Protein Side-chain Placement through MAP Estimation and Problem-Size Reduction

Eun-Jong Hong and Tomás Lozano-Pérez

Computer Science and Artificial Intelligence Lab, MIT,
Cambridge, MA 02139, USA
{eunjong,tlp}@mit.edu

Abstract. We present an exact method for the global minimum energy conformation (GMEC) search of protein side-chains. Our method consists of a branch-and-bound (B&B) framework and a new subproblem-pruning scheme. The pruning scheme consists of upper/lower-bounding methods and problem-size reduction techniques. We explore a way of using the tree-reweighted max-product algorithm for computing lower-bounds of the GMEC energy. The problem-size reduction techniques are necessary when the size of the subproblem is too large to rely on more accurate yet expensive bounding methods. The experimental results show our pruning scheme is effective and our B&B method exactly solves protein sequence design cases that are very hard to solve with the dead-end elimination.

1 Introduction

A computational approach to the protein structure prediction problem is to solve the "inverse folding problem": to find a sequence or conformation that will fold to the target structure [3]. In this approach, the search of the minimum energy conformation is an important computational challenge. Two major applications where finding the minimum energy conformation is useful and necessary are the conformation modeling (homology modeling) problem [17] and the sequence design problem [9]. In finding the minimum energy conformation, the problem is discretized and simplified by computing the interaction energies only for some finite number of fixed side-chain conformations of each residue type [10]. These conformations are chosen by their statistical significance and are called rotamers. With the rotamer model, the energy function of a protein sequence folded into a specific template structure can be described in terms of [2]: (1) $E_{template}$ – the self-energy of a backbone template, (2) $E(i_r)$ – the interaction energy between the backbone and rotamer conformation r at ith position, (3) $E(i_r j_s)$ – the interaction energy between rotamer conformation r at position i and rotamer conformation s at position j, $i \neq j$. Then, the energy of a protein sequence in a specific template structure and conformation $C = \{i_r\}$ is written as $\mathcal{E}(C) = E_{template} + \sum_i E(i_r) + \sum_i \sum_{j>i} E(i_r j_s)$. Note that $E_{template}$ is constant by definition, and therefore can be ignored when minimizing $\mathcal{E}(C)$. A conformation that minimizes $\mathcal{E}(C)$ is often called the global minimum energy

conformation (GMEC). In this work, we call the problem of finding the GMEC for a given set of rotamers and energy terms as the "GMEC problem".

The GMEC problem is a strongly NP-hard optimization problem. 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) [2]. Popularly used elimination procedures such as Goldstein's conditions [6] combined with splitting [16], the magic bullet heuristic [7], and unification [6] are often able to reduce the problem size dramatically, while demanding only reasonable computational resources. However, we still find sequence design cases where DEE requires impractical amount of time and space. Other than DEE, there exist various exact approaches for the GMEC problem. Gordon and Mayo [8] used a variant of the branch-and-bound (B&B) method. Althaus et al. [1], Eriksson et al. [5], and Kingsford et al. [12] present integer linear programming (ILP) approaches. Leaver-Fay et al. [15] and Xu [20] describe methods based on tree-decomposition. Xie and Sahinidis [19] describes several residue-reduction and rotamer-reduction techniques. Each approach has advantages depending on the characteristics of the data, but most of them have not attempted to solve hard protein design cases, where there exist interactions between all possible pairs of positions and a large number of similar rotamers are allowed for each position. There also exist approximate approaches such as Yanover and Weiss [21] who used belief-propagation methods to solve side-chain placement problems.

In this work, we present an alternative exact solution method for the GMEC problem. Figure 1 illustrates the method. Our method consists of a B&B framework and a new subproblem-pruning scheme. The pruning scheme consists of upper/lower-bounding methods and problem-size reduction techniques. The basis for our upper/lower-bounding method is approximate *maximum-a-priori* (MAP) estimation. Particularly, we explore a way of using the tree-reweighted max-product algorithm (TRMP) [18]. The problem-size reduction techniques are necessary when TRMP can only compute weak bounds but the size of the subproblem is too large to rely on more accurate yet expensive bounding methods. Through an iterative use of several reduction techniques, we can obtain a problem of reasonable size that can be effectively lower-bounded. Such reduction techniques guarantee that the given subproblem can be pruned against an upper-bound U if the reduced subproblem can be pruned against U. On the other hand, if we are lucky, a subproblem can be also quickly solved using DEE only. The experimental results show that the running time of our pruning scheme is comparable to linear programming (LP) but our method is more effective in pruning subproblems than LP. We also find our B&B method exactly solves sequence design cases that are very hard to solve with DEE.

2 GMEC problem as MAP estimation

Probabilistic inference problems [11] involve a vector of random variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ characterized by a probability distribution $p(\mathbf{x})$. In this work, the GMEC problem is formulated as a MAP estimation problem that asks to find the

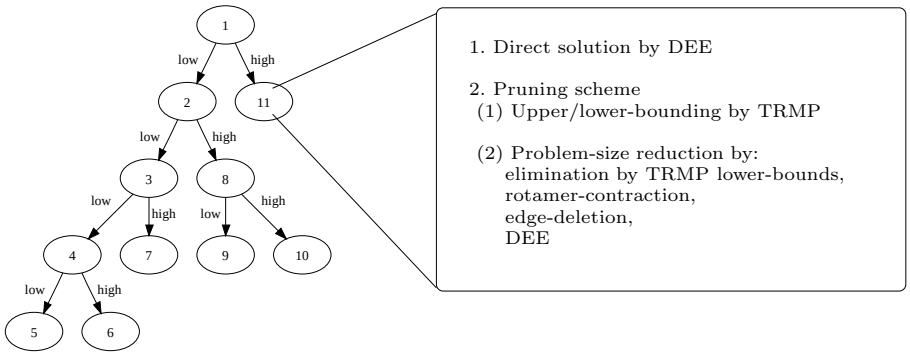


Fig. 1. An overview of the exact method for the GMEC problem. The method consists of a branch-and-bound framework and a pruning scheme, which in turn is composed of bounding by TRMP and a collection of problem-size reduction techniques. Labels on branches are related to the splitting scheme, and the numbers marked on the nodes correspond to the order by which the nodes are visited.

maximum a posteriori (MAP) assignment $\mathbf{x}^*$ such that $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x})$, 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 $\mathcal{X} = R_1 \times \ldots \times R_n$ as

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

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 $p(\mathbf{x})$. A probability distribution over a random vector can be related to a graphical model [11]. An undirected graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of vertices $\mathcal{V}$ for random variables and a set of edges $\mathcal{E}$ connecting some pairs of vertices. In the MAP estimation equivalent of our GMEC problem, the graphical model is generally a complete graph with n vertices.

Wainwright et al. [18] presents an algorithm called tree-reweighted max-product algorithm that can find a MAP assignment for loopy graphical models. The basic idea of the tree-reweighted max-product algorithm is to use a set of spanning-trees $\mathcal{T}$ such that every vertex and edge of $\mathcal{G}$ are covered by some $T \in \mathcal{T}$. Kolmogorov noted [13] that we may define $\mathcal{T}$ as a set of (not necessarily spanning) trees that cover the graph. In what follows, we will use a variant of Wainwright et al.'s algorithm that lets us use an arbitrary tree cover, and call it TRMP without presenting the details of the algorithm. Although TRMP is not guaranteed to always find the optimal solution, it can be used as an upper-bounding tool for the GMEC problem in the same way that the conventional max-product algorithm is used as an upper-bounding tool on loopy graphs [21].

In addition, it can also provide useful lower-bounds for the GMEC problem, which will be explained in Section 4.1.

3 General pair-flags

We use general pair-flags to constrain the conformation space $\mathcal{X}$. For example, if the pair-flag for (i_r, j_s) is set, all conformations in $\mathcal{Z} = \{\mathbf{x} \in \mathcal{X} | (x_i, x_j) = (r, s)\}$ are excluded from the search space, i.e. the GMEC problem is solved over $\mathcal{X} \backslash \mathcal{Z}$. However, unlike in DEE, this does not generally imply $\min_{\{\mathbf{x}|(x_i,x_j)=(r,s)\}} e(\mathbf{x}) > \min_{\mathbf{x}} e(\mathbf{x})$. We will denote the set of pair-flags for the given GMEC problem as $\tilde{P}$ (possibly empty) and define pair-flag functions from $\tilde{P}$ as $\tilde{g}_{ij}(r, s, \tilde{P}) = 1$ if $(i_r, j_s) \in \tilde{P}$, and 0 otherwise. We also let $\tilde{g}(\mathbf{x}, \tilde{P}) = \sum_{i,j \in \mathcal{V}, i \neq j} \tilde{g}_{ij}(x_i, x_j, \tilde{P})$. By defining $P(\{e\}, U) \stackrel{def}{=} \{(i_r, j_s) \mid \min_{\{\mathbf{x}|(x_i,x_j)=(r,s)\}} e(\mathbf{x}) > U\}$, we have the following lemma regarding minimization under pair-flag constraints:

Lemma 1. *For any $\tilde{P}$ and $\tilde{P}'$ such that $\tilde{P} \subset \tilde{P}'$ and $\tilde{P}' \backslash \tilde{P} \subset P(\{e\}, U)$, $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P}')=0\}} e(\mathbf{x})$ is either infeasible or greater than U if and only if $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x})$ is either infeasible or greater than U.*

The implication of Lemma 1 is that given a subproblem $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x})$ in the B&B-tree, the subproblem can be pruned if and only if the modified subproblem $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P}')=0\}} e(\mathbf{x})$ can be pruned. In addition, we can also show $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P}')=0\}} e(\mathbf{x}) = \min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x})$ if $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x}) \leq U$.

In what follows, when we need to mention pair-flag information, we will implicitly assume we have some $\tilde{P}$, and use the notation $\tilde{g}(\mathbf{x})$ instead of $\tilde{g}(\mathbf{x}, \tilde{P})$ where specifying $\tilde{P}$ is not particularly necessary. The following condition on pair-flags can be maintained without loss of generality and will be used in Section 4:

Condition 1 *For all $r \in R_i$ and $i \in \mathcal{V}$, there exists $s \in R_j$ for each $j \in \mathcal{V}, j \neq i$ such that $(i_r, j_s) \notin \tilde{P}$.*

4 Problem-size reduction techniques

4.1 Elimination by TRMP lower-bounds

We can exploit the properties of TRMP in computing a lower-bound of the minimum conformation energy for some given set of conformations. If such a lower-bound is greater than U, we can eliminate corresponding conformations from the problem while conserving the inequality relation between $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x})=0\}} e(\mathbf{x})$ and U. In addition, if $\min_{\{\mathbf{x}|\tilde{g}(\mathbf{x})=0\}} e(\mathbf{x}) \leq U$, the elimination does not change the optimal value. In this section, we first review the key properties of Wainwright et al.'s algorithm – ρ-reparameterization and tree-consistency of pseudo-max-marginals, before presenting how to compute the lower-bounds. Note that TRMP shares these properties.

Single max-marginals μ_i [18] are defined as the maximum of $p(\mathbf{x})$ when one of the variable x_i is fixed, 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 $\mu_{ij}(x_i, x_j) = \kappa_{ij} \max_{\{\mathbf{x}'|(x'_i,x'_j)=(x_i,x_j)\}} p(\mathbf{x}')$. Note that κ_i and κ_{ij} are constants that can vary depending on i or j. It is known that any tree-distribution $p(\mathbf{x})$ can be factored in terms of its max-marginals as $p(\mathbf{x}) \propto \prod_{i\in\mathcal{V}} \mu_i(x_i) \prod_{(i,j)\in\mathcal{E}} \frac{\mu_{ij}(x_i,x_j)}{\mu_i(x_i)\mu_j(x_j)}$. Max-marginals for tree-distributions can be exactly computed by the conventional max-product algorithm. Wainwright et al. [18] use the notion of pseudo-max-marginals. By construction, pseudo-max-marginals $\nu = \{\nu_i, \nu_{ij}\}$ from the tree-reweighted max-product algorithm satisfy ρ-reparameterization, i.e.

$$p(\mathbf{x}) \propto \prod_{T\in\mathcal{T}} \left[\prod_{i\in\mathcal{V}(T)} \nu_i(x_i) \prod_{(i,j)\in\mathcal{E}(T)} \frac{\nu_{ij}(x_i,x_j)}{\nu_i(x_i)\nu_j(x_j)} \right]^{\rho(T)}, \tag{2}$$

where $\rho(T) = \frac{|\{T\in\mathcal{T}\}|}{|\mathcal{T}|}$. A tree-distribution $p^T(\mathbf{x};\nu)$ for given pseudo-max-marginals can be defined as

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

Then, we have $p(\mathbf{x}) \propto \prod_{T\in\mathcal{T}} \{p^T(\mathbf{x};\nu)\}^{\rho(T)}$ from (2). On the other hand, pseudo-max-marginals ν^* at convergence of the tree-reweighted max-product algorithm satisfy tree-consistency condition with respect to every spanning tree $T \in \mathcal{T}$. More precisely, ν^* is tree-consistent with respect to a spanning tree T if it satisfies $\nu_i^*(x_i) \propto \max_{x_j\in R_j} \nu_{ij}^*(x_i, x_j)$ for all $x_i \in R_i$ and $(i,j) \in \mathcal{E}(T)$.

In what follows, we assume ν is in a normal form [13], i.e. $\max_{r\in R_i} \nu_i(r) = 1$ for all $i \in \mathcal{V}$, and $\max_{(r,s)\in R_i\times R_j} \nu_{ij}(r,s) = 1$ for all $(i,j) \in \mathcal{E}$. Then, since ν always satisfies ρ-reparameterization, rearranging the terms of (2) gives, for some constant $\nu_c > 0$,

$$p(\mathbf{x}) = \nu_c \prod_{i\in\mathcal{V}} \nu_i(x_i)^{\rho_i} \prod_{(i,j)\in\mathcal{E}} \left(\frac{\nu_{ij}(x_i,x_j)}{\nu_i(x_i)\nu_j(x_j)} \right)^{\rho_{ij}},$$

where $\rho_{ij} = \frac{|\{T\in\mathcal{T} \mid (i,j)\in\mathcal{E}(T)\}|}{|\mathcal{T}|}$ and $\rho_i = \frac{|\{T\in\mathcal{T} \mid i\in\mathcal{V}(T)\}|}{|\mathcal{T}|}$.

The following lemmas show how we can compute lower-bounds for some sets of conformations. For example, Lemma 2 combined with (1) can provide rotamer lower-bounds i.e. a lower-bound of $\min_{\{\mathbf{x}|x_\zeta=r\}} e(\mathbf{x})$ for each $r \in R_\zeta$ and $\zeta \in \mathcal{V}$:

Lemma 2. *When ν satisfies the tree-consistency condition, we have, for all $r \in R_\zeta$, $\zeta \in \mathcal{V}$, $\max_{\{\mathbf{x}|x_\zeta=r\}} p(\mathbf{x}) \le \nu_c \nu_\zeta(r)^{\rho_\zeta}$.*

For rotamer-pair lower-bounds, i.e. to lower-bound $\min_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} e(\mathbf{x})$, we use $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p(\mathbf{x}) \le \nu_c \prod_{T\in\mathcal{T}} \left[\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x})\right]^{\rho(T)}$, where $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x})$ for each T can be easily solved using Lemma 3 when we let $\mathcal{T} = \mathcal{S}$, a set of stars:

Lemma 3. *When ν satisfies the tree-consistency condition, the following inequalities hold:*

1. *if $\zeta, \eta \notin \mathcal{V}(T)$, then $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x}) = 1$.*

2. *if* $\zeta \in \mathcal{V}(T)$ *and* $\eta \notin \mathcal{V}(T)$, *then* $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x}) = \nu_\zeta(r)$.
3. *if* $(\zeta,\eta) \in \mathcal{E}(T)$, *then* $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x}) = \nu_{\zeta\eta}(r,s)$.
4. *if* $\zeta,\eta \in \mathcal{V}(T)$ *and* $(\zeta,\eta) \notin \mathcal{E}(T)$ *for a star* T *(let* ξ *be the center of* T*), then* $\max_{\{\mathbf{x}|(x_\zeta,x_\eta)=(r,s)\}} p^T(\mathbf{x}) = \max_{x_\xi \in R_\xi} \frac{\nu_{\xi\zeta}(x_\xi,r)\nu_{\xi\eta}(x_\xi,s)}{\nu_\xi(x_\xi)}$.

If we use pair-flags, we may improve rotamer lower-bounds by the inequality $\max_{\{\mathbf{x}|x_\zeta=r,\tilde{g}(\mathbf{x})=0\}} p(\mathbf{x}) \leq \nu_c \prod_{T\in\mathcal{T}} \left[\max_{\{\mathbf{x}|x_\zeta=r,\tilde{g}(\mathbf{x})=0\}} p^T(\mathbf{x})\right]^{\rho(T)} \leq \nu_c \nu_\zeta(r)^{\rho_\zeta}$, which holds for tree-consistent ν. Let n_{rot} be the average number of rotamers per position. If we use a naive search, it takes $O(n_{rot}^2 n)$ comparison operations to exactly solve $\max_{\{\mathbf{x}|x_\zeta=r,\tilde{g}(\mathbf{x})=0\}} p^T(\mathbf{x})$. Therefore, computing an improved lower-bound for a rotamer takes $O(n_{rot}^2 n^2)$ since $|\mathcal{T}| = O(n)$.

4.2 Rotamer-contraction

The idea of rotamer contraction is to reduce the number of rotamers at one selected position by first clustering similar rotamers of the position and replacing all rotamers in each cluster with one rotamer-aggregate. Let ζ be the position whose rotamers we partition into a number of clusters $C_1,\ldots,C_l$, $l < |R_\zeta|$. Then, we contract all rotamers $r \in C_k$ as one rotamer-aggregate c_k. The contracted GMEC problem has a new conformation space $\mathcal{X}^{rc}$, which is same as $\mathcal{X}$ except that R_ζ is replaced by $\{c_1,\ldots,c_l\}$. Then, we define a new energy function $e^{rc}(\mathbf{x})$ over $\mathcal{X}^{rc}$ and the set of pair-flags $\tilde{P}^{rc}$ so that the optimal value of the contracted problem $\min_{\{\mathbf{x}\in\mathcal{X}^{rc}|\tilde{g}(\mathbf{x},\tilde{P}^{rc})=0\}} e^{rc}(\mathbf{x})$ is a lower-bound of $\min_{\{\mathbf{x}\in\mathcal{X}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x})$. One way of choosing $e^{rc}(\mathbf{x})$ for a given clustering is given by *contract-rotamers* in Algorithm 1. We use notation $e^{rc}(\mathbf{x},\tilde{P})$ to indicate the function is also defined by $\tilde{P}$. A lower-bounding technique similar to rotamer-contraction is used by Koster et al. [14] for the frequency assignment problem. We have the following lemma on *contract-rotamers*:

Lemma 4. *For any given clustering of rotamers of* $\zeta \in \mathcal{V}$, *if* $\{\mathbf{x} \in \mathcal{X}|\tilde{g}(\mathbf{x},\tilde{P}) = 0\} \neq \phi$, *then* $\min_{\{\mathbf{x}\in\mathcal{X}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x}) \geq \min_{\{\mathbf{x}\in\mathcal{X}^{rc}|\tilde{g}(\mathbf{x},\tilde{P}^{rc})=0\}} e^{rc}(\mathbf{x},\tilde{P})$.

In rotamer-contraction, how we cluster rotamers of position ζ determines the quality of resulting lower-bounds. Our approach is a greedy scheme that keeps placing rotamers in a cluster as long as the decrease in the optimal value is less than or equal to a specified amount. However, it is hard to exactly know the decrease $\min_{\{\mathbf{x}\in\mathcal{X}|\tilde{g}(\mathbf{x},\tilde{P})=0\}} e(\mathbf{x}) - \min_{\{\mathbf{x}\in\mathcal{X}^{rc}|\tilde{g}(\mathbf{x},\tilde{P}^{rc})=0\}} e^{rc}(\mathbf{x},\tilde{P})$. In addition, it is generally not feasible to bound the decrease since rotamer-contraction may even turn an infeasible subproblem into a feasible one. We instead upper-bound $\Delta OPT^{rc} \stackrel{def}{=} \min_{\mathbf{x}\in\mathcal{X}} e(\mathbf{x}) - \min_{\{\mathbf{x}\in\mathcal{X}^{rc}|\tilde{g}(\mathbf{x},\tilde{P}^{rc})=0\}} e^{rc}(\mathbf{x},\tilde{P})$. Let $U^{rc}_{\Delta OPT}(\tilde{P}) = \max_{k=1,\ldots,l} \min_{r\in C_k} \sum_{j\in\Gamma(\zeta)} \max_{\{s\in R_j|(\zeta_{c_k},j_s)\notin\tilde{P}^{rc}\}} \{e_{\zeta j}(r,s) + \frac{e_\zeta(r)}{|\Gamma(\zeta)|} - e^{rc}_{\zeta j}(c_k,s,\tilde{P})\}$. Then, we have the following lemma:

Lemma 5. *For any given clustering of rotamers of* $\zeta \in \mathcal{V}$, *we have* $\Delta OPT^{rc} \leq U^{rc}_{\Delta OPT}(\tilde{P}) \leq U^{rc}_{\Delta OPT}(\phi)$

Algorithm 1: contract-rotamers

Data: ζ, $C_1, \ldots, C_l$, $\mathcal{X}$, $\{e\}$, $\tilde{P}$
Result: $\mathcal{X}^{rc}, \{e^{rc}\}, \tilde{P}^{rc}$
begin
  $\mathcal{X}^{rc}$ is same with $\mathcal{X}$ except R_ζ is replaced with $\{c_1, \ldots, c_l\}$
  $\tilde{P}^{rc} \leftarrow \tilde{P} \backslash \{(\zeta_r, j_s), j \in \mathcal{V}, j \neq \zeta\}$
  foreach C_k, $k = 1, \ldots, l$ **do**
    foreach $s \in R_j$, $j \in \mathcal{V}, j \neq \zeta$ **do**
      $e^{rc}_{\zeta j}(c_k, s, \tilde{P}) \leftarrow \min_{\{r \in C_k, (\zeta_r, j_s) \notin \tilde{P}\}} e_{\zeta j}(r, s) + \frac{e_\zeta(r)}{|\Gamma(\zeta)|}$,
      if $(\zeta_r, j_s) \in \tilde{P}$ *for all* $r \in C_k$ **then** $\tilde{P}^{rc} \leftarrow \tilde{P}^{rc} \cup (\zeta_{c_k}, j_s)$
    $e^{rc}_\zeta(c_k) \leftarrow 0$.
  define $e^{rc}(\mathbf{x})$ same as $e(\mathbf{x})$ for other terms
end

Note that $U^{rc}_{\Delta OPT}(\tilde{P})$ has a finite value due to Condition 1. Lemma 4 and Lemma 5 suggests rotamer-contraction may benefit from the use of pair-flags by smaller decrease in the optimal value, and better upper-bounding of the decrease. We include a rotamer in a cluster if $U^{rc}_{\Delta OPT}(\tilde{P})$ from the inclusion is less than some constant Δ^{rc}. When $\min_{\mathbf{x}} e(\mathbf{x}) > U$, Δ^{rc} can be allowed to be at most $\min_{\mathbf{x}} e(\mathbf{x}) - U$ or some fraction of it. Since we do not know the exact value of $\min_{\mathbf{x}} e(\mathbf{x})$, Δ^{rc} is heuristically set as a fraction of the difference between an upper-bound of $\min_{\mathbf{x}} e(\mathbf{x})$ and U. Both upper-bounds are obtained by TRMP.

4.3 Edge-deletion

In edge-deletion, we first identify a pair of positions (ζ, η) such that the deviation in $e_{\zeta\eta}(r, s)$ for all $(r, s) \in R_\zeta \times R_\eta$ is small, then set all the pairwise energies of (ζ, η) to the minimum of the pairwise energies. That is, the new energy function $e^{ed}(\mathbf{x})$ will be defined to be the same as $e(\mathbf{x})$ except $e^{ed}_{\zeta\eta}(r, s) = \min_{\{(r,s) \in R_\zeta \times R_\eta | (\zeta_r, \eta_s) \notin \tilde{P}\}} e_{\zeta\eta}(r, s)$, for all $(r, s) \in R_\zeta \times R_\eta$. Since $e^{ed}_{\zeta\eta}(x_\zeta, x_\eta)$ is constant, we can ignore the interaction of (ζ, η) and replace $\mathcal{E}$ by $\mathcal{E} \backslash (\zeta, \eta)$. The same idea is explored by Xie and Sahinidis [19] as an approximation procedure. Some advantages of doing edge-deletion are: (1) when the graph becomes sparse, we may use direct solution techniques such as dynamic programming. (2) Empirically, being able to cover the graph with fewer trees is favorable for obtaining tighter lower-bounds from TRMP. (3) Rotamer-contraction may obtain fewer clusters for the same Δ^{rc}. The pair-flags are kept intact through edge-deletion even for the edge being deleted. Then, it is straightforward to obtain similar properties for edge-deletion as Lemma 4 and 5.

5 Branch-and-bound framework

We split a subproblem by dividing rotamers of a position into two groups by their rotamer lower-bounds. If the conformation space of the current subprob-

lem F^i is defined by rotamer sets $\{R_i\}$ and we decide to split it into $F^{i,low}$ (low-subproblem) and $F^{i,high}$ (high-subproblem), we can define the conformation space for each with $\{R_i^{low}\}$ and $\{R_i^{high}\}$, respectively, where R_i^{low} and R_i^{high} are defined the same as R_i except $R_\zeta^{low} \cup R_\zeta^{high} = R_\zeta$, $|R_\zeta^{low}| \approx |R_\zeta^{high}|$, and $LB(\zeta_r) \leq LB(\zeta_s)$ for all $r \in R_\zeta^{low}$, $s \in R_\zeta^{high}$ ($LB(\zeta_r)$ is a rotamer lower-bound for ζ_r). The goal of such a splitting scheme is to make the optimal value of $F^{i,low}$ likely to be less than that of $F^{i,high}$. We prefer a splitting position ζ whose difference between maximum and minimum rotamer lower-bounds is large. Subproblems are selected by a mix of what are called "best-first" and "depth-first" strategies: (1) follow the depth-first strategy, (2) always dive into $F^{i,low}$ first when the current subproblem F^i is split. The goal is first to find a good upper-bound by following depth-first through the low-subproblems from the first series of splittings, then to prune the remaining subproblems using the upper-bound. Figure 1 shows an example B&B-tree that can result from our splitting scheme and subproblem-selection strategy, where optimal solution from node 5 is supposed to provide a near-optimal upper-bound.

6 Experimental Results

In our numerical experiments, a Linux workstation with a 2.2 GHz Intel Xeon processor and 3.5 GBytes of memory was used. Table 1 shows 12 protein design cases used in the experiments. DEE on each case was performed with the following options: Goldstein's singles elimination, splitting with split flags ($s = 1$), Goldstein's pair elimination with one magic bullet, and unification allowing maximum 6,000 rotamers per position. `E-9` was finished in 4.8 hours but none of others were solved within 48 hours.

Table 1. Test cases facts. All cases are from antigen-antibody model system. Each case repacks either the antigen protein or the antibody, or both. Each column represents (1) case name, (2) number of positions, (3) maximum number of rotamers offered at a position, (4) number of total rotamers, (5) $\sum_{i=1}^{n} \log_{10} |R_i|$, (6) case composition ('m' – #positions allowed to mutate, 'n' – #positions only wild-types are allowed, 'w' – #water molecules to be oriented at the interface). In the case names, `R` uses the standard rotamer library, and `E` multiplies each of χ_1 and χ_2 by a factor of 3 by adding $\pm 10^\circ$. `E-1` were offered only hydrophobic residues while others were offered both hydrophobic and polar residues. All energies were calculated using the CHARMM package and the parameter set 'param22'

Case	n	max $\lvert R_i \rvert$	$\sum \lvert R_i \rvert$	$\log_{10} conf$	Composition	Case	n	max $\lvert R_i \rvert$	$\sum \lvert R_i \rvert$	$\log_{10} conf$	Composition
`R-1`	34	125	1422	30.0	34 m	`E-5`	24	1344	9585	49.6	24 m
`R-2`	30	133	1350	40.2	30 m	`E-6`	36	1984	8543	59.1	4 m, 32 n
`E-1`	19	617	3675	38.1	19 m	`E-7`	10	2075	5201	21.9	5 m, 3 n, 2 w
`E-2`	23	1370	9939	52.3	23 m	`E-8`	10	1915	5437	20.7	4 m, 4 n, 2 w
`E-3`	23	1320	8332	49.1	23 m	`E-9`	15	2091	5700	25.1	3 m, 6 n, 6 w
`E-4`	15	1361	7467	33.9	15 m	`E-10`	23	1949	9837	42.5	7 m, 7 n, 9 w

We first show an example use of TRMP lower-bounds in eliminating rotamers or rotamer-pairs of subproblems from `E-10`. In the following, we use the notation "`i-high`" to denote the high-subproblem at depth i spawned from the first depth-first dive along the low-subproblems (root node is at depth 1).

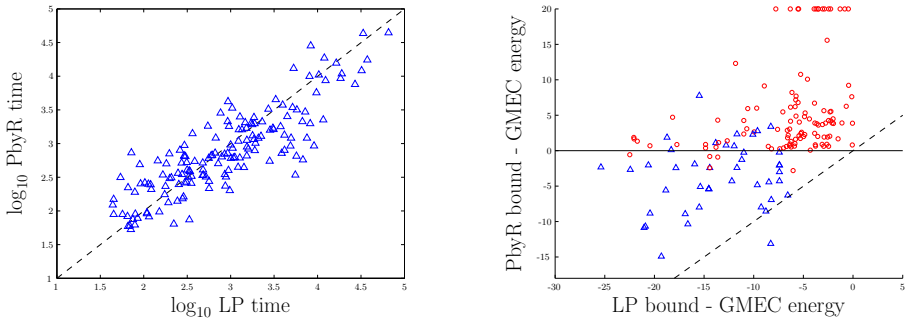
For example, in Figure 1, node 11 is `2-high` and node 6 is `5-high`. Table 2 shows lower-bounding results for subproblems at depth 2 to 11. In `2-high`, simple rotamer lower-bounds were able to eliminate 39% of rotamers. However, we obtain even more elimination when we use rotamer lower-bounds computed using pair-flags. This is due to massive flagging of rotamer-pairs by rotamer-pair lower-bounds. Large elimination obtained for subproblems at small depth are due to our splitting scheme of dividing rotamers by their lower-bounds.

Table 2. TRMP lower-bounding results for subproblems of `E-10`. The meaning of each column is, in order: (1) subproblem, (2) number of rotamers, (3) number of rotamer-pairs, (3) median rotamer lower-bound(lb) when not using pair-flags, (4) number of rotamers such that $lb > U$, (5) median rotamer lower-bound when using pair-flags, (5) number of rotamers such that $lb > U$, (6) median rotamer-pair lower-bound, (7) number of rotamer-pairs such that $lb > U$. The value of U is -325.038. $+\infty$ implies the lower-bounding problem turned out to be infeasible due to pair-flags.

			Rot-lb's w/o pair-flags		Rot-lb's w/ pair-flags		Rot-pair lb's	
Subprob.	#rots	#rot-pairs	med. lb	#rots $lb > U$	med. lb	#rots $lb > U$	med. lb	#pairs $lb > ub$
`2-high`	3,345	4,769,691	-332.831	1,301	$+\infty$	2,220	-312.544	4,071,145
`3-high`	3,022	3,879,787	-315.025	1,134	$+\infty$	2,141	-313.614	3,328,424
`4-high`	2,665	3,036,834	-315.689	1,380	$+\infty$	2,273	-313.276	2,799,272
`5-high`	2,281	2,299,981	-336.173	81	-336.173	920	-323.780	1,292,520
`6-high`	2,171	2,071,431	-343.019	0	-343.019	200	-330.363	590,576
`7-high`	1,964	1,702,980	-342.556	8	-342.556	215	-329.554	508,750
`8-high`	1,848	1,499,857	-344.865	0	-344.636	42	-335.640	218,324
`9-high`	1,669	1,223,065	-337.791	0	-337.791	289	-329.037	384,812

To evaluate our pruning scheme, we compared it (call it PbyR: *prune-by-reduction*) against linear programming (LP). We used subproblems of various sizes generated while solving the design cases of Table 1 with our B&B method. We used the LP formulation given by Wainwright et al. [18] and solved it with a C++ procedure using CPLEX 8.0 library. In PbyR, we alternated rotamer-contraction and edge-deletion at every iteration. At every 8th reduction, we applied DEE to see if we could solve the reduced problem or only to flag more rotamer/rotamer-pairs. (Note that we adapted DEE to make it compatible with general pair-flags.) We computed TRMP lower-bounds at every 24th reduction and flagged rotamers/rotamer-pairs. We allowed at most 300 reductions until we find a lower-bound greater than U or exactly solve the reduced problem. Figure 6 shows the result for the 156 subproblems remaining after excluding the subproblems that could be solved quickly by DEE alone. The bounding times of the two methods are comparable although LP is slightly faster in small to medium-sized subproblems. However, Figure 6 (b) shows that the bounds from PbyR are greater than LP bounds except for the one data point below the $y = x$ line. Note that a PbyR bound for a subproblem is not generally a lower-bound of the subproblem's optimal value since rotamer/rotamer-pair elimination by TRMP lower-bounds can also increase the optimal value. However, a PbyR bound is greater than U only if the original subproblem's optimal value is greater than U. Therefore, if we had U equal to the GMEC energy for each design case, we could immediately prune the subproblems corresponding to the data points over the horizontal solid line in Figure 6 (b). There was no such case with LP among

the tested subproblems. Figure 6 (b) suggests that performing reductions more than 50 times often resulted in lower-bounds that were useless for pruning.



(a) $\log_{10}$ LP time vs. $\log_{10}$ PbyR time.

(b) (LP bound - GMEC energy) vs. (PbyR bound - GMEC energy).

Fig. 2. Comparison of LP and PbyR in pruning subproblems from B&B. In (b), a circle represents the PbyR bound was computed using less than 50 reductions. Also in (b), points such that PbyR bound - GMEC energy ≥ 20 were all clamped at 20

Finally, we used the B&B method of Figure 1 to solve each design case. The branch-and-bound method was implemented in C++ using the PICO-library [4] as a sequential B&B framework. At each node of the B&B method, we first eliminated rotamers using DEE with the same set of options mentioned earlier. When singles-elimination condition of DEE fails to eliminate any rotamer, we let TRMP lower-bounds eliminate more rotamers. Then, we used the reduction techniques iteratively in the same mix as we used for comparison test against LP, but limited the number of reductions to be at most four times the depth of the node in the B&B-tree. When branching was necessary, the subproblem located at the end of the first dive usually had $\sum_{i=1}^{n} \log_{10} |R_i| \leq 13$ and was exactly solved by DEE. Table 3 shows the result. We were able to solve six cases at the root node without branching. Considering DEE couldn't finish five of them for 48 hours, rotamer/rotamer-pair elimination using TRMP lower-bounds enormously reduced the solution time. All cases were also solved efficiently except `E-10` where the upper-bounds (from TRMP) of the subproblems were often very close to the GMEC energy. However, in all cases, the number of total branching is only moderately larger than that from the first dive. In all cases where branching was necessary, the upper-bound obtained at the end of the first dive was equal to the GMEC energy, confirming that our branching scheme and subproblem-selection strategy meets expectations.

Table 3. Solving the design cases using our B&B method. Each column represents (1) case name, (2) number of branches, (3) number of branches from the first depth-first dive along the low-subproblems, (4) total solution time

Case	# Br.	#F.D.Br.	Time (h)	Case	# Br.	#F.D.Br.	Time (h)	Case	# Br.	#F.D.Br.	Time (h)
R-1	0	0	1.1	E-3	0	0	6.4	E-7	15	12	6.9
R-2	14	14	2.7	E-4	0	0	4.2	E-8	17	12	13.7
E-1	28	28	9.7	E-5	8	8	27.2	E-9	0	0	3.3
E-2	0	0	6.2	E-6	0	0	5.6	E-10	202	35	139.1

7 Conclusion

In this work, we presented an exact solution method for the GMEC problem. Our branch-and-bound method using the suggested pruning scheme was able to solve hard sequence design cases that DEE couldn't solve within practical resources levels. There is certainly a decision-making flavor in using our proposed pruning scheme since a trade-off between the amount of pruning effort and the quality of the final bound should be considered in deciding when to stop the pruning attempt and to split the subproblem. Therefore, future work may include a systematic allocation of pruning effort throughout the B&B-tree for faster solution.

Acknowledgment The authors would like to thank Bruce Tidor for suggesting the problem and for helpful advice. Shaun Lippow, Alessandro Senes, and Michael Altman gave freely of the test cases, and the DEE code.

References

1. E. Althaus, O. Kohlbacher, H.-P. Lenhof, and P. Müller. A combinatorial approach to protein docking with flexible side-chains. *Journal of Computational Biology*, 9(4):597–612, 2002.
2. J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356:539–542, 1992.
3. K. E. Drexler. Molecular engineering: an approach to the development of general capabilities for molecular manipulation. *PNAS (USA)*, 78:5275–5278, 1981.
4. J. Eckstein, C. A. Phillips, and W. E. Hart. Pico: an object oriented framework form parallel branch and bound. Technical report, RUTCOR, 2001.
5. O. Eriksson, Y. Zhou, and A. Elofsson. Side chain-positioning as an integer programming problem. In *WABI '01*, volume 2149 of *LNCS*, pages 128–141. Springer.
6. R. F. Goldstein. Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophysical Journal*, 66:1335–1340, 1994.
7. D. B. Gordon and S. L. Mayo. Radical performance enhancements for combinatorial optimization algorithms based on the dead-end elimination theorem. *Journal of Computational Chemistry*, 13:1505–1514, 1998.
8. D. B. Gordon and S. L. Mayo. Branch-and-terminate: a combinatorial optimization algorithm for protein design. *Structure with Folding and Design*, 7(9):1089–1098, 1999.
9. H. W. Hellinga and F. M. Richards. Optimal sequence selection in proteins of known structure by simulated evolution. *PNAS (USA)*, 91:5803–5807, 1994.

10. J. Janin, S. Wodak, M. Levitt, and B. Maigret. Conformation of amino-acid side-chains in proteins. *Journal of Molecular Biology*, 125:357–386, 1978.
11. M. I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, 19:140–155, 2004.
12. C. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1036, 2005.
13. V. Kolmogorov. Convergence tree-reweighted message passing for energy minimization. Technical Report MSR-TR-2005-38, Microsoft Research, June 2005.
14. A. M. Koster, S. P. van Hoesel, and A. W. Kolen. Lower bounds for minimum interference frequency assignment problems. Technical Report RM 99/026, Maastricht University, October 1999.
15. A. Leaver-Fay, B. Kuhlman, and J. Snoeyink. An adaptive dynamic programming algorithm for the side-chain placement problem. In *Pacific Symposium on Biocomputing*, pages 16–27, Singapore, 2005. World Scientific.
16. N. A. Pierce, J. A. Spriet, J. Desmet, and S. L. Mayo. Conformational splitting: a more powerful criterion for dead-end elimination. *Journal of Computational Chemistry*, 21:999–1009, 2000.
17. M. Vasquez. Modeling sidechain conformation. *Current Opinion in Structural Biology*, 6:217–221, 1996.
18. M. J. Wainwright, T. S. Jaakola, and A. S. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. Technical Report UCB/CSD-3-1269, Computer Science Division (EECS), UC Berkeley, 2003.
19. W. Xie and N. V. Sahinidis. Residue-rotamer-reduction algorithm for the protein side-chain conformation problem. *Bioinformatics*, 22(2):188–194, 2006.
20. J. Xu. Rapid protein side-chain packing via tree decomposition. In *RECOMB '05*, volume 3500 of *LNCS*, pages 423–439. Springer.
21. C. Yanover and Y. Weiss. Approximate inference and protein-folding. In *Proceedings of Neural Information Processing Systems*, 2002.