Rotamer Optimization for Protein Design through MAP Estimation and Problem-Size Reduction

EUN-JONG HONG,[1,2] SHAUN M. LIPPOW,[1,3]* BRUCE TIDOR,[1,2,4] TOMÁS LOZANO-PÉREZ[1,2]

[1]*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

[2]*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

[3]*Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

[4]*Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

Received 27 February 2008; Revised 22 September 2008; Accepted 10 November 2008
DOI 10.1002/jcc.21188
Published online in Wiley InterScience (www.interscience.wiley.com).

Abstract: The search for the global minimum energy conformation (GMEC) of protein side chains is an important computational challenge in protein structure prediction and design. Using rotamer models, the problem is formulated as a NP-hard optimization problem. Dead-end elimination (DEE) methods combined with systematic A^* search (DEE/A^*) has proven useful, but may not be strong enough as we attempt to solve protein design problems where a large number of similar rotamers is eligible and the network of interactions between residues is dense. In this work, we present an exact solution method, named BroMAP (branch-and-bound rotamer optimization using MAP estimation), for such protein design problems. The design goal of BroMAP is to be able to expand smaller search trees than conventional branch-and-bound methods while performing only a moderate amount of computation in each node, thereby reducing the total running time. To achieve that, BroMAP attempts reduction of the problem size within each node through DEE and elimination by lower bounds from approximate maximum-a-posteriori (MAP) estimation. The lower bounds are also exploited in branching and subproblem selection for fast discovery of strong upper bounds. Our computational results show that BroMAP tends to be faster than DEE/A^* for large protein design cases. BroMAP also solved cases that were not solved by DEE/A^* within the maximum allowed time, and did not incur significant disadvantage for cases where DEE/A^* performed well. Therefore, BroMAP is particularly applicable to large protein design problems where DEE/A^* struggles and can also substitute for DEE/A^* in general GMEC search.

 J Comput Chem 00: 000–000, 2009

Key words: dead-end elimination; side-chain placement; branch-and-bound; protein design; combinatorial optimization; global minimum energy conformation; maximum-a-posteriori estimation

Introduction

Determining low-energy placements for side chains on a fixed backbone is an important problem in both protein structure prediction and protein design. A typical approach to the protein structure prediction is homology modeling [1–3] followed by refinement of the model through determination of the side-chain conformations. Determining the side-chain conformation for a given backbone structure and an amino acid sequence is called "side-chain placement" and is solved through finding the minimum energy conformation. In addition, in protein design problems, also referred as the "inverse folding problem" ,[4–6] an amino acid sequence that will stably fold to the target backbone structure is to be found. Given a backbone structure and energy functions, the protein design problem is also solved as a generalized side-chain placement problem, that is, by finding the minimum energy conformation of side chains, drawing from a range of amino acid types at each residue position .[7,8] If the backbone structure is not assumed to be fixed, one can still design with a flexible backbone by using iterative steps, where a side-chain

*Present address: Codon Devices, Inc., One Kendall Square, Building 300, Cambridge, Massachusetts 02139.

Correspondence to: T. Lozano-Pérez; e-mail: tlp@csail.mit.edu

placement problem is solved for each perturbed fixed backbone structure .[9] The search for the minimum energy conformation is, therefore, one of the most important computational challenges in computational protein design.

In finding the minimum energy conformation, the search space can be simplified by allowing only some finite number of fixed side-chain conformations, called rotamers .[10,11] With the rotamer model, the energy function of a protein sequence folded onto a specific backbone template can be described in terms of :[12]

1. the self-energy of the backbone template from the interactions within the backbone [denoted as E_{template}];
2. the singleton interaction energy between the backbone and rotamer conformation r at position i of the sequence [denoted as $E(i_r)$];
3. the pairwise interaction energy between rotamer conformation r at position i and rotamer conformation s at position j, $i \neq j$ [denoted as $E(i_r, j_s)$].

Then, the energy of a protein sequence of length n in a specific backbone template structure and conformation $C = \{C_1, \ldots, C_n | C_i$ is the conformation of position $i\}$ can be written in a functional form as

$$\mathcal{E}(C) = E_{\text{template}} + \sum_{i=1}^{n} E(C_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(C_i, C_j). \qquad (1)$$

Energy terms $E(i_r)$ and $E(i_r, j_s)$ can be computed for a given backbone template and the set of allowed rotamers using coordinates of atoms and specified molecular force fields, such as AMBER ,[13–15] CHARMM ,[16,17] MMFF,[18] or OPLS .[19] The conformation C that minimizes the energy function $\mathcal{E}(C)$ is often called the global minimum energy conformation (GMEC). In this work, we consider the problem of finding the GMEC when given a backbone conformation, a set of rotamers, and energy terms, and call such a problem "the GMEC problem". Note that E_{template} is constant by definition and can be ignored when we minimize $\mathcal{E}(C)$.

The GMEC problem is a strongly NP-hard optimization problem as one can readily show by reduction from the satisfiability problem .[20] Despite the theoretical hardness, one finds that many instances of the GMEC problem are easily solved by the exact method of dead-end elimination (DEE) .[12] Elimination procedures such as Goldstein's conditions and unification ,[21] logical singles-pairs elimination ,[22] the magic bullet pairs heuristic ,[23] splitting,[24] generalized elimination conditions ,[25] hybrid optimization through scheduling of various elimination conditions ,[26] and more recently divide-and-conquer enhancement to DEE [27] are often able to reduce the problem size dramatically, while demanding only reasonable computational power.

Other than DEE, there exist various approaches to solve the GMEC problem exactly. Leach and Lemon ,[28] Gordon and Mayo ,[29] and Wernisch et al.[30] describe a branch-and-bound method. Eriksson et al. ,[31] Althaus et al.,[32] and Kingsford et al. [33] present integer linear programming approaches. Leaver-Fay et al. [34] describe a dynamic programming approach based on tree-decomposition. Xu [35] describes another method based on tree-decomposition and presents a tree-decomposition algorithm for protein backbone structures. Xie and Sahinidis [36] describe a method that combines several residue-reduction and rotamer-reduction techniques. Yanover et al. [37] use a tree-reweighted belief propagation algorithm as a linear-program solver with better scalability, and Weiss et al. [38] extend this approach by suggesting a search scheme for an integral solution when the solution of the linear program is fractional. Each exact approach may have some advantages over others depending on the characteristics of the problem being considered. For example, for a simplified version of the problem where the number of rotamers per position is limited or interactions between residue positions are sparse, even deterministic algorithms with guaranteed time bounds exist. However, it is known that protein structures and stabilities can be predicted better with more side-chain flexibility, that is, by using a larger rotamer library .[39,40] In addition, the network of interactions between residue positions can be dense as is often observed in protein cores. Therefore, we are interested in protein design problems where all possible pairs of positions are assumed to interact and a large number of similar rotamers is offered at each position. To our knowledge, only DEE-like methods or DEE followed by branch-and-bound methods have shown success in solving such hard protein design cases exactly.

There also exist approximate approaches for the GMEC problem. Koehl and Delarue [41] present the self-consistent mean field theory. Desjarlais and Handel [42] and Jones[43] use genetic algorithms. Jiang et al. [44] use simulated annealing and Monte Carlo sampling. Wernisch et al. [30] describe a heuristic for protein design. Yanover and Weiss [45] use belief-propagation methods. However, inaccuracy during GMEC search may introduce uncertainty in the analysis step where correction of energy functions or modification of the design protocol is to be made. Therefore, we are primarily interested in finding the exact GMEC and will not further consider approximate methods in this work.

Enhanced DEE [26] performs well for some of the hard protein design cases of interest to us. However, finding dead-ends using the known elimination conditions does not always eliminate as many rotamers or rotamer pairs as necessary. In case the remaining conformational space after DEE application is too large to literally enumerate, a systematic search method such as A^* algorithm [28,46] is often followed to find the GMEC (call the combined method DEE/A^*). However, such a combined scheme will not be useful unless DEE reduces the size of conformational space to the point where a systematic search is applicable.

Here we describe a new exact solution method for the GMEC problem that can substitute for DEE/A^*, especially in solving hard design cases. Our method, named BroMAP (branch-and-bound rotamer optimization using MAP estimation), is based on the branch-and-bound (BnB) framework and a new subproblem-pruning method. We present lower-bounding methods and problem-size reduction techniques, organized into a BnB framework, so that BroMAP is guaranteed to find an optimal solution.

Our numerical experiments confirm the utility of BroMAP in GMEC search for large protein design problems, including ones that are challenging for DEE/A^*. In our experiments, all cases solved by DEE/A^* were also solved by BroMAP, and using BroMAP did not incur significant disadvantage over DEE/A^*. Moreover, BroMAP excelled on the cases where DEE/A^* did not perform well; for each

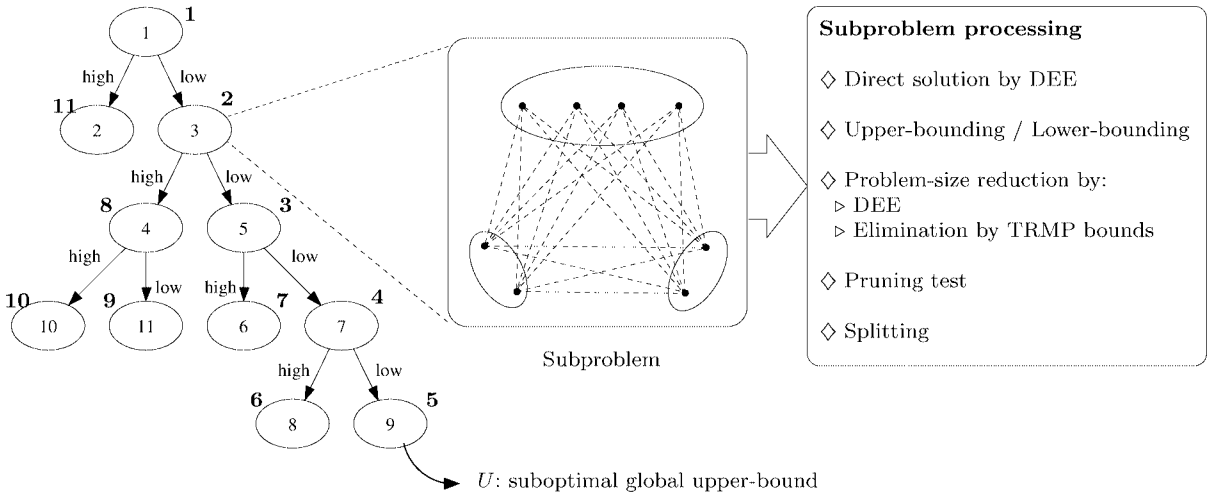

Figure 1. Top branch-and-bound framework of BroMAP. In the search tree, node numbers (inside the ellipses) correspond to the order of subproblem creation. Numbers shown next to ellipses represent the order of node expansion. Labels "low" and "high" marked on the branches indicate the types of child subproblems. As shown by the diagram in the middle, each subproblem is another instance of the GMEC problem; the ellipses represent the residue positions in the subproblem, and the filled dots represent available rotamer choices at each position. The lines connecting rotamers at different positions represent possible interactions between pairs of rotamers. The text box on the right side lists types of computations executed when a node is expanded.

case, which took longer than one hour but was eventually solved by DEE/A^*, BroMAP took at most 33% of the DEE/A^* running time. Among 68 test cases of various types and sizes, we found BroMAP failed to solve three cases within the 7-day allowed time whereas DEE/A^* failed to solve 17 of them.

Compared with DEE, BroMAP has an advantage that it can attack smaller subproblems separately using various problem-size reduction or lower-bounding techniques instead of having to keep the problem as a whole. Meanwhile, the use of DEE as one of the problem-size reduction techniques in BroMAP allows the strengths of DEE for protein design problems to be transferred to BroMAP.

BroMAP has the advantage of reducing the search trees over conventional BnB approaches in two ways. First, it uses problem-size reduction techniques within each node so that the effect of problem-size reduction from branching is often larger than that of a conventional BnB method. Hence, the depth of the resulting search tree is also smaller. Second, it quickly finds a strong upper-bound (at the end of the first depth-first dive) with the help of informed branching and subproblem selection. This facilitates effective pruning of nodes that follow, and therefore often results in sparse search trees growing mostly in one direction. BroMAP achieves these advantages without excessive computation by using new inexpensive lower-bounding methods and limiting the effort spent by bounding or problem-size reduction.

Followings are the contributions made in this work:

1. Development of lower-bounding methods for minimum conformation energy of individual rotamers and rotamer pairs using a maximum-a-posteriori estimation method called tree-reweighted max-product algorithm[47];
2. Adoption of problem-size reduction techniques (DEE and elimination by lower-bounds) within the BnB framework;
3. Use of rotamer lower-bounds in branching and subproblem selection for fast discovery of strong upper-bounds;
4. Extensive evaluation of BroMAP and DEE/A^* on various types and sizes of protein design problems.

Overview of the Method

In this section, we present an overview of BroMAP in a top-down manner. We start with a brief description of the branch-and-bound method as the framework of BroMAP. Then, the pruning scheme used by BroMAP is discussed in more detail.

Branch-and-Bound Framework

Figure 1 shows an overview of BroMAP. It is organized at the top level as a branch-and-bound method (BnB), a general problem-solving technique particularly effective for combinatorial problems .[48] The basic idea of BnB is to partition the original problem recursively and solve these smaller subproblems. In the resulting search tree, each subproblem is another instance of the GMEC problem, with a different number of rotamers or residue positions from the original problem at the root node.

BnB solves the GMEC problem as a kind of tree search problem. It maintains a global upper-bound U, which is the energy of the best conformation found so far. The initial value of U is set to the energy of an arbitrary conformation. BroMAP can be recursively described as follows:

1. Select a subproblem from the queue.
2. Can the subproblem be fully solved within limited time and memory? If so, (a) compute the minimum energy; (b) set U to the minimum energy if it is less than U; (c) return to step 1.

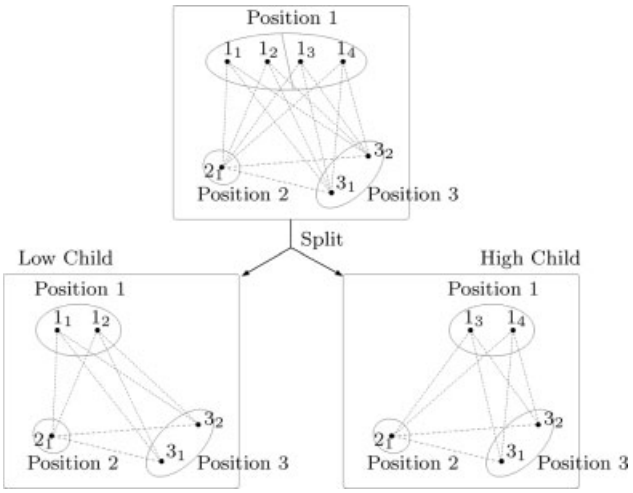

Figure 2. Splitting a subproblem. Rotamers at a position are divided into two groups and each child of the subproblem takes only one group of rotamers.

3. Compute a lower bound and an upper bound on the minimum energy for this subproblem. If the upper bound is less than U, set U to the upper bound.
4. If the lower bound exceeds the current global upper-bound U, then discard (prune) this subproblem and return to step 1.
5. When possible, exclude ineligible conformations from the search space.
6. Pick one residue and split its rotamers into two groups; define two child subproblems based on this split (see Fig. 2).
7. Add the child subproblems to the queue and return to step 1.

A node is said to be "expanded" (i.e. processed) by steps 2 to 7. This description leaves many details unspecified: how to attempt solutions, how to obtain bounds, how to identify ineligible conformations, how to choose the residue and rotamers for the node split, and what order to solve the subproblems. We provide these details in the subsequent sections.

The key advantage of BnB over naive enumeration-based methods comes from being able to approximately solve subproblems, that is, to obtain bounds on the answer that allow many subproblems to be pruned, thus avoiding exploration of the entire solution space. If the bounds are weak, BnB may end up generating too many subproblems to be effective. The purpose of branching in a BnB method is to reduce the size of the subproblems so that they can be either solved or pruned effectively with limited resources.

In our BnB formulation, the branching rule (splitting the rotamers of a residue) only brings about a modest reduction in the search space of each child subproblem compared to its parent subproblem. Furthermore, there is no net reduction in the total search space when one considers both children. A critical component of our approach is to reduce the size of the total search space, by eliminating ineligible conformations, before splitting. This is in the spirit of the dead-end elimination algorithm or "branch-and-terminate" [29] but employing additional elimination by our new lower bounds.

Solving Subproblems

There are two well-known approaches to solving the GMEC problem exactly. One is DEE [12,21,24] and the other is integer linear programming (ILP) .[48] Both of these methods are guaranteed to solve the GMEC problem given unbounded resources but have unpredictable running times as a function of the problem size.

DEE is an iterative method that eliminates a non-GMEC rotamer by comparing its energetics with those of other rotamers at the same position. The same rules are also applied to eliminate rotamer pairs. When a rotamer can be eliminated from consideration, this can be represented by reducing the set of rotamers at a residue position. Eliminated rotamer pairs, on the other hand, are tracked via "pair flags", which indicate ineligible assignments for pairs of positions. When the numerical properties of the energy terms are favorable or when the problem size is relatively small, DEE successfully eliminates many non-GMEC rotamers or rotamer pairs so that the GMEC can be easily found from the remaining small conformational space. In general, we will need to perform a systematic search of the remaining conformational space; the A^* heuristic search algorithm [46] is usually used for this purpose. However, DEE may fail to reduce the size of the conformational space to the point where it is practical to search for the GMEC using A^*. This is what motivates our BnB approach.

ILP is a popular approach to solving combinatorial optimization problems but we have found that direct application of general ILP solvers to protein design problems is generally impractical (see Appendix B). Furthermore, as we discuss below, DEE has the additional advantage of reducing the size of the conformational space at each subproblem, even when it fails to completely solve the subproblem. Therefore, we have used a DEE-based solver as our method for solving subproblems.

Bounding Subproblems

In addition to completely solving subproblems, we also need a way of obtaining lower bounds to prune nodes more efficiently. The classical approach for obtaining bounds for a combinatorial optimization problem is via the relaxation to linear programming (LP) after formulating the problem as ILP. For example, we obtain LP by treating the integer-valued variables in the ILP formulation of the GMEC problem, i.e. eqs. (B1)–(B5) of Appendix B, as real. Although LP problems are solvable in polynomial time, it is still the case that the LP problems resulting from the relaxation of typical protein design problems are often too large and thus require impractical amounts of computing time and memory.

The less expensive lower-bounding method that we use in this work is the tree-reweighted max-product algorithm (TRMP) ,[47] which will be introduced later in this paper. TRMP lower bounds are known to be no better than the LP lower bounds, and there are no guarantees of how close to the LP bound a TRMP bound will be. However, the relatively low computational cost and its good performance in practice makes TRMP an excellent lower-bounding tool.

Another key advantage of TRMP is that, like DEE, it can be used to compute lower-bounds for parts of the conformational space efficiently and to eliminate them as discussed below.

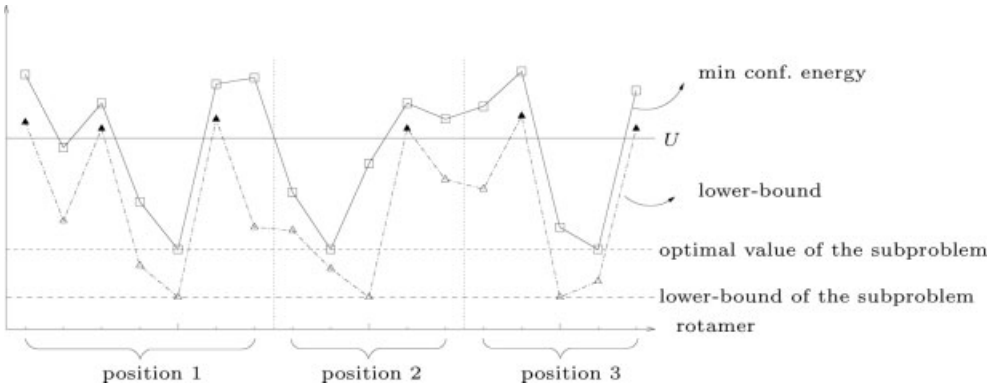

Figure 3. Elimination by rotamer lower bounds. The x-axis lists all rotamers of the subproblem in an arbitrary order. The vertical dotted lines indicate division of rotamers by positions they belong to. Two types of y-values are plotted for each rotamer i_r: (1) minimum energy that a conformation including i_r can have, (2) a lower bound of (1) obtained by a lower-bounding method. Three horizontal lines are also depicted, each representing (a) an upper bound U, (b) the optimal value of the subproblem, (c) a lower bound of (b) obtained from the same lower-bounding method. Rotamers that can be eliminated by comparison against U are indicated by filled triangles.

On the other hand, the upper bounds are also obtained by TRMP for the subproblems that are not exactly solved. This is based on a heuristic use of TRMP, but often produces stronger upper bounds than random sampling of conformations. We present the details on upper-bounding by TRMP later in the article.

Reducing Subproblem Size

As we mentioned earlier, a critical component of our BnB methodology is that we attempt to reduce the size of the search space for each subproblem by removing ineligible conformations. Smaller subproblems are easier to solve and to bound. We use two techniques to accomplish this: DEE discussed above and elimination by lower bounds. The latter is illustrated in Figure 3 and discussed below.

For each rotamer r at an arbitrary position i, we can think of an assignment of rotamers in other positions such that no other assignment can give a lower conformational energy when position i is fixed to r. We call the energy corresponding to such an assignment the minimum conformational energy of i_r. Similarly, we can define the minimum conformational energy for an arbitrary pair of rotamers (i_r, j_s) such that $i \neq j$.

Suppose we know a lower-bound $L(i_r)$ of the minimum conformational energy of i_r and a global upper-bound U such that $L(i_r) > U$. Then, rotamer i_r can be eliminated from the subproblem without affecting whether the subproblem is prunable or not. Similarly, if we have a lower bound of the minimum conformational energy of a rotamer pair greater than U, the rotamer pair can also be eliminated. Figure 4 illustrates the problem-size reduction by elimination of rotamers and rotamer pairs.

The problem is obtaining useful lower bounds for each rotamer or rotamer pair. If we use LP relaxation, we would need to solve LP problems as many times as the number of rotamers or rotamer pairs, and each LP problem can be still very large. A more practical solution follows from the theoretical properties of TRMP, which allow us to obtain the lower bounds for all rotamers and rotamer pairs in one TRMP convergence plus post-processing time at most cubic of the number of rotamers. We will discuss how we can obtain these lower bounds using TRMP later in the paper.

When a rotamer pair is eliminated by a TRMP lower bound, we mark the rotamer pair with a pair flag, as done in DEE. However, such a pair flag is more general than the pair flags used in conventional DEE since the elimination is done relative to the current global upper-bound U. Thus, it is possible for TRMP to flag rotamer pairs belonging to the minimum energy conformation of the subproblem in case the optimal value of the subproblem is greater than U. When this happens, the optimal value of the subproblem after the elimination can be greater than before the elimination. However, if the optimal value is less than or equal to U, elimination by lower bounds is guaranteed to produce reduced subproblems with unchanged optimal value.

If enough pairs are eliminated by TRMP lower bounding, it may be that some positions may not have any remaining valid assignments. In this situation, the whole subproblem is infeasible and can be pruned.

Conventional DEE never flags rotamer pairs that belong to the minimum energy conformation. Therefore, the interaction of DEE with these general pair flags should be carefully considered to avoid illegal elimination by DEE. In our work, this is done by numerically enforcing the pair flags, that is, by replacing the pair flags with very large (artificial) pairwise energies. This guarantees correct elimination by DEE conditions based on energy comparison (e.g. Goldstein's conditions). Meanwhile, when logical elimination is attempted (e.g. logical singles-pairs elimination or unification), general pair flags are used as if they are conventional pair flags.

Note that we use elimination by lower bounds together with the modified DEE in each node of the search tree. In a previous work ,[29] lower bounds were used in the BnB framework to "terminate" singles, but DEE is only used as a preprocessing procedure before applying the BnB method. In another work ,[26] elimination by lower bounds was applied in conjunction with DEE to the whole problem, but no branching was used. The lower bounds used there were also computed differently, by fixing conformations for a subset of

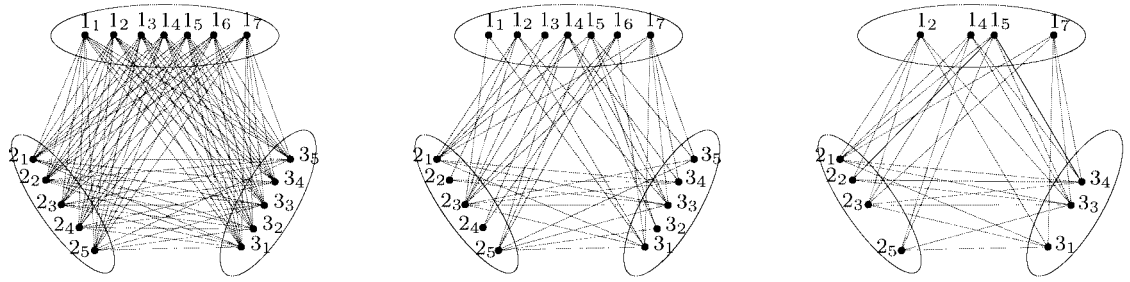

Figure 4. Reduction by elimination of rotamers and rotamer pairs. Although elimination of rotamers brings explicit reduction of the problem size, elimination of rotamer pairs will be implicitly represented by pair flags. Rotamer eliminations in (c) were made consistent with bounds of Figure 3. (a) Original subproblem. (b) Rotamer-pair elimination. (c) Rotamer elimination.

positions and finding minimum values over decomposed sets of positions.

Subproblem Splitting and Selection

Our strategy of subproblem selection is depth-first search (DFS), where one selects the deepest subproblem to expand, breaking ties by choosing the node with the smallest lower bound. The goal is to first find a good upper-bound by following DFS through the children with the lower bounds, then to prune the remaining subproblems using that upper-bound. To implement this strategy, we need to split subproblems so that they have substantially different lower bounds.

As discussed earlier, we can compute inexpensive lower bounds for individual rotamers by TRMP. Therefore, we can split a subproblem by dividing rotamers of a selected position into two groups according to their rotamer lower bounds, so that the maximum rotamer lower bound of one group is less than or equal to the minimum rotamer lower bound of the other group. We call the child from the former group "the low child" and the other as "the high child". The low child is more likely to have an optimal value less than that of the high child. A splitting position is selected so that difference between maximum and minimum rotamer lower bounds is large. This splitting scheme will also tend to make the high child easier to prune than the low child.

The leftmost diagram in Figure 1 illustrates our subproblem selection strategy. We can see that the tree first grows along the line of low-subproblems then the high-subproblems are traversed. We call the DFS along all low-branches until the first leaf node is reached as "the first depth-first dive". If the splitting is successful and nonoptimal nodes are pruned effectively, the search tree should be highly skewed toward low-branches.

Bounding the GMEC Energy through MAP Estimation

In this section, we formulate the GMEC problem as a maximum-a-posteriori (MAP) estimation problem and introduce the MAP estimation method, particularly TRMP, as a lower-bounding tool for the GMEC energy.

Problem Formulation

Probabilistic inference problems ,[49] including the MAP estimation problem, involve a random vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ characterized by a probability distribution that maps a sample $\mathbf{x} \in \mathcal{X}$ to a probability $p(\mathbf{x})$. The MAP estimation problem asks to find a MAP assignment $\mathbf{x}^*$ such that $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x})$, where $\mathcal{X}$ is the sample space for $\mathbf{x}$. In the GMEC problem, we number the sequence positions by $i = 1, \ldots, n$, and associate with each position i a discrete random variable x_i that ranges over R_i, a set of allowed rotamers at position i. Then, we can define a probability distribution $p(\mathbf{x})$ over $\mathcal{X} = R_1 \times \cdots \times R_n$ as

$$p(\mathbf{x}) = \frac{1}{Z} \exp\{-e(\mathbf{x})\}, \tag{2}$$

for a normalization constant Z and $e(\mathbf{x}) = \sum_{i=1}^{n} e_i(x_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} e_{ij}(x_i, x_j)$, where $e_i(r) = E(i_r)$ for $r \in R_i$, and $e_{ij}(r, s) = E(i_r j_s)$ for $(r, s) \in R_i \times R_j$. Therefore, the GMEC problem for minimizing $e(\mathbf{x})$ is equivalent to the MAP estimation problem for $p(\mathbf{x})$, that is, the assignment that maximizes the probability minimizes the energy. Note that the value of Z is conventionally determined so that $\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) = 1$. However, computing the exact value of Z that satisfies this condition is not necessary in finding the MAP assignment of $p(\mathbf{x})$ because $1/Z$ simply scales the exponential function of eq. (2). We will see later that our algorithm does not depend on the value of Z.

A probability distribution over a random vector can be related to a graphical model .[49] An undirected graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of vertices $\mathcal{V}$ that represent random variables and a set of edges $\mathcal{E}$ connecting some pairs of vertices. The structure of a graphical model is determined by conditional independencies among the random variables. That is, a probability distribution $p(\mathbf{x})$ can be represented by an undirected graphical model $\mathcal{G}$ if $p(\mathbf{x})$ can be factorized into non-negative functions (called compatibility functions), each of which is defined over variables in a clique of $\mathcal{G}$. The typical motivation for using the graphical model is finding as simple a model as possible that captures conditional independencies among variables. However, we generally consider a complete graph with n vertices as the graphical model for the GMEC problem, that is, the protein design problems we are interested in have molecular interactions between every pair of positions.

In the following sections, we will often describe distributions by their associated graphical model; for example, a "tree distribution" refers to a distribution represented by a tree graphical model.

Max-Marginals and Max-Product Algorithm

Wainwright et al. [50] define (singleton) max-marginals μ_i as the maximum of $p(\mathbf{x})$ when one of the variables x_i is constrained to a specific value, i.e. $\mu_i(x_i) = \kappa_i \max_{\{\mathbf{x}'|x'_i=x_i\}} p(\mathbf{x}')$. Similarly, pairwise max-marginals μ_{ij} are defined as $\mu_{ij}(x_i, x_j) = \kappa_{ij} \max_{\{\mathbf{x}'|x'_i=x_i, x'_j=x_j\}} p(\mathbf{x}')$, the maximum of $p(\mathbf{x})$ when a pair of the variables are constrained to a specific pair of values. Note that κ_i and κ_{ij} are constants that can vary depending on i and j. Hereafter, we will simply denote all the constants as κ. It is known that any tree distribution $p(\mathbf{x})$ can be factorized in terms of its max-marginals as $p(\mathbf{x}) \propto \prod_{i\in\mathcal{V}} \mu_i(x_i) \prod_{(i,j)\in\mathcal{E}} \frac{\mu_{ij}(x_i,x_j)}{\mu_i(x_i)\mu_j(x_j)}$.[49] If we knew the max-marginals of a tree distribution $p(\mathbf{x})$, we could easily compute the maximum value of $p(\mathbf{x})$.

Example 1. (Max-marginals).[50] Let $\mathbf{x} \in \{0,1\}^3$ be a random vector defined by a graphical model of Figure 5 and compatibility functions ψ such that

$$\psi_i(x_i) = 1, \text{ for all } x_i \in \{0,1\} \text{ and } i \in \{1,2,3\}, \tag{3}$$

and

$$\psi_{ij}(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 4 & \text{otherwise} \end{cases} \quad \text{for all } (i,j) \in \{(1,2),(2,3)\}. \tag{4}$$

That is, $p(\mathbf{x}) = \frac{1}{50}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)$.

Then, it is easy to verify $\max_{\{\mathbf{x}'|x'_1=x_1\}} p(\mathbf{x}') = 4^2/50$ for all $x_1 \in \{0,1\}$. Therefore, we can define max-marginals $\mu_1(x_1) = 1$ for all $x_1 \in \{0,1\}$, i.e. $\max_{\{\mathbf{x}'|x'_1=x_1\}} p(\mathbf{x}') = \frac{4^2}{50}\mu_1(x_1)$ and $\kappa_1 = \frac{50}{4^2}$. Since $\mu_2(x_2)$ and $\mu_3(x_3)$ can be defined similarly, we obtain $\mu_i(x_i) = 1$ for all $x_i \in \{0,1\}$ and $i \in \{1,2,3\}$.

Likewise, we can verify $\max_{\{\mathbf{x}'|(x'_1,x'_2)=(x_1,x_2)\}} p(\mathbf{x}')$ is $4/50$ if $x_1 = x_2$, and $4^2/50$ otherwise. Since we obtain the same result when maximizing under fixed (x_2, x_3) values, we can define $\mu_{ij}(x_i, x_j)$ as

$$\mu_{ij}(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 4 & \text{otherwise} \end{cases} \quad \text{for all } (i,j) \in (1,2),(2,3). \tag{5}$$

i.e. $\max_{\{\mathbf{x}'|(x'_i,x'_j)=(x_i,x_j)\}} p(\mathbf{x}') = \frac{4}{50}\mu_{ij}(x_i,x_j)$ and $\kappa_{ij} = \frac{50}{4}$.

In this example, we realize $\mu_i(x_i) = \psi_i(x_i)$ and $\mu_{ij}(x_i,x_j) = \psi_{ij}(x_i,x_j)$ for all i,j, and also $\mu_{ij}(x_i,x_j) = \psi_{ij}(x_i,x_j)\psi_i(x_i)\psi_j(x_j)$. This makes us easily verify that $p(\mathbf{x})$ is factorized by max-marginals:

$$\begin{aligned} p(\mathbf{x}) &= \frac{1}{50}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3) \\ &\quad \times \frac{\psi_{12}(x_1,x_2)\psi_1(x_1)\psi_2(x_2)}{\psi_1(x_1)\psi_2(x_2)} \frac{\psi_{23}(x_2,x_3)\psi_2(x_2)\psi_3(x_3)}{\psi_2(x_2)\psi_3(x_3)} \quad (6) \\ &= \frac{1}{50}\mu_1(x_1)\mu_2(x_2)\mu_3(x_3) \frac{\mu_{12}(x_1,x_2)}{\mu_1(x_1)\mu_2(x_2)} \frac{\mu_{23}(x_2,x_3)}{\mu_2(x_2)\mu_3(x_3)}. \quad (7) \end{aligned}$$

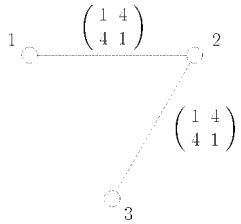

Figure 5. The diagram shows the graphical model and pairwise compatibility functions $\psi_{12}(x_1,x_2)$ and $\psi_{23}(x_2,x_3)$ of the distribution used in Example 1.

Now, assume that we are given $p(\mathbf{x})$ for every $\mathbf{x}$, and the max-marginals $\{\mu_i, \mu_{ij}\}$. We illustrate how max-marginals can be used to compute $\max_{\mathbf{x}} p(\mathbf{x})$. We know $p(\mathbf{x}) = \frac{1}{Y}\mu_1(x_1)\mu_2(x_2)\mu_3(x_3) \frac{\mu_{12}(x_1,x_2)}{\mu_1(x_1)\mu_2(x_2)} \frac{\mu_{23}(x_2,x_3)}{\mu_2(x_2)\mu_3(x_3)}$ for some Y. The value of Y can be easily computed by comparing both sides of the equation for some specific assignment, e.g. $(0,0,0)$. In this example, we obtain $Y = 50$ as shown in eq. (7). Assuming $\mathbf{x}^*$ is a MAP assignment, we have

$$\begin{aligned} \max_{\mathbf{x}} p(\mathbf{x}) = p(\mathbf{x}^*) = \frac{1}{50}\mu_1(x_1^*)\mu_2(x_2^*)\mu_3(x_3^*) \frac{\mu_{12}(x_1^*,x_2^*)}{\mu_1(x_1^*)\mu_2(x_2^*)} \\ \times \frac{\mu_{23}(x_2^*,x_3^*)}{\mu_2(x_2^*)\mu_3(x_3^*)}. \quad (8) \end{aligned}$$

Since we know x_i^* and (x_i^*, x_j^*) should be a maximizer of $\mu_i(x_i)$ and $\mu_{ij}(x_i,x_j)$, respectively, the maximum value of $p(\mathbf{x})$ can be obtained simply by finding the maximum value of each $\mu_i(x_i)$ and $\mu_{ij}(x_i,x_j)$ without needing to find the actual assignment $\mathbf{x}^*$. Therefore, $\max_{\mathbf{x}} p(\mathbf{x}) = 4^2/50$.

Max-marginals are also useful in finding a MAP assignment for a tree distribution .[50] We can easily determine a MAP assignment value for the root node of the tree by finding a value that maximizes the singleton max-marginals of the root. Then, the MAP assignment is determined for the rest of the nodes in the order of tree traversal from the root to leaves; for each pair of parent and child nodes and a given assignment for the parent node, the child node assignment is a value that maximizes the corresponding pairwise max-marginals.

For a distribution over a nontree (cyclic) graphical model, knowing the exact max-marginals does not necessarily imply a MAP assignment or the maximum value of $p(\mathbf{x})$ can be easily found. There are special cases that allow efficient computation of MAP assignments for cyclic distributions using max-marginals. For example, when each singleton max-marginals factor has a unique maximizer, the assignment consisting of these maximizers is the unique MAP assignment. More generally, an assignment that maximizes every max-marginals factor of the distribution is a MAP assignment .[47] Such an assignment can be more efficiently found by restricting the search to a subgraph derived from singleton factors that have multiple maximizers .[38] However, this search is still very large in case there are many maximizers of each singleton max-marginals factor and the subgraph is densely connected.

The ordinary max-product (also known as max-plus or min-sum) algorithm [49] is an iterative algorithm that estimates a MAP

assignment by propagating a series of messages along the edges of the graphical model. The algorithm exactly computes a MAP assignment for tree distributions, but it does not guarantee finding one for cyclic distributions. It is known that the ordinary max-product algorithm applied to a tree distribution can be interpreted as computing max-marginals exactly and efficiently .[50] For general cyclic distributions, there is no known method that efficiently computes max-marginals; it can be as expensive as the original MAP estimation problem.

Pseudo-Max-Marginals

Instead of attempting to compute max-marginals, Wainwright et al. [47] used the notion of pseudo-max-marginals in their tree-reweighted max-product (message-passing) algorithm. Pseudo-max-marginals are defined so that they become max-marginals for each tree distribution used in the algorithm, and the original distribution is represented as a convex combination of these tree distributions.

The basic idea of the tree-reweighted max-product algorithm is to express a cyclic distribution as a convex combination of distributions over a set of spanning trees. This convex combination of tree distributions is used to upper bound the MAP probability, that is, to lower bound the energy. It can be shown that the upper bound is tight if and only if every tree distribution shares a common MAP configuration, i.e. tree agreement .[47] The tree-reweighted max-product algorithm tries to induce this tree agreement by factorizing each tree distribution with factors called pseudo-max-marginals and having pseudo-max-marginals converge to the max-marginals of each tree distribution.

Let us assume we use the tree-reweighted max-product algorithm with $\mathcal{T}$, a set of spanning trees of $\mathcal{G}$, and some non-negative constant $\rho(T)$ for each $T \in \mathcal{T}$ such that $\sum_{T\in\mathcal{T}} \rho(T) = 1$. The tree-reweighted max-product algorithm requires that every vertex and edge of $\mathcal{G}$ be covered by $\mathcal{T}$, i.e. each vertex and edge in $\mathcal{G}$ is in some tree T in $\mathcal{T}$ such that $\rho(T) > 0$. Then, by construction, pseudo-max-marginals $\nu = \{\nu_i, \nu_{ij}\}$ from the tree-reweighted max-product algorithm satisfy "ρ-reparameterization", that is described as:

$$p(\mathbf{x}) \propto \prod_{T\in\mathcal{T}} \left[\prod_{i\in\mathcal{V}(T)} \nu_i(x_i) \prod_{(i,j)\in\mathcal{E}(T)} \frac{\nu_{ij}(x_i,x_j)}{\nu_i(x_i)\nu_j(x_j)} \right]^{\rho(T)} = \prod_{i\in\mathcal{V}} \nu_i(x_i)^{\rho_i} \prod_{(i,j)\in\mathcal{E}} \left[\frac{\nu_{ij}(x_i,x_j)}{\nu_i(x_i)\nu_j(x_j)} \right]^{\rho_{ij}}, \quad (9)$$

where ρ_{ij} is an edge coefficient such that $\rho_{ij} = \sum_{T\in\mathcal{T}:(i,j)\in\mathcal{E}(T)} \rho(T)$ defined for all $(i,j) \in \mathcal{E}$, and ρ_i is a vertex coefficient such that $\rho_i = \sum_{T\in\mathcal{T}:i\in\mathcal{V}(T)} \rho(T)$ defined for all $i \in \mathcal{V}$. Note that, if $\mathcal{T}$ is a set of spanning trees, then ρ_i is 1 for all $i \in \mathcal{V}$.

A tree distribution $p^T(\mathbf{x};\nu)$ for some $T \in \mathcal{T}$ and given pseudo-max-marginals can be defined as

$$p^T(\mathbf{x};\nu) = \prod_{i\in\mathcal{V}(T)} \nu_i(x_i) \prod_{(i,j)\in\mathcal{E}(T)} \frac{\nu_{ij}(x_i,x_j)}{\nu_i(x_i)\nu_j(x_j)}. \quad (10)$$

Then, we have $p(\mathbf{x}) \propto \prod_{T\in\mathcal{T}} \{p^T(\mathbf{x};\nu)\}^{\rho(T)}$ from eq. (9). The pseudo-max-marginals ν^* at convergence of the tree-reweighted max-product algorithm satisfy the "tree-consistency condition" with respect to every tree $T \in \mathcal{T}$. That is, the pseudo-max-marginals converge to the max-marginals of each tree distribution.

Example 2. (Pseudo-max-marginals) .[47] Let $\mathbf{x} \in \{0,1\}^3$ be a random vector on a graphical model illustrated in Figure 6a. Let $p(\mathbf{x}) = \frac{1}{98}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{31}(x_3,x_1)$, where $\psi_i(x_i)$ and $\psi_{ij}(x_i,x_j)$ are defined same as in Example 1. We define pseudo-max-marginals $\hat{\nu}$ as follows:

$$\hat{\nu}_i(x_i) = 1, \text{ for all } x_i \in \{0,1\} \text{ and } i \in \{1,2,3\}, \quad (11)$$

$$\hat{\nu}_{ij}(x_i,x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 8 & \text{otherwise} \end{cases} \quad \text{for all } (i,j) \in \{(1,2),(2,3),(3,1)\}. \quad (12)$$

Figures 6b–6d illustrates the trees used for the convex combination and pseudo-max-marginals on each tree. It can be easily verified that pseudo-max-marginals on each tree are in fact max-marginals. Thus, the pseudo-max-marginals are tree-consistent. The distribution for each tree is given by eq. (10). For example, the distribution for Figure 6b is

$$p^1(\mathbf{x};\hat{\nu}) = \hat{\nu}_1(x_1)\hat{\nu}_2(x_2)\hat{\nu}_3(x_3)\frac{\hat{\nu}_{12}(x_1,x_2)}{\hat{\nu}_1(x_1)\hat{\nu}_2(x_2)}\frac{\hat{\nu}_{23}(x_2,x_3)}{\hat{\nu}_2(x_2)\hat{\nu}_3(x_3)}. \quad (13)$$

Then, by letting $\rho(T) = 1/3$ for all three trees, we obtain

$$\frac{1}{98}p^1(\mathbf{x};\hat{\nu})^{1/3}p^2(\mathbf{x};\hat{\nu})^{1/3}p^3(\mathbf{x};\hat{\nu})^{1/3} = \frac{1}{98}\psi_1(x_1)\psi_2(x_2)\psi_3(x_3)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{31}(x_3,x_1) = p(\mathbf{x}), \quad (14)$$

from $\psi_i(x_i) = \hat{\nu}_i(x_i)^{-1/3}$ and $\psi_{ij}(x_i,x_j) = \hat{\nu}_{ij}(x_i,x_j)^{2/3}$. This verifies the pseudo-max-marginals satisfy ρ-reparameterization as well.

TRMP

Algorithm 1 in Appendix A describes "edge-based reparameterization updates" [47] defining $\mathcal{T}$ as a set of (not necessarily spanning) trees in $\mathcal{G}$, as used by Kolmogorov .[51] Hereafter, we will call this algorithm TRMP in short. Note that, although we define $\mathcal{T}$ as a set of general trees covering all vertices and edges of $\mathcal{G}$, it can be easily verified that all the analyses done by Wainwright et al. [47] can be applied to TRMP in exactly the same way, to show TRMP has the same properties owned by the original edge-based reparameterization updates.

TRMP can sometimes guarantee the optimality of an assignment found at convergence for cyclic distributions. Even if TRMP does not find the exact MAP assignment, we can easily compute the exact maximum value for each tree distribution at TRMP convergence as pseudo-max-marginals converge to max-marginals for each tree distribution. Then, we can combine these to get an upper bound for

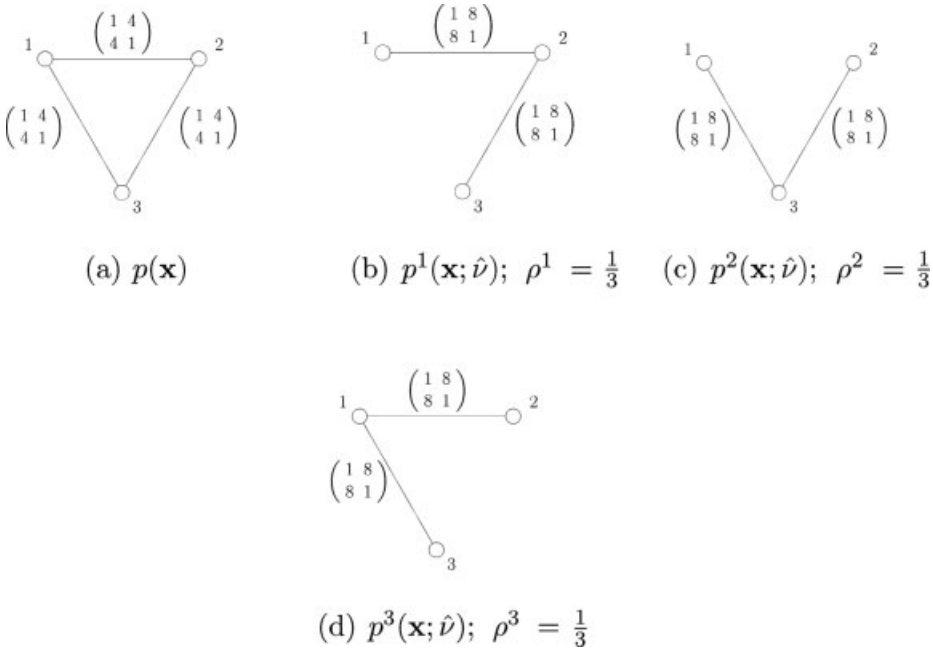

Figure 6. Illustration of pseudo-max-marginals and ρ-reparameterization. (a) Original distribution. (b)–(d) Pseudo-max-marginals on each tree used by convex combination. (a) $p(\mathbf{x})$ (b) $p^1(\mathbf{x};\hat{\nu})$; $\rho^1 = \frac{1}{3}$. (c) $p^2(\mathbf{x};\hat{\nu})$; $\rho^2 = \frac{1}{3}$. (d) $p^3(\mathbf{x};\hat{\nu})$; $\rho^3 = \frac{1}{3}$.

the original, cyclic distribution (thereby obtaining a lower bound on the energy).

We are free to choose any set of trees $\mathcal{T}$ and $\rho(\cdot)$ as long as each vertex and edge is covered by some $T \in \mathcal{T}$ with $\rho(T) > 0$. In this work, we consistently use a set of maximal stars $\mathcal{S}$ in place of $\mathcal{T}$ for the convenience of implementation and the simplicity in computing rotamer-pair lower bounds. A star is a tree where at most one vertex is not a leaf. We denote the center of star S as $\gamma(S)$. A maximal star is a star that is not a subset of another star. Figure 7 illustrates covering a graph by a set of maximal stars; all vertices and edges of graph (a) are covered by $\mathcal{S}$ consisting of three maximal stars. In general, covering dense graphs such as complete graphs requires $\mathcal{O}(n)$ maximal stars. As explained in Lemma 3, computing a rotamer-pair lower bound involves solving a constrained maximization problem for each tree distribution. Therefore, using $\mathcal{S}$ allows us to address only $\mathcal{O}(n)$ maximization problems in computing a rotamer-pair lower bound. In addition, because of the structure of a star, maximization of each tree (i.e. star) distribution can be simplified to one of the four cases of eq. (21).

Following the terminology of Kolmogorov ,[51] we say ν is in a normal form if it satisfies $\max_{r\in R_i} \nu_i(r) = 1$ for all $i \in \mathcal{V}$, and $\max_{(r,s)\in R_i\times R_j} \nu_{ij}(r,s) = 1$ for all $(i,j) \in \mathcal{E}$. Hereafter, we assume ν of Algorithm 1 is always in a normal form. Then, from eq. (2) and (9), and by introducing a positive constant ν_c, we obtain the following equation:

$$\exp\{-e(\mathbf{x})\} = \nu_c \prod_{S\in\mathcal{S}} \{p^S(\mathbf{x};\nu)\}^{\rho(S)}. \tag{15}$$

The value of ν_c can be computed by comparing both sides of eq. (15) for any assignment $\mathbf{x} \in \mathcal{X}$. Equivalently, $p(\mathbf{x})$ can be expressed as follows:

$$p(\mathbf{x}) = \frac{\nu_c}{Z} \prod_{S\in\mathcal{S}} \{p^S(\mathbf{x};\nu)\}^{\rho(S)}. \tag{16}$$

Bounding the GMEC Energy with TRMP

We also make heuristic use of TRMP to obtain upper bounds for the GMEC energy. At convergence of TRMP, we occasionally find an exact MAP configuration. TRMP provides an easy evaluation condition called optimum specification (OS) criterion such that an assignment is guaranteed to be a MAP configuration if it satisfies the OS criterion. However, such an assignment may not exist for a given reparameterization or it could be computationally expensive to find. Therefore, in our upper bounding, instead of trying to find an assignment that satisfies the OS criterion, we simply find an assignment that maximizes the tree distribution for some star $S \in \mathcal{S}$ at TRMP convergence, using dynamic programming .[50] Another possible upper-bounding method is to randomly pick a maximizer for each singleton max-marginals at TRMP convergence regardless of the trees. Although neither of these procedures guarantees the quality of the upper bounds, the resulting upper bounds are empirically close to the optimal values. The procedures can be repeated for different trees or different random selection of maximizers to improve the upper bounds.

A lower bound for the GMEC energy $\min_{\mathbf{x}} e(\mathbf{x})$ can be easily obtained at the convergence of TRMP with the following lemma:

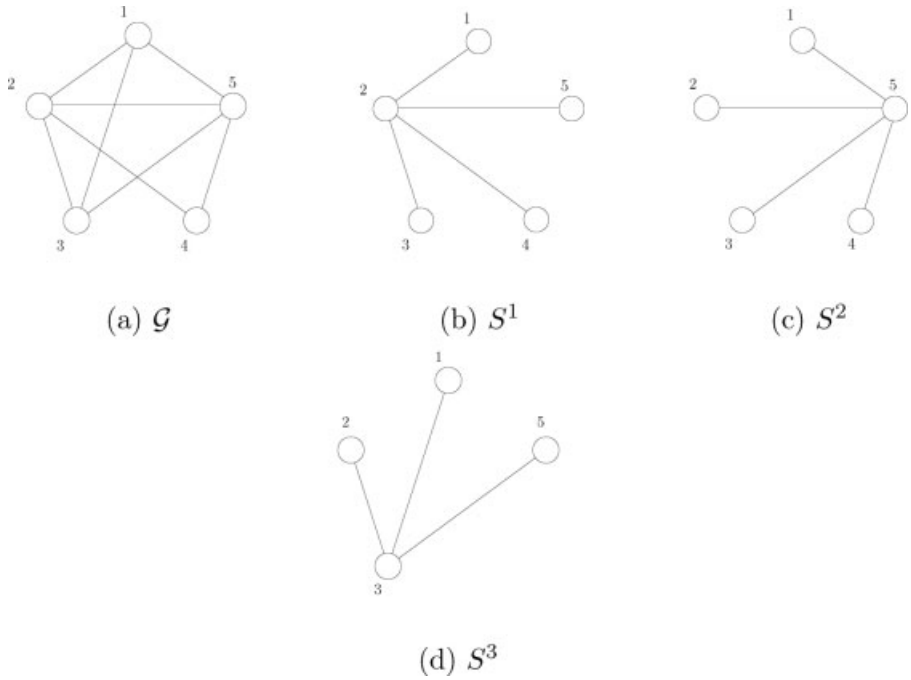

Figure 7. Example of covering a graph by maximal stars: $\mathcal{G}$ of (a) is completely covered by S^1, S^2, and S^3. (a) $\mathcal{G}$, (b) S^1, (c) S^2, (d) S^3.

Lemma 1. *When ν and ν_c of* (16) *in a normal form satisfy the tree-consistency condition, the MAP probability is upper bounded by*

$$\max_{\mathbf{x}} p(\mathbf{x}) \leq \frac{\nu_c}{Z}. \tag{17}$$

Therefore, the GMEC energy $\min_{\mathbf{x}} e(\mathbf{x})$ is lower bounded by $\min_{\mathbf{x}} e(\mathbf{x}) \geq -\ln \nu_c$ from (2). This lower bound of the GMEC energy is independent of the normalization constant Z because, in (15), the product $\prod_{S\in\mathcal{S}} p^S(\mathbf{x};\nu)$ purely depends on the normalized pseudo-max-marginals, that are generated without any reference to Z. Note that Lemma 1 is true not only for star covers but for general tree covers.

Example 3. To upper bound $\max_{\mathbf{x}} p(\mathbf{x})$ using Lemma 1 and the pseudo-max-marginals given in Example 2, we first need to normalize pairwise pseudo-max-marginals. Since the maximum value of $\hat{\nu}_{ij}(x_i, x_j)$ for all (i,j) are 8, normalized pairwise pseudo-max-marginals are as follows:

$$\nu_{ij}(x_i, x_j) = \begin{cases} 1/8 & \text{if } x_i = x_j \\ 1 & \text{otherwise} \end{cases} \quad \text{for all } (i,j) \in \{(1,2),(2,3),(3,1)\}. \tag{18}$$

Singleton pseudo-max-marginals are already in a normal form. Given the normalized pseudo-max-marginals, and $p(\mathbf{x})$ for every $\mathbf{x}$, we can easily compute $\nu_c/Z = 64/98$ from (16) due to the small-scale nature of this toy problem. Then, by Lemma 1, the upper bound of the MAP probability is 64/98. It is easy to see $\max_{\mathbf{x}} p(\mathbf{x})$ is equal to 16/98 attained by any of $(x_1, x_2, x_3) = (0,0,1), (0,1,0)$, etc. The upper bound of the MAP probability (thereby the resulting lower bound of the GMEC energy) is not tight in this example, but the quality of bounds from Lemma 1 can be stronger depending on pseudo-max-marginals from TRMP. In this example, on the other hand, a tight lower-bound of $p(\mathbf{x})$ (therefore a tight upper-bound of the GMEC energy) is easily obtained by finding a MAP assignment for any of the trees in $\mathcal{T}$. For instance, $(x_1, x_2, x_3) = (0,1,0)$ is a MAP assignment for tree distribution $p^1(\mathbf{x};\nu)$, and also for $p(\mathbf{x})$.

Elimination by TRMP Lower Bounds

We can exploit the tree-consistency of ν at TRMP convergence in computing various lower bounds for a set of conformations. If a lower bound greater than a global upper-bound U is obtained, we can eliminate corresponding conformations from the subproblem while conserving the inequality relation between the minimum energy of the subproblem and U. We make a more precise argument for what we call rotamer-pair elimination and rotamer elimination as follows. Let $\tilde{P}$ be the set of flagged rotamer pairs in the subproblem of our interest. Then, given conformational space $\mathcal{X}$, we define $\mathcal{L}(\mathcal{X}, \tilde{P})$ as the set of all legal conformations containing no flagged rotamer pairs.

1. Rotamer-pair elimination: Suppose we have a lower-bound $\mathrm{LB}(\zeta_r, \eta_s)$ of the minimum conformational energy for $\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}$, the set of all conformations including (ζ_r, η_s), such that $\min_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} e(\mathbf{x}) \geq \mathrm{LB}(\zeta_r, \eta_s) > U$. Elimination of (ζ_r, η_s) can be represented by the set of pair-flags $\tilde{P}' = \tilde{P} \cup (\zeta_r, \eta_s)$. We know $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P}')} e(\mathbf{x})$ is prunable if and

only if $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P})} e(\mathbf{x})$ is prunable. Therefore, we use $\tilde{P}'$ as the updated set of pair flags.

2. Rotamer elimination: Suppose we have a lower-bound $\mathrm{LB}(\zeta_r)$ of the minimum conformational energy for $\{\mathbf{x}|x_\zeta = r\}$, the set of all conformations including ζ_r, such that $\min_{\{\mathbf{x}|x_\zeta=r\}} e(\mathbf{x}) \geq \mathrm{LB}(\zeta_r) > U$. Elimination of ζ_r can be represented by the set of pair-flags $\tilde{P}' = \tilde{P} \cup \{(\zeta_r, j_s)|s \in R_j, j \in \mathcal{V}, j \neq \zeta\}$, which includes all rotamer pairs stemming from ζ_r. Again, we know $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P}')} e(\mathbf{x})$ is prunable if and only if $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P})} e(\mathbf{x})$ is prunable. Therefore, we use $\tilde{P}'$ as the updated set of pair flags.

In both cases, the optimal value of $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P})} e(\mathbf{x})$ does not change if $\min_{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P})} e(\mathbf{x}) \leq U$.

The lower-bounds $\mathrm{LB}(\zeta_r)$ and $\mathrm{LB}(\zeta_r, \eta_s)$ can be, for example, obtained by directly solving an LP relaxation of the ILP given in Appendix B. However, solving LP may not be practical when the problem size is large. In addition, solving LP for every rotamer or rotamer pair will multiply the lower-bounding time by the number of rotamers or rotamer pairs. Here, we use upper-bounding inequalities for the singleton and pairwise max-marginals to obtain lower bounds for minimum conformational energies of rotamers and rotamer pairs. Such lower bounds are at best as tight as the bounds from solving the LP discussed in Appendix B ,[47] but requires computation time for one TRMP run until convergence (no guaranteed time bound) plus post-processing time at most cubic of the number of rotamers. The rest of this section explains how we can efficiently compute the rotamer and rotamer-pair lower bounds.

We have the following lemma on upper-bounding the singleton max-marginals:

Lemma 2. *When ν and ν_c of eq.* (16) *in a normal form satisfy the tree-consistency condition, it is true for all $r \in R_\zeta$, $\zeta \in \mathcal{V}$ that*

$$\max_{\{\mathbf{x}|x_\zeta=r\}} p(\mathbf{x}) \leq \frac{\nu_c}{Z} \nu_\zeta(r)^{\rho_\zeta}. \tag{19}$$

Example 4. From Lemma 2 and the normalized pseudo-max-marginals given in Example 3, we find an upper bound for the maximum probability of $p(\mathbf{x})$ when $x_1 = 0$ as $(\nu_c/Z)\nu_1(0)^{1/3} = 64/98 \times 1^{1/3}$. The bound is not tight because $\max_{\{\mathbf{x}|x_1=0\}} p(\mathbf{x}) = 16/98$, but the tightness may change depending on the pseudo-max-marginals from TRMP. Even when the resulting bound is not tight, it could be still strong enough to eliminate the corresponding rotamer through comparison against a global upper-bound U.

Lemma 2 combined with eq. (2) provides a rotamer lower-bound $\mathrm{LB}(\zeta_r)$ for each $r \in R_\zeta$ and $\zeta \in \mathcal{V}$ as $\min_{\{\mathbf{x}|x_\zeta=r\}} e(\mathbf{x}) \geq \mathrm{LB}(\zeta_r) = -\ln \nu_c - \rho_\zeta \ln \nu_\zeta(r)$.

To upper bound the pairwise max-marginals, we use the general inequality

$$\max_{\{\mathbf{x}|x_\zeta=r,x_\eta=s\}} p(\mathbf{x}) \leq \frac{\nu_c}{Z} \prod_{S\in\mathcal{S}} \left[\max_{\{\mathbf{x}|x_\zeta=r,x_\eta=s\}} p^S(\mathbf{x};\nu) \right]^{\rho(S)}. \tag{20}$$

The maximization problem $\max_{\{\mathbf{x}|x_\zeta=r,x_\eta=s\}} p^S(\mathbf{x};\nu)$ can be easily solved using the following lemma:

Lemma 3. *When ν and ν_c of eq.* (16) *in a normal form satisfy the tree-consistency condition,*

$$\max_{\{\mathbf{x}|x_\zeta=r,x_\eta=s\}} p^S(\mathbf{x};\nu) = \begin{cases} 1 & \text{if } \zeta, \eta \notin \mathcal{V}(S) \\ \nu_\zeta(r) & \text{if } \zeta \in \mathcal{V}(S) \text{ and } \eta \notin \mathcal{V}(S) \\ \nu_{\zeta\eta}(r,s) & \text{if } (\zeta,\eta) \in \mathcal{E}(S) \\ \max_{x_\xi \in R_\xi} \frac{\nu_{\xi\zeta}(x_\xi, r)\nu_{\xi\eta}(x_\xi, s)}{\nu_\xi(x_\xi)} & \text{else (let } \xi = \gamma(S)) \end{cases} \tag{21}$$

Example 5. Let us bound $\max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p(\mathbf{x})$ using the normalized pseudo-max-marginals given in Example 3. As discussed above, we have to solve maximization problem for each star:

1. $p^1(\mathbf{x};\nu)$ and $p^3(\mathbf{x};\nu)$ (Figures 6b and 6d): this corresponds to the third case of eq. (21). Therefore, $\max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p^1(\mathbf{x};\nu) = \max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p^3(\mathbf{x};\nu) = \nu_{12}(0,0) = 1/8.$
2. $p^2(\mathbf{x};\nu)$ (Figure 6c): this corresponds to the fourth case of eq. (21). Therefore,

$$\max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p^2(\mathbf{x};\nu) = \max_{x_3} \frac{\nu_{3,1}(x_3,0)\nu_{3,2}(x_3,0)}{\nu_3(x_3)} = 1. \tag{22}$$

By combining the above results in eq. (20), we obtain

$$\max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p(\mathbf{x}) \leq (64/98) \times (1/8)^{1/3} \times (1/8)^{1/3} \times 1^{1/3} = 16/98. \tag{23}$$

This bound is tight from $\max_{\{\mathbf{x}|(x_1,x_2)=(0,0)\}} p(\mathbf{x}) = 16/98$ attained by $x_3 = 1$. Note that the same pseudo-max-marginals that yielded weak upper bounds in Examples 3 and 4, led to a tight upper bound for the rotamer pair, a more constrained bounding problem.

$\mathrm{LB}(\zeta_r, \eta_s)$, a lower bound for the minimum conformation energy of rotamer-pair (ζ_r, η_s), is given by $\mathrm{LB}(\zeta_r, \eta_s) = -\ln \nu_c - \sum_{S\in\mathcal{S}} \rho_S \ln \max_{\{\mathbf{x}|x_\zeta=r,x_\eta=s\}} p^S(\mathbf{x};\nu)$.

Note that there can be at most $\mathcal{O}(n)$ stars that correspond to the fourth case of eq. (21) for each position pair (ζ, η). If we let n_{rot} be the average number of rotamers per position, the maximization problem corresponding to the fourth case of eq. (21) requires $\mathcal{O}(n_{\mathrm{rot}})$ operations. Therefore, it will take $\mathcal{O}(n_{\mathrm{rot}}n)$ post-processing operations to compute an upper bound for each rotamer pair using Lemma 3, and $O(n_{rot}^3 n^3)$ for all rotamer pairs.

In computing the rotamer lower bound for a rotamer ζ_r, we can also use pair-flags information to obtain a lower bound, $\mathrm{LB}'(\zeta_r)$, for the constrained problem $\min_{\{\mathbf{x}\in\mathcal{L}(\mathcal{X},\tilde{P})|x_\zeta=r\}} e(\mathbf{x})$. If we have $\mathrm{LB}'(\zeta_r) > U$, then conformations, $\{\mathbf{x} \in \mathcal{L}(\mathcal{X},\tilde{P})|x_\zeta = r\}$ can be excluded from the search space. This is equivalent to eliminating rotamer ζ_r because all conformations containing $x_\zeta = r$ are in effect excluded. Computing $\mathrm{LB}'(\zeta_r)$ will take additional polynomial time compared to $\mathrm{LB}(\zeta_r)$, but it is particularly advantageous to leverage the pair flags when there exist a large number of flagged rotamer pairs. We used a simple search-based method to compute $\mathrm{LB}'(\zeta_r)$

as follows; let $\hat{p} = \prod_{S \in \mathcal{S}} [\max_{\{\mathbf{x} \in \mathcal{L}(\mathcal{X}, \tilde{P}) | x_\zeta = r\}} p^S(\mathbf{x}; \nu)]^{\rho(S)}$ for tree-consistent ν in a normal form. Then, it is easy to see $(\nu_c/Z)\hat{p}$ is an upper bound of $\max_{\{\mathbf{x} \in \mathcal{L}(\mathcal{X}, \tilde{P}) | x_\zeta = r\}} p(\mathbf{x})$. To compute $\hat{p}$, we first build a table for $\max_{x_j} \nu_{i,j}(r, x_j)$ for every $(i, j) \in \mathcal{E}$, and $r \in R_i$. This takes $\mathcal{O}(n_{rot}^2 n^2)$ time. Using this table, it takes $\mathcal{O}(n_{rot} n)$ additions to compute $\max_{\{\mathbf{x} \in \mathcal{L}(\mathcal{X}, \tilde{P}) | x_\zeta = r\}} p^S(\mathbf{x}; \nu)$; when the rotamer of $\gamma(S)$ is fixed, it takes $\mathcal{O}(n)$ additions to find the maximum value of $p^S(\mathbf{x}; \nu)$, and there are $\mathcal{O}(n_{rot})$ rotamers in $\gamma(S)$. Therefore, it takes $\mathcal{O}(n_{rot} n^2)$ time to exactly compute $\hat{p}$. Finally, the rotamer lower bound is computed as $\mathrm{LB}'(\zeta_r) = -\ln \nu_c - \ln \hat{p}$. To compute $LB'(\zeta_r)$ for all $\zeta \in \mathcal{V}$, and $r \in R_\zeta$, it will take $\mathcal{O}(n_{rot}^2 n^3)$ post-processing time.

Overall, lower-bound computation for all rotamers and rotamer pairs requires post-processing time of $\mathcal{O}(n_{rot}^3 n^3)$.

Results and Discussions

We performed computational experiments to evaluate the performance of BroMAP. We used a set of various protein design cases to measure and compare the running times of BroMAP and a fast implementation of DEE/A^* that includes most of the state-of-art techniques .[26] In the following, to distinguish the modified version of DEE used in BroMAP from the DEE used in DEE/A^*, we will call the former as DEE-gp (DEE for general pair flags). The two main questions we are interested in investigating with the experiments are (1) whether BroMAP can solve design cases previously unsolved by DEE/A^*, and (2) whether we can use BroMAP generally as an alternative to DEE/A^* without being restricted to specific types of design cases. Although we are mainly interested in the overall performance of BroMAP here, Hong and Lozano-Pérez [52] evaluate the effectiveness of our pruning method by comparing it against linear programming.

DEE/A Implementation*

We used an in-house implementation of DEE/A^* written in the C programming language .[53] DEE/A^* was performed with the following options and order:

1. Eliminate singles using Goldstein's condition .[21] Repeat until elimination is unproductive.
2. Eliminate singles using split flags ($s = 1$) .[24] Repeat until elimination is unproductive.
3. Do logical singles-pairs elimination.[22]
4. Eliminate pairs using Goldstein's condition with one magic bullet.[23]
5. Do logical singles-pairs elimination.
6. If unification is possible, do unification ,[21] and go to (1).
7. Do A^*.[46]

For unification, the pair of positions that has the largest fraction of flagged rotamer pairs is picked. However, because the energy terms and pair flags must be stored in machine memory, we capped the total number of rotamers that would result to be no greater than a unification option C_{uni}. Therefore, any pair of positions that would create a larger number of rotamers when unified than C_{uni} was not considered, and the pair with the next-largest fraction of flagged rotamer pairs was considered. We experimented with different values of C_{uni}, i.e. 6000, 8000, 10,000, 12,000, and 14,000, to obtain the best running time for each test case. Note that this gives DEE/A^*, the competing method an advantage over BroMAP in comparing their running times, because it will give better DEE/A^* times than consistently using one of the C_{uni} values. Increasing C_{uni} and thus the allowance for large unification can facilitate solving otherwise difficult or unsolvable cases. However, for small to medium cases, larger values of C_{uni} often result in slower solution times.

Our DEE implementation uses a full table of energies. That is, if there are $q = \sum_{i=1}^{n} |R_i|$ rotamers in the problem, DEE allocates memory for q^2 floating point numbers.

When the DEE/A^* procedure described above using various C_{uni} values failed to solve a test case, we also tried singles-elimination using split flags with $s = 2$ instead of $s = 1$, or allowed the number of magic bullets to increase up to the number of positions.

BroMAP Implementation

BroMAP was implemented in C++. We used the PICO-library [54] for the BnB framework. The PICO-library provides the data structures and methods to create/delete nodes and to search the tree. It also provides procedure skeletons, for instance, for upper/lower-bounding methods.

In BroMAP, we restricted the amount of effort spent by DEE-gp instead of allowing it to keep iterating singles/pairs-flagging and unification until it finally solved the subproblem. This was done by limiting the maximum number of iterations of singles/pairs-flagging and also by using a smaller fixed C_{uni} value for unification than those used by DEE/A^*.

Other than performing *DEE-gp* and TRMP bounding for each subproblem, we also allowed rotamer-contractions .[52] Rotamer-contraction reduces the size of a subproblem by grouping similar rotamers at a residue position as a cluster and replacing the cluster by a new single rotamer. It also defines the pairwise energies for the new rotamer so that the optimal value of the reduced subproblem is always a lower bound of the optimal value of the subproblem before the rotamer-contraction. Rotamer-contraction was iteratively performed until we obtained a lower bound greater than U or the number of executed rotamer-contractions reached a pre-determined limit. We used a heuristic boundability index (BI) multiplied by a positive integer P_{rc} as such limits. The BI for a specific node is equal to the number of 'high' branches on the path from the root to the node. For example, in the search tree of Figure 1, assuming the BI of the root node is equal to 0, BI's are 0 for nodes 1, 3, 5, 7, 9, and 1 for nodes 2, 4, 6, 8, 11, and 2 for node 10. In these experiments, we let $P_{\mathrm{rc}} = 16$ after exploring the overall effect of different values of P_{rc} on running times of BroMAP.

In case rotamer-contractions were performed multiple times in bounding a subproblem as described earlier, we also performed additional DEE-gp and TRMP periodically on the subproblem reduced by rotamer-contractions. After every P_{DEE} consecutive rotamer-contractions, we applied DEE-gp to see if we could solve the reduced problem or only to flag more rotamers or rotamer pairs. TRMP was also run until convergence after every P_{TRMP} consecutive rotamer-contractions to compute a lower bound for the subproblem or to flag rotamers or rotamer-pairs using the TRMP lower bounds. In this experiment, we let $P_{\mathrm{DEE}} = 8$, and $P_{\mathrm{TRMP}} = 16$.

Along the first depth-first dive, that is, until we exactly solve a subproblem for the first time, we performed only DEE-gp, TRMP bounding, and subproblem splitting, once respectively, but did not use any rotamer-contraction. As with DEE/A^*, BroMAP also used

the A^* search algorithm when DEE-gp could not eliminate any more rotamers or rotamer pairs and the subproblem was considered small, i.e. contained less than 200,000 rotamer pairs.

The BroMAP implementation needs to hold TRMP data, whose size is of the order of the number of rotamer pairs. This corresponds to $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|R_i||R_j|$ floating point numbers, and is roughly half the memory required by DEE/A^*. Because BroMAP also performs DEE-gp, it requires additional memory of $(\sum_{i=1}^{n}|R_i|)^2$ floating point numbers for the full DEE energy table. Therefore, the maximum memory requirement of BroMAP is $(\sum_{i=1}^{n}|R_i|)^2 + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|R_i||R_j|$ floating point numbers, which is roughly 1.5 times larger than that of DEE/A^*.

Platform

We used a Linux workstation with two dual-core 2 GHz AMD Opteron 246 processors and 2 Gbytes of memory for the experiment. The C/C++ codes for BroMAP and DEE/A^* were compiled using Intel C/C++ Compiler Version 9.1 for Linux. During compile, OpenMP directives were enabled to parallelize the execution of DEE, DEE-gp, and TRMP over two CPU cores. All other procedures, including A^*, were executed over a single core. Note that BroMAP or DEE/A^* was allowed to use the whole system memory but only one processor at a time.

Test Cases

We used 68 test cases whose energy files are smaller than 300 Mbytes. An energy file contains floating point numbers representing singleton and pairwise energies. We found energy files larger than 300 Mbytes are not handled well with the current implementation of BroMAP on our workstation due to the memory requirement of BroMAP.

We used three different model systems in obtaining test cases:

1. FN3: Derived from protein ^{10}Fn3, the tenth human fibronectin type III domain .[55] It is a 94-residue β-sheet protein with an immunoglobulin-like fold. Besides its natural *in vivo* role, the protein has been engineered as an antibody mimic to bind with high affinity and specificity to arbitrary protein targets.
2. D44.1[56] and D1.3 :[57] antibodies that bind to hen egg-white lysozyme (HEL), though they bind different HEL epitopes. Each has low nanomolar binding affinity, and was originally isolated after murine immunization. For the D1.3 core designs, we redesigned the core of the lysozyme protein, absent of the antibody.
3. EPO: Human erythropoietin (Epo) protein complexed to its receptor (EpoR) .[58] One Epo binds to two EpoR with one high-affinity and one low-affinity binding sites. Our EPO interface designs addressed the high-affinity binding site while our core designs addressed the core of the EpoR from the high-affinity site.

Each case corresponds to one of three types of protein regions:

1. INT: protein–protein binding interface.
2. CORE: protein core, i.e. side chains that are not solvent-exposed.
3. CORE++: protein core plus boundary positions that are partially exposed to solvent.

We varied the types of amino acids offered at design positions of each case as follows:

1. H: Hydrophobic amino acids (A, F, G, I, L, M, W, V).
2. HP: Hydrophobic plus polar amino acids (A, F, G, I, L, M, W, V, H, N, Q, S, T, Y).
3. A: All type of amino acids, excluding proline and cysteine.

For CORE, we used both H and HP, and for CORE++, we used HP (with both neutral tautomers of histidine allowed in each case). For INT, we used A, and allowed both neutral tautomers and the protonated form of histidine. For all designs, if the wild-type amino acid was not part of the library, it was added at that position. For some test cases, the total number of positions in the search was greater than the number of design positions. At these other positions, the native amino-acid type was retained and its conformation was varied.

Each case was made using one of two different rotamer libraries:

1. REG: Standard rotamer library. This is based on the backbone-independent May 2002 library .[59] This library was supplemented with three histidine rotamers for an unsampled ring flip, and two asparagine rotamers to increase sampling of the final dihedral angle rotation.
2. EXP: Expanded rotamer library. This was created by expanding both χ_1 and χ_2 of rotamers in REG by $\pm 10^\circ$. The hydroxyls of serine, threonine, and tyrosine were sampled every 30 degrees. For some INT cases of D1.3, D44.1, and EPO, crystallographic water molecules were allowed conformational freedom. The oxygen atom location was fixed to that of the crystal structure and the hydrogen atoms were placed to create 60 symmetric water molecule rotations.

For all libraries and cases, each crystallographic wild-type rotamer was added in a position-specific manner to the library, using the complete Cartesian representation of the side chain, rather than just the dihedral angles.

The singleton/pairwise energies of rotamers were computed using the energy function of CHARMM PARAM22 all-atom parameter set with no cut-offs for nonbonded interactions and a $4r$ distance-dependent dielectric constant. All energy terms were used (bond, angle, Urey-Bradley, dihedral, improper, Lennard-Jones, and electrostatic). Rotamers that clashed with the fixed protein regions were eliminated during case generation if their singleton energies were greater than the smallest singleton energy at that position by at least 50 kcal/mol. Further details on design methods and test case construction can be found from Lippow et al. [60]

Table 1 lists composition and problem-size information of each test case. Its last column also summarizes the experimental results presented in the following.

Running Time Comparison

Among the 68 cases, BroMAP solved 65 cases within the 7-days allowed time, whereas DEE/A^* solved 51 cases for the same allowed time. There were no cases DEE/A^* solved but BroMAP was not able to solve. The 14 cases solved by BroMAP but not by DEE/A^* suggest that BroMAP can be an alternative to DEE/A^* for hard design cases where DEE/A^* performs poorly.

Table 1. Test Case Facts.

No.	Model	Region	AA	Lib	n	n_D	w	$\sum \lvert R_i \rvert$	Pairs	log*conf*	Solved by
1	fn3	core	HP	REG	14	14	0	743	2.5 E 5	50.2	Limited DEE
2	fn3	core++	HP	REG	20	20	0	1778	1.5 E 6	83.7	Bro & DEE
3	fn3	core++	HP	REG	23	23	0	1894	1.7 E 6	94.1	Bro & DEE
4	fn3	core++	HP	REG	25	25	0	2048	2.0 E 6	102.9	Bro & DEE
5	fn3	core++	HP	REG	27	27	0	2083	2.1 E 6	108.6	Bro & DEE
6	fn3	core	HP	EXP	14	14	0	8774	3.5 E 7	82.4	Limited DEE
7	D44.1	int	A	REG	7	4	0	476	8.5 E 4	21.6	Limited DEE
8	D44.1	int	A	REG	7	7	0	822	2.8 E 5	28.7	Limited DEE
9	D44.1	int	A	REG	8	8	0	965	4.0 E 5	33.4	Bro & DEE
10	D44.1	int	A	REG	9	9	0	1019	4.5 E 5	37.1	Bro & DEE
11	D44.1	int	A	REG	10	10	0	1133	5.6 E 5	40.6	Bro & DEE
12	D44.1	int	A	REG	11	11	0	1376	8.4 E 5	46.4	Bro
13	D44.1	int	A	REG	16	14	2	2020	1.9 E 6	70.1	None
14	D44.1	int	A	EXP	7	4	0	5026	9.5 E 6	36.4	Limited DEE
15	D44.1	int	A	EXP	7	5	0	7019	1.9 E 7	39.9	Bro & DEE
16	D44.1	int	A	EXP	7	6	0	7910	2.6 E 7	42.9	Bro
17	D44.1	int	A	EXP	7	7	0	8771	3.2 E 7	42.9	Bro
18	D1.3	int	A	REG	6	4	2	450	8.3 E 4	21.7	Limited DEE
19	D1.3	int	A	REG	11	8	3	767	2.6 E 5	38.5	Limited DEE
20	D1.3	int	A	REG	23	7	9	1618	1.2 E 6	78.8	Limited DEE
21	D1.3	int	A	EXP	6	4	2	3599	4.8 E 6	28.7	Limited DEE
22	D1.3	int	A	EXP	7	5	2	3616	4.8 E 6	28.7	Limited DEE
23	D1.3	int	A	EXP	8	6	2	4070	6.3 E 6	34.4	Bro & DEE
24	D1.3	int	A	EXP	11	4	3	4612	8.0 E 6	42.6	Bro & DEE
25	D1.3	int	A	EXP	11	6	3	4987	9.7 E 6	45.1	Bro & DEE
26	D1.3	int	A	EXP	11	7	3	5461	1.2 E 7	47.4	Bro & DEE
27	D1.3	int	A	EXP	11	7	3	5891	1.4 E 7	50.5	Bro
28	D1.3	int	A	EXP	11	8	3	6365	1.7 E 7	52.8	Bro
29	D1.3	core	H	REG	16	16	0	342	5.4 E 4	44.1	Limited DEE
30	D1.3	core	H	REG	20	20	0	430	8.6 E 4	54.6	Limited DEE
31	D1.3	core	H	REG	26	26	0	503	1.2 E 5	66.7	Limited DEE
32	D1.3	core	H	REG	34	34	0	567	1.5 E 5	81.4	Limited DEE
33	D1.3	core	HP	REG	16	16	0	980	4.4 E 5	59.5	Bro & DEE
34	D1.3	core	HP	REG	20	20	0	1228	7.1 E 5	74.1	Bro & DEE
35	D1.3	core	HP	REG	26	26	0	1431	9.7 E 5	92.3	Bro & DEE
36	D1.3	core	HP	REG	34	34	0	1582	1.2 E 6	112.7	Bro & DEE
37	D1.3	core	H	EXP	13	13	0	1844	1.5 E 6	56.3	Bro & DEE
38	D1.3	core	H	EXP	16	16	0	2734	3.5 E 6	75.7	Bro
39	D1.3	core	H	EXP	20	20	0	3370	5.3 E 6	91.8	Bro
40	D1.3	core	H	EXP	26	26	0	3894	7.1 E 6	111.6	Bro
41	D1.3	core	H	EXP	34	34	0	4444	9.4 E 6	142.0	Bro
42	epo	int	A	REG	5	5	0	466	7.1 E 4	16.6	Limited DEE
43	epo	int	A	REG	6	6	0	419	6.8 E 4	17.0	Limited DEE
44	epo	int	A	REG	11	11	0	1005	4.4 E 5	39.4	Bro & DEE
45	epo	int	A	REG	21	11	3	1503	1.0 E 6	67.5	Bro & DEE
46	epo	int	A	REG	21	15	3	1999	1.9 E 6	79.6	Bro
47	epo	int	A	REG	21	18	3	2138	2.1 E 6	87.5	None
48	epo	int	A	EXP	5	5	0	5001	8.4 E 6	26.5	Limited DEE
49	epo	int	A	EXP	6	6	0	4170	6.8 E 6	26.3	Bro & DEE
50	epo	int	A	EXP	8	8	0	7544	2.3 E 7	46.4	Bro & DEE
51	epo	int	A	EXP	9	9	0	8724	3.2 E 7	53.4	Bro & DEE
52	epo	core	H	REG	17	17	0	291	3.9 E 4	43.5	Limited DEE
53	epo	core	H	REG	22	22	0	395	7.4 E 4	58.1	Limited DEE
54	epo	core	H	REG	28	28	0	433	8.9 E 4	65.4	Limited DEE
55	epo	core	H	REG	33	33	0	573	1.6 E 5	82.7	Limited DEE
56	epo	core	H	REG	41	41	0	727	2.6 E 5	103.3	Limited DEE

(*continued*)

Table 1. (*Continued*)

57	epo	core	HP	REG	17	17	0	827	3.2 E 5	60.1	Bro & DEE
58	epo	core	HP	REG	22	22	0	1103	5.8 E 5	79.9	Bro & DEE
59	epo	core	HP	REG	28	28	0	1208	7.0 E 5	92.6	Bro & DEE
60	epo	core	HP	REG	33	33	0	1615	1.3 E 6	115.9	Bro & DEE
61	epo	core	HP	REG	36	36	0	1827	1.6 E 6	128.4	Bro
62	epo	core	HP	REG	38	38	0	1956	1.9 E 6	136.6	Bro
63	epo	core	HP	REG	41	41	0	1999	1.9 E 6	143.1	Bro
64	epo	core	H	EXP	17	17	0	2307	2.4 E 6	73.5	Limited DEE
65	epo	core	H	EXP	22	22	0	3006	4.2 E 6	99.0	Bro & DEE
66	epo	core	H	EXP	28	28	0	3213	4.8 E 6	111.1	Bro & DEE
67	epo	core	H	EXP	33	33	0	4322	8.9 E 6	140.0	Bro
68	epo	core	H	EXP	41	41	0	5712	1.6 E 7	175.0	None

Each column represents (1) No.: case number, (2) Model: model system, (3) Region: protein regions being considered, (4) AA: type of amino acids offered for design positions, (5) Lib: types of rotamer library used, (6) n: number of positions, (7) n_D: number of design positions, (8) w: number of mobile water molecules considered, (9) $\sum |R_i|$: total number of rotamers, (10) Pairs: total number of rotamer pairs, (11) log$conf$: $\sum_{i=1}^{n} \log |R_i|$, (12) Solved by: methods that solved the case ("Limited DEE" implies the case was solved by both BroMAP and DEE/A^*, but only DEE-gp was necessary for BroMAP. "Bro" and "DEE" abbreviate BroMAP and DEE/A^*, respectively).

Among the 51 cases solved by both BroMAP and DEE/A^*, solving 23 cases by BroMAP required only the DEE-gp part of BroMAP. As BroMAP only acted as a light DEE for these cases, comparing the running times of BroMAP and DEE/A^* on them is not meaningful. After eliminating these 23 cases, we are left with 28 cases for which we are interested in comparing the running times of BroMAP and DEE/A^*. The running times for these 28 cases are shown in Table 2. Additionally, the table lists results for 14 cases that only BroMAP solved.

Figure 8 plots the ratio of BroMAP running time to DEE/A^* running time vs. DEE/A^* running time. Note that the plotted ratios for cases solved only by BroMAP are upper bounds on actual ratios because actual DEE/A^* running times should be more than 7 days. Overall, the plot suggests BroMAP gains advantage for cases as DEE/A^* takes longer. For all cases that DEE/A^* took more than one hour to solve, the maximum ratio was 0.33. Together with the 14 cases solved by BroMAP only, the experiment supports that BroMAP can be an alternative to DEE/A^* for hard design cases. There are 5 cases for which the BroMAP solution time is at least 10% longer than DEE/A^* solution time. Considering four of them (cases 45, 58, 65, and 66) were almost ideally solved by BroMAP (the GMEC was found at the end of the first depth-first dive and there was no branching after the first depth-first dive), we find more aggressive DEE conditions such as larger C_{uni} were critical in obtaining shorter running times on them. In terms of the total running time, however, none of these five cases needed more than 130 min to be solved by BroMAP. Therefore, using BroMAP did not impractically slow down the solution time for cases in Table 1.

For large hard cases, the system memory can be a limiting factor on the performance of DEE/A^* because the performance of DEE/A^* often greatly depends on the unification procedure that requires a large amount of memory. Although this implies larger system memory could have given advantage to DEE/A^* over BroMAP in terms of running time, our results suggests that the memory constraints experienced by DEE/A^* can be alleviated through the use of BroMAP.

Table II lists the percentage of time used for each component of BroMAP. In most cases, DEE-gp, A^*, and TRMP turned out to be major contributors to the running time. If we sum running times of BroMAP for all cases, it is found that 42% of the total time was spent on DEE-gp and A^*, and 45% on TRMP. On the other hand, considering the proportion between the total running time of BroMAP and A^* time, a great amount of time was spent on A^* for cases 11 and 12. This could be avoided by further restricting the size of the subproblem for which A^* is allowed to run.

Among cases in Table 2, BroMAP was able to solve six cases at the root node without splitting. All other cases required BroMAP to branch but many of them needed very little branching other than those performed during the first depth-first dive. This trend is observed through the skewness of the search tree, defined as $\frac{(\text{number of low-subproblems split})}{(\text{total number of branchings})-1}$. The ratio varies between 0 and 1 and is larger than 0.5 if there are more low-subproblems split than high-subproblems. We computed skewness for 36 cases where BroMAP required more than one split. The minimum skewness from these cases is 0.69 and 17 cases had skewness equal to 1, that is, needed only low-subproblem splittings.

Figure 9 shows actual search trees generated by BroMAP during solution of three cases. Box-shaped (shaded) nodes in each search tree represent the subproblems that were exactly solved and resulted in an upper bound less than or equal to the current best upper bound. Therefore, the box-shaped node that is expanded latest is a node where the GMEC is found in the search tree. Note that, for 27 cases out of 42 cases in Table 2, an upper bound equal to the GMEC energy was found at the end of the first depth-first dive. However, early discovery of the GMEC did not necessarily lead to fast completion of BroMAP. For example, in Figure 9c, we can see the lower bounding was not effective for large subproblems although they were expanded after the optimal upper-bound was found, resulting in a large number of branchings.

Table 2. Results of Solving the Non-"Limited DEE" Cases with BroMAP and DEE/A^*.

No.	Bro	DEE	T-Br	F-Br	Skew	F-Ub	Leaf	Rdctn	RC	%DE	%A^*	%TR
2	2.6 E 3	3.1 E 4	31	25	0.90	0.49	30.7	2.12	36	42.8	0.3	56.3
3	2.4 E 3	2.3 E 4	31	26	0.93	0.49	27.7	2.55	32	46.2	0.6	52.6
4	2.8 E 3	1.3 E 4	23	23	1	0	33.7	3.01	0	43.9	0.3	55.5
5	2.7 E 3	2.1 E 4	26	26	1	0.55	27.4	3.12	0	37.2	0.4	62.2
9	1.2 E 2	4.8 E 2	3	3	1	0	27.6	1.93	0	8.9	74.1	17.0
10	4.6 E 2	1.3 E 3	13	10	0.75	0.37	26.9	1.02	74	7.6	70.4	14.4
11	5.7 E 3	3.5 E 4	109	17	0.81	0.36	26.2	0.85	663	3.8	78.9	11.2
15	2.9 E 2	3.5 E 2	0	0	NA	0	NA	NA	0	94.6	0.4	4.7
23	1.5 E 2	2.6 E 2	0	0	NA	0	NA	NA	0	86.7	0	12.6
24	3.2 E 2	3.1 E 2	4	4	1	0	25.3	4.33	0	62.3	15.1	21.6
25	2.9 E 2	1.2 E 3	0	0	NA	0	NA	NA	0	89.6	0	10.4
26	1.4 E 3	1.7 E 3	11	11	1	0.89	29.2	1.65	0	46.1	0.4	53.2
33	4.1 E 2	2.1 E 3	13	13	1	0	27.9	2.43	0	34.7	4.5	59.8
34	1.1 E 3	3.7 E 3	19	19	1	0	30.0	2.32	0	32.2	2.7	64.8
35	2.8 E 3	4.1 E 4	21	21	1	0	28.7	3.03	0	50.7	0.6	48.6
36	4.6 E 3	2.3 E 4	25	25	1	0	27.9	3.39	0	53.2	0.7	45.9
37	2.5 E 2	2.5 E 2	0	0	NA	0	NA	NA	0	76.0	2.4	21.2
44	2.2 E 2	3.8 E 1	8	6	0.71	0.54	28.2	1.87	17	8.2	75.5	14.1
45	8.8 E 2	2.0 E 2	8	8	1	0	26.2	5.16	8	48.6	23.8	25.4
49	3.3 E 2	5.0 E 2	4	4	1	0	19.8	1.63	0	51.1	11.5	37.5
50	1.2 E 3	1.1 E 3	7	7	1	0	22.3	3.44	12	72.2	7.0	17.1
51	5.7 E 4	2.8 E 5	666	25	0.85	0.58	27.6	1.03	5656	16.7	21.2	41.8
57	4.6 E 1	2.7 E 2	0	0	NA	0	NA	NA	0	84.8	0	15.2
58	1.5 E 3	1.0 E 3	19	19	1	0	28.8	2.69	0	42.5	0.2	57.1
59	4.4 E 2	4.0 E 3	0	0	NA	0	NA	NA	0	70.6	0	29.1
60	1.5 E 4	4.6 E 4	32	32	1	0	37.3	2.46	0	30.1	0.1	69.7
65	4.6 E 3	1.7 E 3	15	15	1	0	22.7	5.09	0	61.9	0	37.8
66	7.7 E 3	2.4 E 3	15	15	1	0	33.9	5.15	0	67.2	0	32.6
					Cases below were solved by BroMAP only							
12	2.0 E 5	NA	2773	23	0.82	7.11	26.2	0.88	3.9 E 4	6.0	59.1	20.1
16	3.5 E 3	NA	12	11	0.91	0	23.6	1.75	30	41.7	6.0	49.3
17	1.1 E 5	NA	298	21	0.84	3.35	26.7	0.77	2576	17.7	28.1	32.7
27	8.0 E 3	NA	23	23	1	0	27.8	0.99	13	32.2	1.1	66.4
28	2.1 E 4	NA	175	25	0.91	0	28.0	0.99	1168	23.8	8.5	57.9
38	1.4 E 4	NA	155	31	0.87	0.50	30.2	1.47	571	30.9	0.4	62.8
39	1.2 E 5	NA	572	43	0.85	0	27.4	1.50	4791	30.4	0.1	58.4
40	1.8 E 5	NA	293	43	0.81	0	29.6	1.91	2440	35.9	0	56.1
41	2.1 E 5	NA	364	41	0.85	0	33	2.66	2771	34.2	0	57.5
46	5.0 E 5	NA	2675	36	0.69	8.28	27.8	1.44	1.4 E 5	18.8	18.8	35.3
61	2.8 E 4	NA	55	49	0.96	0.36	28.2	2.04	15	49.0	0	50.8
62	3.6 E 5	NA	232	58	0.88	0.27	30.1	1.84	1119	43.5	0	50.2
63	1.1 E 5	NA	143	53	0.85	0.29	32.8	2.08	506	41.4	0	55.4
67	1.3 E 5	NA	37	37	1	0	35.6	2.82	0	51.5	0	48.5

Columns (1) No.: case number, (2) Bro: BroMAP solution time in seconds, (3) DEE: DEE/A^* solution time in seconds, (4) T-Br: total number of branchings (i.e. splits), (5) F-Br: number of branchings during the first depth-first dive, (6) Skew: skewness of the search tree defined as $\frac{(\text{number of low-subproblems split})}{(\text{total number of splits})-1}$, (7) F-Ub: $U - OPT$, i.e. difference between the upper bound from the first depth-first dive and the GMEC energy, (8) Leaf: $\sum_i \log_{10} |R_i|$ of the node at the end of the first depth-first dive, (9) Rdctn: average reduction of $\sum_i \log_{10} |R_i|$ during the first depth-first dive, i.e. (log*conf* – Leaf)/(F-Br), where log*conf* is defined in Table 1, (10) RC: number of rotamer-contractions performed, (11) %DE: BroMAP time percentage used for *DEE-gp*, (12) %A^*: BroMAP time percentage used for A^*, (13) %TR: BroMAP time percentage used for TRMP. Note that columns 11–13 may not sum to 100% because of time spent on rotamer-contraction and overhead of using the branch-and-bound framework.

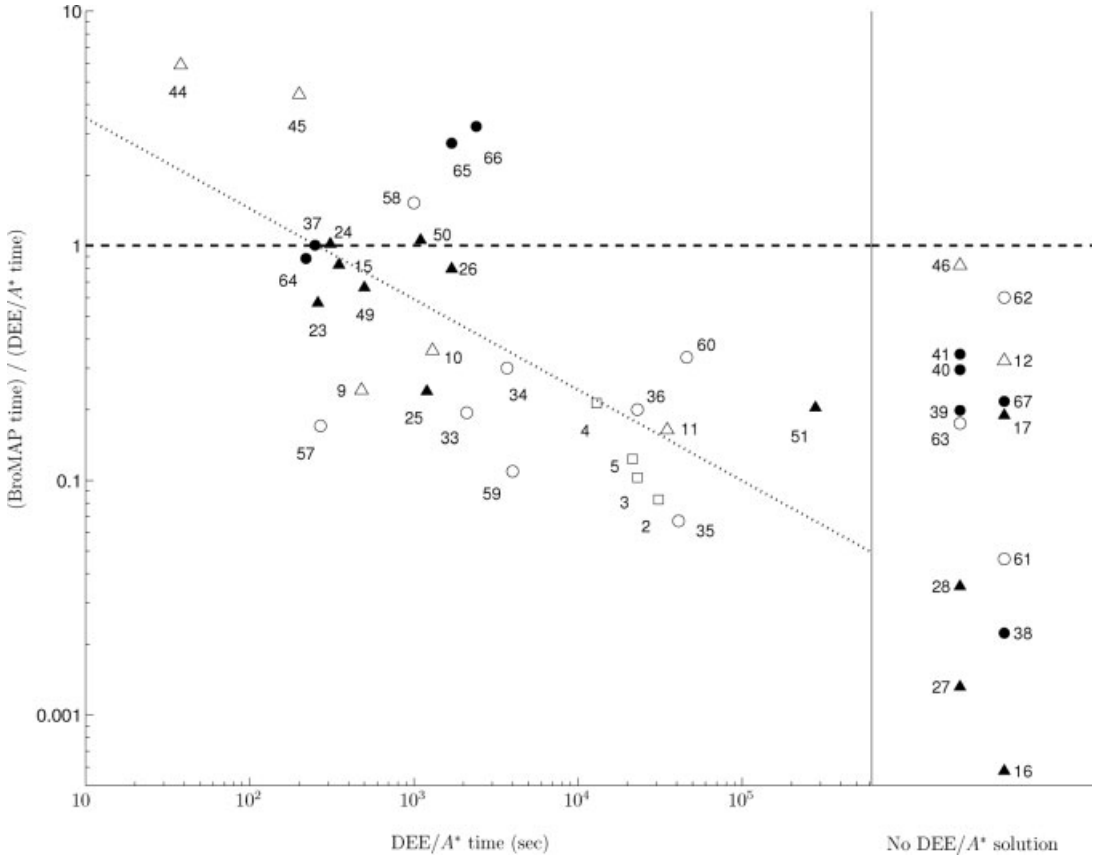

Figure 8. Ratio of BroMAP time to DEE/A^* time vs. DEE/A^* time for 42 cases in Table 2. Labels next to data points are case numbers from Table 1. The 14 cases solved by BroMAP only are shown in the narrow pane on the right side. The running time ratios for these cases were calculated by assuming the DEE/A^* time for each of them is 7 days although they were not solved by DEE/A^* within 7 days. The trend line represents a robust fit for the 28 cases that were solved by both BroMAP and DEE/A^*. The horizontal dashed line represents the ratio equal to 1. Different symbols are used to represent each case depending on the type of protein region (CORE, CORE++, or INT) and the type of library used (REG or EXP): (1) ∘ = CORE, □ = CORE++, Δ = INT, (2) empty = REG, filled = EXP.

Table 2 suggests that the search trees of BroMAP have smaller depths than those from conventional BnB methods would have. A simple branching without reduction within a node would only reduce the problem size by a factor of two. That is, a child subproblem will have $\sum_i \log_{10} |R_i|$ value smaller by $\log_{10} 2 \approx 0.30$ than its parent subproblem. However, column "Rdctn" shows the average reduction was far greater. Excluding the cases solved without branching, the average of the average reduction of $\sum_i \log_{10} |R_i|$ along the first depth-first dive was 2.32, a factor of 7 speed-up over reduction by conventional BnB methods. It should be noted that the reduction within a node can be even greater after a strong upper bound is found. This is evidenced by highly skewed shapes of search trees. Overall, the reduced depth and high skewness of BroMAP search trees suggest the number of nodes expanded by BroMAP is exponentially smaller than that of conventional BnB methods using simple branching. Meanwhile, the effect of smaller search trees will be transferred to shorter running times as well; the experimental results presented by Hong and Lozano-Pérez [52] show that the node processing time by DEE-gp and TRMP is similar to the bounding time for solving a linear programming (LP) problem, a typical bounding method used in BnB methods, but the LP produces weaker bounds.

The plots in Figure 10 provide interesting insights on the hardness of test cases. In Figure 10a, categorizing all test cases by their solvability reveals cases with higher ratios of log conformation to the number of design positions tend to be harder to solve. Figure 10b uses gray scale to represent the running times of BroMAP. Although the performance of BroMAP is not particularly dependent on protein regions, it is noted that INT cases are smaller than CORE cases. This is because we excluded small CORE cases from the experiment because they are often too easy for either BroMAP or DEE/A^*, and also excluded large INT cases for the opposite reason. There are two main reasons that INT cases are harder than CORE. First, INT cases are offered more rotamers per position because we allowed 8–14 amino acids for CORE cases whereas 18 amino acids including R, K, D, and E were allowed for INT cases. These four additional amino acids offer even more rotamers per amino acid than average because of their long side chains. Second, whereas CORE cases are constrained by side-chain/side-chain interactions as

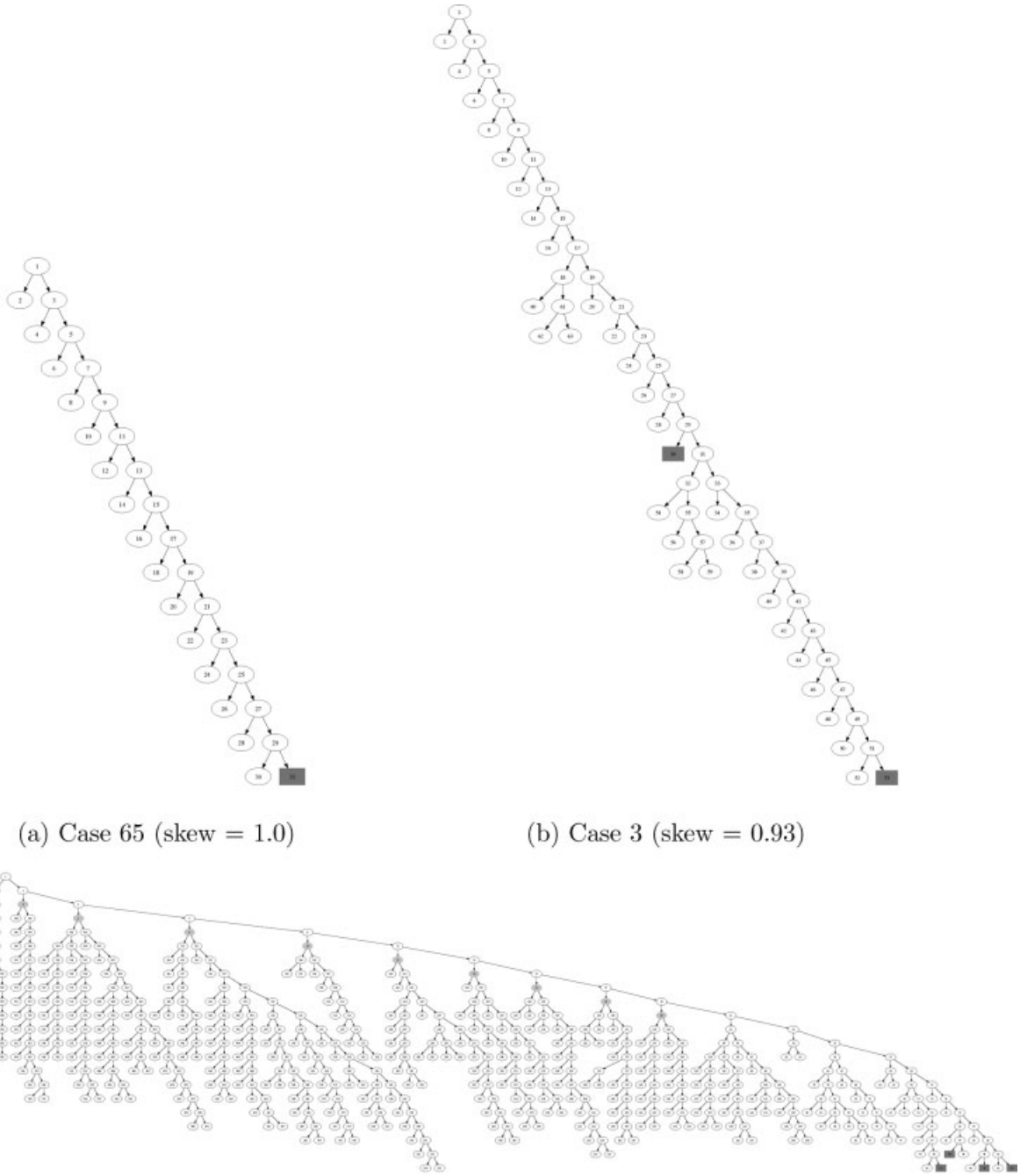

Figure 9. Search trees of BroMAP for three cases. For each branching, the low-subproblem is placed on the right-hand side, and the high-subproblem on the left-hand side. Shaded box-shaped nodes represent the subproblems that were exactly solved and resulted in an update of the global upper bound. (a) Case 65 (skew = 1.0). (b) Case 3 (skew = 0.93). (c) Case 17 (skew = 0.84).

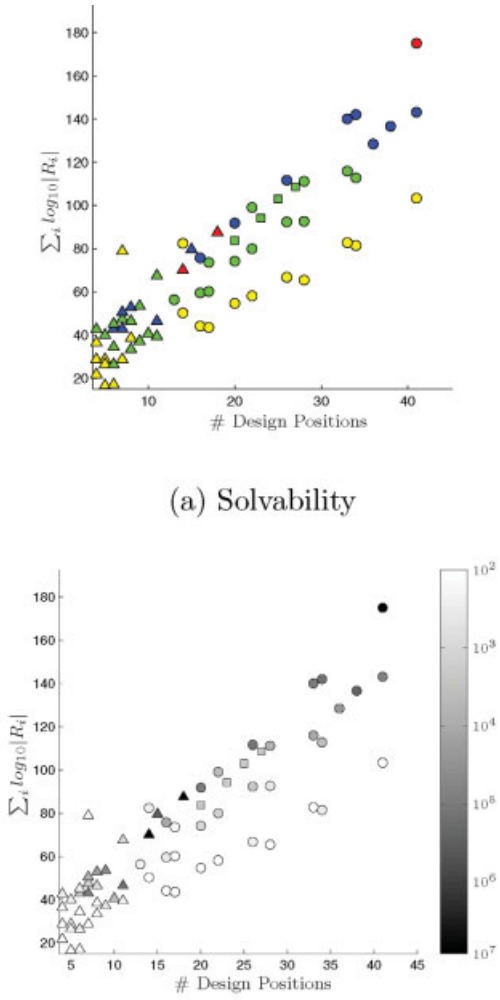

Figure 10. Each case is plotted by the number of design positions and log number of conformations. In both (a) and (b), different symbols were used for different protein regions: (1) Δ = INT, (2) $\circ$ = CORE, (3) $\square$ = CORE++. In (a), cases were marked with different colors depending on their solvability: (1) yellow, solved by limited DEE; (2) green, solved by BroMAP and DEE/*A**; (3) blue, solved by BroMAP only; (4) red, solved by none. In (b), the BroMAP running time on each case was used to color the corresponding symbol. The color bar on the right side shows mapping between a color and a running time in seconds. (a) Solvability. (b) Protein region and BroMAP running time.

well as side-chain/backbone interactions, INT cases are generally less constrained by side-chain/backbone interactions, and therefore there exist a larger number of compatible conformations.

TRMP Lower Bounds

We present a numerical example to illustrate the utility of TRMP lower bounds in rotamer/rotamer-pair elimination. For this purpose, we use subproblems of Case 17 at depth 2 to 11 (root node is at depth 1). These subproblems correspond to node numbers $2, 4, 6, \ldots, 20$, and are colored in light gray in Figure 9c (nodes in the search tree are numbered by the order of creation using depth-first search). Table 3 lists the lower-bounding result.

In each node, we obtain more elimination using rotlb_2 (rotamer lower bound from using pair flags) than using rotlb_1 (rotamer lower bound from not using pair flags). This is due to massive flagging of rotamer pairs by rplb (rotamer-pair lower bound). Figure 11 shows rotamers ordered by the value of rotlb_1 on x-axis and their rotlb_1, rotlb_2 values on y-axis for the subproblem of node 2. The difference between rotlb_1 and rotlb_2 for the same rotamer shows pair-flags information can strengthen the lower bounds, thereby doubling the number of eliminated rotamers in this example.

Large elimination obtained for subproblems at small depth are suspected to come from our splitting scheme of dividing rotamers by their lower bounds. As we go deeper down the search tree, we expect such distinction between rotamer lower bounds to become less clear. The trend is observed by the median value of rotlb_1 and the percentage of eliminated rotamers and rotamer pairs for nodes at different depths.

Computing rotlb_2 takes more time than rotlb_1, but Table 3 shows that the difference is relatively insignificant compared to the time for computing rplb. The time for computing rplb for every rotamer pair was typically at least double the time for TRMP convergence, suggesting that an efficiency improvement of rotamer-pair lower-bound computation would significantly contribute to reducing the running time of BroMAP.

Conclusions

In this work, we presented an exact solution method (BroMAP) for the global minimum energy conformation search. Particularly, BroMAP was designed to substitute the DEE/*A** approach for large protein design problems where a large number of rotamers is offered at each position and there exist side-chain interactions between all pairs of residue positions. BroMAP uses a branch-and-bound (BnB) framework and performs problem-size reduction within each subproblem using DEE and elimination by TRMP lower bounds. BroMAP also exploits TRMP lower bounds in branching and subproblem selection. We performed computational experiments to evaluate BroMAP on various types and sizes of protein design problems in comparison with DEE/*A**. The experimental results show that BroMAP solved hard protein design cases faster than DEE/*A**, and that BroMAP also solved many cases that DEE/*A** failed to solve within allowed time and memory. In addition, using BroMAP on cases where DEE/*A** performed well did not incur significant disadvantage in running time.

The performance advantage of BroMAP over DEE/*A** or conventional BnB methods can be attributed to three factors. First, the search trees are radically smaller than those from conventional BnB methods. Problem-size reduction performed within each node results in reduced depths of search trees, and early discovery of suboptimal upper bounds allows effective pruning of nodes that follow. Second, on top of fast reduction by DEE within each node, BroMAP can perform additional elimination and informed branching using lower bounds from inexpensive computation. Third, the general BnB framework of BroMAP allows additional lower-bounding techniques such as rotamer-contraction to be easily incorporated.

Table 3. TRMP Lower-Bounding Results for Subproblems of case 17.

				Rot lb's w/o p-flags			Rot-pair lb's			Rot lb's w/ p-flags		
Node	No. of rots	No. of pairs	T_{TR}	med	%el	time (s)	med	%el	time	med	%el	time
2	4504	8.4 E 6	148	−71.69	11	0	−40.21	74	1162	−70.91	26	7
4	3837	6.3 E 6	203	−70.98	9	0	−46.18	68	774	−70.61	21	3
6	3570	5.4 E 6	238	−80.98	3	0	−53.45	54	607	−80.56	7	4
8	3269	4.5 E 6	190	−84.57	1	1	−58.82	44	463	−84.35	3	1
10	2969	3.7 E 6	99	−73.98	12	0	−47.80	62	353	−73.59	17	2
12	2704	3.1 E 6	105	−84.00	7	0	−59.17	45	261	−83.81	8	1
14	2504	2.6 E 6	77	−84.10	13	0	−51.16	50	202	−83.93	14	1
16	2173	2.0 E 6	65	−82.25	5	0	−61.36	42	138	−82.13	8	0
18	1878	1.5 E 6	71	−88.56	5	0	−66.31	40	92	−88.47	9	0
20	1725	1.3 E 6	16	−86.09	7	0	−65.68	38	74	−85.94	8	0

The meaning of each column is, in order: (1) node number, (2) number of rotamers in the subproblem, (3) number of rotamer pairs in the subproblem, (4) time (s) for TRMP convergence, (5) median rotamer lower bound when not using pair flags (rotlb_1), (5) percentage of rotamers such that $\text{rotlb}_1 > U$, (6) time (s) for computing rotlb_1 for all rotamers, (7) median rotamer-pair lower bound (rplb), (8) percentage of rotamer pairs such that $rplb > U$, (9) time (s) for computing rplb for all rotamer pairs, (10) median rotamer lower bound when using pair flags (rotlb_2) after rotamer pairs were flagged by rplb, (11) percentage of rotamers such that $\text{rotlb}_2 > U$, (12) time (s) for computing rotlb_2 for all rotamers. In the Table, time for TRMP convergence was excluded from time for computing rotlb_1, rotlb_2, or rplb. The value of U is −55.13, which is equal to the optimal value and was available as a global upper bound for each node in the table by the time they were expanded.

It could be argued that the performance comparison between BroMAP and DEE/A^* was not thorough or fair because DEE can be faster depending on what elimination conditions are used, how they are combined ,[26] or how much memory is available for unification.

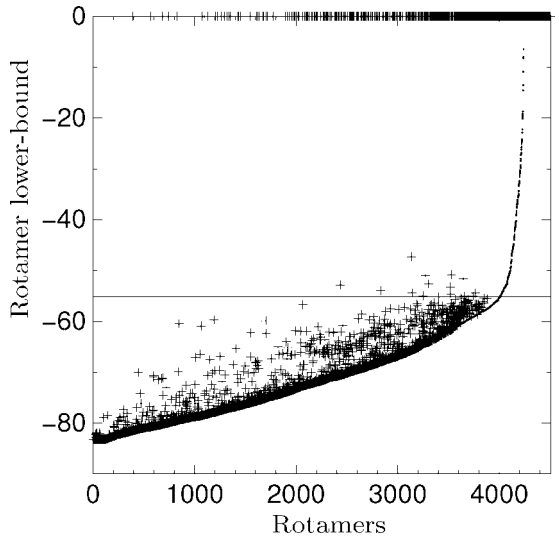

Figure 11. Plots of TRMP lower bound vs. rotamer, for all (4504) rotamers in the subproblem of node 2 in solving case 17 by BroMAP; rotlb_1 is represented by a dot and rotlb_2 by a '+' symbol. Rotamers are sorted on x-axis by the increasing order of rotlb_1. All rotamers with lower bounds greater than or equal to 0 were clipped at $y = 0$. The horizontal line at $y = -55.13$ represents U. By comparing rotlb_1 against U, 497 rotamers ($4,008^{\text{th}}$ to $4,504^{\text{th}}$ in the order) were eliminated. Using rotlb_2 instead increased the number of eliminated rotamers to 1171.

However, it should be noted that BroMAP also exploits DEE, and that BroMAP can be regarded as an added structure to DEE/A^* to allow a more effective use of it in a general framework. As a result, if a better implementation of DEE/A^* is given or a better system environment is allowed, the performance of BroMAP is also expected to benefit from it.

In our experiment, using rotamer-contraction did not always improve the total running time of BroMAP, although it tends to reduce the number of nodes expanded by BroMAP. However, among the 14 test cases that were solved by BroMAP with rotamer-contraction but not by DEE/A^*, two could not be solved by BroMAP without rotamer-contraction within the 7-day time limit. In addition, for the 51 test cases used for comparison of BroMAP and DEE/A^*, the total running time of BroMAP was reduced by 17% on average simply by using rotamer-contraction. Therefore, there is a question of how much effort should be spent on rotamer-contraction to maximize the performance of BroMAP. On the other hand, observing the behavior of BroMAP on many random instances to parameterize its solution time by problem characteristics will be interesting and may help improve the performance of BroMAP, because no direct correlation between the problem size and the BroMAP solution time has been found. Finally, a substantial fraction of BroMAP's running time is spent on post-processing of TRMP to compute rotamer-pair lower bounds. Therefore, a speed-up of BroMAP could be made through efficiency improvement of this postprocessing procedure. Our future investigation will address these problems to extend the applicability of BroMAP to larger protein design cases.

Acknowledgment

The authors thank the current and past members of Tidor group, especially Michael Altman for his DEE/A^* code and Alessandro Senes for his advice and sparing his time.

Appendix A: TRMP

Algorithm 1 describes "edge-based reparameterization updates" [47] for a set of general trees $\mathcal{T}$. In line 2, 3, 5, and 6, κ_i^n and κ_{ij}^n are constants that can be arbitrarily set as long as they are positive. $\Gamma(i)$ is the set of vertices neighboring i in $\mathcal{G}$ for $i \in \mathcal{V}(\mathcal{G})$.

Algorithm 1: TRMP (edge-based reparameterization updates[47;51])

Data: $\epsilon > 0$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\{\rho\}$, $\{\epsilon\}$

Result: tree-consistent pseudo-max-marginals $\{\nu\}$

1 **begin**

2 $\quad \nu_i^0(x_i) \leftarrow \kappa_i^0 \exp\left(-\frac{\epsilon_i(x_i)}{\rho_i}\right), \quad i \in \mathcal{V}$

3 $\quad \nu_{ij}^0(x_i, x_j) \leftarrow \kappa_{ij}^0 \exp\left(-\frac{\epsilon_{ij}(x_i,x_j)}{\rho_{ij}} - \frac{\epsilon_i(x_i)}{\rho_i} - \frac{\epsilon_j(x_j)}{\rho_j}\right), \quad (i,j) \in \mathcal{E}$

4 $\quad$ **repeat** update pseudo-max-marginals

5 $\quad\quad \nu_i^{n+1}(x_i) \leftarrow \kappa_i^{n+1} \nu_i^n(x_i) \prod_{j \in \Gamma(i)} \left(\frac{\max_{x_j \in R_j} \nu_{ij}^n(x_i,x_j)}{\nu_i^n(x_i)}\right)^{\frac{\rho_{ij}}{\rho_i}}, \quad i \in \mathcal{V}$

6 $\quad\quad \nu_{ij}^{n+1}(x_i, x_j) \leftarrow \kappa_{ij}^{n+1} \frac{\nu_{ij}^n(x_i,x_j) \nu_i^{n+1}(x_i) \nu_j^{n+1}(x_j)}{\max_{x_j' \in R_j} \nu_{ij}^n(x_i,x_j') \max_{x_i' \in R_i} \nu_{ij}^n(x_i',x_j)}, \quad (i,j) \in \mathcal{E}.$

7 $\quad$ **until** $|\nu^{n+1} - \nu^n| < \epsilon$, update $n \leftarrow n+1$

8 **end**

Appendix B. ILP Formulation

The ILP formulation for the GMEC problem referred in this article is as follows:

$$\min_{\{x_{i_r}\},\{x_{i_r j_s}\}} \left[\sum_{i \in \mathcal{V}} \sum_{r \in R_i} E(i_r) x_{i_r} + \sum_{(i,j) \in \mathcal{E}} \sum_{(r,s) \in R_i \times R_j} E(i_r j_s) x_{i_r j_s} \right] \tag{B1}$$

$$\sum_{r \in R_i} x_{i_r} = 1, \quad \forall i \in \mathcal{V} \tag{B2}$$

$$\sum_{s \in R_j} x_{i_r j_s} = x_{i_r}, \quad \forall (i,j) \in \mathcal{E}, \quad \forall r \in R_i, \tag{B3}$$

$$x_{i_r} \in \{0,1\}, \quad \forall i \in \mathcal{V}, \quad r \in R_i, \tag{B4}$$

$$x_{i_r j_s} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{E}, \quad (r,s) \in R_i \times R_j. \tag{B5}$$

An LP relaxation can be obtained by simply replacing integer constraints (B4,B5) with interval constraints [0, 1]. The resulting LP is equivalent to the tree-relaxed LP .[47]

Appendix C. Proof of Lemma 1

From eq. (16), we have

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{\mathbf{x}} \frac{\nu_c}{Z} \prod_{S \in \mathcal{S}} \{p^S(\mathbf{x};\nu)\}^{\rho(S)} \le \frac{\nu_c}{Z} \prod_{S \in \mathcal{S}} \left\{\max_{\mathbf{x}} p^S(\mathbf{x};\nu)\right\}^{\rho(S)} \tag{C1}$$

Since ν is tree-consistent with every $S \in \mathcal{S}$, we can easily find a MAP assignment $\mathbf{x}^S$ such that $\mathbf{x}^S \in \arg\max_{\mathbf{x}} p^S(\mathbf{x};\nu)$ using dynamic programming .[50] Then, due to the assumption that ν is tree-consistent with S and is in a normal form, we have the following properties:

$$\nu_i(x_i^S) = 1, \quad \text{for all } i \in \mathcal{V}(S), \tag{C2}$$

$$\nu_{ij}(x_i^S, x_j^S) = 1, \quad \text{for all } (i,j) \in \mathcal{E}(S). \tag{C3}$$

Therefore,

$$\max_{\mathbf{x}} p^S(\mathbf{x};\nu) = p^S(\mathbf{x}^S;\nu) = \prod_{i \in \mathcal{V}(S)} \nu_i(x_i^S) \prod_{(i,j) \in \mathcal{E}(S)} \frac{\nu_{ij}(x_i^S, x_j^S)}{\nu_i(x_i^S)\nu_j(x_j^S)} = 1. \tag{C4}$$

Since (C4) is true for every $S \in \mathcal{S}$ and $\sum_{S \in \mathcal{S}} \rho(S) = 1$, we obtain $\max_{\mathbf{x}} p(\mathbf{x}) \le \nu_c / Z$ from (C1).

Appendix D. Proof of Lemma 2

From eq. (16), we have

$$\begin{aligned}
\max_{\{\mathbf{x}|x_\zeta = r\}} p(\mathbf{x}) &= \max_{\{\mathbf{x}|x_\zeta = r\}} \frac{\nu_c}{Z} \prod_{S \in \mathcal{S}} \{p^S(\mathbf{x};\nu)\}^{\rho(S)} \\
&\le \frac{\nu_c}{Z} \prod_{S \in \mathcal{S}} \left\{\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu)\right\}^{\rho(S)} \\
&= \frac{\nu_c}{Z} \prod_{S \in \mathcal{S}: \zeta \in \mathcal{V}(S)} \left\{\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu)\right\}^{\rho(S)} \\
&\quad \times \prod_{S \in \mathcal{S}: \zeta \notin \mathcal{V}(S)} \left\{\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu)\right\}^{\rho(S)}
\end{aligned} \tag{D1}$$

By the definition of max-marginals and the assumption that ν is tree-consistent, for $S \in \mathcal{S}$ such that $\zeta \in \mathcal{V}(S)$, we have

$$\nu_\zeta(r) = \kappa_\zeta \max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu), \tag{D2}$$

for some constant κ_ζ. We know there exists $r^* \in R_\zeta$ such that

$$\nu_\zeta(r^*) = \max_{x_\zeta \in R_\zeta} \nu_\zeta(x_\zeta) = \kappa_\zeta \max_{\{\mathbf{x}|x_\zeta = r^*\}} p^S(\mathbf{x};\nu). \tag{D3}$$

Then, since ν is in a normal form, $\nu_\zeta(r^*) = 1$. We know from (C4) in the proof of Lemma 1, that $\max_{\{\mathbf{x}|x_\zeta = r^*\}} p^S(\mathbf{x};\nu) = \max_{\mathbf{x}} p^S(\mathbf{x};\nu) = 1$. Therefore, $\kappa_\zeta = 1$, and

$$\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu) = \nu_\zeta(r). \tag{D4}$$

On the other hand, for all $S \in \mathcal{S}$ such that $\zeta \notin \mathcal{V}(S)$, we know $\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu) = \max_{\mathbf{x}} p^S(\mathbf{x};\nu) = 1$. Plugging the obtained values of $\max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x};\nu)$ and $\sum_{S \in \mathcal{S}: \zeta \in \mathcal{V}(S)} \rho(S) = \rho_\zeta$ into (D1), we obtain $\max_{\{\mathbf{x}|x_\zeta = r\}} p(\mathbf{x}) \le (\nu_c / Z) \nu_\zeta(r)^{\rho_\zeta}$.

Appendix E. Proof of Lemma 3

1. If $\zeta, \eta \neq \mathcal{V}(S)$, we know $\max_{\{\mathbf{x}|x_\zeta = r, x_\eta = s\}} p^S(\mathbf{x}; \nu) = \max_{\mathbf{x}} p^S(\mathbf{x}; \nu)$. Then, since ν is tree-consistent and in a normal form, $\max_{\{\mathbf{x}|x_\zeta = r, x_\eta = s\}} p^S(\mathbf{x}; \nu) = 1$ from (C4).
2. If $\zeta \in \mathcal{V}(S)$, and $\eta \notin \mathcal{V}(S)$, we know $\max_{\{\mathbf{x}|x_\zeta = r, x_\eta = s\}} p^S(\mathbf{x}; \nu) = \max_{\{\mathbf{x}|x_\zeta = r\}} p^S(\mathbf{x}; \nu)$. Then, from (D4), $\max_{\{\mathbf{x}|x_\zeta = r, x_\eta = s\}} p^S(\mathbf{x}; \nu) = \nu_\zeta(r)$
3. If $(\zeta, \eta) \in \mathcal{E}(S)$, by the definition of max-marginals and the assumption that ν is tree-consistent with every S, we have $\nu_{\zeta\eta}(r, s) = \kappa_{\zeta\eta} \max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} p^S(\mathbf{x}; \nu)$, for some constant $\kappa_{\zeta\eta}$. We also know there exists $(r^*, s^*) \in R_\zeta \times R_\eta$ such that $\nu_{\zeta\eta}(r^*, s^*) = \max_{(x_\zeta, x_\eta) \in R_\zeta \times R_\eta} \nu_{\zeta\eta}(x_\zeta, x_\eta) = \kappa_{\zeta\eta} \max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r^*, s^*)\}} p^S(\mathbf{x}; \nu)$. Then, since ν is in a normal form, i.e. $\nu_{\zeta\eta}(r^*, s^*) = 1$ and we have $\max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r^*, s^*)\}} p^S(\mathbf{x}; \nu) = \max_{\mathbf{x}} p^S(\mathbf{x}; \nu) = 1$ from (C4), we obtain $\kappa_{\zeta\eta} = 1$. Therefore, $\max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} p^S(\mathbf{x}; \nu) = \nu_{\zeta\eta}(r, s)$.
4. If $\zeta, \eta \in \mathcal{V}(S)$ and $(\zeta, \eta) \notin \mathcal{E}(S)$, let $\xi = \gamma(S)$. Then,

$$\begin{aligned}
&\max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} p^S(\mathbf{x}; \nu) \\
&= \max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} \nu_\xi(x_\xi)\nu_\zeta(r)\nu_\eta(s) \frac{\nu_{\xi\zeta}(x_\xi, r)}{\nu_\xi(x_\xi)\nu_\zeta(r)} \\
&\quad \times \frac{\nu_{\xi\eta}(x_\xi, s)}{\nu_\xi(x_\xi)\nu_\eta(s)} \prod_{j \in \mathcal{V}(S)\backslash\{\zeta,\eta,\xi\}} \nu_j(x_j) \frac{\nu_{\xi j}(x_\xi, x_j)}{\nu_\xi(x_\xi)\nu_j(x_j)} \qquad \text{(E1)} \\
&= \max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} \frac{\nu_{\xi\zeta}(x_\xi, r)\nu_{\xi\eta}(x_\xi, s)}{\nu_\xi(x_\xi)} \prod_{j \in \mathcal{V}(S)\backslash\{\zeta,\eta,\xi\}} \frac{\nu_{\xi j}(x_\xi, x_j)}{\nu_\xi(x_\xi)} \qquad \text{(E2)} \\
&= \max_{x_\xi} \left\{ \frac{\nu_{\xi\zeta}(x_\xi, r)\nu_{\xi\eta}(x_\xi, s)}{\nu_\xi(x_\xi)} \prod_{j \in \mathcal{V}(S)\backslash\{\zeta,\eta,\xi\}} \max_{x_j} \frac{\nu_{\xi j}(x_\xi, x_j)}{\nu_\xi(x_\xi)} \right\}. \qquad \text{(E3)}
\end{aligned}$$

From the tree-consistency of ν, we have $\nu_\xi(x_\xi) = \beta_{\xi j} \max_{x_j} \nu_{\xi j}(x_\xi, x_j)$ for some $\beta_{\xi j} > 0$. We can also easily find $\mathbf{x}^S$ such that $\mathbf{x}^S \in \arg\max_{\mathbf{x}} p^S(\mathbf{x}; \nu)$ using the tree-consistency of ν. Then, since ν is in a normal form, we have $\nu_\xi(x_\xi^S) = 1$, and $\max_{x_j} \nu_{\xi j}(x_\xi^S, x_j) = \nu_{\xi j}(x_\xi^S, x_j^S) = 1$. Therefore, $\beta_{\xi j} = 1$, and $\nu_\xi(x_\xi) = \max_{x_j} \nu_{\xi j}(x_\xi, x_j)$. So the maximization over x_j in the parentheses of (E3) is equal to 1 for all $j \in \mathcal{V}(S)\backslash\{\zeta, \eta, \xi\}$. Therefore,

$$\max_{\{\mathbf{x}|(x_\zeta, x_\eta) = (r,s)\}} p^S(\mathbf{x}; \nu) = \max_{x_\xi} \frac{\nu_{\xi\zeta}(x_\xi, r)\nu_{\xi\eta}(x_\xi, s)}{\nu_\xi(x_\xi)}.$$

References

1. Chothia, C.; Lesk, A. M. EMBO J 1986, 5, 823.
2. Ring, C. S.; Cohen, F. E. FASEB J 1993, 7, 783.
3. Baker, D.; Sali, A. Science 2001, 294, 93.
4. Drexler, K. E. Proc Natl Acad Sci USA 1981, 78, 5275.
5. Pabo, C. Nature 1983, 301, 200.
6. Godzik, A.; Kolinski, A.; Skolnick, J. J Comput Aided Mol Des 1993, 7, 397.
7. Hellinga, H. W.; Richards, F. M. Proc Natl Acad Sci USA 1994, 91, 5803.
8. Dahiyat, B. I.; Mayo, S. L. Protein Sci 1996, 5, 895.
9. Kuhlman, B.; Dantas, G.; Ireton, G. C.; Varani, G.; Stoddard, B. L.; Baker, D. Science 2003, 302, 1364.
10. Ponder, J. W.; Richards, F. M. J Mol Biol 1987, 193, 775.
11. Dunbrack, R. L.; Karplus, M. J Mol Biol 1993, 230, 543.
12. Desmet, J.; De Maeyer, M.; Hazes, B.; Lasters, I. Nature 1992, 356, 539.
13. Weiner, S. J.; Kollman, P. A.; Case, D. A.; Singh, U. C.; Ghio, C.; Alagona, G.; Profeta, S. Jr.; Weiner, P. J Am Chem Soc 1984, 106, 765.
14. Weiner, S. J.; Kollman, P. A.; Nguyen, D. T.; Case, D. A. J Comput Chem 1986, 7, 230.
15. Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M. Jr.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. J Am Chem Soc 1995, 117, 5179.
16. Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. J Comput Chem 1983, 4, 187.
17. Mackerell, A. D.; Bashford, D.; Bellott, M.; Dunbrack, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-Mccarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D. T.; Prodhom, B.; Reiher, W. E.; Roux, B.; Schlenkrich, M.; Smith, J. C.; Stote, R.; Straub, J.; Watanabe, M.; Wiorkiewicz-Kuczera, J.; Yin, D.; Karplus, M. J Phys Chem B 1998, 102, 3586.
18. Halgren, T. A. J Comput Chem 1996, 17, 490.
19. Jorgensen, W. L.; Tirado-Rives, J. J Phys Chem 1996, 100, 14508.
20. Korte, B.; Vygen, J. In Combinatorial Optimization: Theory and Algorithms, 2nd ed., Springer-Verlag, New York, 2002.
21. Goldstein, R. F. Biophys J 1994, 66, 1335.
22. Lasters, I.; De Maeyer, M.; Desmet, J. Protein Eng 1995, 8, 815.
23. Gordon, D. B.; Mayo, S. L. J Comput Chem 1998, 13, 1505.
24. Pierce, N. A.; Spriet, J. A.; Desmet, J.; Mayo, S. L. J Comput Chem 2000, 21, 999.
25. Looger, L. L.; Hellinga, H. W. J Mol Biol 2001, 307, 429.
26. Gordon, D. B.; Hom, G. K.; Mayo, S. L.; Pierce, N. A. J Comput Chem 2003, 24, 232.
27. Georgiev, I.; Lilien, R. H.; Donald, B. R. Bioinformatics 2006, 22, 174.
28. Leach, A. R.; Lemon, A. P. Proteins 1998, 33, 227.
29. Gordon, D. B.; Mayo, S. L. Structure 1999, 7, 1089.
30. Wernisch, L.; Hery, S.; Wodak, S. J. J Mol Biol 2000, 301, 713.
31. Eriksson, O.; Zhou, Y.; Elofsson, A. In Proceedings of WABI 2001, Vol. 2149 (LNCS); Springer, Aarhus, Denmark, 2001; pp. 128–141.
32. Althaus, E.; Kohlbacher, O.; Lenhof, H.-P.; Müller, P. J Comput Biol 2002, 9, 597.
33. Kingsford, C.; Chazelle, B.; Singh, M. Bioinformatics 2005, 21, 1028.
34. Leaver-Fay, A.; Kuhlman, B.; Snoeyink, J. In Pacific Symposium on Biocomputing 10, World Scientific Publishing, Hawaii, U.S.A., 2005; pp. 16–27.
35. Xu, J. In Proceedings of RECOMB 2005; Springer, Cambridge, MA, 2005; pp. 423–439.
36. Xie, W.; Sahinidis, N. V. Bioinformatics 2006, 22, 188.
37. Yanover, C.; Meltzer, T.; Weiss, Y. J Mach Learn Res 2006, 7, 1887.
38. Weiss, Y.; Yanover, C.; Meltzer, T. In Proceedings of UAI, Vancouver, B.C., Canada, 2007, 2007.
39. Desjarlais, J. R.; Handel, T. M. J Mol Biol 1999, 289, 305.
40. Peterson, R. W.; Dutton, P. L.; Wand, A. J. Protein Sci 2004, 13, 735.
41. Koehl, P.; Delarue, M. J Mol Biol 1994, 239, 249.
42. Desjarlais, J. R.; Handel, T. M. Protein Sci 1995, 4, 2006.
43. Jones, D. T. Protein Sci 1994, 3, 567.
44. Jiang, X.; Farid, H.; Pistor, E.; Farid, R. Protein Sci 2000, 9, 403.
45. Yanover, C.; Weiss, Y. In Proceedings of NIPS 2002, Vancouver, B.C., Canada, 2002.

46. Hart, P. E.; Nilsson, N. J.; Raphael, B. IEEE Trans Syst Sci Cyber 1968, 2, 100.
47. Wainwright, M. J.; Jaakkola, T. S.; Willsky, A. S. IEEE Trans Inform Theory 2005, 51, 3697.
48. Nemhauser, G. L.; Wolsey, L. A. In Integer and Combinatorial Optimization; Wiley: NY, 1988.
49. Cowell, R. G.; Dawid, A. P.; Lauritzen, S. L.; Spiegehalter, D. J. In Probabilistic Networks and Expert Systems; Springer-Verlag: New York, 1999.
50. Wainwright, M. J.; Jaakkola, T.; Jordan, M. I. Stat Comput 2004, 14, 143.
51. Kolmogorov, V. IEEE Trans Pattern Anal 2006, 28, 1568.
52. Hong, E.-J.; Lozano-Pérez, T. In Proceedings of WABI 2006, Vol. 4175 (LNCS); Springer, Zurich, Switzerland, 2006; pp. 219–230.
53. Altman, M. D. Computational ligand design and analysis in protein complexes using inverse methods, combinatorial search, and accurate solvation modeling. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.
54. Eckstein, J.; Phillips, C. A.; Hart, W. E. Pico: An object oriented framework form parallel branch and bound Technical report, RUTCOR, Piscataway, NJ, 2001.
55. Main, A. L.; Harvey, T. S.; Baron, M.; Boyd, J.; Campbell, I. D. Cell 1992, 71, 671.
56. Braden, B. C.; Souchon, H.; Eisele, J. L.; Bentley, G. A.; Bhat, T. N.; Navaza, J.; Poljak, R. J. J Mol Biol 1994, 243, 767.
57. Bhat, T. N.; Bentley, G. A.; Boulot, G.; Greene, M. I.; Tello, D.; Dallacqua, W.; Souchon, H.; Schwarz, F. P.; Mariuzza, R. A.; Poljak, R. J. Proc Natl Acad Sci USA 1994, 91, 1089.
58. Syed, R.; Reid, S.; Li, C.; Cheetham, J.; Aoki, K.; Liu, B.; Zhan, H.; Osslund, T.; Chirino, A.; Zhang, J.; Finer-Moore, J.; Elliott, S.; Sitney, K.; Katz, B.; Matthews, D.; Wendoloski, J.; Egrie, J.; Stroud, R. Nature 1998, 395, 511.
59. Dunbrack, R. L. Curr Opin Struct Biol 2002, 12, 431.
60. Lippow, S. M.; Wittrup, K. D.; Tidor, B. Nat Biotechnol 2007, 25, 1171.