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

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

[1] Department of Computer Science, Duke University, Durham, NC, USA
[2] Department of Biochemistry, Duke University Medical Center, Durham NC, USA
[3] Computational Biology and Bioinformatics Program, Duke University, Durham, NC, USA
[4] Division of Infectious Diseases, Duke University Medical Center, Durham, NC, USA

Abstract. Protein design algorithms that compute binding affinity search for sequences with an energetically favorable free energy of binding. Recent work shows that the following design principles improve the biological accuracy of protein design: *ensemble-based design* and *continuous conformational flexibility*. Ensemble-based algorithms capture a measure of entropic contributions to binding affinity, K_a. Designs using backbone flexibility and continuous side-chain flexibility better model conformational flexibility. 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). However, previous provable methods that model ensembles and continuous flexibility are *single-sequence* algorithms, which are very costly: linear in the number of sequences and thus exponential in the number of mutable residues. 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. A key innovation of *BBK** is the *multi-sequence* (MS) bound: *BBK** efficiently computes a single provable upper bound to approximate K_a for *a combinatorial number of sequences*, and entirely avoids single-sequence computation for all provably suboptimal sequences. Thus, to our knowledge, *BBK** is the first provable, ensemble-based K_a algorithm to run in time *sublinear* in the number of sequences. Computational experiments on 204 protein design problems show that *BBK** finds the tightest binding sequences while approximating K_a for up to 10^5-fold fewer sequences than exhaustive enumeration. Furthermore, for 51 protein-ligand design problems, *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, but also efficiently performs designs that are too large for previous methods.

† These authors contributed equally to the work.
* Corresponding author: Bruce R. Donald, Tel: 919-660-6583, Fax: 919-660-6519, Email: brd+recomb17@cs.duke.edu

1 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 functional design into a structural optimization problem whose goal is to find amino acid sequences that fold into specified three-dimensional structures. 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 [1]. Protein design algorithms [5, 9] 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 [7, 37, 41] and to design enzymes [3, 13, 32, 49], new drugs [19], inhibitors of protein-protein interactions [15, 42], epitope-specific antibody probes [12], and even neutralizing antibodies [17, 45].

Computational methods can potentially search a large number of sequences to predict the proteins that bind most tightly to a target ligand in less time and with fewer resources than *in vitro* methods such as phage display [2, 38]. 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 [5,15,18,32]. Second, for each sequence, finding the lowest energy conformations that most influence binding affinity is NP-hard [26, 39, 40, 55, 56], making algorithms that guarantee optimality expensive for larger designs. Third, mutating a protein sequence 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; see also Supplementary Information [36] (**SI**) Section 2.

Global Minimum Energy Conformation (GMEC)-based algorithms [4, 10, 20, 43, 53] assume that the lowest energy conformation accurately predicts binding affinity. However, GMEC-based designs cannot accurately model entropic

change due to binding [6] and can disproportionately favor sequences with energetically favorable GMECs over sequences with tight binding affinity [3, 15, 32, 42, 46, 47]. Many protein design algorithms [29, 48, 50, 51, 53] rely on a simplified, discrete model of side-chain flexibility. 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 [10, 15, 21, 22, 42, 43] that incorporate continuous side-chain flexibility [10, 15]. Another important aspect of design algorithms is the quality of the computed results. Whereas GMEC-based design is NP-hard [26,40,55,56], computation of thermodynamic ensembles and associated partition functions is #P-hard [35, 52, 53]. Provable protein design algorithms either return the optimal sequences or conformations [4, 10, 23, 28, 43, 48, 50, 51, 53] with respect to the input model, or return provably good approximate solutions [15, 21, 22, 32, 42]. Non-provable algorithms such as Metropolis Monte Carlo methods [27, 29, 30] instead use stochastic methods to rapidly sample the space described by the input model. These algorithms return solutions without any guarantees. Thus, ensemble-based design, a realistic model of structural flexibility, and provable optimality of the computed sequences with respect to the input model improve the predictive power of protein design algorithms. However, each of these design principles also increases the cost of protein design (as a function of the number of sequences).

Single-sequence algorithms, which explicitly evaluate each possible sequence, are powerful and versatile. Molecular dynamics [31, 57], for instance, is frequently applied to design for binding affinity. The approach models ensembles and continuous flexiblity. Provable algorithms have also been developed to model these two phenomena. The K^* algorithm [15, 32, 42] in OSPREY [10, 11, 13–15, 20–23, 25, 32, 42, 43] uses a combination of dead-end elimination pruning [10, 15] and A^* [24, 28, 44] gap-free conformation enumeration to provably and efficiently approximate K_a. K^* and all previous provable ensemble-based algorithms that model continuous side-chain flexibility [21, 22] 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. Thus, designs with many mutable residues rapidly become intractable when using single-sequence algorithms. The COMETS algorithm [20] in OSPREY is the only provable multi-state design algorithm that is more efficient than single-sequence algorithms. However, its binding predictions are GMEC-based rather than ensemble-based. Additional background and references are located in **SI** [36], Sec. 1.

To efficiently search large sequence spaces while retaining all the benefits of provable guarantees, ensemble-based design, and continuous side-chain flexibil-

ity, we present a new, provable algorithm: Branch and Bound over K^* (*BBK**). The key innovation of *BBK** is the *multi-sequence* (MS) bound: *BBK** efficiently computes upper and lower bounds on the binding affinities of *partial sequences*, which are shared by a combinatorial number of full sequences. By avoiding costly single-sequence computation, *BBK** runs in time *sublinear* 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 and efficiently enumerates sequences in a gap-free decreasing order of binding affinity. 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 empirically combinatorially faster, returns a gap-free list of sequences in decreasing order of binding affinity, and runs in time sublinear 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 ε-approximation to the desired partition functions.
4. 255 protein designs showing that *BBK** approximates binding affinity for full 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 [10, 11, 13–15, 20–23, 25, 32, 42, 43] protein design software package, available for download as free software.

2 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 thus be cast as an optimization problem [5]. 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* [10, 15]. We define $E_X(c)$ to be the energy of c for structure X after energy-minimizing the side-chains of mutable residues.

2.1 *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. 1, the K^* approximation and, by extension, the full partition function, are #P-hard to compute [35, 52, 53]. 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 [54] or non-provably [8]. An efficient ε-approximation to $Z_X(s)$ is computed in [15,32,42]. 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 take time 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, in 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 a specific 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. 3, pruning these sequences circumvents prohibitive computational costs required to compute many single-sequence K^* scores.

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

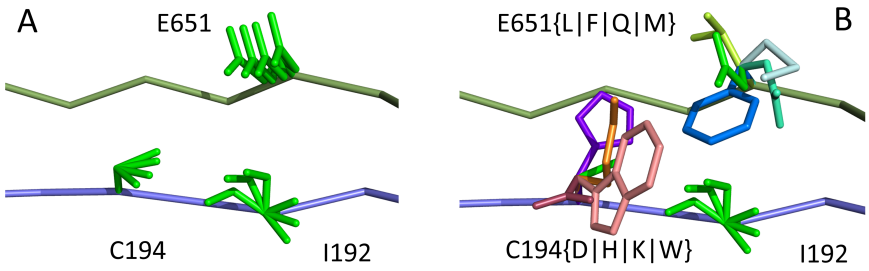

Fig. 1. A toy protein design problem in which conformational ensembles (A) and optimal mutations (B) must be computed at 3 residues. Residues of the fibronectin F1 module (Fn, blue ribbon), and of a fragment of *S. aureus* fibronectin binding protein A (FNBPA-5, green ribbon) are shown (PDB id: 2RL0). 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 Fn:FNBPA-5 design problem is described in Sec. 4.3.

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^* score 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'. 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. Details of the algorithm, proofs of its space and time complexity, and comparison to iMinDEE/A^*/K^* are provided in Appendix A of the **SI** [36]. An additional enhancement, pruning by fold stability compared to wild type, is described in Appendix A.7 of the **SI** [36].

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 [15, 32, 42]. 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 $\mathcal{O}(q^a(a^2 + q^2 t^2 un))$, 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 $\mathcal{O}(q^{a+2} t^2 un)$ 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. 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 $\mathcal{O}(q^{a+2} t^2 un)$ 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 of the **SI** [36]. 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

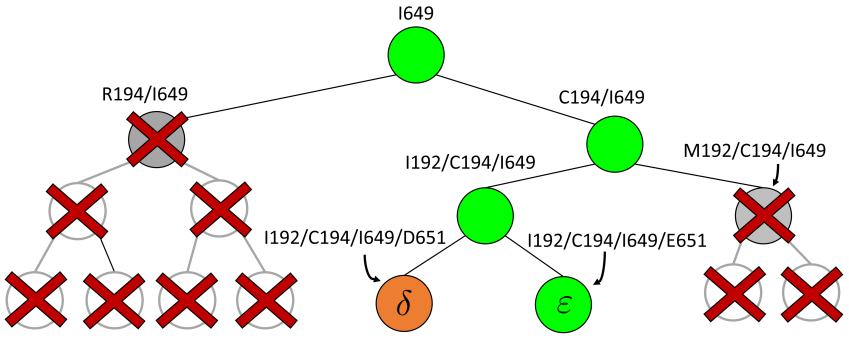

Fig. 2. *BBK** pruning efficiently explores the sequence space. An example design of residues 192 and 194 of the fourth fibronectin F1 module, and residues 649 and 651 of a fragment *S. aureus* fibronectin binding protein A 5 is shown (Fig. 1, PDB Id: 2RL0). As *BBK** searches the sequence space (tree above) its multi-sequence bounds provably prune 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 early, after computing a mere δ*-approximation bound* (orange leaf node), which is cheaper, and not as tight as an 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.

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 full 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 is provided in Appendix A.3 of the **SI** [36]. 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 [28] and to provably approximate the partition function of single sequences [5,15,32,42]. 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.

3.1 Algorithm Overview

BBK* bounds all possible sequences either with the MS bounds described in Sec. 3, or by computing a single-sequence bound as described in [13, 32, 42]. In brief, to bound a single sequence, *BBK** computes a gap-free list of conformations whose statistical mechanical 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 [15, 16] as *SS-δ bounds*. As the gap-free list of conformations 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 a provable ε-approximation, which we will refer to as 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 [15].

BBK* maintains a max heap whose node values correspond to either full or partial sequences and whose node keys are an upper bound on all K^* scores (Eq. 2) in the sequence space represented by the node. The heap is initialized with a node representing 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 m additional conformations ($m = 8$ in our study), tightening the SS-δ bound. x is reinserted into the heap, with the updated SS-δ bound as its key. Computing this tighter SS-δ bound is important for pruning sequences, as shown by our computational experiments in Sec. 4.2.

3. *Return*. If node x contains a full sequence, whose bound is an SS-ε bound, then the sequence in x has the best K^* score of all unenumerated sequences (as with A^*, all better sequences are guaranteed to have already been enumerated). The sequence of x is returned.

BBK* terminates when it has enumerated the top k sequences (by SS-ε bound), where k is a user-specified number. A detailed description of the algorithm is provided in Appendix A.8 of the **SI** [36].

4 Computational Experiments

We implemented *BBK** in our laboratory's open source OSPREY [11] protein design package and compared our algorithm to the previous state-of-the-art single-sequence iMinDEE/A^*/K^* algorithm [15, 32, 42]. 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 [10, 43]. As in [10, 15], rotamers from the Penultimate Rotamer Library [33] 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) [15], followed by either iMinDEE/A^*/K^* or *BBK**. The initial pruning window [10] was 0.1 kcal/mol, and the SS-ε bound accuracy was 0.683 (details are provided in Appendix A.9 of the **SI** [36]). 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 of the **SI** [36].

4.1 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 sublinear in the number of sequences. We first measured the efficiency

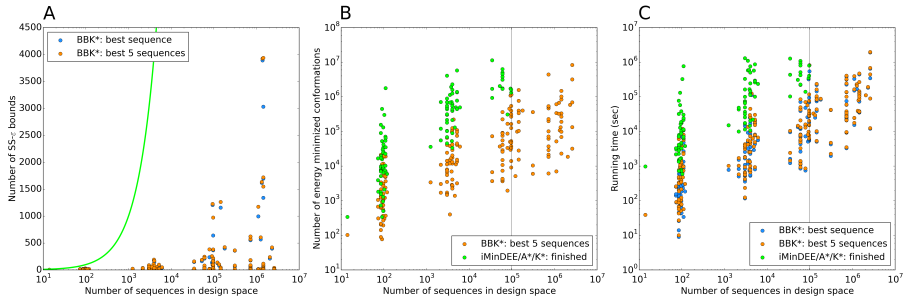

Fig. 3. *BBK** is up to five orders of magnitude more efficient than iMinDEE/A^*/K^*. *BBK** completed all 204 designs within a 30 day limit, while iMinDEE/A^*/K^* completed only 107. (A) The number of SS-ε bounds performed vs. the number of sequences in the design space. Results are shown for computing 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 limit. 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^*.

of *BBK** using the number of SS-ε bounds computed. Next, we measured efficiency using the number of conformation energy minimizations performed. Last, we compared the running times of *BBK** to those of 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 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 (viz. 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 [2, 38].

4.2 Sequence Space Pruning

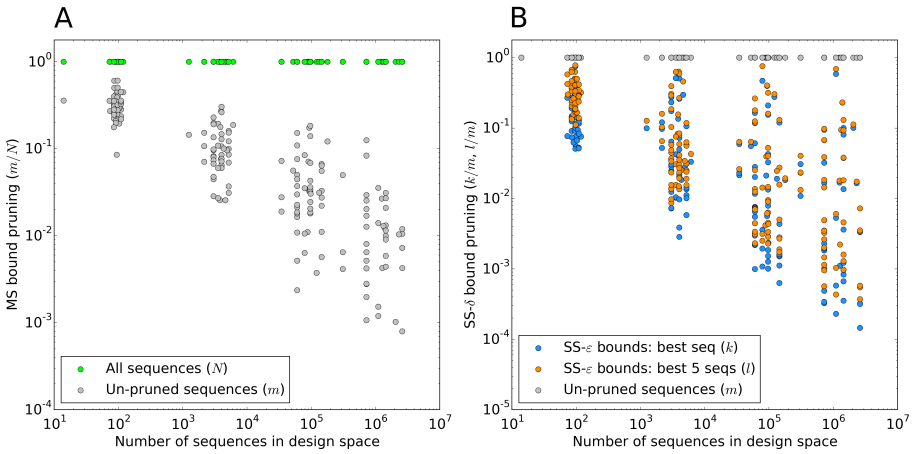

Fig. 4. *BBK** used MS and SS-δ bounds to prune up to 99.99999% of the sequence space. (A) Sequence space reduction due to MS pruning. The fraction of un-pruned sequences (gray) normalized to the total number of sequences (green). *BBK** used MS bounds to provably prune up to 99.9% of the sequence space. *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** calculated 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.

BBK* owes its efficiency to two complementary modes of sequence pruning: MS bound pruning and SS-δ bound pruning. 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. 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. Further details are provided in Appendices A.10 and B.1of the **SI** [36].

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 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 protein-protein interface. *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. See **SI** [36] Section 5.2 for details.

4.3 Design with Coupled Continuous Side-Chain and Backbone Flexibility

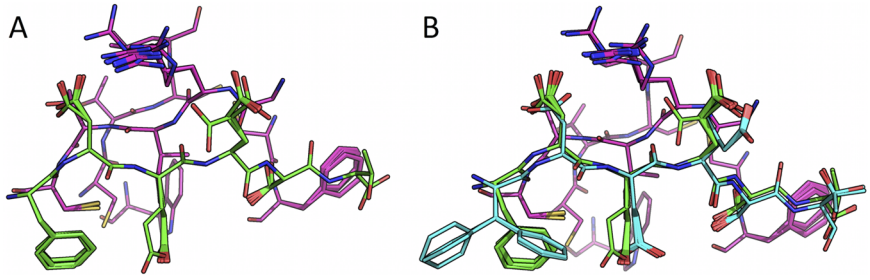

Fig. 5. *BBK** efficiently handles coupled continuous side-chain and local backbone flexibility. Selected residues from ensembles, computed by *BBK**, of human fibronectin F1 modules 4-5 (magenta) in complex with a fragment of *S. aureus* fibronectin binding protein A 5 (FNBPA-5, PDB id: 2RL0, [34]). 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 [58]). 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.

To determine whether design with a *fixed* backbone and continuous rotamers predicts tight binding in the same sequences as does a model with both *local backbone flexibility* and continuous rotamers, we used *BBK** to redesign the Human Fibronectin F1:*Staphylococcus aureus* FNBPA-5 interface [34] (PDB id: 2RL0) for binding affinity. As we will discuss below, the flexible backbone model favors binding in different sequences than the fixed backbone model does. Details of our experimental protocol are provided in Appendix C.3 of the **SI** [36].

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 only 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 more tightly than WT 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 Section B.2 of the **SI** [36]). 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.

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 optimization. Thus, *BBK** provides provable methods to analyze the difference in predicted sequences between different models of side-chain and backbone flexibility.

5 Conclusion

BBK* fills an important *lacuna* in protein design: we presented a novel algorithm that can search not over the energies of single-conformations, but instead over the binding affinity of sequences. *BBK** is, to our knowledge, the first provable, ensemble-based algorithm to search over binding affinity and run in time *sublinear* 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 sequence 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. *BBK** enables computational protein design by providing new K_a algorithms, with provable guarantees, for these large-scale protein designs.

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

References

1. F. E. Boas, P. B. Harbury. *Curr Opin Struct Biol* **17**, 199 (2007).
2. S. Carmen, L. Jermutus. *Brief Funct Genomic Proteomic* **1**, 189 (2002).
3. C.-Y. Chen *et al*. *Proc Natl Acad Sci U S A* **106**, 3764 (2009).
4. J. Desmet *et al*. *Nature* **356**, 539 (1992).
5. B. R. Donald. *Algorithms in Structural Molecular Biology* (MIT Press, Cambridge, MA, 2011).
6. S. J. Fleishman *et al*. *Protein Sci* **20**, 753 (2011).
7. K. M. Frey *et al*. *Proc Natl Acad Sci U S A* **107**, 13707 (2010).
8. M. Fromer, C. Yanover. *Bioinformatics* **24**, i214 (2008).
9. P. Gainza, H. M. Nisonoff, B. R. Donald. *Curr Opin Struct Biol* **39**, 16 (2016).
10. P. Gainza, K. E. Roberts, B. R. Donald. *PLoS Comput Biol* **8**, e1002335 (2012).
11. P. Gainza *et al*. *Methods Enzymol* **523**, 87 (2013) (Program, user manual, and source code are available at www.cs.duke.edu/donaldlab/software.php).
12. I. Georgiev *et al*. *Retrovirology* **9** (2012).
13. I. Georgiev, B. R. Donald. *Bioinformatics* **23**, i185 (2007).

14. I. Georgiev, R. H. Lilien, B. R. Donald. *Bioinformatics* **22**, e174 (2006).
15. I. Georgiev, R. H. Lilien, B. R. Donald. *J Comput Chem* **29**, 1527 (2008).
16. I. S. Georgiev. Novel algorithms for computational protein design, with applications to enzyme redesign and small-molecule inhibitor design. Ph.D. thesis. Duke University. http://hdl.handle.net/10161/1113 (2009).
17. I. S. Georgiev *et al*. *J Immunol* **192**, 1100 (2014).
18. M. K. Gilson *et al*. *Biophys J* **72**, 1047 (1997).
19. M. J. Gorczynski *et al*. *Chem Biol* **14**, 1186 (2007).
20. M. A. Hallen, B. R. Donald. *J Comput Biol* **23**, 311 (2016).
21. M. A. Hallen, P. Gainza, B. R. Donald. *J Chem Theory Comput* **11**, 2292 (2015).
22. M. A. Hallen, J. D. Jou, B. R. Donald. *J Comput Biol* **Epub ahead of print** (2016).
23. M. A. Hallen, D. A. Keedy, B. R. Donald. *Proteins* **81**, 18 (2013).
24. P. Hart, N. N.J., B. Raphael. *IEEE Trans on SSC* **4**, 100 (1968).
25. J. D. Jou *et al*. *J Comput Biol* **23**, 413 (2016).
26. C. L. Kingsford, B. Chazelle, M. Singh. *Bioinformatics* **21**, 1028 (2005). PMID: 15546935.
27. B. Kuhlman, D. Baker. *Proc Natl Acad Sci U S A* **97**, 10383 (2000).
28. A. R. Leach, A. P. Lemon. *Proteins* **33**, 227 (1998).
29. A. Leaver-Fay *et al*. *Methods Enzymol* **487**, 545 (2011).
30. C. Lee, M. Levitt. *Nature* **352**, 448 (1991).
31. J. Leech, J. F. Prins, J. Hermans. *Computational Science and Engineering* **3**, 38 (1996).
32. R. H. Lilien *et al*. *J Comput Biol* **12**, 740 (2005).
33. S. C. Lovell *et al*. *Proteins* **40**, 389 (2000).
34. S. K. Lower *et al*. *Proc Natl Acad Sci U S A* **108**, 18372 (2011).
35. H. Nisonoff. *B.S. Thesis. Department of Mathematics, Duke University. http://hdl.handle.net/10161/9746* (2015).
36. A. A. Ojewole *et al*. Supplementary information: BBK* (Branch and Bound over K*): A provable and efficient ensemble-based algorithm to optimize stability and binding affinity over large sequence spaces for sparse approximations of computational protein design, (2015) (Available at http://www.cs.duke.edu/donaldlab/Supplementary/recomb17/bbkstar).
37. A. Ojewole *et al*. *Methods Mol. Biol.* **1529**, 291 (2017). PMID: 27914058.
38. G. Pál *et al*. *J Biol Chem* **281**, 22378 (2006).
39. J. Peng *et al*. *arXiv:1504.05467 [q-bio.BM]* (2015).
40. N. A. Pierce, E. Winfree. *Protein Eng* **15**, 779 (2002).
41. S. M. Reeve *et al*. *Proc Natl Acad Sci U S A* **112**, 749 (2015).
42. K. E. Roberts *et al*. *PLoS Comput Biol* **8**, e1002477 (2012).
43. K. E. Roberts, B. R. Donald. *Proteins* **83**, 1151 (2015).
44. K. E. Roberts *et al*. *Proteins* **83**, 1859 (2015).
45. R. S. Rudicell *et al*. *J Virol* **88**, 12669 (2014).
46. D. Sciretti *et al*. *Proteins* **74**, 176 (2009).
47. N. W. Silver *et al*. *J Chem Theory Comput* **9**, 5098 (2013).
48. D. Simoncini *et al*. *J Chem Theory Comput* **11**, 5980 (2015).
49. B. W. Stevens *et al*. *Biochemistry* **45**, 15495 (2006).
50. S. Traoré *et al*. *Bioinformatics* **29**, 2129 (2013).
51. S. Traoré *et al*. *J Comput Chem* **37**, 1048 (2016).
52. L. G. Valiant. *Theoretical computer science* **8**, 189 (1979).
53. C. Viricel *et al*. *The 22nd International Conference on Principles and Practice of Constraint Programming* (2016).
54. M. J. Wainwright, T. S. Jaakkola, A. S. Willsky. *CoRR* **abs/1301.0610** (2013).
55. J. Xu. *9th Annual International Conference, RECOMB* **3500**, 423 (2005).
56. J. Xu, B. Berger. *The Journal of the ACM* **53**, 533 (2006).
57. F. Zheng *et al*. *J Am Chem Soc* **130**, 12148 (2008).
58. J. Zhou, G. Grigoryan. *Protein Sci* **24**, 508 (2015).