Chapter 3

An Efficient Algorithm for Multi-State Protein Design Based on FASTER

The text of this chapter was adapted from a manuscript coauthored with Stephen L. Mayo.

Abstract

Most of the methods that have been developed for computational protein design involve the selection of side-chain conformations in the context of a single, fixed main-chain structure. In contrast, multi-state design (MSD) methods allow sequence selection to be driven by the energetic contributions of multiple structural or chemical states simultaneously. This methodology is expected to be useful when the design target is an ensemble of related states rather than a single structure, or when a protein sequence must assume several distinct conformations to function. MSD can also be used with explicit negative design to suggest sequences with altered structural, binding, or catalytic specificity. We report implementation details of an efficient multi-state design optimization algorithm based on FASTER (MSD-FASTER). We subjected the algorithm to a battery of computational tests and found it to be generally applicable to various multi-state design problems; designs with a large number of states and many designed positions are completely feasible. A direct comparison of MSD-FASTER and multi-state-design Monte Carlo indicated that MSD-FASTER discovers low-energy sequences much more consistently. MSD-FASTER likely performs better because amino acid substitutions are chosen on an energetic basis rather than randomly, and because multiple substitutions are applied together. Through its greater efficiency, MSD-FASTER should allow protein designers to test experimentally better-scoring sequences, and thus accelerate progress in the development of improved scoring functions and models for computational protein design.

Introduction

The field of computational protein design provides software tools that facilitate the identification of amino acid sequences with specific desired properties. Most protein design protocols choose amino acid types and side-chain conformations in the context of a single, fixed, main-chain conformation. Given this simplifying approximation, one can precompute all pairwise interaction energies between possible side-chain conformations at different positions and then optimize this system of interactions to find sequences expected to stabilize the fold.[1, 2] The most common optimization algorithms employed for this purpose are based on Monte Carlo with simulated annealing (MC),[3–5] the dead-end elimination theorem (DEE),[5–7] genetic algorithms,[5, 8] and Fast and Accurate Side-Chain Topology and Energy Refinement (FASTER).[9, 10] These single-state design methods have produced several notable successes, when used on their own or in conjunction with main-chain optimization techniques.[1, 3, 11–14] However, single-state design is not necessarily sufficient when design objectives require the explicit consideration of multiple states at once.[15]

For example, we might desire a sequence that is able to assume two distinct folds under different conditions; the single-state design methodology described above does not provide a mechanism for selecting sequences that are simultaneously compatible with both folds. Similarly, single- state design methods do not provide a way to explicitly alter binding specificity, since only one binding partner may be modeled during sequence selection. Likewise, enzyme design methods might be enhanced through the explicit modeling of the substrate, transition state, and product, rather than only one of these at a time. Finally, we note that NMR-derived solution structures have been neglected as

targets for protein design because typical structure determination methods give an ensemble rather than a single set of coordinates.[16] To the extent that the structural diversity of an NMR ensemble reflects realistic conformational flexibility, it will be interesting to investigate the effects of using such an ensemble as the basis for design.

Each of the design goals given above requires sequence selection to be informed by multiple structural or chemical states simultaneously, in what we call multi-state design (MSD). The optimization strategy we apply to MSD problems comprises an outer routine that suggests possible amino acid sequences, and an inner routine that assesses the fitness of a sequence by performing rotamer optimization on each state and combining the individual state energies to yield an overall score. This basic approach has been used by others to design specificity into a self-associating coiled-coil system, to generate a molecular switch, and to recover sequences that bind their cognate ligands with high affinity.[15, 17, 18] Here, we describe a generalization of these strategies that is applicable to any number of states and compatible with any type of scoring function that might be used to combine the energies of sequences threaded on the target states.

For a design problem with *n* states to consider, we use *n* processors of a computer cluster to calculate one optimization trajectory. Each processor holds in memory the pairwise energy matrix for one state, and is responsible for evaluating the energies of candidate sequences in the context of that state only. In general, a candidate sequence is evaluated by performing rotamer optimization using a side-chain placement algorithm based on MC, DEE, or FASTER. One of the processors (the boss) is additionally responsible for identifying amino acid sequences to be scored, communicating this information to the others, collecting the results, and combining the energies to yield an

overall fitness score. Here, we provide implementation details for MSD optimization algorithms with amino acid selection schemes based on MC and FASTER, and give quantitative comparisons of their performance for a variety of multi-state design problems.

Results and discussion

Scoring functions

To solve the multi-state design problem, we employ an extension of the methodology that has been developed for single-state design. In single-state design, the cost function to be optimized is the energy E of the rotamer configuration R. The energy is computed by summing the rotamer/template energies E_i for each of the N residue positions and the interaction energies E_{ij} between all pairs of rotamers at residue positions i and j. Typically, the rotamer configuration is optimized without regard to the amino acid types of the rotamers available at each position.

$$E(R) = \sum_{i=1}^{N} E_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} E_{ij} \qquad (1)$$

In multi-state design, the score σ to be optimized is a function of the amino acid sequence A. In general, an amino acid sequence will not assume the same side-chain conformations in the various states being modeled. If there are n states, then the score is computed using a function of the following form:

$$\sigma(A) = \sigma(E_1(A), E_2(A), ..., E_n(A)) \qquad (2)$$

Each $E_s(A)$ corresponds to the energy of the sequence A threaded on state s, and is computed by single-state rotamer optimization using equation 1. Different energy combination functions σ may be appropriate for different types of design problems. For

example, in a case where the designed sequence is meant to satisfy n distinct states equally well, the simplest scoring function simply computes the average energy across all states:

$$\sigma(A) = \frac{1}{n}\sum_{s=1}^{n} E_s(A) \qquad (3)$$

When the design target is an ensemble of similar states, such as an NMR solution structure, the requirement that a sequence satisfy all states may be too stringent; it cannot be assumed that every member of the ensemble would be significantly populated or relevant for the designed sequence. In this case, a scoring function that applies Boltzmann-weighted averaging may be more useful:

$$\sigma(A) = -kT\log\left(\sum_{s=1}^{n} e^{-Es(A)/kT}\right) \qquad (4)$$

Use of equation 4 prevents sequences that fail to satisfy a few states from being severely penalized. If the design goal is to alter conformational, binding, or catalytic specificity, a scoring function for explicit negative design is warranted. Given one positive design state ρ and one negative design state η, one might apply the following scoring function:

$$\sigma(A) = \Delta E_\rho(A) - W\Delta E_\eta(A) \qquad (5)$$

Here, W is a weighting factor used to control the balance of ρ-state stabilization and η-state destabilization. Each $\Delta E_s(A)$ in equation 5 is the excess energy of sequence A when threaded on state s compared to the optimal sequence A_0 for that state as determined by single-state design:

$$\Delta E_s(A) = E_s(A) - E_s(A_0) \qquad (6)$$

Because $E_s(A_0)$ is the minimum energy of any sequence threaded on state s, $\Delta E_s(A) \geq 0$. The $\Delta E_s(A)$ terms are intended to normalize the energies of the sequences being selected

and to allow a single value of W to be used with various energy functions and design targets.

Over the course of a negative design calculation, sequences may be found that cannot be threaded on the negative design target structure without causing severe van der Waals clashes; use of equation 5 in a multi-state design calculation will cause such sequences to be preferred. Any predicted clash must surely be alleviated by a shift in the distribution of conformational states assumed by a real protein. However, we hypothesize that variants with native states perturbed in this manner will tend to be destabilized, especially when multiple clashes are predicted together. Because the energies assigned to these clashes by a standard Leonard-Jones potential depend strongly on several approximations (such as discrete side-chain rotamers and a fixed main chain), we threshold all rotamer-template and rotamer-rotamer energies on the negative design target state to a positive constant. This effectively causes sequences with a greater number of clashes to be preferred over sequences with a smaller number of larger-magnitude clashes, as desired.[19]

A more rigorous approach to explicit negative design would be to maximize the probability with which the target state is assumed over all explicitly modeled competing states, as computed according to basic statistical mechanics. This approach has been applied to the design of specificity in self-associating and ligand-binding systems.[15, 18] The success of this method relies on the availability of atomic models that accurately represent all target and competing states; unfortunately, general methods for the construction of these models have not yet been developed and validated. For the computational tests reported here, we have sidestepped issues of model construction by

applying equation 5 to a system with one crystal structure as the positive design target, and another as the competing state for negative design.

Multi-state Monte Carlo

Monte Carlo with simulated annealing (MC) is an efficient stochastic optimization technique that is heavily used in computational protein design.[3–5] When used for rotamer optimization, MC can produce high-quality approximate solutions quickly and find low-energy variants in the vicinity of an existing solution.[5] MC is easily applied as the outer routine in multi-state design by making perturbations at the level of amino acid sequence only. In each step of multi-state design MC (MSD-MC, Figure 1), a residue position is picked at random, and a random amino acid substitution is made at that position. The new sequence is scored on each state by rotamer optimization. The decision to accept or reject the perturbation is made based on the change in the score σ and the simulated annealing temperature, which is cycled up and down over the course of the optimization to allow traversal of local maxima and exploration around local minima.

We have applied two enhancements to MSD-MC in an attempt to improve its performance. In the first, random perturbations are chosen uniformly from a list of all allowed amino acid substitutions, without respect the positions at which they occur. This prevents positions that have fewer allowed amino acids than others from being the focus of a disproportionate number of substitution attempts. In the second enhancement, rotamer optimization after a substitution is limited to those positions within a specified C_α- C_α distance cutoff from the perturbed position, reducing the amount of time required

for rotamer optimization and allowing more steps of MSD-MC to be completed per unit time.

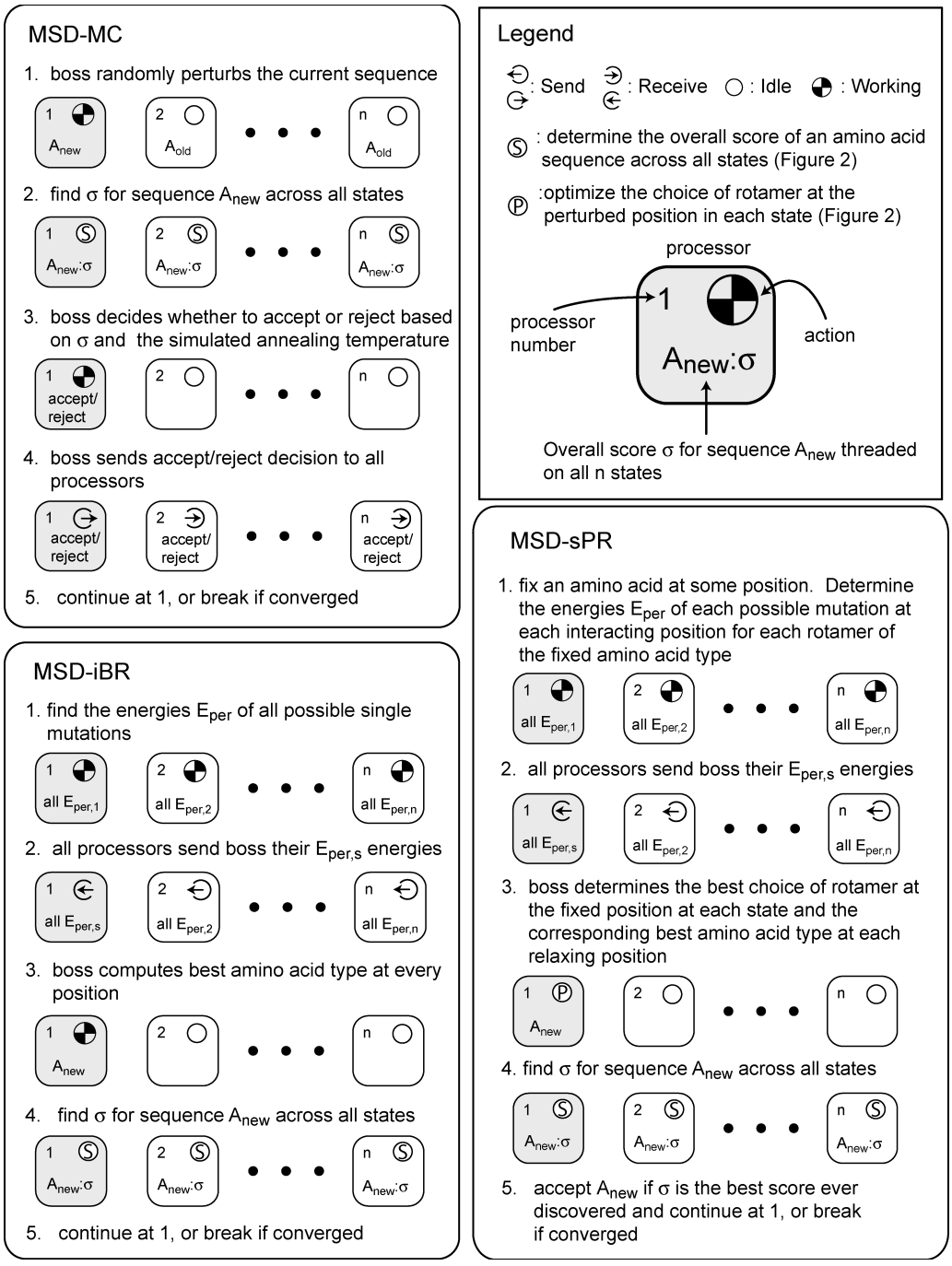

Figure 1: Graphical depictions of the three MSD sequence selection routines described in the text. Legend (upper right panel): explains the symbols used to depict a parallel algorithm. Each box represents a single processor that performs energy calculations on a single state. Fields within the box identify the processor by number, show the current action, and explain the relevant data that the processor holds in memory. The boss processor is shaded in grey. The subroutines **S** and **P** are depicted in Figure 2 and described in the text. Depicted here are: one step of MSD-MC (upper-left panel), one round of MSD-iBR (lower-left panel), and one perturbation in MSD-sPR (lower-right panel).

Multi-state FASTER

Like Monte Carlo, FASTER is a stochastic optimization algorithm that makes perturbations to existing solutions and accepts or rejects them based on their energetic consequences.[10] The two algorithms differ chiefly in the methods by which perturbations are chosen. FASTER has two main components that we have modified for MSD: iterative batch relaxation (iBR), and single perturbation and relaxation (sPR). In each component, amino acid substitutions at several positions are chosen independently and applied together to yield a new solution. Each component is applied iteratively until convergence is detected. In MSD-iBR, convergence is signaled when the user-defined limit for the number of nonproductive rounds (i.e., rounds that fail to improve the energy) is reached. In MSD-sPR, convergence can occur either when the user-defined limit for total rounds is reached or when an entire round has elapsed without an improved solution being found. One trajectory of MSD-FASTER is performed by generating a random initial sequence, applying MSD-iBR until convergence, and then applying MSD-sPR until convergence.

Multi-state iBR

During a round of single-state iBR, the best rotamer at each position of the protein is determined independently in the context of the current rotameric configuration at all other positions. Then, the new rotamers at each position are all updated simultaneously, and the resulting updated configuration of the system is retained regardless of the change in energy. iBR is applied iteratively until a user-defined limit for nonproductive rounds

has been reached. After the detection of convergence, the lowest-energy configuration ever found during the rounds of iBR is selected to move on to sPR.

During a round of MSD-iBR, the best amino acid at each position must be chosen considering all states simultaneously (Figure 1). For each possible amino acid substitution at each position, each processor determines for its own state the best possible total energy of the system when that substitution is made with the current rotamer configuration fixed at all other positions, and sends this information to the boss. If there are p positions and a amino acid types allowed at each position, then each processor needs to communicate pa floating-point values. For each position, the boss computes the overall score of each possible substitution across all states using these values and a scoring function σ. The amino acid identity at each position is then updated with the best-scoring substitution found by the boss in the previous step. Each processor rescores the resulting sequence for its state by rotamer optimization and these energies are again combined to produce an overall score. This process is repeated until convergence, as in single-state iBR.

Multi-state sPR

In a step of single-state sPR, one position is forced to assume a particular rotamer (is "perturbed"), the other positions are allowed to relax independently in the context of the current rotamer configuration, and the rotamers at all relaxing positions are updated at once. The resulting relaxed rotamer configuration is accepted only if its energy is better than any previously observed. In a step of single-state sPR, amino acid substitutions can occur at the perturbed position and also at the relaxing positions, since rotamers are

sampled without regard to their amino acid types. In one round of single-state sPR, each rotamer at each position will be fixed exactly once; positions to fix are picked in random order. Rounds of sPR are performed until an entire round fails to produce a better solution, or until a user-defined limit is reached.

Several significant complications arise when adapting sPR for multi-state design. We would like to fix a particular amino acid at some position and choose the resulting best amino acid substitution at each independently relaxing position (Figure 1). Typically, there will be multiple available rotamers of the fixed amino acid type at the perturbed position in each state. Each of these rotamers will lead to a distinct set of energies for the possible amino acid substitutions at the relaxing positions. Thus, an explicit choice of fixed rotamer at the perturbed position must be made for each state in order to determine the best-scoring amino acid types at the relaxing positions when all states are considered simultaneously. Unfortunately, each processor cannot simply determine the best fixed rotamer in its own state and send the corresponding substitution energies to the boss to be scored. To improve the overall score across all states, a given state may be forced to accept a substitution that is suboptimal when that state is considered by itself. To score that suboptimal substitution correctly, the state may be forced to employ a rotamer at the perturbed position that is different from the one that leads to the best substitutions for that state in isolation. Thus, each processor must communicate substitution energies corresponding to all of the available rotamers of the fixed amino acid type at the perturbed position, and not just of the ones that seem optimal in the context of its own state.

For each rotamer of the fixed amino acid type at the perturbed position, each processor must send the total energy of each possible amino acid substitution at each of the relaxing positions. If there are *r* rotamers of the fixed amino acid type at the perturbed position, *p* relaxing positions, and *a* amino acid types available at each of the relaxing positions, then each processor must send *rpa* floating-point values to the boss.

A given assignment of fixed rotamers to states allows a preferred amino acid substitution at each relaxing position and its MSD score to be computed using a σ function, as described in the MSD-iBR section above. Thus, if there are *n* relaxing positions allowed, there will be *n* separate MSD score values σ_r. In order to determine the best relaxed sequence given an amino acid perturbation, we optimize the sum of these σ_r (subroutine **P** in Figure 2). The optimization comprises a quick Monte Carlo run of 10,000 steps along a linear temperature gradient from 4000 K to 1 K with a nonproductive steps limit of 100. In each step of MC, a random state is chosen, a random fixed rotamer for that state is selected, and the corresponding sum of MSD substitution scores at the relaxing positions is determined; the new fixed rotamer configuration is accepted or rejected based on the Boltzmann criterion. This protocol generates a favorable choice of fixed rotamer for each state and incurs negligible computational expense. After the amino acids at the relaxing positions are chosen, each processor evaluates the energy of the new sequence threaded on its state by rotamer optimization. The energies are then combined into an overall score using a σ function as described above.

Although the technique just described is expected to perform well for most MSD problems, there is some reason to believe that it may be inadequate when used in the

context of explicit negative design. Because subroutine **P** attempts to choose rotamers of the fixed amino acid type that result in designed sequences that minimize σ, it preferentially selects sequences that clash with the chosen fixed rotamers in competing states, even though these clashes might be relaxed away during the subsequent rotamer optimization step. This single-minded focus on sequences that clash most strongly prior to rotamer optimization could inhibit the ability of the algorithm to find those sequences with the most favorable scores after rotamer optimization. To address these concerns, we have implemented and tested two modifications that allow the fixed rotamer configuration (and resulting relaxed amino acid sequence) to be chosen completely randomly, or randomly from one of the top *r* configurations found during subroutine **P**. Comparison with these simple modifications should allow the overall utility of the original procedure to be assessed.

We recently reported that the efficiency of single-state FASTER can be improved by allowing only the positions that interact most strongly with the perturbed position to be relaxed.[9] When applied to MSD-sPR, this improvement also limits the amount of data that must be communicated between processors and improves the efficiency with which the optimal fixed rotamers for each state can be determined. In MSD-sPR, the potential relaxing positions are ranked according to the absolute values of the σ_r scores calculated from their interactions with the perturbed position. The initial rotamer configurations in each state prior to the perturbation are used to assess these interactions.

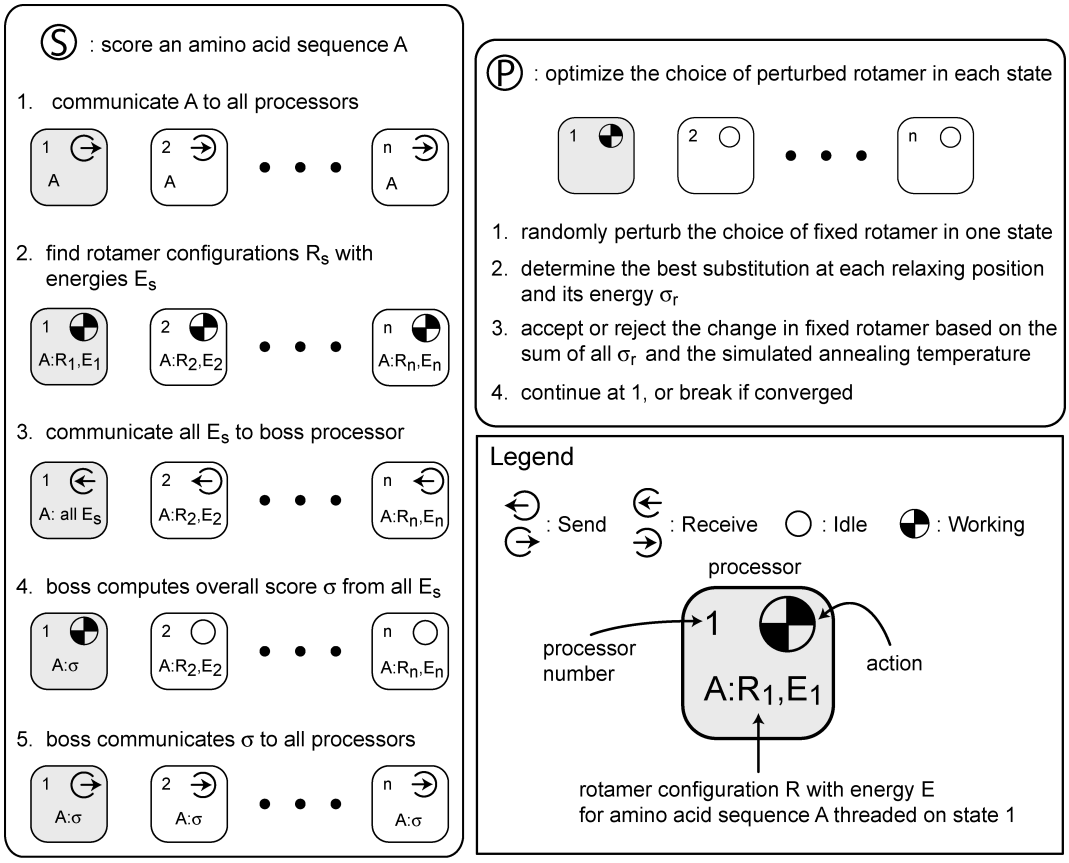

Figure 2: Subroutines used by the MSD sequence selection algorithms. **S**: the subroutine used to assign an overall score to a given amino acid sequence based on input from all of the states. **P:** the subroutine used to determine an optimized choice of fixed rotamer at the perturbed position in each state during MSD-sPR. The boss processor runs this routine using data accumulated from all processors.

Rotamer optimization (RO) algorithms

The MSD sequence selection algorithms described above require that the energy of specific sequences be evaluated in the context of each target state (subroutine **S** in Figure 2). Any of the rotamer optimization (RO) algorithms that have been developed for single-state protein design and side-chain placement, such as MC, DEE, and FASTER, can be used to evaluate these energies. When used for rotamer optimization in this work, one cycle of MC comprised a simulated annealing schedule that varied linearly from high temperature to low. When FASTER was used, rounds of iBR and then of sPR were applied in series; each pass was terminated when convergence was detected or the user-defined rounds limit was reached. In a step of sPR, the set of positions allowed to relax in response to the perturbation was limited to the ten that interact most strongly with the perturbed position.[9] DEE-based rotamer optimizations were performed as previously described,[7] except that the split-DEE and bounding steps were omitted. For some amino acids sequences, DEE failed to converge to a single solution; in these cases, FASTER was automatically invoked to find an approximate solution instead.

When performing rotamer optimization using MC or FASTER, an initial rotamer configuration is required. During multi-state design, RO is applied in subsequent rounds to amino acid sequences that differ at only a few positions; our implementation of MSD exploits this situation to provide better initial rotamer configurations for optimization. In MSD-MC, the amino acid identity at exactly one position will have changed since the most recent rotamer optimization. The rotamer at this position is initialized randomly, while the initial rotamer configuration at each of the unchanged positions is taken directly

from the previous solution. In MSD-sPR, rotamer optimization occurs after each processor has determined the best energies for each amino acid type at several relaxing positions, given a fixed amino acid at a perturbed position. The rotamers at positions that are neither fixed nor relaxed are taken from the previous solution. The rotamer at the fixed position in each state is chosen as described in the section on MSD-sPR above. Reasonable rotamers for each amino acid type at the relaxing positions are also already known; the energies of these rotamers were used to select the sequence being scored. The rotamer solution taken from these three sources can be used to determine directly the energy of the sequence, or additional RO may be performed using it as an initial solution. We refer to the routine that directly determines the energies on each state without further optimization as the Null rotamer optimizer. However, our results below indicate that the Null routine is insufficient for effective MSD sequence optimization.

In each MSD calculation, we employ two different RO modules that we refer to as "weak" and "strong". During rounds of MSD-MC and MSD-sPR, an initial rotamer configuration for each state is available for input to the rotamer optimization routines as described above. Thus, we start from these initial solutions and perform a limited number of rounds of rotamer optimization to save time (weak RO). On the other hand, good initial solutions are not available at the beginning of a round of any MSD algorithm, or at any time during MSD-iBR due to the large number of substitutions that can be made during each round. In these cases, we start from random rotamer configurations and apply more rounds of rotamer optimization to increase our confidence in the resulting energies (strong RO). When DEE is used, it is employed with the same parameters for

both the strong and weak rotamer optimization, because initial solutions cannot be exploited in our implementation of DEE.

Test cases for multi-state design

We tested the performance of the algorithms described here with several different multi-state design problems. The MSD-MC and MSD-FASTER amino acid selection schemes are stochastic and provide no guarantee that the global minimum energy solution will ever be found. We therefore perform many optimization trajectories with different random number seeds, and assess the algorithms based on the distribution of solutions given by these trajectories. When a significant fraction of the trajectories report the same best solution ever found, we take that solution to be optimal. Given the fraction of trajectories f that find the optimal solution, and the average processor-time in minutes t required to compute a trajectory, we compare algorithms using according to the value $S = t / f$. This score represents the total number of processor-minutes required on average to find the optimal solution; smaller values are better. We previously used this metric to analyze the performance of single-state design optimization algorithms.[9]

Single-state design problems

When a MSD algorithm is applied to a design problem with only one target state, its accuracy and efficiency may be compared to well-characterized single-state design algorithms, such as single-state design FASTER (SSD-FASTER). We optimized four full sequence designs that were previously used as test cases for the single-state versions of Monte Carlo and FASTER: 1AAY, 1PIN, 1PGA, and 1C9O. These designs have from

28 to 66 designed positions, and the average number of rotamers per position is 212; a more complete description of these designs is available elsewhere.[9] Because each of these designs had only one target state, the MSD scoring function was simply $\sigma(A) = E(A)$, which is consistent with equations 3 and 4 when $n = 1$.

For each design, we computed 1000 trajectories of MSD-FASTER and MSD-MC with a variety of different weak RO algorithms: Null, MC, iBR, FASTER, and DEE. We refer to a particular pairing of MSD and RO algorithms in a/b format: MSD-FASTER used with FASTER for weak rotamer optimization is called MSD-FASTER/FASTER. For the parameters used in each optimization algorithm formulation, see the materials and methods.

SSD test cases: MSD-FASTER

The results of the MSD-FASTER calculations (Table 1) indicate that the MSD algorithm easily finds the optimal solution (as determined by SSD-FASTER) for each design when paired with weak RO routines based on FASTER, iBR, or MC. For the two smaller designs, 1AAY and 1PIN, MSD-FASTER was actually able to find the lowest-energy solution 20–80% *more* efficiently than SSD-FASTER, because a greater fraction of its trajectories were able to find the optimal solution without requiring significantly more compute time. When applied to the larger and more difficult designs, 1PGA and 1C9O, the performance of MSD-FASTER deteriorated to between 8–18% of the efficiency of SSD-FASTER. This deterioration stemmed both from an increase in the time required to perform simulation trajectories, and a decrease in the fraction of trajectories that were able to find the optimal solution. Ultimately, we were pleased to

discover that, despite the limitations imposed on the algorithm by the requirements of multi-state design, MSD-FASTER can effectively find optimal amino acid sequences among sets of at least 10^{56} alternatives (1C9O). Although MSD-FASTER does not seem to scale to larger problem sizes as well as SSD-FASTER, its performance should allow for the rigorous investigation of new ideas in multi-state computational protein design.

When the results for all four designs are considered simultaneously, the most favorable comparison with SSD-FASTER is offered by MSD-FASTER/FASTER, which allows significant relaxation after each round of MSD-iBR and each step of MSD-sPR. MSD-FASTER also yielded satisfactory performance when MC was used as the weak RO routine, although the number of correct trajectories found per unit time was always fewer than when FASTER was used. As a quicker but less accurate alternative, iBR allowed fewer correct trajectories to be found, but reduced significantly the time required to compute each trajectory, leading to similar overall performance when compared to FASTER and MC. For the 1AAY and 1PIN designs, the most correct trajectories were found when using DEE for rotamer optimization. However, this greater accuracy came at the cost of significantly more processor time required. Furthermore, MSD-FASTER was unable to complete trajectories for the 1PGA design in a reasonable time when RO was performed by DEE (> 100 minutes each), and so the run was aborted. Although DEE-based rotamer optimization may be too slow for sequence selection in nontrivial design problems, it can still be useful to rescore a list of sequences produced using a quicker but more approximate RO method. When no weak RO was performed at all (MSD-FASTER/Null), the optimal solution was found for the 1AAY and 1PIN designs, but not the two larger ones. We note that the average time per trajectory for these designs was

only slightly lower than when iBR was used, indicating that most of the time in MSD-FASTER/iBR is spent choosing sequences to score rather than scoring them by rotamer optimization. Rotamer optimization of some kind seems to be required for the efficient convergence of nontrivial multi-state design problems using MSD-FASTER.

Table 1: **Performance of MSD-FASTER when applied to four difficult single-state design problems**

Design	Size[a]	Opt[b]	% correct[c] ($f \times 100$)	t[d]	S[e]	p[f]
1AAY	28	SSD-FASTER	1.0	0.5	46	1.00
		MSD-FASTER/Null	0.2	0.3	153	0.30
		MSD-FASTER/MC	1.8	0.6	35	1.31
		MSD-FASTER/iBR	1.3	0.4	32	1.44
		MSD-FASTER/FASTER	2.5	0.6	25	1.84
		MSD-FASTER/DEE	3.8	3.6	94	0.49
1PIN	34	SSD-FASTER	2.1	0.6	28	1.00
		MSD-FASTER/Null	1.5	0.4	26	1.08
		MSD-FASTER/MC	3.0	0.7	23	1.22
		MSD-FASTER/iBR	2.5	0.5	20	1.40
		MSD-FASTER/FASTER	3.2	0.7	23	1.22
		MSD-FASTER/DEE	3.6	4.3	118	0.24
1PGA	56	SSD-FASTER	4.2	1.9	46	1.00
		MSD-FASTER/Null	0	3.1	—	—
		MSD-FASTER/MC	1.1	6.2	562	0.08
		MSD-FASTER/iBR	1.5	4.9	327	0.14
		MSD-FASTER/FASTER	3.3	8.5	258	0.18
		MSD-FASTER/DEE	—[g]	—[g]	—	—
1C9O	66	SSD-FASTER	2.0	1.4	71	1.00
		MSD-FASTER/Null	0.0	2.5	—	—
		MSD-FASTER/MC	0.9	5.7	629	0.11
		MSD-FASTER/iBR	0.7	4.3	610	0.12
		MSD-FASTER/FASTER	1.5	7.6	507	0.14
		MSD-FASTER/DEE	1.1	16.4	1486	0.05

a. The number of variable positions in the design
b. The optimization strategy that was used, as described in the text. The term after the slash indicates the weak rotamer optimization routine that was used.
c. The percentage of trajectories that found the best known solution ($f \times 100$), as determined by SSD-FASTER. 1000 total trajectories were computed in each MSD or SSD calculation.
d. The average time, in minutes, required to perform each trajectory on one processor
e. The score $S = t / f$, as described in the text. Smaller values are better, indicating that the optimal solution can be found more quickly. "—" indicates that S is undefined because $f = 0$.
f. The multiplicative factor p measures the deterioration in performance compared to SSD-FASTER. For example, $p = 0.17$ indicates that the MSD algorithm was 17% as efficient as the SSD algorithm.
g. When optimizing the 1PGA design using MSD-FASTER/DEE, the runs were aborted when it was determined that trajectories would take longer than 100 minutes each to complete.

SSD test cases: MSD-MC

To compare the performance of MSD-MC to MSD-FASTER, we repeated the single-state test designs using Null, iBR, MC, and FASTER for rotamer optimization. In the course of these test calculations, it was determined that MSD-MC performed the best when applied with uniform sampling of amino acid substitutions and with the positions to be optimized after a substitution limited to those within 15 Å C_α-C_α of the substituted position, as described above. For brevity, we report only the results of this best MSD-MC formulation here. To make the comparison between MSD-MC and MSD-FASTER as fair as possible, we adjusted the number of Monte Carlo steps in MSD-MC so that the average time per trajectory would be similar to when MSD-FASTER was used (see materials and methods); many more amino acid substitutions can be attempted per unit time if the total time for rotamer optimization per substitution is reduced.

Even using this best formulation, the ability of MSD-MC to find correct solutions to these SSD problems was dramatically worse than that of MSD-FASTER (Table 2). When paired with the Null rotamer optimizer or with iBR, MSD-MC was able to find the optimal solutions to the two smaller design problems, albeit with much lower frequency than MSD-FASTER despite longer sampling times. The relative success of MSD-MC with less rigorous rotamer optimization routines reflects the fact that MSD-MC is strongly limited by the number of amino acid substitutions it is able to test; implementations with less expensive rotamer optimization can afford to test more sequences per unit time, and therefore perform better.

The optimal solutions to the two larger design problems were never found using any implementation of MSD-MC. Because the *S* and *p* scores that were used to compare

the efficiencies of the MSD-FASTER algorithms are undefined when the fraction of correct trajectories is zero, we report two different metrics for MSD-MC. ΔE is the difference in simulation energy between the best sequence found by the MSD-MC algorithm and the optimal sequence found by SSD-FASTER; N_m is the number of positions that differ between the two sequences. Although the 1PGA and 1C9O calculations were not able to find the optimal solution, they can be evaluated based on how close they came (i.e., how close ΔE and N_m are to zero). In terms of ΔE and N_m, these two larger designs showed significant deviations, with differences in simulation energy of 2–4 kcal/mol and 4–7 mutations away from the best-scoring sequence found using SSD-FASTER and MSD-FASTER. Even these suboptimal sequences were found only a few times in the aggregate simulation run, rather than the numerous times the optimal sequence was found by the MSD-FASTER protocols. In addition to various combinations of uniform sampling and restricted sets of positions for rotamer optimization, we attempted various simulated annealing schedules and temperature ranges in MSD-MC, as well as applying fewer trajectories of longer length, all to no avail (data not shown). Compared to MSD-FASTER, the optimization ability of MSD-MC is clearly unacceptable for designs of this difficulty.

Table 2: The performance of MSD-MC when applied to four difficult single-state design problems

Design	Size[a]	Opt[b]	% correct[c] ($f \times 100$)	ΔE[d]	N_m[e]	t[f]
1AAY	28	SSD-FASTER	1.0	0.0	0	0.5
		MSD-MC/Null	0.2	0.0	0	2.5
		MSD-MC/MC	0.2	0.0	0	8.4
		MSD-MC/iBR	0.8	0.0	0	3.2
		MSD-MC/FASTER	0.0	0.7	2	2.6
1PIN	34	SSD-FASTER	2.1	0.0	0	0.6
		MSD-MC/Null	0.3	0.0	0	3.0
		MSD-MC/MC	0.0	0.5	5	9.7
		MSD-MC/iBR	0.1	0.0	0	3.8
		MSD-MC/FASTER	0.0	1.2	9	3.3
1PGA	56	SSD-FASTER	4.2	0.0	0	1.9
		MSD-MC/Null	0.0	3.9	7	5.5
		MSD-MC/MC	0.0	7.8	16	18.1
		MSD-MC/iBR	0.0	1.5	5	16.7
		MSD-MC/FASTER	0.0	11.2	12	9.9
1C9O	66	SSD-FASTER	2.0	0.0	0	1.4
		MSD-MC/Null	0.0	1.6	4	6.7
		MSD-MC/MC	0.0	5.6	14	24.3
		MSD-MC/iBR	0.0	2.0	5	22.7
		MSD-MC/FASTER	0.0	12.4	20	11.0

a. The number of variable positions in the design
b. The optimization strategy that was used, as described in the text. The term after the slash indicates the weak rotamer optimization routine that was used. The number of steps of MSD-MC was adjusted for each algorithm combination so that the average times per trajectory would be similar to those for MSD-FASTER (Table 1).
c. The percentage of trajectories that found the optimal solution ($f \times 100$), as determined by SSD-FASTER. 1000 total trajectories were computed in each MSD or SSD calculation.
d. The difference in simulation energy (kcal/mol) between the best sequence found by MSD-MC and the optimal sequence found by SSD-FASTER
e. The number of residue positions that differ between the best sequence found by MSD-MC and the optimal sequence found by SSD-FASTER
f. The average time, in minutes, required to perform each trajectory on one processor

Multi-state design of protein G

To compare MSD-FASTER and MSD-MC in the context of positive design, we designed two separate areas of 1GB1, a 60-member NMR ensemble of the β1 domain of streptococcal protein G.[20] Single-state designs based on the crystal structure of this protein have found several stabilized variants,[13, 21] but to our knowledge no designs based on an NMR ensemble of this molecule have yet been characterized experimentally. In the first design, we varied all 25 non-Gly positions classified as core or boundary, and in the second we varied all 27 non-Gly positions classified as surface.

For the MSD-FASTER calculations, we dispensed with the evaluation of the several possible rotamer optimization routines, and relied on FASTER only for this purpose. However, given our concerns about potential problems with fixed rotamer selection schemes during MSD-sPR, we tested three implementations in MSD-FASTER. In two cases, (r = 1 and r = 5 in Table 3), the choice of fixed rotamer in each state was determined as described above; the relaxed amino acid sequence to be scored by rotamer optimization was either produced from the best fixed rotamer configuration found, or was produced from a randomly chosen member of the top five configurations found, respectively. In the final case (r = rand), the fixed rotamer optimization was skipped entirely, and the relaxed amino acid sequence to be rescored was determined with fixed rotamers of the perturbed amino acid type chosen randomly for each state. Calculation parameters for MSD-FASTER and the strong and weak rotamer optimization routines were identical to those described for the single-state design test cases above.

We tested a variety of formulations of MSD-MC in an attempt to find one that would compare favorably to MSD-FASTER when applied to many target states

simultaneously. Implementation details that were varied included the type of rotamer optimization performed, the application of uniform sampling of amino acid substitutions, and the use of the distance-based cutoff to limit the expense of rotamer optimization; several of these combinations are shown in Table 3.

In contrast to the SSD test cases described above, the optimal solutions to these two MSD problems are not known except through the calculations we report here. In the absence of additional information, we sampled as rigorously as possible with each MSD algorithm and assumed the best-scoring sequence ever found to be optimal. We typically use this strategy when optimizing single-state designs with stochastic algorithms as well.[9]

For the core+boundary design, all the formulations of MSD-FASTER and MSD-MC we tested found the same lowest-energy solution (Table 3). All three implementations of MSD-FASTER achieved essentially identical performance, indicating that method used to choose fixed rotamers in MSD-sPR was not a significant determinant of optimization power in this design problem. Among the MSD-MC formulations we tested, MSD-MC/iBR performed slightly better than any of the MSD-FASTER implementations, whereas all other performed significantly worse. The preference for a rotamer optimization routine of intermediate expense is consistent with the results of our SSD test calculations (Table 2). It illustrates that, for efficient sampling in MSD-MC to be achieved, a delicate balance must be struck between the accuracy of sequence-rescoring and the number of individual sequences that are evaluated.

Analysis of the surface design calculations shows a stark contrast between the performance of MSD-FASTER and MSD-MC. Whereas all three MSD-FASTER implementations each found the same top sequence in a significant fraction of the

attempted trajectories, this sequence was never found by any of the MSD-MC formulations we tried, despite their greater computational expense. This more difficult design problem also allowed differentiation between the three MSD-FASTER implementations; randomly chosen fixed rotamers (r = rand) resulted in a 5-fold drop in optimization efficiency compared to the use of fixed-rotamer optimization in MSD-sPR (r = 1).

When the states in a MSD calculation are very similar, one might ask whether the MSD-optimal solution could have been found by performing single-state design on each state and rescoring the resulting SSD-derived sequences using MSD. In the case of the core+boundary design described here, the MSD-optimal sequence was never found during single-state design of the individual states; the MSD-optimal sequence for the surface design was also the SSD-optimal sequence for only one of the 60 states. Use of the MSD strategy thus seems warranted for design problems with multi-state requirements; the SSD-based strategy cannot be generally relied upon to produce the same sequences as a true MSD procedure.

The results of the 1GB1 designs show that both MSD-MC and MSD-FASTER can efficiently find low-energy sequences based on a large NMR structural ensemble. Although one formulation of MSD-MC performed slightly better than MSD-FASTER in the core+boundary design, the failure of all MSD-MC formulations when applied to the surface design prompts greater confidence in the consistency and general utility of MSD-FASTER. When applying MSD-FASTER to a large conformational ensemble, the optimization of fixed rotamer choice in MSD-sPR may help to improve the efficiency of sampling in some design problems, and can be recommended on this basis.

Table 3: Multi-state design of 1GB1, a 60-member NMR ensemble of protein G

Design	Size[a]	Opt[b]	r[c]	% correct[e] ($f \times 100$)	t[f]	S[g]
Core	25	MSD-FASTER	1	5.4	3.0	55
+		MSD-FASTER	5	4.8	2.9	60
Boundary		MSD-FASTER	rand	4.1	2.3	56
			US/CPL[d]			
		MSD-MC/FASTER	no	0.7	4.2	593
		MSD-MC/FASTER	yes	2.9	4.3	147
		MSD-MC/Null	yes	0.2	3.4	1712
		MSD-MC/iBR	yes	9.1	4.2	46
			r[c]			
Surface	27	MSD-FASTER	1	5.6	2.8	50
		MSD-FASTER	5	3.8	2.8	75
		MSD-FASTER	rand	1.0	2.6	261
			US/CPL[d]			
		MSD-MC/FASTER	no	0.0	4.2	—
		MSD-MC/FASTER	yes	0.0	4.4	—
		MSD-MC/Null	yes	0.0	3.4	—
		MSD-MC/iBR	yes	0.0	4.3	—

a. The number of variable positions in the design
b. The optimization strategy that was used, as described in the text
c. After optimizing the choice of fixed rotamer in all states during a step of sPR, the amino acid sequence to score by rotamer optimization is chosen randomly from the top r fixed rotamer configurations. "rand" indicates that the fixed rotamer optimization step is skipped, and the amino acid sequence to score results from randomly chosen fixed rotamers in each state.
d. Indicates whether or not uniform substitution sampling is applied in MSD-MC and a close position limit of 15 Å is applied during each rotamer optimization.
e. The percentage of trajectories that found the optimal MSD solution, as defined in the text. 1000 trajectories were computed for each design.
f. The average time, in minutes, required to perform each trajectory using 60 processors
g. The score $S = t / f$, as described in the text. Smaller values are better, indicating that the optimal solution can be found more quickly. "—" indicates that S is undefined because $f = 0$.

Negative design of calmodulin

Calmodulin (CaM) is a second messenger protein that, in the presence of Ca^{2+}, binds to different recognition sequences on various proteins with high affinity and low specificity.[22] CaM variants with increased specificity have been engineered by performing single-state design on a crystal structure of CaM bound to a target peptide from smooth muscle myosin light chain kinase (smMLCK).[23, 24] Experimentally, the variants bound the smMLCK peptide with similar affinity to wild type, and bound most other target peptides with weaker affinity than wild type. Although those experiments showed that single-state design was sufficient to alter binding specificity in this system, we anticipate that more delicate control over such properties may be allowed through the use of explicit negative design. To assess the utility of MSD-FASTER and MSD-MC for negative design, we attempted to design CaM sequences that would bind smMLCK and fail to bind another natural CaM target, CaM kinase I (CaMKI). This sequence selection was performed via a two-state design with a smMLCK-CaM crystal structure as the positive design target state (1CDL),[25] and a CaMKI-CaM crystal structure as the negative design target state (1MXE).[26]

Table 4 compares the application of SSD-FASTER, MSD-FASTER, and MSD-MC to this simple negative formulation of negative design. First, we evaluated the previously published technique for implicit computational negative design. In this case, we applied SSD-FASTER to the positive design target state only, rescored the resulting best sequence against the negative design target state by rotamer optimization, and combined these two energies into an overall score using equation 5. These calculations

indicate a partial clash when the SSD-optimal sequence is threaded on the negative design target state, and a predicted increase in binding specificity.

As with the protein G NMR ensemble calculations, we dispensed with the evaluation of each rotamer optimization routine in the context of MSD-FASTER, and relied on FASTER only. Furthermore, we again tested the fixed rotamer selection schemes during MSD-sPR corresponding to $r = 1$, $r = 5$, and $r =$ rand.

Interestingly, all three techniques found the same best-scoring sequence in 15–20% of their trajectories, and all three incurred roughly the same amount of computational expense. According to the simulations, this sequence is destabilized by only 0.4 kcal/mol in the context of the positive design target state compared to the optimal sequence for that state, and is predicted to clash more significantly when threaded on the negative design target than the sequence found using SSD-FASTER alone. The similarity between the results and performance of the three implementations of MSD-FASTER/FASTER tested here inspires confidence that the utility of MSD-FASTER does not hinge on the particulars of the scheme used to choose rotamers of the fixed amino acid type during MSD-sPR.

We also tested the same set of formulations for MSD-MC as we did for the 1GB1 designs described above, in an attempt to find one that would compare favorably to MSD-FASTER for explicit negative design (Table 4). Despite our best efforts, and even with substantially more computational time devoted to the problem, no version of MSD-MC was able to find the solution produced by MSD-FASTER even once. Furthermore, no MSD-MC calculation converged on any particular consensus solution, indicating that either much longer simulation times or a much better algorithm formulation would be

required for a user to have confidence in the results produced by MSD-MC for this design problem. The best solutions that were found using MSD-MC all exhibited destabilization in the context of the positive design target state in addition to several clashes in the negative design target state; however, only extensive experimental validation will conclusively show whether these differences in simulation energy are meaningful in the context of the potential functions and rigid structural models we have used here. To the extent that predicted clashes correlate with destabilization of the negative design target state, both MSD algorithms are expected to be more useful than single-state design for the explicit manipulation of specificity. Based on our results, MSD-FASTER should be preferred over MSD-MC due to the higher efficiency with which it is able to discover favorable sequences and the greater confidence inspired by its ability to repeatedly discover the optimal solution.

Table 4: Explicit negative design to increase the binding specificity of calmodulin

Opt[a]		% correct[d] ($f \times 100$)	t^e	$\Delta E_P{}^f$	$\Delta E_N{}^g$	σ^h	N^i
SSD-FASTER		0.0	0.9	0.0	37.6	-1.5	2
	r^b						
MSD-FASTER/FASTER	1	18.5	13.9	0.4	54.4	-1.8	0
MSD-FASTER/FASTER	5	19.5	13.4	0.4	54.4	-1.8	0
MSD-FASTER/FASTER	rand	15.1	13.7	0.4	54.4	-1.8	0
	US/CPL[c]						
MSD-MC/FASTER	no	0.0	24.1	4.2	92.0	0.5	6
MSD-MC/FASTER	yes	0.0	27.3	4.0	110.6	-0.4	2
MSD-MC/Null	yes	0.0	14.1	6.0	100.2	2.0	6
MSD-MC/iBR	yes	0.0	15.2	5.7	139.8	0.1	6

a. The optimization strategy that was used, as described in the text. In SSD-FASTER, sequences were optimized in the context of the positive design target only, and then rescored against both targets.
b. After optimizing the choice of fixed rotamer in all states during a step of sPR, the amino acid sequence to score by rotamer optimization is chosen randomly from the top r fixed rotamer configurations. "rand" indicates that the fixed rotamer optimization step is skipped, and the amino acid sequence to score results from randomly chosen rotamers of the fixed amino acid type in each state.
c. Indicates whether or not uniform substitution sampling is applied for MSD-MC and a close position limit of 15 Å is applied during each rotamer optimization.
d. The percentage of trajectories that found the optimal MSD solution, as defined in the text. 1000 trajectories were performed for each MSD calculation, and 6400 were performed for the SSD-FASTER calculation.
e. The average time, in minutes, required to perform each trajectory using 2 processors (MSD), or 1 processor (SSD)
f. The excess energy of the best sequence threaded on the positive design target (equation 6)
g. The excess energy of the best sequence threaded on the negative design target (equation 6). The pairwise energies that are summed to yield this value are each capped at 50 kcal/mol.
h. The overall score of the best sequence found (equation 5)
i. The number of amino acid differences between this sequence and the best designed sequence determined using MSD-FASTER

Conclusions

We have presented implementation details of a new optimization algorithm for multi-state protein design based on FASTER, determined acceptable parameters for its use, and compared its performance to a multi-state implementation of Monte Carlo. Accurate scoring of sequences suggested by the MSD algorithms is required for efficient multi-state optimization; rotamer optimization routines for side-chain placement based on MC, FASTER, and iBR can all provide acceptable performance. Our results indicate that both MSD algorithms can find favorable sequences in realistic test cases for positive and negative design. Both algorithms can accommodate design problems with many states; even a 60-member NMR ensemble was designed without difficulty. In our hands, MSD-MC scales poorly compared to MSD-FASTER as the complexity of the design problem increases; the observed difference is much more pronounced than what has been reported for the single-state versions of these algorithms.[9] Due to this effect, the efficiency and consistency of MSD-FASTER was better than MSD-MC in every class of design problem we tested. In most cases, MSD-MC could not ever find the low-energy consensus solutions produced by MSD-FASTER. Given that the evaluation of each sequence is relatively time-consuming in MSD, MSD-FASTER likely performs better because it tends to make multiple substitutions simultaneously, and because substitutions are selected for scoring based on energetic considerations rather than randomly.

Although the general approach to multi-state design used by these MSD algorithms has met with several experimental successes already,[15, 17, 18] rigorous evaluation of energy functions and multi-state scoring functions will be required to prove

and improve the usefulness of this methodology. Realistic design procedures based on the explicit modeling of many native and non-native conformational states cannot be implemented without efficient optimization techniques to drive them. We hope that the greater optimization power of MSD-FASTER will help to accelerate progress in this area via its improved speed and accuracy compared to alternative methods.

Materials and methods

Design parameters: single-state design test cases

The energy functions and designed positions used for the single-state design problems were as previously described.[9]

For rotamer optmization, four of the weak RO algorithms (Null, MC, iBR, and FASTER) were paired with a strong rotamer optimizer utilizing two trajectories of FASTER with a maximum of 5 rounds of iBR and 3 rounds of sPR. When DEE was used as the weak rotamer optimizer, it was also used as the strong rotamer optimizer, as explained above. For the weak RO algorithms iBR and FASTER, the maximum number of nonproductive iBR rounds was 5. For FASTER, the iBR pass was followed by exactly one round of sPR. For those sequences for which DEE failed to converge, the strong FASTER rotamer optimization routine described above was automatically employed to find a reasonable approximate solution. The simulated annealing regimen for MC when used for weak RO comprised 1 cycle of 2.0×10^4 steps with a high temperature of 400 K and a low temperature of 1 K.

In MSD-FASTER, the FASTER parameters for sequence selection were: maximum nonproductive rounds in iBR, 5, maximum rounds in sPR, 5, and number of

relaxing positions in each step of sPR, 10.[9] In every MSD-MC calculation, the high and low temperatures for sequence selection were also set to 400 K and 1 K, respectively. The number of cycles and steps of MSD-MC was set in each calculation so that total time used by MSD-FASTER and MSD-MC would be comparable. The following simulated annealing schedules were used for sequence selection in each algorithm combination: MSD-MC/Null, 10 cycles of 1.0×10^6 steps; MSD-MC/MC, 1 cycle of 2.5×10^4 steps; MSD-MC/iBR, 1 cycle of 1.0×10^5 steps; MSD-MC/FASTER, 1 cycle of 1.5×10^4 steps.

Design parameters: 1GB1

The 1GB1 ensemble of protein G[20] was prepared and designed as follows. Hydrogens were removed from each ensemble member and added back in optimized positions using REDUCE.[27] Each structure was then standardized via 50 steps of conjugate-gradient minimization with the DREIDING force field.[28] All positions were classified as core, boundary, or surface as described previously[1] based on the coordinates of the crystal structure (1PGA).[29] The core+boundary design comprised positions 1, 3, 5, 7, 11, 12, 16, 18, 20, 23, 25, 26, 27, 29, 30, 33, 34, 37, 39, 43, 45, 50, 52, 54, and 56; the surface design comprised positions 2, 4, 6, 8, 10, 13, 15, 17, 19, 21, 22, 24, 28, 31, 32, 35, 36, 40, 42, 44, 46, 47, 48, 49, 51, 53, and 55. In the core+boundary design, the amino acid types Ala, Val, Leu, Ile, Phe, Tyr, and Trp were allowed at each designed core position; Ala, Val, Leu, Ile, Phe, Tyr, Trp, Ser, Thr, Asn, Gln, Asp, Glu, His, Lys, and Arg were allowed. In the surface design, Ala, Ser, Thr, Asn, Gln, Asp, Glu, His, Lys, and Arg were allowed. For each design, we used rotamers from the Dunbrack backbone-dependent rotamer library.[30] There were an average of 3634 total rotamers per state with

rotamer/template energies better than 20 kcal/mol for the core+boundary design, and 5617 for the surface design. Pairwise energies were computed using energy functions as previously described,[7] except the polar hydrogen burial term was omitted.

For the core+boundary design, the following parameters were used for each MSD-MC algorithm combination: MSD-MC/FASTER (no US/CPL), 1 cycle of 2.0×10^4 steps; MSD-MC/FASTER, 1 cycle of 3.5×10^4 steps; MSD-MC/Null, 1 cycle of 5.0×10^5 steps; MSD-MC/iBR, 1 cycle of 1.0×10^5 steps.

For the surface design, the following parameters were used: MSD-MC/FASTER (no US/CPL), 1 cycle of 6.0×10^3 steps; MSD-MC/FASTER, 1 cycle of 1.3×10^4 steps; MSD-MC/Null, 1 cycle of 5.0×10^5 steps; MSD-MC/iBR, 1 cycle of 6.5×10^4 steps.

The number of MSD-MC steps in each case was chosen to make the average time per trajectory similar to MSD-FASTER. Equation 4 was used with kT = 300 kcal/mol to combine the energies from all 60 ensemble members into overall scores.

Design parameters: CaM

The two CaM structures were prepared and minimized as described above for the 1GB1 structures. Chains B and F were used from the 1CDL structure and chains A and E were used from the 1MXE structure. The amino acid types Ala, Val, Leu, Ile, Phe, Tyr, Trp, Met, and Glu were allowed at each of the following designed positions on the CaM chain: 7, 8, 11, 14, 15, 28, 32, 35, 47, 51, 64, 67, 68, 80, 84, 87, 88, 101, 104, 105, 108, 120, 124, 140, and 141. The 19 positions of the smMLCK peptide in the positive design state and the 25 positions of the CaMKI peptide in the negative design state were allowed to vary side-chain conformation but not amino acid identity. Side-chain conformations at

the variable positions were from the Dunbrack backbone-dependent rotamer library with expansions of ±1 standard deviation about $\chi 1$ and $\chi 2$. The same energy functions were used to compute pairwise energies as for the 1GB1 designs described above. For the multi-state design calculations, all rotamer-backbone and rotamer-rotamer energies on the negative design target state were capped at 50 kcal/mol. To compute σ during the optimizations, equation 5 was used with $W = 0.04$. The single-state design optimizations were performed as described,[9] without the initial elimination of rotamers using DEE.

The following parameters were used for each MSD-MC algorithm combination: MSD-MC/FASTER (no US/CPL), 1 cycle of 2.0×10^3 steps; MSD-MC/FASTER, 1 cycle of 6.0×10^3 steps; MSD-MC/Null, 25 cycles of 1.0×10^6 steps; MSD-MC/iBR, 1 cycle of 3.0×10^4 steps.

Acknowledgements

The authors thank Kyle Lassila, Christina Vizcarra, Jennifer Keeffe, and an anonymous reviewer for their insightful comments. This work was supported by the Howard Hughes Medical Institute, the Ralph M. Parsons Foundation, an IBM Shared University Research Grant, and the Defense Advanced Research Projects Agency.

References

1. Dahiyat, B. I.; Mayo, S. L., De novo protein design: Fully automated sequence selection. *Science* **1997,** *278* (5335), 82–87.

2. Gordon, D. B.; Marshall, S. A.; Mayo, S. L., Energy functions for protein design. *Current Opinion in Structural Biology* **1999,** *9* (4), 509–513.

3. Kuhlman, B.; Dantas, G.; Ireton, G. C.; Varani, G.; Stoddard, B. L.; Baker, D., Design of a novel globular protein fold with atomic-level accuracy. *Science* **2003,** *302* (5649), 1364–1368.

4. Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E., Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* **1953,** *21* (6), 1087–1092.

5. Voigt, C. A.; Gordon, D. B.; Mayo, S. L., Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *Journal of Molecular Biology* **2000,** *299* (3), 789–803.

6. Desmet, J.; Demaeyer, M.; Hazes, B.; Lasters, I., The Dead-End Elimination Theorem and Its Use in Protein Side-Chain Positioning. *Nature* **1992,** *356* (6369), 539–542.

7. Gordon, D. B.; Hom, G. K.; Mayo, S. L.; Pierce, N. A., Exact rotamer optimization for protein design. *Journal of Computational Chemistry* **2003,** *24* (2), 232–243.

8. Desjarlais, J. R.; Handel, T. M., De-Novo Design of the Hydrophobic Cores of Proteins. *Protein Science* **1995,** *4* (10), 2006–2018.

9. Allen, B. D.; Mayo, S. L., Dramatic performance enhancements for the FASTER optimization algorithm. *Journal of Computational Chemistry* **2006,** *27* (10), 1071–1075.

10. Desmet, J.; Spriet, J.; Lasters, I., Fast and Accurate Side-Chain Topology and Energy Refinement (FASTER) as a new method for protein structure optimization. *Proteins-Structure Function and Genetics* **2002,** *48* (1), 31–43.

11. Jiang, L.; Althoff, E. A.; Clemente, F. R.; Doyle, L.; Rothlisberger, D.; Zanghellini, A.; Gallaher, J. L.; Betker, J. L.; Tanaka, F.; Barbas, C. F.; Hilvert, D.; Houk, K. N.; Stoddard, B. L.; Baker, D., De novo computational design of retro-aldol enzymes. *Science* **2008,** *319* (5868), 1387–1391.

12. Looger, L. L.; Dwyer, M. A.; Smith, J. J.; Hellinga, H. W., Computational design of receptor and sensor proteins with novel functions. *Nature* **2003,** *423* (6936), 185–190.

13. Malakauskas, S. M.; Mayo, S. L., Design, structure and stability of a hyperthermophilic protein variant. *Nature Structural Biology* **1998,** *5* (6), 470–5.

14. Rothlisberger, D.; Khersonsky, O.; Wollacott, A. M.; Jiang, L.; DeChancie, J.; Betker, J.; Gallaher, J. L.; Althoff, E. A.; Zanghellini, A.; Dym, O.; Albeck, S.; Houk, K. N.; Tawfik, D. S.; Baker, D., Kemp elimination catalysts by computational enzyme design. *Nature* **2008,** *453* (7192), 190–U4.

15. Havranek, J. J.; Harbury, P. B., Automated design of specificity in molecular recognition. *Nature Structural Biology* **2003,** *10* (1), 45–52.

16. Wuthrich, K., Protein structure determination in solution by NMR spectroscopy. *Journal of Biological Chemistry* **1990,** *265* (36), 22059–22062.

17. Ambroggio, X. I.; Kuhlman, B., Computational design of a single amino acid sequence that can switch between two distinct protein folds. *Journal of the American Chemical Society* **2006,** *128* (4), 1154–61.

18. Boas, F. E.; Harbury, P. B., Design of protein-ligand binding based on the molecular-mechanics energy model. *Journal of Molecular Biology* **2008,** *380* (2), 415–424.

19. Bolon, D. N.; Grant, R. A.; Baker, T. A.; Sauer, R. T., Specificity versus stability in computational protein design. *Proceedings of the National Academy of Sciences of the United States of America* **2005,** *102* (36), 12724–12729.

20. Gronenborn, A. M.; Filpula, D. R.; Essig, N. Z.; Achari, A.; Whitlow, M.; Wingfield, P. T.; Clore, G. M., A novel, highly stable fold of the immunoglobulin binding domain of streptococcal protein G. *Science* **1991,** *253* (5020), 657–61.

21. Dahiyat, B. I.; Mayo, S. L., Probing the role of packing specificity in protein design. *Proceedings of the National Academy of Sciences of the United States of America* **1997,** *94* (19), 10172–10177.

22. O'Neil, K. T.; DeGrado, W. F., How calmodulin binds its targets: sequence independent recognition of amphiphilic alpha-helices. *Trends in Biochemical Sciences* **1990,** *15* (2), 59–64.

23. Shifman, J. M.; Mayo, S. L., Modulating calmodulin binding specificity through computational protein design. *Journal of Molecular Biology* **2002,** *323* (3), 417–423.

24. Shifman, J. M.; Mayo, S. L., Exploring the origins of binding specificity through the computational redesign of calmodulin. *Proceedings of the National Academy of Sciences of the United States of America* **2003,** *100* (23), 13274–13279.

25. Meador, W. E.; Means, A. R.; Quiocho, F. A., Target enzyme recognition by calmodulin: 2.4 A structure of a calmodulin-peptide complex. *Science* **1992,** *257* (5074), 1251–1255.

26. Clapperton, J. A.; Martin, S. R.; Smerdon, S. J.; Gamblin, S. J.; Bayley, P. M., Structure of the complex of calmodulin with the target sequence of calmodulin-dependent protein kinase I: studies of the kinase activation mechanism. *Biochemistry* **2002,** *41* (50), 14669–14679.

27. Word, J. M.; Lovell, S. C.; Richardson, J. S.; Richardson, D. C., Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation. *Journal of Molecular Biology* **1999,** *285* (4), 1735–1747.

28. Mayo, S. L.; Olafson, B. D.; Goddard, W. A., Dreiding — a Generic Force-Field for Molecular Simulations. *Journal of Physical Chemistry* **1990,** *94* (26), 8897–8909.

29. Gallagher, T.; Alexander, P.; Bryan, P.; Gilliland, G. L., 2 Crystal-Structures of the B1 Immunoglobulin-Binding Domain of Streptococcal Protein-G and Comparison with NMR. *Biochemistry* **1994,** *33* (15), 4721–4729.

30. Dunbrack, R. L.; Cohen, F. E., Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Science* **1997,** *6* (8), 1661–1681.