COMETS (Constrained Optimization of Multistate Energies by Tree Search): A Provable and Efficient Algorithm to Optimize Binding Affinity and Specificity with Respect to Sequence

Mark A. Hallen[1] and Bruce R. Donald[1,2(✉)]

[1] Department of Computer Science, Duke University, Durham, NC 27708, USA
[2] Department of Biochemistry, Duke University Medical Center, Durham, NC 27710, USA
brd+recomb15@cs.duke.edu

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

1 Introduction

Protein design is the prediction and selection of protein sequences with desired properties, generally some combination of structure stability, binding to desired ligands, and lack of binding to undesired ligands. The gold standard for protein design is natural evolution, in which protein mutations confer fitness advantages only if several desired properties are all present: mutants must be sufficiently stable, effective at binding or catalysis, and selective for their fitness-conferring function [6]. Researchers have tried to emulate this process by directed evolution experiments [1]. But methods to optimize these properties computationally [5] allow enormous sequence spaces to be searched without enormous resource expenditures, and thus greatly expand the space of possible designs. Such searches require algorithms that do not analyze each candidate sequence separately: large sequence spaces are too expensive to analyze one by one. Computational protein designers have used three different strategies to achieve the desired properties with their new

T. Przytycka (Ed.): RECOMB 2015, LNBI 9029, pp. 122–135, 2015.
DOI: 10.1007/978-3-319-16706-0_14

sequences: energy minimization of a single desired protein or complex structure ("single-state design"); heuristic minimization of some function combining multiple desired properties ("traditional multistate design methods"); and analysis of one sequence at a time in detail ("single-sequence analysis").

Single-state design is the most developed class of dedicated protein design algorithms. It is commonly used to improve fold stability by selecting mutants that minimize the protein's total energy [4,5,10,17,24], and to increase binding affinity by selecting mutants that minimize the energy of a complex [15,22]. Some of these methods are provable: given a sequence space to search, a model of the protein's conformational space, and an energy function, they are guaranteed to return the lowest-energy sequence and conformation (the global minimum-energy conformation, or GMEC). The dead-end elimination (DEE) [4] and A* [25] algorithms have this guarantee. In their original form, they assume a discrete conformational space, but they have been extended to include both continuous sidechain [10,15] and backbone [13,19] flexibility. Provable single-state methods can also enumerate either a gap-free list of the lowest-energy sequences and conformations [10,19,25], or of the sequences with the lowest-energy optimal conformations [33]. Other single-state methods are not provable, most prominently Metropolis Monte Carlo-based methods [24,27], but are popular for reasons of computational speed. All these methods use some simplified model of protein conformational flexibility. A popular but highly approximate model is to allow the conformation of each amino acid to be selected from a discrete set, referred to as *rotamers*. This model can be made substantially more accurate by allowing small, continuous conformational adjustments around the rotameric conformations, which can be incorporated while maintaining provable accuracy [10,15,19].

Single-state design can be thought of as the stabilization of a desired "state" of a protein—essentially, its fold, overall conformation, and ligand-binding mode. This paradigm can be extended to include multiple states, possibly with different ligands, in order to specify multiple desired properties for the designed sequence. This strategy is known as multistate protein design [3]. DEE has been extended to multistate design in the type-dependent DEE algorithm [39]. This algorithm prunes rotamers that are guaranteed not to be part of the optimal conformation of a state of the protein. It offers a significant advantage in efficiency, but does not reduce the number of sequences that must be considered, because it only eliminates rotamers by comparison to more favorable rotamers of the same amino-acid type. On the other hand, non-provable methods have also been developed to try to optimize objective functions based on the energies of multiple states, without considering each sequence separately. Genetic algorithms have been used to optimize differences in energy between states [29] as well as other objective functions [26], and belief propagation has been used to optimize sums of energies of different states, in order to design a binding partner appropriate for multiple ligands [7–9]. Type-dependent DEE can also be combined with such techniques, to reduce the conformational space that is searched heuristically [8,39]. However, previous multistate design algorithms cannot provide any guarantees about the optimality of their designed sequences without an exhaustive search over sequence space.

Methods that consider each candidate sequence explicitly are another important and highly versatile category of computational protein design methods. They are the most similar to natural evolution, in the sense that natural selection generates each "candidate" mutation explicitly and then subjects it to various selective pressures. However, the computational costs can be very high—linear in the number of sequences, and thus exponential in the number of simultaneously mutable positions. Molecular dynamics can be applied for single-sequence analysis in protein design [28,40], using simulations over time to investigate the properties of a candidate sequence. Molecular dynamics readily models all types of protein flexibility with many different energy functions, including effects like solvent polarization [36] or explicit solvent. It also allows the user to account for entropic contributions to binding energies. More recent algorithms account for entropy without the steep costs of simulation over time. The K^* algorithm in OSPREY [10,11,15,30] predicts the binding of a mutant protein sequence to a ligand by computing an ensemble of low-energy protein states to provably approximate the binding constant K_a within a desired relative error for the user-specified flexibility model and energy function. Though it provides a vast speedup relative to exhaustive search over all conformations at each sequence, it does require explicit consideration of each sequence sufficient to bound the energies in its ensemble. K^* in OSPREY [11] has yielded several multistate protein designs that were successful experimentally. The calculations have involved both comparisons of the bound and unbound states of a single complex [18,34,35] and of multiple complexes [2,6,12,37], and the OSPREY-designed proteins have performed well *in vitro* [2,6,12,18,34,35,37] and *in vivo* [6,18,34,35] as well as in non-human primates [35].

We now present an algorithm distinct from these three traditional strategies that combines advantages from all three: COMETS. Like other multistate methods, it optimizes an energy measure that considers multiple states: for example, it can directly optimize the binding energy (the difference in energy between the bound and unbound states), or the difference in binding energy between two different ligands. Like single-sequence analysis, it allows consideration of a wide variety of stability, affinity, and specificity requirements during sequence selection. This is facilitated by its accommodation of optimization constraints: for example, it can optimize binding to one ligand while constraining binding energy for other ligands. It provably returns the best sequence for its specified optimization problem, without performing an exhaustive search over the possible sequences. Some previous methods can do this for single-state design problems, but before COMETS it was impossible for multistate problems. As a result, COMETS provides a vast performance improvement over the previous state-of-the-art for provable multistate design, which is exhaustive search over sequence space.

By presenting COMETS, this paper makes the following contributions:

1. A novel and versatile framework for multistate protein design, allowing constrained optimization of any linear combinations of state energies.
2. An algorithm to solve problems in this framework that provably obtains the same results as exhaustive search over sequences but is combinatorially

faster than this exhaustive search, as shown by empirical measurements on 52 protein design problems.
3. Support for continuous sidechain and backbone flexibility in COMETS.
4. The ability to enumerate as many constraint-satisfying sequences as desired, in a gap-free list in ascending order of the desired objective function.
5. An implementation of COMETS in our laboratory's open-source OSPREY protein-design software package [2,6,15], available for download at [16] as free software.

2 Methods

2.1 Problem Formulation

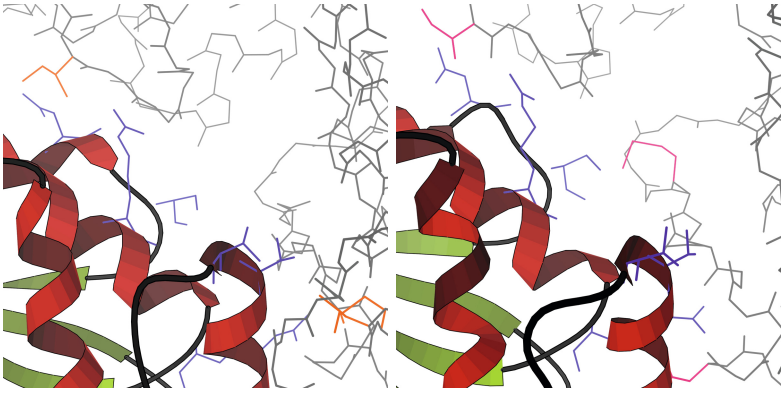

Fig. 1. Flexible and mutable residues in a design for specificity. The apoptotic regulator CED4 forms two different dimers, one to block apoptosis (left; PDB id 2a5y [38]) and one to induce it (right; PDB id 3lqr [32]). We want to design for specificity (to block apoptosis), so we allow mutations to some residues in the binding site (blue). To accurately model the conformational changes induced by the mutations, we also model residues on the other side of each interface that interact with the mutable residues as flexible (orange, pink). Analysis of this calculation and others is described in Section 3.

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

$$f(\mathbf{s}) = c_0 + \sum_{a \in A} c_a E_a(\mathbf{s}) \tag{1}$$

where the c_a are real coefficients. We call these functions *linear multistate energies* (LMEs). COMETS is an algorithm to minimize any LME $f(\mathbf{s})$ with respect

to sequence $\mathbf{s}$, with constraints of the form $c_i(\mathbf{s}) < 0$, where each c_i is also an LME. LMEs are suitable for representing stability, affinity, and selectivity requirements in protein design. For example, to optimize a binding energy, we set A to consist of the bound state b and the unbound state u, and optimize $f(\mathbf{s}) = E_b - E_u$. That is, we set $c_b = 1$, $c_u = -1$ and $c_0 = 0$ for our objective function. A highly simplified, "toy" example of this setup is in **Supplementary Information (SI)** A [20].

The choice of objective function and constraints defines the physical problem we wish to solve. We require a computational model of proteins to convert this into a computational problem. To model protein flexibility, we use the very general model of the DEEPer algorithm [19] in OSPREY. The protein in each state is allowed to have any number of degrees of freedom, which can be either continuous or discrete, and which fully specify both the sequence and conformation of the protein. Each residue in each state has a set of "residue conformations" (RCs) [19]. An RC is a portion of conformational space defined by bounds on every conformational degree of freedom available to the residue. A residue conformation is associated with a specific amino acid type. Residue conformations are chosen to be small enough that once a residue conformation is assigned to every residue, the energy minimum over this limited conformational space can be found by local minimization. This framework is suitable for accommodating both continuous sidechain and backbone flexibility, but it reduces to the model of continuous sidechain flexibility of [10,15] if only sidechain dihedrals are used as continuous degrees of freedom. If each sidechain dihedral is confined to a single value within each residue conformation, then this special case is just the commonly used rigid-rotamer approximation [4,25]. In both of these special cases, each residue conformation represents a single sidechain rotamer.

The model of flexibility may differ between states; in fact, different residues may be made flexible. For example, in a calculation with a bound and an unbound state of a protein, the ligand will have flexibility in the bound state, but will be absent from the unbound state (Fig. 1). But all states have the same set of mutable residues, and the same set of allowed amino-acid types at each mutable residue. This way, COMETS outputs a sequence applicable to all states.

To model energy, we must have an "energy function" that estimates the energy of a given sequence and conformation. Our implementation of COMETS uses a pairwise additive energy function, meaning that it is a sum of terms that depend on the conformations of at most two residues. This property is only used in the computation of lower bounds for LMEs over subsets of the sequence space and state conformational spaces (Section 2.2; **SI** B [20]), so a non-pairwise energy function that admits such lower-bound computations would also be compatible with COMETS. COMETS will return optimal results for the given model of flexibility and energy function.

2.2 A* Over Sequences

COMETS uses the A* [21] search algorithm to search sequence space. In most previous applications of A* to protein design [15,25], nodes of the tree correspond

to partially defined conformations. Each partially defined conformation is specified by RC assignments for one or more residues. Thus, each node corresponds to the conformational space made up of all conformations consistent with the partial definition. A node's score is a lower bound on all the conformational energies in this space. COMETS is similar, but nodes correspond to partially defined *sequences* and thus to a sequence space. A node's score is a lower bound on the objective function for all sequences in the node's sequence space (Fig. 2).

In A*, we repeatedly process the lowest-scoring node in the tree. Processing a node means either splitting it into several nodes that partition its sequence space, or computing a higher score (i.e., tighter bound) for it (that is still a valid lower bound). Score computation may involve conformational search (Fig. 2), and some nodes will be processed until their sequence is fully defined and the optimal conformation for each state is fully determined. These nodes are termed *fully processed*, and their objective function and constraint LMEs can be evaluated exactly. When the lowest-scoring node is fully processed, we can return its sequence as optimal, because its objective function value (at optimal conformations for each state) is better than any sequence in any of the sequence spaces of the other nodes in the tree. This is because the other nodes' scores are lower bounds on their optimal objective function values.

Types of Nodes. We will store two types of nodes in our tree (Fig. 2). Examples of each type of node in the toy example are given in **SI** A [20].

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

The second type of node has a fully defined sequence: an amino-acid type assigned for each mutable residue. At each such node, for each state, we store an A* tree expanding the conformational space for that sequence. These trees are identical to those used in DEEPer in OSPREY [19]: their nodes each represent a subset of conformational space, defined by RC assignments to some of the residues, which restrict the values of the proteins' degrees of freedom to the bounds associated with the assigned RCs. The score of each node is a lower bound on the energy of all conformations in its allowed conformational space. If a node has a fully-defined sequence and the lowest-scoring node of each of its conformational trees has an RC assignment to each residue, then the lowest node score in each conformational tree will be the optimal energy of its state for the node's sequence. Thus, by evaluating the objective function and constraints using these optimal state energies, we obtain the exact values of the objective function and constraint LMEs. So the node is fully processed, and will be removed from consideration if it violates any constraints.

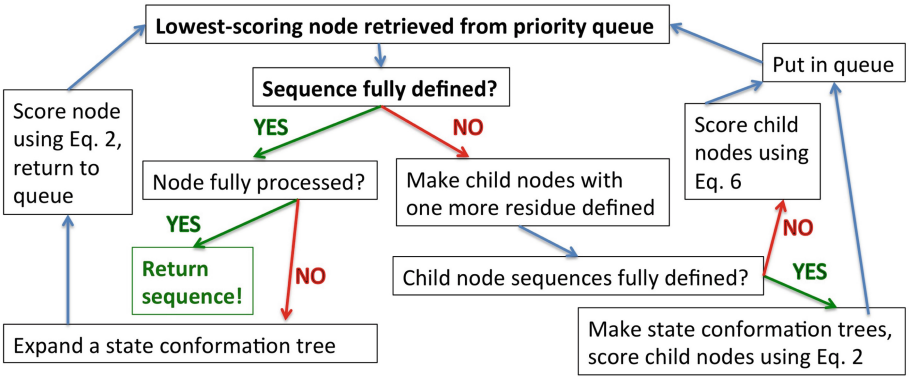

Fig. 3. COMETS is a sequence of node-processing operations

Node-processing Operations. For either type of node, node processing consists of two steps: an "expansion" step and a "bounding" step (Fig. 3). Every time we draw a node from the priority queue, meaning it is the lowest score in the tree, we choose the appropriate processing operation and perform it (Fig. 3).

Expansion step. For a node without a fully defined sequence, the expansion step splits the node n into several nodes whose sequence spaces partition the sequence space of n. If the first mutable residue without an amino-acid type assigned in n is residue r, then this partition can be performed by creating a node for each amino-acid type a allowed at r. These child nodes will each have a sequence space identical to that of n, except with the amino-acid a assigned to residue r. For a node n with fully defined sequence, we split the lowest-scoring node in one of n's conformational trees: each child node has a different RC assignment for a residue whose RC is not assigned at the parent node. This is the same type of split used in DEEPer [19], and essentially as in previous protein design applications of A*.

Bounding step. In the bounding step, a lower bound is computed for the objective function and for each of the constraint LMEs. If the lower

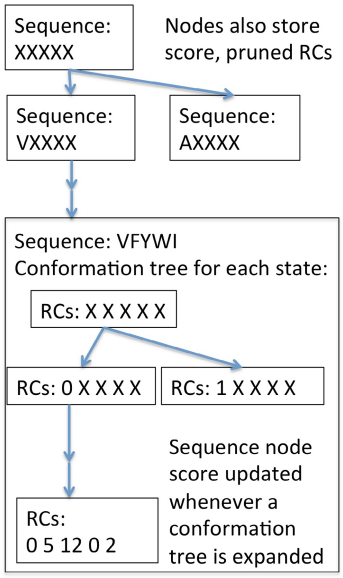

Fig. 2. Expansion steps during node processing generate nodes with partially and then fully defined sequences. Once a node has a fully defined sequence, conformational trees are built for it for all states. Then conformational tree expansions lead to fully processed nodes. X, unassigned amino acid or RC; V, Val; A, Ala; F, Phe; Y, Tyr; W, Trp; I, Ile.

bound for any of the constraint LMEs c_i is greater than 0, then we know all sequences at the node violate that constraint, and we eliminate the node. Otherwise, the node score is set to be the lower bound on the objective function. Previous A*-based protein design algorithms include methods to compute a lower bound on the energy of a single protein state over a sequence space [15,19,21]. These methods can be modified to provide a lower bound on an LME over a sequence space, with complexity as follows:

Theorem 1. *For any sequence space S defined by specifying the allowed set of amino acid types $S(i)$ at each mutable residue i, the lower bound on the LME Eq. (1) can be computed in time $O(n^2r^2s)$, where n is the number of flexible or mutable residues in the system, s is the number of states, and r is the maximum number of RCs available at a given residue.*

Details of the method for computing lower bounds, including a proof of Theorem 1, are provided in **SI** B [20].

For nodes without fully-defined sequences, we update the list of pruned RCs for the child node before computing bounds. Pruning is performed by type-dependent DEE [39]—in our implementation, the various pruning algorithms available in OSPREY [11,14,15] are used.

2.3 Starting and Finishing the Calculation

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

Each operation will define either the sequence or the conformation in one of the states at a residue where it was previously not defined, so in a finite number of steps, we will achieve a fully processed node: that is, a node whose sequence is fully defined, and whose conformation trees are sufficiently expanded to be fully processed (see **SI** Fig. S1 for a toy example). If our lower-bounding techniques are adequate, very few sequences will need to be fully processed in this way, so this sequence A* tree will return the optimal sequence with great efficiency compared to exhaustive search over sequences.

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

3 Results

Protein design calculations were performed in order to measure the efficiency of COMETS and its ability to design proteins with properties undesignable by single-state methods. Systems of four types were used: designs for specificity on a protein that can form two or more different complexes; optimization of the binding energy for a single complex; stabilization of a single protein robust to

choice of force field; and stabilization of the reduced form of angiotensinogen relative to the oxidized form or vice versa. Details of these test cases are in **SI** C [20].

3.1 Measurement of Efficiency

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

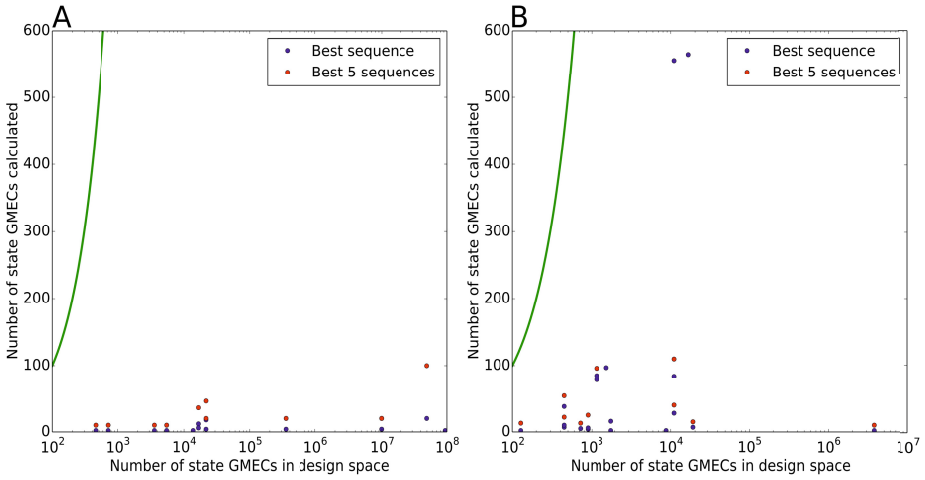

Fig. 4. Number of state GMECs calculated in COMETS runs with (A) rigid or (B) continuous flexibility (g), compared to the number sN of state GMECs in the entire design space (sN is the number of sequences in the design space times the number of states). Results are shown both for calculation of the best sequence and for enumeration of the best five, when possible under the design constraints. Exhaustive search would have to calculate all state GMECs (green curve).

Exhaustive search, the only other provable algorithm for multistate design, must calculate the GMEC for each sequence in each state. For a s-state design space with N sequences, this means that N sequences must be considered explicitly and sN state GMECs must be calculated—a formidable proposition, since N grows exponentially with the number of mutable residues and each state GMEC calculation is NP-hard [31]. To measure the ability of COMETS to avoid these calculations, the number g of state GMECs calculated by each run of COMETS was measured and compared to sN. Also, COMETS provably need not even consider each sequence explicitly, even briefly. To determine if this reduced consideration of sequences provides a significant advantage in efficiency, the number m of

sequence tree nodes created in each COMETS run was measured and compared to N. Hence, m is the number of partial sequences explicitly considered in a COMETS run.

Many provable algorithms, including A* [25] and integer linear programming [23], and non-provable methods like Monte Carlo [24] can perform the GMEC calculation using an exhaustive search over sequences without also exhaustively searching over conformations. So even without COMETS there is no need for an exhaustive search over *conformational* space. However, all previous provable methods must still compute the GMEC *for every sequence* when performing *multistate design*, because they are intended to calculate the minimum of an energy function (with respect to sequence and conformation). In contrast, COMETS calculates the *constrained minimum* (over all sequences) of a linear combination of minima (over all conformations) of energy functions. Hence, in this paper, we measure the ability of COMETS to *avoid* computing GMECs for most of the sequences, and sometimes even to avoid any explicit consideration of most of the remaining sequences. These are the main novel abilities of COMETS.

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

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

Provably Finding Unsatisfiable Constraints. The statistics above exclude runs for which no sequences can satisfy the constraints. COMETS can provably verify the absence of satisfying sequences, usually more quickly than it finds the best sequence for similar design spaces (likely because pruning can take place at early tree levels). It did so for 8 of the 27 continuous runs and 5 of the 25 rigid runs. Also, several runs with a constraint-satisfying sequence (9 of 19 continuous; 3 of 20 rigid) had less than five constraint-satisfying sequences. This indicates

that satisfaction of biophysically relevant energy constraints can depend on small alterations to the sequence, highlighting the importance of a provable design algorithm that will return the optimal sequence for each problem.

3.2 Differences in Sequences Returned by Multistate Designs and Single-state Proxies

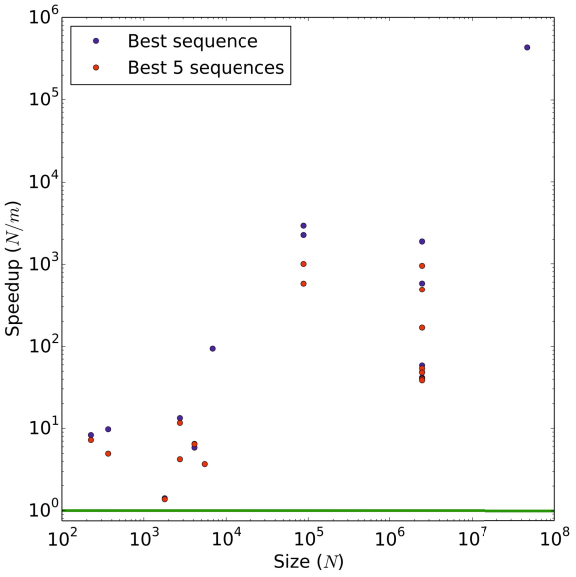

Fig. 5. Speedup due to reduced explicit consideration of sequences in COMETS, compared to exhaustive search (green line), for designs with rigid rotamers. m: number of sequence tree nodes created in COMETS. N: number of sequences in the design space. Magnifying this speedup, COMETS handles sequences that it considers explicitly very efficiently (Fig. 4).

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

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

Further details on the test cases are provided in **SI** C [20].

These results show that explicit, provable multistate design provides significant advantages in the calculation of optimal sequences for a wide range of problems, and that COMETS provides an efficient way to perform such designs.

The number of sequences and of state GMECs considered could likely be reduced substantially further by improved bounding heuristics. Thus, COMETS liberates provable multistate protein design from the efficiency barrier imposed by exhaustive search.

4 Conclusions

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

Acknowledgments. We would like to thank Dr. Ivelin Georgiev for helpful discussions and for providing useful multistate protein design problems; Dr. Kyle Roberts for helpful discussions and advice on the algorithms; Dr. Kyle Roberts and Pablo Gainza for providing PDB files and scripts for testing; all members of the Donald lab for helpful comments; and the PhRMA foundation (MAH) and NIH (grant 2R01-GM-78031-05 to BRD) for funding.

References

1. Arnold, F.H.: Design by directed evolution. Accounts of Chemical Research **31**(3), 125–131 (1998)
2. Chen, C.-Y., et al.: Computational structure-based redesign of enzyme activity. PNAS **106**(10), 3764–3769 (2009)
3. Davey, J.A., et al.: Multistate approaches in computational protein design. Protein Science **21**(9), 1241–1252 (2012)
4. Desmet, J., et al.: The dead-end elimination theorem and its use in protein side-chain positioning. Nature **356**, 539–542 (1992)
5. Donald, B.R.: Algorithms in Structural Molecular Biology. MIT Press (2011)
6. Frey, K.M., et al.: Predicting resistance mutations using protein design algorithms. PNAS **107**(31), 13707–13712 (2010)
7. Fromer, M.: A Probabilistic Approach to the Design of Structural Selectivity of Proteins. PhD thesis, Hebrew University of Jerusalem (2010)
8. Fromer, M., et al.: SPRINT: Side-chain prediction inference toolbox for multistate protein design. Bioinformatics **26**(19), 2466–2467 (2010)
9. Fromer, M., et al.: Design of multispecific protein sequences using probabilistic graphical modeling. Proteins: Structure, Function, and Bioinformatics **78**(3), 530–547 (2010)
10. Gainza, P., et al.: Protein design using continuous rotamers. PLoS Computational Biology **8**(1), e1002335 (2012)
11. Gainza, P., et al.: OSPREY: Protein design with ensembles, flexibility, and provable algorithms. Methods in Enzymology **523**, 87–107 (2013)

12. Georgiev, I., et al.: Design of epitope-specific probes for sera analysis and antibody isolation. Retrovirology **9**(Suppl. 2), P50 (2012)
13. Georgiev, I., et al.: Dead-end elimination with backbone flexibility. Bioinformatics **23**(13), i185–i194 (2007)
14. Georgiev, I., et al.: Improved pruning algorithms and divide-and-conquer strategies for dead-end elimination, with application to protein design. Bioinformatics **22**(14), e174–e183 (2006)
15. Georgiev, I., et al.: The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. Journal of Computational Chemistry **29**(10), 1527–1542 (2008)
16. Georgiev, I., et al.: OSPREY (Open Source Protein Redesign for You) user manual (2009). http://www.cs.duke.edu/donaldlab/software.php; Updated, 2015. 96 pages
17. Georgiev, I.S., et al.: Antibodies VRC01 and 10E8 neutralize HIV-1 with high breadth and potency even with Ig-framework regions substantially reverted to germline. The Journal of Immunology **192**(3), 1100–1106 (2014)
18. Gorczynski, M.J., et al.: Allosteric inhibition of the protein-protein interaction between the leukemia-associated proteins Runx1 and CBFβ. Chemistry and Biology **14**, 1186–1197 (2007)
19. Hallen, M.A., et al.: Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility. Proteins: Structure, Function and Bioinformatics 81(1), 18–39 (2013)
20. Supplementary material. http://www.cs.duke.edu/donaldlab/Supplementary/recomb15/comets/
21. Hart, P.E., et al.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
22. Karanicolas, J., et al.: Computational design of affinity and specificity at protein-protein interfaces. Current Opinion in Structural Biology **19**(4), 458–463 (2009)
23. Kingsford, C.L., et al.: Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics **21**(7), 1028–1039 (2005)
24. Kuhlman, B., et al.: Native protein sequences are close to optimal for their structures. PNAS **97**(19), 10383–10388 (2000)
25. Leach, A.R., et al.: Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. Proteins: Structure, Function, and Bioinformatics **33**(2), 227–239 (1998)
26. Leaver-Fay, A., et al.: A generic program for multistate protein design. PLoS One **6**(7), e20937 (2011)
27. Lee, C., et al.: Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. Nature **352**, 448–451 (1991)
28. Leech, J., et al.: SMD: Visual steering of molecular dynamics for protein design. Computational Science and Engineering **3**(4), 38–45 (1996)
29. Lewis, S.M., et al.: Generation of bispecific IgG antibodies by structure-based design of an orthogonal Fab interface. Nature Biotechnology **32**, 191–198 (2014)
30. Lilien, R.H., et al.: A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase A phenylalanine adenylation enzyme. Journal of Computational Biology **12**(6), 740–761 (2005)
31. Pierce, N.A., et al.: Protein design is *NP*-hard. Protein Engineering **15**(10), 779–782 (2002)

32. Qi, S., et al.: Crystal structure of the *Caenorhabditis elegans* apoptosome reveals an octameric assembly of CED-4. Cell **141**(3), 446–457 (2010)
33. Roberts, K.E.: Novel Computational Protein Design Algorithms with Applications to Cystic Fibrosis and HIV. PhD thesis, Duke University (2014)
34. Roberts, K.E., et al.: Computational design of a PDZ domain peptide inhibitor that rescues CFTR activity. PLoS Computational Biology **8**(4), e1002477 (2012)
35. Rudicell, R.S., et al.: Enhanced potency of a broadly neutralizing HIV-1 antibody *in vitro* improves protection against lentiviral infection *in vivo*. Journal of Virology (2014); Published online, 2014
36. Sitkoff, D., et al.: Accurate calculation of hydration free energies using macroscopic solvent models. Journal of Physical Chemistry **98**, 1978–1988 (1994)
37. Stevens, B.W., et al.: Redesigning the PheA domain of gramicidin synthetase leads to a new understanding of the enzyme's mechanism and selectivity. Biochemistry **45**(51), 15495–15504 (2006)
38. Yan, N., et al.: Structure of the CED-4-CED-9 complex provides insights into programmed cell death in *Caenorhabditis elegans*. Nature **437**, 831–837 (2005)
39. Yanover, C., et al.: Dead-end elimination for multistate protein design. Journal of Computational Chemistry **28**(13), 2122–2129 (2007)
40. Zheng, F., et al.: Most efficient cocaine hydrolase designed by virtual screening of transition states. Journal of the American Chemical Society **130**, 12148–12155 (2008)