, γ. Single-sequence methods compute a K^* ratio for each sequence s. In contrast, ***BBK***** prunes any partial sequence $s' \subset s$ that violates the constraint $U_{_P}(s') \leq \gamma$ (see Eq. 22 for the definition of $U_{_P}(s')$). Because $U_{_P}(s') \geq U_{_P}(s) \geq Z_{_P}(s)$ (see Appendix A.2.4), it follows from $U_{_P}(s') \leq \gamma$ that $Z_{_P}(s) \leq \gamma$ for the combinatorial number of sequences s consistent with s'. By pruning s', ***BBK***** avoids costly single-sequence computation for a combinatorial number of sequences. Therefore, ***BBK***** uses MS bounds to prune a combinatorial number of sequences using not only ensemble-based binding affinity criteria, but also ensemble-based sequence stability criteria.

A.8 *BBK** Algorithm Description

In this section we include pseudocode describing ***BBK*****. Algorithm 1 outlines the ***BBK***** algorithm.

A.9 Using ε to Bound the Value of K^*

Let s be a full sequence. As described in [17, 34, 44], the ε-approximate partition function ratio to approximate K_a for s is:

$$K^\dagger(s) = \frac{(1-\varepsilon)Z_{_{PL}}(s)}{(1-\varepsilon)^2 Z_{_P}(s)Z_{_L}(s)}. \tag{40}$$

For $X \in \{PL, P, L\}$, we define the partition function, $Z_{_X}(s)$, over the conformational states of s as

$$Z_{_X}(s) = Z^*_{_X}(s) + Z'_{_X}(s) \tag{41}$$

where $Z^*_{_X}(s)$ is the partition function contribution from enumerated and energy-minimized conformations, and $Z'_{_X}(s)$ is the upper bound of the partition function contribution from un-enumerated conformations. We define ε as

$$0 \leq \varepsilon = \frac{Z'_{_X}(s)}{Z^*_{_X}(s) + Z'_{_X}(s)} \leq 1. \tag{42}$$

Algorithm 1 *BBK** Procedure

procedure NEXTBESTSEQUENCE
Initialization:
    $h \leftarrow$ empty heap
    insert $(\emptyset, \infty)$ into h
Loop:
    while h is not empty **do**
        remove the max node (s^*, f^*) from h
        if s^* has a provable ε-approximation **then return** s^*
        if s^* is a complete sequence **then**
            compute tighter bound f' on $K^*(s^*)$
            $f^* \leftarrow f'$
            insert (s^*, f^*) into h
        else
            pick an unassigned mutable residue u
            for each allowed AA t at u **do**
                $s_a \leftarrow s^* \cup \{u, t\}$
                insert new node $(s_a, K_a^+(s_a))$ into h

In our experiments in Sec. 5, the details of which are can be found in Appendix B.1, we specified an ε-approximation accuracy of 0.683. Below, we justify the claim that $\varepsilon = 0.683$ guarantees that our calculated binding affinity is within one order of magnitude of $K^\dagger(s)$.

It is established in [17, 34, 44] that

$$L = (1-\varepsilon)^2 K^\dagger \leq K^\dagger \leq \frac{K^\dagger}{(1-\varepsilon)^2} = U. \tag{43}$$

Proposition 10. *Let* $0 \leq \varepsilon \leq 1$ *and* $\alpha \geq 1$. *If* $U \leq \alpha K^\dagger$, *then* $\varepsilon \leq 1 - 1/\sqrt{\alpha}$.

Proof. $U \leq \alpha K^\dagger$ implies that

$$\frac{K^\dagger}{(1-\varepsilon)^2} \leq \alpha K^\dagger \tag{44}$$

which is equivalent to $\varepsilon \leq 1 - 1/\sqrt{\alpha}$, as desired. □

Proposition 11. *Let* $0 \leq \varepsilon \leq 1$ *and* $\beta \geq 1$. *If* $L \geq \frac{K^\dagger}{\beta}$, *then* $\varepsilon \leq 1 - 1/\sqrt{\beta}$.

Proof. $L \geq \frac{K^\dagger}{\beta}$ implies that

$$L = (1-\varepsilon)^2 K^\dagger \geq \frac{K^\dagger}{\beta} \tag{45}$$

which reduces to $\varepsilon \leq 1 - 1/\sqrt{\beta}$, as desired. □

Thus, per Props. 10 and 11, to compute binding affinity $K^\ddagger$ such that $\frac{K^\dagger}{10} \leq K^\ddagger \leq 10K^\dagger$, it is sufficient to have $\varepsilon = 1 - 1/\sqrt{10} \approx 0.683$.

A.10 A Tighter Two-pass Partition Function Bound for More Efficient SS-ε Bound Computation

In this section, we contrast SS-ε bound computation (where ε is the user-specified approximation accuracy) in *BBK** to iMinDEE/A^*/K^*, our previous best algorithm. In particular, we show that, compared to iMinDEE/A^*/K^*, *BBK** computes a tighter, more efficient partition function bound, which allows *BBK** to tighten a δ-approximate partition function bound to an ε-approximate bound more efficiently. Per Eq. (1), $Z_{_X}(s)$ is the partition function of sequence s in state $X \in \{P, L, PL\}$. For simplicity, we will refer to $Z_{_X}(s)$ as $Z(s)$.

iMinDEE/A^*/K^* computes a δ-approximation of $Z(s)$ by using A^* to enumerate a gap-free list of conformations in increasing order of energy bound [17, 34, 44]. Each energy bound is a lower bound on the minimized energy of the full conformation. As described in Appendix A.1.1, $\mathbf{Q}(s)$ is the set of conformations consistent with s. Since iMinDEE/A^*/K^* computes a running sum to approximate $Z(s)$, at any point in time there will be a set of enumerated conformations, and a set of un-enumerated conformations. Let $A \subset \mathbf{Q}(s)$ be the set of *enumerated conformations*. We define the running partition function, $Z^*(s)$, as

$$Z^*(s) = \sum_{a \in A} \exp(-E(a)/RT), \tag{46}$$

where $E(a)$ is the *minimized* energy of conformation a. Let conformation l, whose minimized energy *lower bound* is $E^{\ominus}(l)$, be the last conformation enumerated in A. A^* enumeration guarantees that $E^{\ominus}(l)$ has the highest energy bound of all conformations in A. Further, let $B = \mathbf{Q}(s) - A$ be the set of remaining, *un-enumerated conformations*, and let $|B|$ be the size of B. iMinDEE/A^*/K^* computes a partition function upper bound for the conformations in B as

$$Z^{'}(s) = |B| \exp(-E^{\ominus}(l)/RT), \tag{47}$$

and computes δ_1 such that

$$\delta_1 = \frac{Z^{'}(s)}{Z^*(s) + Z^{'}(s)}. \tag{48}$$

As conformations in $\mathbf{Q}(s)$ are enumerated, conformations are removed from B and added to A. So, $|A|$ and $E^{\ominus}(l)$ increase, but $|B|$ decreases. As a result, $Z^*(s)$ increases but $Z^{'}(s)$ decreases until $\delta_1 \leq \varepsilon$, and $Z^*(s)$ becomes an ε-approximation of $Z(s)$.

$Z^{'}(s)$ is a loose partition function bound over B because iMinDEE/A^*/K^* optimistically assumes that the energy bound for all conformations in B is equal to $E^{\ominus}(l)$. However, A^* guarantees that conformations are enumerated in order of increasing energy bound, so successive conformations in B actually have *increasingly higher* energy bounds than $E^{\ominus}(l)$.

We use this fact to construct a tighter partition function upper bound $Z^{\dagger}(s)$, (which we call the *two-pass bound*) over conformations in B as follows:

$$Z^{\dagger}(s) = \sum_{b \in B} \exp(-E^{\ominus}(b)/RT) \tag{49}$$

and compute δ_2 as

$$\delta_2 = \frac{Z^{\dagger}(s)}{Z^*(s) + Z^{\dagger}(s)}. \tag{50}$$

A^* enumeration guarantees that, for all conformations $b \in B$, $E^{\ominus}(b) > E^{\ominus}(l)$, so $\exp(-E^{\ominus}(b)/RT) < \exp(-E^{\ominus}(l)/RT)$. As a result, $Z^{\dagger}(s) < Z^{'}(s)$, and $\delta_2 < \delta_1$. Therefore, the δ_2 approximation, which is

computed by *BBK**, is tighter than the δ_1 approximation computed by iMinDEE/A^*/K^*.

The tighter δ_2 bound allows *BBK** to compute $Z^*(s)$ to ε-approximation accuracy more efficiently than iMinDEE/A^*/K^*. *BBK** is more efficient for two reasons. First, the per-conformation cost of computing $Z^*(s)$ exceeds that of $Z^\dagger(s)$. In particular, the per-conformation cost of computing any $E(a)$ for $Z^*(s)$ is $\mathcal{O}(In^2)$, where I is the number of iterations required for energy minimization [23]. In contrast, the cost per conformation to compute any $E^\ominus(b)$ for $Z^\dagger(s)$ is merely $\mathcal{O}(n^2)$. Consequently, *BBK** computes $Z^\dagger(s)$ efficiently. Second, because *BBK**'s δ_2 bound is tighter than iMinDEE/A^*/K^*'s δ_1 bound, in practice *BBK** avoids high costs by minimizing fewer conformations to compute $Z^*(s)$ and obtain an ε-approximation of $Z(s)$ (see Fig. 6(A) in Appendix B.1). Thus, the ε-approximate value of $Z^*(s)$ computed by *BBK** is not only tighter than the corresponding value computed by iMinDEE/A^*/K^*, but also more efficient to compute (Appendix B.1, Fig. 6(B)). Therefore, *BBK** computes SS-ε bounds more efficiently than iMinDEE/A^*/K^*.

B Supplementary Data

In this section, we provide data to further support claims made in Sec. 5. We first provide data showing that *BBK** computes SS-ε bounds more efficiently than does our previous best algorithm, iMinDEE/A^*/K^*. We also provide data from the protein design runs in Sec. 5.3, where we compared designs with a fixed backbone to designs with a flexible backbone.

B.1 Computational Experiments: Computing an SS-ε Bound in *BBK**

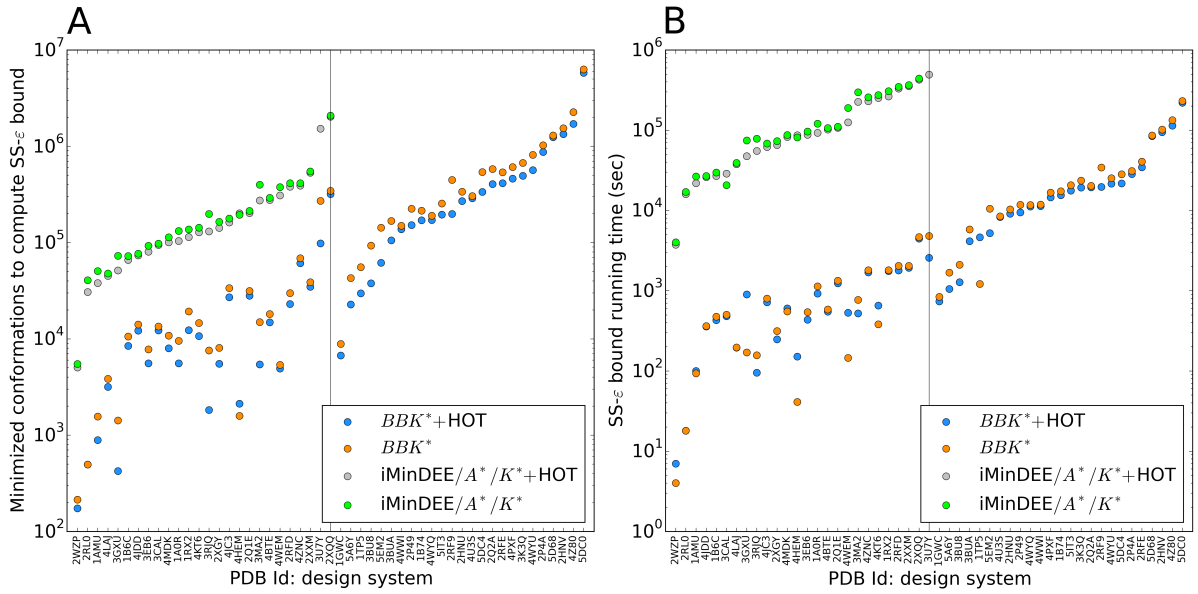

Figure 6: ***BBK** computes SS-ε bounds orders of magnitude more efficiently than iMinDEE/A^*/K^*.** (A) The number of energy-minimized conformations required to compute SS-ε bound for 51 design systems. For the 26/51 designs completed by iMinDEE/A^*/K^* within 7 days (left of vertical line), HOT improves the efficiency of iMinDEE/A^*/K^* by an average 1.2-fold, while *BBK** improves efficiency by 25-fold, on average. The combination of *BBK**+HOT is 38-fold more efficient than iMinDEE/A^*/K^* on average. (B) SS-ε bound running times for 51 design systems. Among designs completed by iMinDEE/A^*/K^* in 7 days (left of the vertical line), HOT improves running times by an average of 1.1-fold, while *BBK** and *BBK**+HOT improve running times by an average of 389-fold and 277-fold, respectively.

Computation of SS-ε bounds is the main performance bottleneck of *BBK**. Each SS-ε bound requires energy minimization of $\mathcal{O}(q^n)$ continuously flexible conformations in the worst case, where n is the number of design positions and q is the maximum number of rotamers per design position. The iMinDEE/A^*/K^*

algorithm [17, 34, 44] avoids exhaustive enumeration of conformations, however, by processing conformations in gap-free increasing order of energy to compute a provably accurate SS-ε bound. In practice, iMinDEE/A^*/K^* computes an SS-ε bound after processing a very small percentage of all possible conformations. Nevertheless, the method can be computationally prohibitive for the largest design problems. To address this limitation, we designed a new two-pass bounding algorithm (Appendix A.10) to accelerate computation of each partition function. To compare the efficiency of our new two-pass SS-ε algorithm to iMinDEE/A^*/K^*, we computed SS-ε bounds for the wild type sequences of the 51 protein-ligand systems described in Appendix C. We modeled continuous flexibility for 8-14 residues at the protein-ligand interface, resulting in conformation spaces ranging from 10^6 to 10^{10} conformations. Each continuous rotamer was permitted to rotate freely within 9° of the modal χ-angle values in the Penultimate Rotamer Library [35]. We performed minimized dead-end elimination pruning (minDEE) [17] with a pruning window [12] of at least 0.1 kcal/mol to prune high energy rotamers and set the SS-ε bound accuracy to 0.683 (see Appendix A.9 for justification). To further improve efficiency, we used higher order tuples (HOT) [24, 45] to reduce the inaccuracy of the pairwise-minimized bounds. We terminated designs that did not complete after 7 days. A detailed description of the 51 protein design systems in our experiments is provided in Appendix C.2.

B.1.1 Performance Comparison

While *BBK** completed all 51 designs, iMinDEE/A^*/K^* completed only 26. We now discuss results for these 26 systems. We first measured efficiency using the number of energy-minimized conformations necessary to compute K^* to SS-ε accuracy for each sequence (Fig. 6(A)). Since each conformation energy minimization is computationally expensive, efficiency is tantamount to reducing the number of minimizations. HOT, which reduces errors in the terms that are used to bound pairwise energies, did little to improve efficiency, yielding a 1.2-fold improvement on average. On the other hand, *BBK** improved efficiency by a minimum of 6-fold, an average of 25-fold, and a maximum of 70-fold. *BBK**+HOT was even more efficient, reducing the number of energy minimized conformations by a minimum of 6-fold, an average of 38-fold, and a maximum of 171-fold.

When we use running times as our efficiency metric (Fig. 6(B)), we also observed that HOT improved efficiency by only 1.1-fold. In contrast, *BBK** improved efficiency by a minimum of 41-fold, an average of 389-fold, and a maximum of 1982-fold. The *BBK**+HOT combination improved running times by similar margins: a minimum of 43-fold, an average of 277-fold, and a maximum of 944-fold. These data show that *BBK** not only substantially reduces the computational bottleneck associated with continuous rotamers in protein design for affinity, but also solves problems that were too large to solve using iMinDEE/A^*/K^*.

B.2 Computational Experiments: Design with Coupled Continuous Side-Chain and Backbone Flexibility

Table 1: Re-design of Human Fibronectin F1:*Staphylococcus aureus* FNBPA-5 interface (PDB Id: 2RL0) for binding affinity using a either a fixed or flexible FNBPA-5 backbone. The sequence space consisted of the wild-type sequence, represented in bold font, and 15 single amino-acid polymorphisms.

Beginning of Table 1				
Mutation	Fixed backbone K^* ratio ($\log_{10}$)	Flexible backbone K^* ratio ($\log_{10}$)	Fixed backbone sequence ranking	Flexible backbone sequence ranking
F156, R191, I192, C194, F649, D650, E651, E652, S653, T654	25.97	25.71	2	3
C194G	24.56	24.30	10	9
I192L	8.87	8.62	16	16
I192V	25.45	25.04	5	5
R191K	25.08	24.42	7	7
F156Y	25.93	25.15	4	4
F156V	25.29	24.35	6	8

Continuation of Table 1				
Sequence	Fixed backbone K^* ratio ($\log_{10}$)	Flexible backbone K^* ratio ($\log_{10}$)	Fixed backbone sequence ranking	Flexible backbone sequence ranking
F156I	24.83	23.89	9	10
F156L	24.98	24.61	8	6
T654S	26.38	25.95	1	1
E651D	23.96	23.69	11	11
D650E	25.95	25.72	3	2
F649Y	22.12	15.10	15	15
F649V	22.79	22.50	13	13
F649I	23.01	22.57	12	12
F649L	22.67	22.28	14	14
End of Table 2				

Table 2: Re-design of Human Fibronectin F1:*Staphylococcus aureus* FNBPA-5 interface (PDB Id: 2RL0) for binding affinity using a either a fixed or flexible FNBPA-5 backbone. The sequence space consisted of the wild-type sequence, represented in bold font, and 25 single amino-acid polymorphisms.

Beginning of Table 2				
Sequence	Fixed backbone K^* ratio ($\log_{10}$)	Flexible backbone K^* ratio ($\log_{10}$)	Fixed backbone sequence ranking	Flexible backbone sequence ranking
F156, R191, I192, C194, F649, D650, E651, E652, S653, T654	25.81	25.83	5	5
C194G	24.40	24.42	15	15
I192L	8.70	8.73	26	26
I192V	25.29	25.08	8	10
R191K	24.92	24.85	11	11
F156Y	25.77	25.63	6	7
F156V	25.13	25.08	10	9
F156I	24.67	24.62	13	13
F156L	24.82	24.79	12	12
T654S	26.23	26.16	3	3
T654N	26.33	26.24	2	2
T654Q	26.48	26.48	1	1
S653T	25.27	25.29	9	8
S653N	26.04	26.02	4	4
S653Q	23.61	23.65	18	17
E652D	23.66	23.64	17	18
E652H	24.43	24.42	14	14
E652G	23.83	23.76	16	16
E651H	22.41	22.41	24	24
D650E	25.77	25.76	7	6
D650H	22.50	22.51	23	23
F649Y	21.98	15.14	25	25
F649V	22.63	22.60	21	21
F649I	22.85	22.81	20	20
F649L	22.50	22.55	22	22
F649M	23.14	23.16	19	19
End of Table 1				

C Details of Protein Designs

C.1 MS Bound Designs

Table 3: Protein design test cases for Section 5. We varied the number of sequences in each protein-ligand system by adjusting the number of simultaneously mutable residues, n, from 1 to 4, yielding sequence spaces whose size s ranged from 10 to 10^7. Each mutable residue was also modeled as continuously flexible, as described in Section 5. All runs were set to enumerate the five best sequences.

Beginning of Table 3						
PDB code	Protein-ligand interface	Mutable / flexible residues	Design 1: n, s	Design 2: n, s	Design 3: n, s	Design 4: n, s
3CAL	Fibronectin f1 modules2-3 / staphylococcus aureus fnbpa-5	3, 40, 42, 100, 101, 103, 104, 105	1, 97	2, 4033	3, 96769	4, 1451521
2RF9	Egfr kinase / mig6 peptide	346, 350, 351, 352, 881, 911, 917	1, 86	2, 3026	3, 60482	4, 725762
4Z80	Toxoplasma gondii ama4 di-dii-egf1 / tgron2l1 peptide	210, 215, 258, 280, 281, 1301, 1312, 1318	1, 119	2, 6092	3, 179641	4, 2612738
2RFE	Egfr kinase / mig6 peptide	352, 357, 358, 359, 360, 904, 906, 924, 928	1, 110	2, 5186	3, 145154	4, 2612738
2RFD	Egfr kinase / mig6 peptide	352, 358, 359, 908, 924, 929	1, 74	2, 2162	3, 34562	4, 311042
2HNV	Q58v mutant of bovine neurophysin-i	244, 267, 295, 307, 336, 349, 376	1, 86	2, 3026	3, 60482	4, 725762
5IT3	Swirm domain of human lsd1	3, 30, 33, 37, 134, 150, 153, 170	1, 91	2, 3538	3, 79273	4, 1108081
2HNU	Bovine neurophysin-i	191, 193, 228, 236, 269, 273, 309, 314, 317	1, 110	2, 5186	3, 145154	4, 2612738
4JDD	Estrogen receptor alpha / 14-3-3 protein	174, 181, 222, 226, 230, 591, 593	1, 85	2, 3025	3, 60481	4, 725761
4PXF	Opsin / finger-loop peptide	72, 75, 76, 135, 139, 250, 310	1, 85	2, 3025	3, 60481	4, 725761
4WEM	F4 fimbrial adhesin faeg / llama antibody v1	29, 62, 281, 283, 900, 902, 905, 909	1, 96	2, 3949	3, 93745	4, 1391041
3MA2	Metalloproteinase mt1-mmp / timp-1	115, 274, 277, 281, 329, 338, 367, 373	1, 98	2, 4034	3, 96770	4, 1451522
1GWC	Glutathione s-transferase	53, 64, 67, 71, 78, 93, 100, 101, 104	1, 14	2, 1249	3, 52417	4, 1257985
5A6Y	Lecb lectin / mannose-alpha1,3mannoside	319, 334, 336, 338, 341, 361, 389, 433	1, 97	2, 4033	3, 96769	4, 1451521
3U7Y	Nih45-46 fab / gp120 of 93th057 hiv	279, 280, 460, 467, 558, 602, 789, 790	1, 97	2, 4033	3, 96769	4, 1451521
3RJQ	Anti-hiv llama vhh antibody a12 / c186 gp120	102, 105, 109, 113, 368, 601, 629, 695	1, 94	2, 3783	3, 87841	4, 1274401
2XQQ	Human dynll2 / ac-srgtqte	4, 6, 61, 62, 73, 77, 84	1, 85	2, 3025	3, 60481	4, 725761
4KT6	Beta-nad+ glycohydrolase/ifs	103, 112, 160, 226, 383, 384, 386	1, 85	2, 3025	3, 60481	4, 725761
3BUA	Trf2 trfh / apollo peptide	84, 118, 119, 120, 502, 506, 509, 510	1, 98	2, 4034	3, 96770	4, 1451522
4HEM	Llama vhh-02 / tp901-1	120, 122, 145, 163, 422, 426, 455, 457	1, 95	2, 3865	3, 90721	4, 1330561
5DC0	Monobody gg3 / abl1 sh2 domain	54, 58, 60, 164, 165, 187, 233, 234	1, 91	2, 3538	3, 79273	4, 1108081
5D68	Krit1 ard-ferm	297, 321, 322, 327, 330, 549, 603	1, 85	2, 3025	3, 60481	4, 725761
1B6C	Tgf-beta receptor / fkbp12	36, 37, 46, 59, 82, 90, 195, 196	1, 96	2, 3949	3, 93745	4, 1391041

Continuation of Table 3						
PDB code	Protein-ligand interface	Mutable / flexible residues	Design 1: n, s	Design 2: n, s	Design 3: n, s	Design 4: n, s
3BU8	Trf2 trfh / tin2 peptide	105, 116, 117, 119, 124, 257, 258	1, 86	2, 3026	3, 60482	4, 725762
5EM2	Erb1-ytm1	113, 151, 318, 446, 481, 483, 488	1, 85	2, 3025	3, 60481	4, 725761
4WWI	Staphylococcal protein a / human igg fc	10, 13, 17, 31, 253, 434, 435	1, 85	2, 3025	3, 60481	4, 725761
3EB6	Ciap2 ring domain / ubch5b	553, 559, 561, 1101, 1104, 1109, 1112, 1115	1, 97	2, 4033	3, 96769	4, 1451521
2XGY	Lentivirus (relik) capsid / cyclophilin a	76, 87, 89, 93, 109, 112, 184, 189, 250	1, 109	2, 5185	3, 145153	4, 2612738
2WZP	Lactococcal phage binding protein	202, 207, 242, 244, 630, 631, 655, 699, 703	1, 103	2, 4618	3, 121717	4, 2059345
4ZNC	Human igg / staphylococcal protein a	30, 36, 44, 47, 373, 398, 400	1, 85	2, 3025	3, 60481	4, 725761
2XXM	Hiv-1 capsid protein / camelid vhh	2, 3, 6, 7, 10, 169, 186, 211	1, 97	2, 4033	3, 96769	4, 1451521
2Q1E	Amyloidogenic kappa1 bence jones protein	254, 266, 314, 318, 418, 425, 426, 427	1, 98	2, 4034	3, 96770	4, 1451522
1A0R	Phosducin / transducin beta-gamma	311, 313, 332, 601, 605, 696, 697, 698, 729	1, 106	2, 4897	3, 133057	4, 2322433
1RX2	Dihydrofolate reductase, folate and NADPH	27, 28, 31, 50, 54, 94, 160	1, 103	2, 4336	3, 98261	4, 1252816
1AMU	Gramicidin synthetase 1 / amp	374, 413, 414, 427, 429, 432, 441, 443	1, 91	2, 3538	3, 79273	4, 1108081
2RL0	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	649, 650, 651, 654, 156, 172, 192, 193	1, 97	2, 4033	3, 96769	4, 1451521
4LAJ	Hiv-1 yu2 envelope gp120 glycoprotein / cd4-mimetic miniprotein	421, 434, 632, 627, 700, 707, 710, 712	1, 97	2, 4033	3, 96769	4, 1451521
1B74	Glutamate racemase	57, 220, 222, 223, 224, 243, 245, 252	1, 97	2, 4033	3, 96769	4, 1451521
5DC4	Monobody as25 / abl1 sh2 domain	35, 44, 47, 75, 77, 81, 82, 158, 234	1, 109	2, 5185	3, 145153	4, 2612737
4WYQ	Dicer-trbp	271, 277, 278, 362, 386, 387	1, 73	2, 2161	3, 34561	4, 311041
4WYU	Scribble pdz34 tandem	2, 3, 5, 6, 26, 29, 81	1, 85	2, 3025	3, 60481	4, 725761
4BTE	Dj-1 cu(i)	79, 83, 89, 95, 111, 115	1, 73	2, 2161	3, 34561	4, 311041
3GXU	Eph receptor / ephrin	27, 43, 135, 614, 618, 624, 625, 628	1, 91	2, 3538	3, 79273	4, 1108081
3K3Q	Llama antibody / c. Botulinum neurotoxin	98, 103, 107, 219, 222, 349, 354, 357	1, 97	2, 4033	3, 96769	4, 1451521
2P49	Camelid single-domain vhh antibody / rnase a	56, 64, 66, 68, 71, 106, 150, 154, 228	1, 110	2, 5186	3, 145154	4, 2612738
4MDK	Cdc34-ubiquitin-cc0651 complex	595, 600, 603, 605, 889, 930, 951	1, 85	2, 3025	3, 60481	4, 725761
2Q2A	Artj	144, 145, 149, 153, 156, 385, 386, 390, 394	1, 109	2, 5185	3, 145153	4, 2612737
4U3S	Coh3scab-xdoc m1scaa	79, 90, 131, 134, 174, 178, 180	1, 85	2, 3025	3, 60481	4, 725761

Continuation of Table 3						
PDB code	Protein-ligand interface	Mutable / flexible residues	Design 1: n, s	Design 2: n, s	Design 3: n, s	Design 4: n, s
2P4A	Camelid affinity matured single-domain vhh antibody / rnase a	132, 135, 140, 142, 377, 380, 381, 385, 387	1, 110	2, 5186	3, 145154	4, 2612738
4JC3	Estrogen receptor alpha / 14-3-3 protein	171, 174, 175, 222, 226, 593, 595	1, 85	2, 3025	3, 60481	4, 725761
1TP5	Pdz3 domain of psd-95 / kketwv	325, 326, 328, 340, 342, 372, 422, 424, 425	1, 109	2, 5185	3, 145153	4, 2612737
End of Table 3						

C.2 SS-ε Bound Designs

Table 4: Protein design test cases for Appendix B.1. 8-14 residues at each protein-ligand interface were modeled as both mutable and flexible. l is the size of the conformation space after dead-end-elimination pruning.

Beginning of Table 4			
PDB code	Protein-ligand interface	Mutable / flexible residues	l
3CAL	Fibronectin f1 modules2-3 / staphylococcus aureus fnbpa-5	3, 39, 40, 42, 100, 101, 103, 104, 105, 106	8566614
2RF9	Egfr kinase / mig6 peptide	337, 346, 350, 351, 352, 881, 885, 888, 908, 911, 917	1887923220
4Z80	Toxoplasma gondii ama4 di-dii-egf1 / tgron2l1 peptide	210, 211, 215, 249, 252, 258, 280, 281, 1301, 1305, 1310, 1312, 1318	3375435880
2RFE	Egfr kinase / mig6 peptide	352, 357, 358, 359, 360, 904, 905, 906, 924, 925, 928	2478051684
2RFD	Egfr kinase / mig6 peptide	352, 357, 358, 359, 360, 904, 908, 920, 924, 928, 929	1094527572
2HNV	Q58v mutant of bovine neurophysin-i	244, 255, 257, 267, 295, 296, 307, 336, 338, 349, 376	335437578
5IT3	Swirm domain of human lsd1	3, 30, 33, 34, 37, 41, 134, 149, 150, 153, 154, 170	37773033760
2HNU	Bovine neurophysin-i	191, 192, 193, 194, 228, 236, 237, 269, 272, 273, 275, 309, 314, 317	2766092544
4JDD	Estrogen receptor alpha / 14-3-3 protein	60, 174, 178, 181, 182, 222, 226, 230, 591, 593	5654890
4PXF	Opsin / finger-loop peptide	72, 75, 76, 77, 135, 138, 139, 249, 250, 253, 310, 311	1046727528
4WEM	F4 fimbrial adhesin faeg / llama antibody v1	28, 29, 31, 62, 281, 283, 900, 902, 905, 906, 909	1703340880
3MA2	Metalloproteinase mt1-mmp / timp-1	115, 274, 277, 281, 328, 329, 336, 338, 367, 373, 375	2105697020
1GWC	Glutathione s-transferase	52, 53, 64, 67, 71, 78, 93, 95, 99, 100, 101, 104	345408644
5A6Y	Lecb lectin / mannose-alpha1,3mannoside	245, 317, 319, 334, 336, 338, 341, 361, 389, 431, 433, 448, 450	3151995360
3U7Y	Nih45-46 fab / gp120 of 93th057 hiv	279, 280, 455, 460, 467, 558, 561, 602, 789, 790	263980885
3RJQ	Anti-hiv llama vhh antibody a12 / c186 gp120	102, 105, 109, 113, 368, 601, 629, 695, 697	118514120
2XQQ	Human dynll2 / ac-srgtqte	4, 6, 61, 62, 64, 73, 75, 77, 82, 84	560770921
4KT6	Beta-nad+ glycohydrolase / ifs	103, 112, 115, 160, 226, 383, 384, 386	137931048
3BUA	Trf2 trfh / apollo peptide	80, 84, 118, 119, 120, 122, 502, 506, 507, 509, 510	1397500768

Continuation of Table 4			
PDB code	Protein-ligand interface	Mutable / flexible residues	l
4HEM	Llama vhh-02 / tp901-1	119, 120, 122, 145, 163, 380, 422, 426, 455, 457	341807264
5DC0	Monobody gg3 / abl1 sh2 domain	17, 54, 58, 60, 164, 165, 187, 223, 231, 233, 234	751347450
5D68	Krit1 ard-ferm	297, 320, 321, 322, 327, 330, 331, 549, 596, 599, 600, 603, 604	494002390
1B6C	Tgf-beta receptor / fkbp12	36, 37, 46, 59, 82, 90, 99, 195, 196, 197, 200	28152780
3BU8	Trf2 trfh / tin2 peptide	102, 105, 116, 117, 119, 124, 127, 257, 258, 266	982110380
5EM2	Erb1-ytm1	113, 151, 181, 196, 227, 318, 390, 446, 481, 483, 484, 486, 488	49761924
4WWI	Staphylococcal protein a / human igg fc	10, 11, 13, 14, 17, 28, 31, 35, 252, 253, 311, 434, 435	1676312640
3EB6	Ciap2 ring domain / ubch5b	553, 559, 561, 1101, 1105, 1108, 1109, 1112, 1115	144471756
2XGY	Lentivirus (relik) capsid / cyclophilin a	76, 87, 89, 93, 109, 112, 184, 189, 250, 251, 255	66722600
2WZP	Lactococcal phage binding protein	202, 207, 242, 244, 630, 631, 655, 699, 703	20490720
4ZNC	Human igg / staphylococcal protein a	30, 33, 36, 43, 44, 47, 51, 373, 398, 400	17225115
2XXM	Hiv-1 capsid protein / camelid vhh	2, 3, 6, 7, 10, 169, 173, 182, 186, 211	194541095
2Q1E	Amyloidogenic kappa1 bence jones protein	254, 266, 256, 309, 314, 316, 318, 365, 418, 423, 425, 426, 427	123252930
1A0R	Phosducin / transducin beta-gamma	42, 311, 313, 332, 601, 605, 696, 697, 698, 729	63737408
1RX2	Dihydrofolate reductase, folate and NADPH	5, 27, 28, 31, 46, 50, 54, 94, 160	71389162
1AMU	Gramicidin synthetase 1 / amp	374, 413, 414, 427, 429, 432, 439, 441, 443, 450	151484688
2RL0	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	649, 650, 651, 654, 156, 172, 191, 192, 193, 194	592909440
4LAJ	Hiv-1 yu2 envelope gp120 glycoprotein / cd4-mimetic miniprotein	419, 421, 434, 632, 627, 700, 702, 707, 710, 712	153348639
1B74	Glutamate racemase	30, 57, 61, 220, 221, 222, 223, 224, 243, 245, 246, 252	508170560
5DC4	Monobody as25 / abl1 sh2 domain	35, 44, 47, 75, 77, 81, 82, 158, 161, 162, 164, 168, 234	7642166616
4WYQ	Dicer-trbp	206, 271, 274, 277, 278, 281, 362, 363, 383, 386, 387	264373872
4WYU	Scribble pdz34 tandem	2, 3, 4, 5, 6, 26, 27, 29, 36, 49, 81, 176	11263055220
4BTE	Dj-1 cu(i)	72, 79, 82, 83, 89, 92, 95, 102, 111, 112, 115	140435652
3GXU	Eph receptor / ephrin	27, 43, 78, 135, 560, 614, 616, 618, 624, 625, 628	222696544
3K3Q	Llama antibody / c. Botulinum neurotoxin	98, 100, 101, 102, 103, 107, 219, 222, 349, 354, 357, 358, 387	1574424348
2P49	Camelid single-domain vhh antibody / rnase a	56, 64, 65, 66, 68, 71, 106, 148, 150, 153, 154, 222, 228	5888750400
4MDK	Cdc34-ubiquitin-cc0651 complex	595, 599, 600, 603, 605, 889, 923, 930, 951, 953	406322880
2Q2A	Artj	144, 145, 146, 149, 152, 153, 156, 385, 386, 390, 393, 394, 397	23784865920
4U3S	Coh3scab-xdoc m1scaa	36, 79, 81, 90, 94, 131, 133, 134, 135, 174, 177, 178, 180	830160768
2P4A	Camelid affinity matured single-domain vhh antibody / rnase a	132, 135, 140, 141, 142, 143, 377, 380, 381, 385, 386, 387, 388	1053662176

Continuation of Table 4			
PDB code	Protein-ligand interface	Mutable / flexible residues	l
4JC3	Estrogen receptor alpha / 14-3-3 protein	49, 122, 171, 174, 175, 222, 226, 230, 593, 595	14794088
1TP5	Pdz3 domain of psd-95 / kketwv	325, 326, 328, 340, 342, 372, 421, 422, 423, 424, 425	966734976
End of Table 4			

C.3 Fixed Backbone vs. Flexible Backbone Protein Designs

Table 5: Protein design test cases for Sec. 5.3. 10 residues at the F1:FNBPA-5 interface were modeled as both mutable and flexible. s is the number of sequences in the design, and l is the size of the conformation space after dead-end-elimination pruning.

Beginning of Table 5					
Design	Protein-ligand interface	Mutable / flexible residues	Backbone PDB code(s)	s	l
1	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	156, 191, 192, 194, 649, 650, 651, 652, 653, 654	2RL0	16	9030
2	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	156, 191, 192, 194, 649, 650, 651, 652, 653, 654	2RL0	26	267964
3	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	156, 191, 192, 194, 649, 650, 651, 652, 653, 654	2RL0, 2RKY	16	12796302
4	Fibronectin f1 modules 4-5 / staphylococcus aureus fnbpa-5	156, 191, 192, 194, 649, 650, 651, 652, 653, 654	2RL0, 2RKY	26	2263554042
End of Table 5					