# Protein side-chain placement: probabilistic inference and integer programming methods

Eun-Jong Hong, Tomás Lozano-Pérez
MIT Computer Science and Artificial Intelligence Laboratory

***Abstract*—The prediction of energetically favorable side-chain conformations is a fundamental element in homology modeling of proteins and the design of novel protein sequences. The space of side-chain conformations can be approximated by a discrete space of probabilistically representative side-chain conformations (called rotamers). The problem is, then, to find a rotamer selection for each amino acid that minimizes a potential energy function. This is called the Global Minimum Energy Conformation (GMEC) problem. This problem is an $NP$-hard optimization problem. The Dead-End Elimination theorem together with the A$^*$ algorithm (DEE/A$^*$) has been successfully applied to this problem. However, DEE fails to converge for some complex instances.**

**In this paper, we explore two alternatives to DEE/A$^*$ in solving the GMEC problem. We use a probabilistic inference method, the max-product (MP) belief-propagation algorithm, to estimate (often exactly) the GMEC. We also investigate integer programming formulations to obtain the exact solution. There are known ILP formulations that can be directly applied to the GMEC problem. We review these formulations and compare their effectiveness using CPLEX optimizers. We also present preliminary work towards applying the branch-and-price approach to the GMEC problem.**

**The preliminary results suggest that the max-product algorithm is very effective for the GMEC problem. Though the max-product algorithm is an approximate method, its speed and accuracy are comparable to those of DEE/A$^*$ in large side-chain placement problems and may be superior in sequence design.**

***Index Terms*—protein side-chain placement, protein design, belief propagation, integer programming, computational biology**

## I. Introduction

THE biological function of a protein is determined by its three-dimensional structure. However, the widely used experimental techniques for determining the structures of proteins, X-ray crystallography and NMR spectroscopy, are currently expensive and time-consuming. Alternative computational approaches for protein structure prediction, such as homology modeling [1], and for the design of novel protein sequences [2] are often based on the prediction of energetically favorable side-chain conformations given a known and fixed protein backbone.

In these approaches the space of side-chain conformations is often approximated by a discrete space of probabilistically representative side-chain conformations (called rotamers) [3]. With the rotamer model, the total energy function of a protein is described in terms of:

1. the self-energy of a backbone template (denoted as $e_{backbone}$)
2. the interaction energy between the backbone and residue $i$ in its rotamer conformation $r$ in the absence of other free side-chains (denoted as $e_{i_r}$)
3. the interaction energy between residue $i$ in the rotamer conformation $r$ and residue $j$ in the rotamer conformation $s$, $i \neq j$ (denoted as $e_{i_r j_s}$)

The total energy of a protein in a specific conformation $C$ can be written as

$$\mathcal{E}_C = e_{backbone} + \sum_i e_{i_r} + \sum_i \sum_{j>i} e_{i_r j_s}. \qquad (1)$$

Then, to determine the most energetically favorable side-chain conformation, we find a rotamer selection for modeled residues that minimizes the energy function $\mathcal{E}_C$, which is often called global minimum energy conformation (GMEC). In this work, we call the problem of finding the energetically optimal rotamer selection, the GMEC problem[1].

One can readily show that the GMEC problem is a strongly $NP$-hard optimization problem (by reduction from $3SAT$). Despite the theoretical hardness, one finds that many instances of the GMEC problem are easily solved by exact methods such as Dead-End Elimination (DEE) combined with A$^*$ (DEE/A$^*$) [4] [5]. However, DEE's elimination criteria are not always powerful enough to reduce a problem's complexity significantly. Though there have been many modifications and improvements over the original elimination conditions [6] [7] [8], there is still room for alternative approaches, especially in applications to sequence design.

In this paper, we explore two additional approaches to solving the GMEC problem. We first use a probabilistic inference method, the max-product (MP) belief-propagation algorithm [9], to infer the GMEC using the energy terms translated into "potentials" on a graphical model [10]. We also explore integer programming formulations to obtain exact solutions. There are known ILP formulations that can be directly applied to the GMEC problem. We review these formulations and compare their effectiveness using CPLEX optimizers. We also present results from preliminary attempts to apply the branch-and-price approach [11] to the GMEC problem. We review the

Both authors are with the M.I.T. Computer Science and Artificial Intelligence Laboratory, 200 Technology Square, Cambridge, MA 02139, USA, email: eunjong@ai.mit.edu, tlp@mit.edu

[1] In this paper, we limit our attention to computational aspects of the GMEC problem, therefore, issues such as protein energy models and characteristics of alternative rotamer libraries are outside our scope.

algorithm and suggest possible decomposition schemes.

The preliminary results suggest that the max-product (MP) algorithm is the most effective among the methods we explored. Though the max-product algorithm is an approximate method, its speed and accuracy are comparable to those of DEE/A* in large side-chain placement problems, and overall superior for sequence design problems. The ILP approach, while not competitive on larger problems, does solve medium size problems exactly.

## II. Related work

Voigt et al. [12] examined the performance of DEE with other well-known methods such as Monte Carlo, genetic algorithms, and self-consistent mean-field and concluded that DEE is the most feasible method. Below we briefly review related work on the two approaches we explore in detail in this paper.

### A. Probabilistic methods

Early work using the self-consistent mean field theory was done by Koehl and Delarue [13]. The method calculates the mean field energy as the sum of interaction energy weighted by the conformational probabilities. The conformational probabilities are related to the mean field energy by the Boltzmann law. Iterative updates of the probabilities and the mean field energy are performed until they converge. At convergence, the rotamer with the highest probability from each residue is selected as the conformation. The method is not exact, but has linear time complexity.

Yanover and Weiss [14] applied belief propagation, generalized belief propagation [15], and the mean field method to finding minimum energy side-chain configuration and compared the results with those from SCWRL [16], a protein-folding program. Their energy function is approximate in that only local interactions between neighboring residues are considered, which results in sparse graphical models. The energies found by each method are compared with those from one another, rather than with optimal values from exact methods.

### B. Integer linear programming (ILP)

The polyhedral approach is a popular technique for solving hard combinatorial optimization problems. The main idea behind the technique is to iteratively strengthen the LP formulation by adding violated valid inequalities.

Althaus et al. [17] presented an ILP approach for side-chain demangling in rotamer representation of the side chain conformation. Using an ILP formulation, they identified classes of facet-defining inequalities and devised a separation algorithm for a subclass of inequalities. On average, the branch-and-cut algorithm was about five times slower than their heuristic approach.

Eriksson et al. [18] also formulated the side chain placement problem as an ILP problem. However, in their computational experiments, they found that the LP relaxation of every test instance has an integral solution and, therefore, the integer programming (IP) techniques were not necessary. They conjecture that the GMEC problem might always have integral solutions in LP relaxation, which we have found not to be true in practice.

## III. Methods

In this section, we present the details of the two methods that we have explored for protein side-chain placement: the max-product algorithm (MP) for probabilistic inference and several formulations of integer linear programming (ILP).

### A. Probabilistic inference

A wide variety of problems, ranging from error-correcting codes, speech recognition and image understanding, can be phrased as probabilistic inference problems [15]. These problems involve a set of random variables that can take on some set of values and a probability distribution over variable assignments, specified via prior probabilities on the individual variable assignments and compatibility functions governing the probability of assignments to, for example, pairs of variables. This type of problem can be represented as a graphical model in which the nodes represent variables and the links represent pairwise compatibility functions. One important inference problem in this context is that of finding the maximum a posteriori (MAP) assignment of a set of unobserved variables given the values of some observed variables. The max-product (MP) algorithm has proven to be extremely effective for this type of problem [9].

We use probabilistic inference methods to infer the GMEC using variables, representing residues, whose values range over rotamers. The energy terms are used to define the prior probabilities of each rotamer value and the compatibility functions for pairwise assignments of rotamers to residues. Probabilistic inference methods are generally approximate methods, when the graphical model has loops, but empirically their computation time is short and the solutions found are usually very close to optimal.

#### A.1 Energy minimization as probabilistic inference

The significant link that connects energy minimization and probabilistic inference is the Boltzmann law, which relates the probability of a state to its potential energy. As we indicated above, probabilistic inference generally requires defining the prior probabilities for variable assignments (possibly based on any observed evidence), denoted $\phi_i(x_i)$, and the compatibility between pairs of variable assignments, denoted $\psi_{ij}(x_i, x_j)$. In attacking the GMEC, these were calculated by the following relation:

$$\psi_{ij}(x_i = r, x_j = s) = e^{-(e_{i_r j_s} - e_{min})}. \quad (2)$$

$$\phi_i(x_i = r) = e^{-(e_{i_r} - e_{min})}, \quad (3)$$

where $e_{min}$ is defined by $e_{min} = \min\{\min_{i,r} e_{i_r}, \min_{i<j,r,s} e_{i_r j_s}\}$. In fact, although these values are all between 0 and 1, they are not actually

probabilities; they do not sum to 1. Although it is straightforward to normalize the $\phi_i$ values, it is not as easy for the compatibilities. In fact, the MP algorithm does not require probabilities for the compatibilities, it simply requires "potentials" that are high for likely pairwise assignments [10].
The joint probability distribution for a given set of variables can be written as

$$p(x) = \frac{1}{Z}\left(\prod_{i\in V}\phi_i(x_i)\prod_{i,j\in E}\psi(x_i,x_j)\right) \quad (4)$$

where $V$ is the set of variables, $E$ is the set of pairs of variables constrained by a compatibility function and $Z$ is a complicated normalization constant. Note that the log of this probability is (except for constant factors) the negative of the protein energy and so maximizing the log of the probability (which is equivaluent to maximizing the joint probability) is the same as minimizing the protein energy. This is the basis of the connection between energy minimization and probabilistic inference.

### A.2 The max-product algorithm

The max-product (MP) algorithm is one of a family of iterative algorithms that operate by propagating a series of "messages" in the graphical model. Node $s$ sends to its neighbor node $i$ the message $m_{si}(x_i)$ which can be interpreted as how likely node $s$ sees node $i$ is in state $x_i$. Then, as shown in [10], the belief at node $i$ is given by the product of $\phi_i(x_i)$ associated with variable $x_i$ and all the messages coming into it:

$$P(x_i) = k\phi_i(x_i)\prod_{s\in N'(i)} m_{si}(x_i), \quad (5)$$

where $k$ is a normalization constant and $N'(i)$ denotes the neighboring nodes of $i$ except the singleton factor node associated with $x_i$.

Suppose $X$ is the set of all possible configurations of the random variable $\{x\}$. Then, the MAP (maximum a posteriori) assignment is given by

$$x_{MAP} = \arg\max_{\{x\}\in X} P(\{x\}). \quad (6)$$

Finding the MAP assignment is generally $NP$-hard. However, if the graphical model is singly connected, the max-product algorithm can efficiently solve the problem.

In the max-product algorithm the messages are updated as follows:

$$m_{si}(x_i) \leftarrow \max_{x_j}\phi_j(x_j)\psi_{ij}(x_i,x_j)\prod_{t\in N'(j)\setminus s} m_{tj}(x_j). \quad (7)$$

The belief update is given by (5). When the belief update converges, the belief for any node $i$ satisfies

$$b_i(x_i) = k\max_{\{x\}\in X} P(\{x\}|x_i), \quad (8)$$

and the MAP assignment can be found by

$$(x_{MAP})_i = \arg\max_{x_i} b_i(x_i). \quad (9)$$

The overall operation of the algorithm is that of a simple loop that updates the messages until the change in the sum of the log probabilities of the variables is sufficiently small, indicating convergence. The MP algorithm may fail to converge for some inputs; in that case, the propagation is terminated after some maximum number of iterations. The conditions for convergence of the MP algorithm remain somewhat obscure as are the general conditions under which it finds the optimal answer. The max-product algorithm does not guarantee finding the MAP assignment in graphs with loops, such as the one for our GMEC problem which is a densely connected graph. Freeman and Weiss [9] have shown that the assignment from the fixed points of the MP algorithm is a neighborhood maximum of the posterior probability.

## *B. Integer linear programming*

The GMEC problem can be solved using integer linear programming (ILP) techniques. We first present an ILP formulation by Erkisson et al., and two similar formulations adopted from related combinatorial optimization problems. Based on these classical ILP formulations, we review the branch-and-price algorithm and consider three different decomposition schemes for column generation.

### B.1 Formulation by Eriksson, Zhou, and Elofsson

In the ILP formulation of the GMEC problem by Eriksson et al. [18], the self-energy of each rotamer is evenly distributed to every interaction energy involving the rotamer. A residue's chosen rotamer interacts with every other residue's chosen rotamer. Therefore, the self-energies can be completely incorporated into the interaction energies without affecting the total energy by modifying the interaction energies as follows:

$$e'_{i_r j_s} = e_{i_r j_s} + \frac{e_{i_r} + e_{j_s}}{n-1}, \quad (10)$$

where $n$ is the number of residues in the protein. Then, the total energy of a given conformation $C$ can be written as

$$\mathcal{E}_C = \sum_i \sum_{j>i} e'_{i_r j_s}. \quad (11)$$

Since the total energy now only depends on the interaction energy, $\mathcal{E}_C$ can be expressed as a set of value assignments on binary variables that decide whether an interaction between two residues in a certain conformation exists or not. We let $x_{i_r j_s}$ be a binary variable, where its value is 1 if residue $i$ is in the rotamer conformation $r$, and residue $j$ is in the rotamer conformation $s$, and 0 otherwise. We also let $R_i$ denote the set of all possible rotamers of residue $i$. Then, the GMEC problem is

$$\min \sum_i \sum_{r\in R_i} \sum_{j>i} \sum_{s\in R_j} e'_{i_r j_s} x_{i_r j_s} \quad (12)$$

$$\sum_{r\in R_i}\sum_{s\in R_j} x_{i_r j_s} = 1 \text{ for all } i \text{ and } j,\ i<j, \tag{13}$$

$$\sum_{q\in R_h} x_{h_q i_r} = \sum_{p\in R_g} x_{g_p i_r} = \sum_{s\in R_j} x_{i_r j_s} = \sum_{t\in R_k} x_{i_r k_t} \tag{14}$$

for all $g,h,i,j,k$ such that $g,h<i<j,k$ and for all $r\in R_i$,

$$x_{i_r j_s} \in \{0,1\}. \tag{15}$$

The constraints guarantee that exactly one interaction between any pair of residues exists and only one rotamer is chosen for a residue. We denote (12) – (15) by `F1`.

Eriksson et al. devised this formulation to be used within the framework of integer programming algorithms, but they found that the LP relaxation of this formulation always gave an integral solution and hypothesized that every instance of the GMEC problem can be solved by linear programming. However, in our experiments, we found some instances have fractional solutions to the LP relaxation of `F1`. However, except two cases, all solved LP relaxations of `F1` had integral solutions.

### B.2 The second formulation

The second formulation is a minor modification of the formulation for the maximum edge-weighted clique problem (MEWCP) by Hunting et al. [19]. The goal of the MEWCP is to find a clique with the maximum sum of edge weights.

If we take each rotamer $r$ for residue $i$ as a node $i_r$, and the interaction between two rotamers $r$ and $s$ for residues $i$ and $j$, respectively, as an edge $(i_r, j_s)$, the GMEC problem reduces to finding the minimum edge-weighted maximum clique of the graph.

We introduce binary variables $x_{i_r}$ for node $i_r$ for all $i$ and $r\in R_i$. We also adopt binary variables $y_{i_r j_s}$ for each edge $(i_r, j_s)$ for all $i$ and $j$, and for all $r\in R_i$ and $s\in R_j$. Variable $x_{i_r}$ takes value 1 if node $i_r$ is included in the selected clique, and 0 otherwise. Variable $y_{i_r j_s}$ takes value 1 if edge $(i_r, j_s)$ is in the clique, and 0 otherwise. We define $V$ and $E$ as follows:

$$V = \{i_r |\ \forall\ r\in R_i, \forall\ i\}, \tag{16}$$

$$E = \{(i_r, j_s) |\ \forall\ i_r, j_s \in V, i\neq j\}. \tag{17}$$

Then, the adapted ILP formulation of the MEWCP is given by

$$\min \sum_{i_r\in V} e_{i_r} x_{i_r} + \sum_{(i_r,j_s)\in E} e_{i_r j_s} y_{i_r j_s} \tag{18}$$

$$y_{i_r j_s} - x_{i_r} \leq 0,\quad \forall (i_r, j_s)\in E, \tag{19}$$

$$y_{i_r j_s} - x_{j_s} \leq 0,\quad \forall (i_r, j_s)\in E, \tag{20}$$

$$x_{i_r} + x_{j_s} - 2y_{i_r j_s} \leq 0,\quad \forall (i_r, j_s)\in E, \tag{21}$$

$$x_{i_r} \in \{0,1\},\quad \forall i_r \in V, \tag{22}$$

$$y_{i_r j_s} \geq 0,\quad \forall (i_r, j_s)\in E. \tag{23}$$

To complete the formulation of the GMEC problem, we add the following set of constraints, which implies that exactly one rotamer should be chosen for each residue:

$$\sum_{r\in R_i} x_{i_r} = 1. \tag{24}$$

We denote (18) – (24) by `F2`. When the CPLEX MIP solver was used to solve a given GMEC instance in both `F1` and `F2`, `F1` was faster than `F2`. This is mainly because `F2` heavily depends on the integrality of variables $x_{i_r}$. On the other hand, `F2` has an advantage when used in the branch-and-cut framework since polyhedral results and Lagrangean relaxation techniques are available for the MEWCP that can be readily applied to `F2`. [19] [20]

### B.3 The third formulation

Koster et al. [21] presented another formulation that incorporates characteristics of two previous formulations. Koster et al. studied solution to the frequency assignment problem (FAP) via tree-decomposition. There are many variants of the FAP, and, interestingly, the minimum interference frequency assignment problem (MI-FAP) studied by Koster et al. has exactly the same problem formulation as the GMEC problem, except that the node weights and edge weights of the MI-FAP are positive integers.

The formulation suggested by Koster et al. uses node variables and combines the two styles of constraints from the previous formulations. If we transform the problem setting of the FAP into that of the GMEC problem, we obtain the following ILP formulation for the GMEC problem:

$$\min \sum_{i_r\in V} e_{i_r} x_{i_r} + \sum_{(i_r,j_s)\in E} e_{i_r j_s} y_{i_r j_s} \tag{25}$$

$$\sum_{r\in R_i} x_{i_r} = 1, \tag{26}$$

$$\sum_{s\in R_j} y_{i_r j_s} = x_{i_r},\quad \forall j\neq i,\ \forall s\in R_j,\ \forall i,\ \forall r\in R_i, \tag{27}$$

$$x_{i_r} \in \{0,1\},\quad \forall i_r \in V, \tag{28}$$

$$y_{i_r j_s} \geq 0,\quad \forall (i_r, j_s)\in E. \tag{29}$$

We denote (25) – (29) by `F3`. In `F3`, (26) restricts the number of selected rotamers for each residue to one as it does in `F2`. On the other hand, (27) enforces that the selection of interactions should be consistent with the selection of rotamers. Koster et al. studied the polytope represented by the formulation, and developed facet defining inequalities [22] [23] [24].

### B.4 Branch-and-price

The branch-and-price (also known as column generation IP) is a branch-and-bound algorithm that performs the bounding operation using LP relaxation [11]. Particularly, the LP relaxation at each node of the branch-and-bound tree is solved using the delayed column generation technique. It is usually based on an IP formulation that introduces a huge number of variables but has a tighter convex

hull in LP relaxation than a simpler formulation with fewer variables. The key idea of the method is splitting the given integer programming problem into the master problem and the subproblem by Dantzig-Wolfe decomposition and then exploiting the problem specific structure to solve the subproblem.

We first review the notion of branch-and-price given by Barnhart et al. [11]. The previous classical ILP formulations can be all captured in the following general form of ILP:

$$\min c'x \tag{30}$$

$$Ax \leq b, \tag{31}$$

$$x \in S, \quad x \in \{0,1\}^n, \tag{32}$$

where $c \in \Re^n$ is a constant vector, and $A$ is an $m \times n$ real-valued matrix. The basic idea of the Dantzig-Wolfe decomposition involves splitting the constraints into two separate sets of constraints – (31) and (32), and representing the set

$$S^* = \{x \in S \mid x \in \{0,1\}^n\} \tag{33}$$

by its extreme points. $S^*$ is represented by a finite set of vectors. In particular, if $S$ is bounded, $S^*$ is a finite set of points such that $S^* = \{y^1, \ldots, y^p\}$, where $y_i \in \Re^n$, $i = 1, \ldots, p$.

Now, if we are given $S^* = \{y^1, \ldots, y^p\}$, any point $y \in S^*$ can be represented as

$$y = \sum_{1 \leq k \leq p} y^k \lambda^k, \tag{34}$$

subject to the convexity constraint

$$\sum_{1 \leq k \leq p} \lambda^k = 1, \tag{35}$$

$$\lambda^k \in \{0,1\}, \quad k = 1, \ldots, p. \tag{36}$$

Let $c^k = c'y^k$, and $a^k = Ay^k$. Then, we obtain the general form of branch-and-price formulation for the ILP given by (30) – (32) as follows:

$$\min \sum_{1 \leq k \leq p} c^k \lambda^k \tag{37}$$

$$\sum_{1 \leq k \leq p} a^k \lambda^k \leq b, \tag{38}$$

$$\sum_{1 \leq k \leq p} \lambda^k = 1, \tag{39}$$

$$\lambda^k \in \{0,1\}, \quad k = 1, \ldots, p. \tag{40}$$

The fundamental difference between the classical ILP formulation (30) – (32) and the branch-and-price formulation (37) – (40) is that $S^*$ is replaced by a finite set of points. Moreover, any fractional solutions to the LP relaxation of (30) – (32) is a feasible solution of the LP relaxation of (37) – (40) if and only if the solution can be represented by a convex combination of extreme points of $conv(S^*)$. Therefore, it is easily inferred that the LP relaxation of (37) – (40) is at least as tight as that of (30) – (32) and more effective in branch-and-bound.

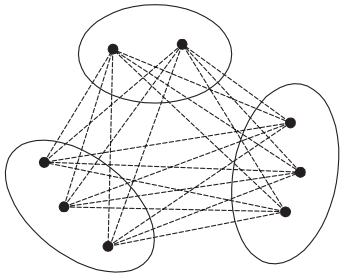

Fig. 1. Problem setting

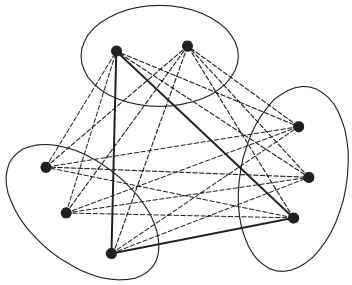

Fig. 2. A feasible solution to the GMEC problem

Now we recognize that success of the branch-and-price algorithm will, to a great extent, depend on the choice of base formulation we use for (30) – (32) and how we decompose it. In this work, we use `F1` as the base ILP formulation for it usually has the smallest number of variables and constraints among `F1`, `F2`, and `F3`, and also has good lower bounds. Designing the decomposition scheme is equivalent to defining $S^*$. Naturally, the constraints corresponding to (31) will be the complement constraints of $S^*$ in the (13) – (15). We consider three different definitions of $S^*$ below.

Figure 1 is a graphical illustration of the problem setting, and Figure 2 shows the feasible solution when all constraints are used. In fact, any maximum clique of the graph is a feasible solution. In this decomposition scheme, we consider only the constraint that one interaction is chosen between every pair of residues. Formally, we have

$$S^* = \{Q \mid Q \text{ is a set of } (i_r, j_s), \exists r \in R_i, \exists s \in R_j, \forall i, \forall j\}. \tag{41}$$

We denote this definition of $S^*$ by $S_1$.

The subproblem is finding out the smallest weight edge between each pair of residues. A solution to the subproblem will generally look like Figure 3. The size of $S_1$ is exponential to the original solution space, but each subproblem can be solved within $O(n)$ time.

We can also define $S^*$ to be the set of paths that starts from node $i_r$ and arrives at the same node by a sequence of edges connecting each pair of residues as illustrated by Figure 4. A column generated this way is a feasible solution to the GMEC problem only if the number of visited rotamers at each residue is one. We denote $S^*$ defined this

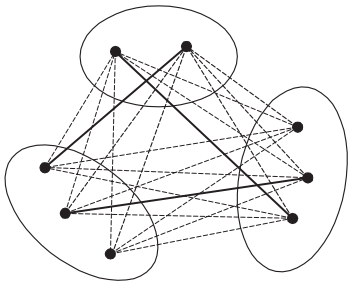

Fig. 3. A set of minimum weight edges

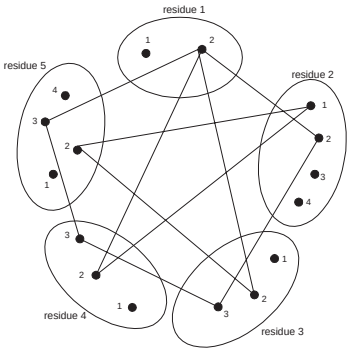


Fig. 4. Path starting from rotamer 2 of residue 1 and arriving at the same rotamer

way as $S_2$.

The size of $S_2$ is approximately the square root of the size of $S_1$. In this case, the subproblem becomes the shortest path problem if we replicate a residue together with its rotamers and add them to the path every time the residue is visited. Figure 5 shows how we can view the graph to apply the shortest path algorithm. The subproblem is less simple than the case $S^* = S_1$, but we can efficiently generate multiple columns at a time if we use the $k$-shortest path algorithm.

The inherent hardness of the combinatorial optimization problem is independent of the base ILP formulation or the decomposition scheme used in the branch-and-price algorithm. Therefore, the decomposition can be regarded as the design of the balance point between the size of $S^*$ and the hardness of the subproblem.

The size of $S_2$ is huge because more than one rotamer can be chosen from every residue on the path but the first. To make the column generation more efficient, we can try fixing several residues' rotamers on the path as we do for the first residue.

Based on the idea of partitioning a feasible solution, or a maximum clique into small cliques, the third decomposition we consider is to define $S_3$ as a set of cliques. Figure 6 illustrates this method. A maximum clique consisting of five nodes is fragmented into two three-node cliques composed of nodes $1_2, 3_3, 5_3$ and nodes $1_2, 4_3, 5_3$ respectively. To complete the maximum clique, four edges represented by dashed lines are added (an edge can be also regarded as a clique of two nodes.)

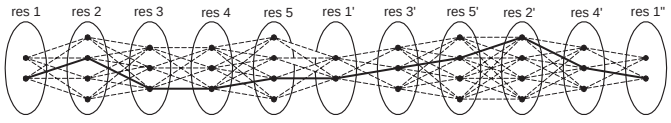


Fig. 5. An elongated graph to calculate shortest paths

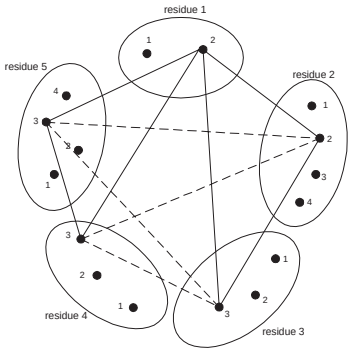


Fig. 6. Fragmentation of the maximum clique

The branch-and-price formulation for this decomposition differs from (37) – (40) in that (39) and (40) are not necessary. All the assembly of small cliques are done by (38) and it is possible we obtain an integral solution when (37) – (38) has a fractional optimal solution from LP relaxation. Therefore, the branching is also determined by examining the integrality of original variables, $x$ of (30) – (32).

The size of $S_3$ is obviously smaller than those of $S_1$ and $S_2$. If we let $n$ be the number of residues, and $m$ be the maximum number of rotamers for one residue, the size of $S_2$ is $O(m^{\frac{n(n-1)}{2}})$. In comparison, the size of $S_3$ is $O((m+1)^n)$.

The subproblem turns out to be the minimum edge-weighted clique problem (MEWCP), which is an NP-hard optimization problem. Macambira and de Souza [20] have investigated the MEWCP with polyhedral techniques, and there is also an Lagrangean relaxation approach to the problem by Hunting et al. [19]. However, it is more efficient to solve the subproblem by heuristics and to use the analytical techniques only if heuristics fail to find any new column.

On the other hand, there exists a pitfall in application of this decomposition; by generating edges as columns, we may end up with a basis for the LP relaxation of the base formulation. To obtain a better lower bound, we need to find a basis that better approximates the convex hull of integral feasible points than that of the LP relaxed base formulation. However, by generating edges as columns necessary to connect the small cliques, the column generation will possibly converge only after exploring the edge columns. One easily conceivable remedy is generating only cliques with at least three nodes, but it is a question whether this is a better decomposition scheme in practice than, say, $S^* = S_2$.

The LP or ILP formulations often involve a large number of

variables and constraints. Considering that the necessary number of variables is roughly at least the square of the total number of rotamers, LP or ILP based methods are infeasible when the problem size grows very large without the support of enormous computing power.

## IV. Results and discussions

### A. Probabilistic inference

We tested our simple implementation of the max-product (MP) algorithm of Section A.2 on several GMEC problem instances. Since MP is relatively fast and converges for most cases, we were able to carry out fairly large side-chain placements as well as some sequence design. All tests were performed on a Debian workstation with a 2.20 GHz Intel Xeon processor and 1 GBytes of memory; the program was written in C and compiled by the GNU compiler, gcc.

#### A.1 Side-chain placement

The initial protein test set for the side-chain placement is presented in Table I. Each test case is distinguished by the protein's PDB code and the number of residues modeled. For simplicity from now on, we will call each test case by combining the PDB code and the number of residues modeled, for example `256b-50`. Note that all the energy calculations for `1bpi`, `1amm`, `256b`, `1arb` were done using the same rotamer library (call it `LIB1`), and the test cases of `2end` and `1i1q` were generated using a different library (call it `LIB2`). `LIB2` consists of denser samples of side-chain torsion angles, and therefore have a larger number of rotamers for each amino acid than `LIB1`. We know the optimal value of each test case using one or more of the CPLEX optimizers, and Altman's DEE/A* implementation [25]. We have very large optimal values for some cases of `1amm` and `1arb`. We believe this results from the clash between rotamers, which might happen because of the non-ideal characteristics of the rotamer library that does not always represent the actual conformation of a side-chain very well. We, however, included, the cases in our test to see whether the characteristics of the rotamer library affects the performance of the methods.

For reference, we included the time taken to solve each test case by different solvers. We used the IP solver (CPLEX MIP Optimizer) only if we could not obtain integral solutions using the LP solver (CPLEX LP Optimizer). For example, in the cases of `256b-70` and `256b-80`, the optimal solutions of the LP relaxation are fractional. On the other hand, for `2end-35`, `2end-40`, and `2end-49`, the LP solver broke down because of the system's limited memory. The IP solver is not expected to handle these cases considering the greater computational requirements to find integral solutions. Altman's DEE/A* implementation was much faster than the other two solvers in all test cases of Table I.

The test result for MP are also shown in Table I. Noticeably, MP outperforms LP/IP and calculates optimal solutions for all test cases but two of `256b`. An interesting point is that MP failed only for the cases where LP relaxation does not have integral solutions ( `256b-70` error 0.33407, `256b-80` error 0.11505 ) but we do not yet understand whether there exists more general connection between the two methods[2]

TABLE I

Protein test set for side-chain placement (optimal: optimal energy, $T_{LP}$: LP/IP solver time (formulation F1) in sec, $T_{DEE}$: DEE/A* time in sec, $T_{MP}$: MP time in sec, symbol $*$ : failed, symbol F : fractional LP solution, symbol ! : MP did not converge).

| protein | #res | optimal | $T_{\mathrm{LP}}$ | $T_{\mathrm{DEE}}$ | $T_{\mathrm{MP}}$ |
|---|---|---|---|---|---|
| `1bpi` | 10 | -75.37721 | 0 | 0 | 0 |
| | 15 | -57.13359 | 0 | 0 | 0 |
| | 20 | -75.87743 | 0 | 0 | 0 |
| | 25 | -93.65443 | 2 | 0 | 0 |
| | 46 | -205.52529 | 30 | 1 | 1 |
| `1amm` | 10 | -62.64278 | 0 | 0 | 0 |
| | 15 | -134.63023 | 0 | 0 | 0 |
| | 20 | -167.31388 | 1 | 0 | 0 |
| | 25 | 138.61011 | 0 | 0 | 0 |
| | 70 | 238.12890 | 44 | 1 | 8 |
| | 80 | 17222.13193 | 100 | 2 | 10 |
| `256b` | 10 | -110.18409 | 0 | 0 | 0 |
| | 15 | -116.81010 | 0 | 0 | 1 |
| | 20 | -209.37536 | 1 | 0 | 0 |
| | 25 | -242.85081 | 2 | 0 | 0 |
| | 30 | -285.57884 | 3 | 0 | 0 |
| | 35 | -352.56265 | 8 | 0 | 1 |
| | 40 | -379.45106 | 10 | 0 | 1 |
| | 50 | -459.25473 | 26 | 1 | 3 |
| | 60 | -564.94275 | 62 | 1 | 12 |
| | 70 | -633.09146 | F 170 | 3 | ! 17 |
| | 80 | -626.06002 | F 974 | 11 | ! 61 |
| `1arb` | 10 | -65.10481 | 0 | 0 | 0 |
| | 20 | -128.35604 | 0 | 0 | 0 |
| | 30 | 999715.70565 | 0 | 0 | 0 |
| | 78 | -460.69454 | 23 | 0 | 8 |
| `2end` | 15 | -182.56152 | 30 | 2 | 2 |
| | 25 | -285.07644 | 286 | 6 | 13 |
| | 35 | -347.29088 | $*$ | 27 | 92 |
| | 40 | -402.92974 | $*$ | 30 | 106 |
| | 49 | -540.60516 | $*$ | 46 | 224 |
| `1i1q` | 7 | -81.23579 | 0 | 0 | 0 |
| | 11 | -172.50967 | 2 | 0 | 0 |

Considering speed and the accuracy of the prediction, MP appears to be an alternative to DEE/A* in the side-chain placement. To have a better comparison of the MP and DEE/A*, we tested MP and DEE/A* on larger cases of side-chain placement. We generated the test cases consisting of $35 \sim 60$ residues using `LIB2`. The length of the modeled protein sequences are around the average of those in Table I, but have more rotamer coformations. The results for these test cases are presented in Table II.

The table shows that MP found the optimal values for all the cases where DEE/A* was succesful. However, DEE/A* failed on two cases during the A* search due to exhausting the available memory. The failure during the A* search implies that DEE's elimination was not effective enough to make the A*'s job doable. MP was able to find

[2] Throughout our entire experiments including those not presented here, we did not observe any other protein energy example that had fractional solution in LP relaxation.

TABLE II

PERFORMANCE COMPARISON OF MP AND DEE/A* ON SIDE-CHAIN PLACEMENT ($E_{DEE}$: SOLUTION VALUE FROM DEE/A*, $E_{MP}$: SOLUTION VALUE FROM MP, $T_{DEE}$: TIME FOR DEE/A* IN SECONDS, $T_{MP}$: TIME FOR MP IN SECONDS, SYMBOL = : SAME ENERGY, SYMBOL * : DEE FAILED, SYMBOL ? : UNKNOWN ENERGY, SYMBOL ! : MP DID NOT CONVERGE).

| protein | #res | $E_{DEE}$ | $E_{\mathrm{MP}}$ | $T_{\mathrm{DEE}}$ | $T_{\mathrm{MP}}$ |
|---|---|---|---|---|---|
| 1cbn | 35 | 999802.90413 | = | 14 | 107 |
| 1isu | 49 | -329.62462 | = | 875 | 1679 |
| 1igd | 50 | -376.93007 | = | 282 | 2054 |
| 9rnt | 50 | -391.89270 | = | 70 | 463 |
| 1whi | 50 | -450.82517 | = | 467 | 2699 |
| 1ctj | 50 | -439.59678 | = | 162 | 1117 |
| 1aac | 50 | -549.36958 | = | 289 | 2020 |
| 1cex | 50 | -519.22413 | = | 179 | ! 3104 |
| 1xnb | 60 | ? | -437.26643 | * | ! 2350 |
| 1plc | 60 | -331.36163 | = | 130 | 2457 |
| 2hbg | 60 | -560.59013 | = | 646 | 5500 |
| 2ihl | 60 | ? | -527.61321 | * | 3555 |

answers for all the cases but it failed to converge properly on two of the cases and was terminated by an arbitrary limit on the number of iterations. In these cases, the result found by MP is less trustworthy. In one of the cases where MP did not converge we see that it found the optimal value, in the other we do not know the optimal value since DEE failed for that case.

In terms of running time, it is clear that the DEE/A* implementation is substantially more efficient than that of MP. It remains to be seen whether a more sophisticated implementation of MP can improve on this.

### A.2 Sequence design

Table III shows the protein test set for sequence design. The optimal solutions were obtained only for three cases, and the remaining three cases were not solved through either LP or DEE/A*. Both CPLEX LP Optimizer and Altman's DEE/A* implementation failed in `R15R23`, `sweet7-NOW`, and `P2P2prime` due to the system's limited memory. On the other hand, the LP solver managed to solve `1i1q-design3` after five hours. DEE/A* broke down for this case during the A* search.

We also used MP to solve these sequence design cases. The results are summarized in Table III. For all the cases with known optimal solutions, MP computed the solutions exactly in extremely short time. In protein design, unlike the side-chain placement, the measure of accuracy is usually the fraction of amino acids that are predicted incorrectly, but MP's results are correct on all the rotamers.

In the other three extremely large cases of sequence design whose optimal values are unknown, MP failed to converge properly on two of them but found plausible values before being stopped due to maximum iteration limit.

It is interesting to see that DEE/A* fails in the design cases whose complexity is not really worse than those of simple rotamer placement. It seems that DEE/A* is more effective when there are fewer rotamers per residue. MP seems less susceptible to this problem. We think combining the two approaches, say, by feeding DEE's incompletely eliminated output to MP as an input may make a better scheme than either one separately. This remains for future work.

TABLE III

THE PROTEIN TEST SET FOR SEQUENCE DESIGN (SYMBOL ? : UNKNOWN OPTIMAL VALUE, SYMBOL * : FAILED, SYMBOL ! : MP DID NOT CONVERGE ).

| Case ID | optimal | $E_{\mathrm{MP}}$ | $T_{\mathrm{LP}}$ | $T_{\mathrm{DEE}}$ | $T_{\mathrm{MP}}$ |
|---|---|---|---|---|---|
| 1i1q-design1 | -186.108150 | -186.108150 | 11 | 2 | 1 |
| 1i1q-design2 | -190.791630 | -190.791630 | 616 | 40 | 3 |
| 1i1q-design3 | -194.170360 | -194.170360 | 19522 | * | 13 |
| R15R23 | ? | -52.93456 | * | * | ! 157 |
| sweet7-NOW | ? | -192.417120 | * | * | ! 285 |
| P2P2prime | ? | -385.134720 | * | * | 767 |

### B. ILP approach

We performed an experimental comparison of the classical ILP formulations `F1`, `F2`, and `F3`. The formulations were implemented in C code using CPLEX Callable Library. The test cases of `1bpi`, `1amm`, `1arb`, and `256b` were generated using a common rotamer library (called by `LIB1` throughout the work), but a denser library (`LIB2`) was used to generate test cases of `2end`. The choice of modeled proteins followed the work by Voigt et al. [12]. We made several cases with different sequence lengths using the same protein to control the complexity of test cases. The energy terms were calculated by the CHARM22 script. All experiment was done on a Debian workstation with a 2.20 GHz Intel Xeon processor and 1 GBytes of memory. We used both CPLEX LP Optimizer and Mixed Integer Optimizer to compare the solution time between the formulations. The results are listed in Table IV.

The result from LP Optimizer tells that `F2` is far less effective in terms of both the lower bounds it provides and the time needed for the LP Optimizer. `F2` obtained only fractional solutions whereas `F1` and `F3` solved most of the cases optimally. `F1` and `F3` show similar performance though `F3` is slightly more effective than `F1` in LP relaxation. Since `F2` showed inferior performance in LP relaxation, we measured Mixed Integer Optimizer's running time only with `F1` and `F3`. As shown in the last two columns of Table IV, Mixed Integer Optimizer mostly took more time for `F1` than `F3`, but no more than a constant factor as was the case with LP Optimizer.

CPLEX optimizers were able to solve the small- to medium-sized cases, but using the optimizers only with a fixed formulation turns out to be not very efficient. On the other hand, DEE/A* is mostly faster than the CPLEX optimizers using the classical formulations, which makes the ILP approach entirely relying on general optimizers look unattractive. We also observe cases that both `F1` and `F3` obtain fractional solutions in LP relaxation. Solving large IP's usually takes significantly more time than solving corresponding LP relaxations. This suggests that we should explore the possible development of a problem specific IP

TABLE IV

COMPARISON OF THREE CLASSICAL FORMULATIONS (PROTEIN: PDB CODE, #RES: NUMBER OF MODELED RESIDUES, LP SOLUTION: I - INTEGRAL, F - FRACTIONAL, $T_{LP}$: TIME FOR CPLEX LP OPTIMIZER, $T_{IP}$: TIME FOR CPLEX MIP OPTIMIZER, SYMBOL – : SKIPPED, SYMBOL ∗ : FAILED).

| | | LP solution | | | $T_{LP}$ (sec) | | | $T_{IP}$ (sec) | |
|---|---|---|---|---|---|---|---|---|---|
| protein | #res | F1 | F2 | F3 | F1 | F2 | F3 | F1 | F3 |
| 1bpi | 10 | I | F | I | 0 | 1 | 0 | 0 | 0 |
| | 20 | I | F | I | 0 | 8 | 1 | 0 | 0 |
| | 25 | I | F | I | 2 | 134 | 0 | 3 | 1 |
| | 46 | I | F | I | 30 | 6487 | 14 | 37 | 16 |
| 1amm | 10 | I | F | I | 0 | 0 | 0 | 0 | 0 |
| | 20 | I | F | I | 1 | 96 | 0 | 1 | 0 |
| | 70 | I | F | I | 44 | 33989 | 29 | 45 | 48 |
| | 80 | I | F | I | 100 | ∗ | 69 | 99 | 102 |
| 1arb | 10 | I | I | I | 0 | 0 | 0 | 0 | 0 |
| | 20 | I | F | I | 0 | 0 | 0 | 0 | 0 |
| | 30 | I | F | I | 0 | 15 | 0 | 1 | 0 |
| | 78 | I | F | I | 23 | 10384 | 21 | 27 | 25 |
| 256b | 30 | I | – | I | 3 | – | 2 | 6 | 3 |
| | 40 | I | – | I | 10 | – | 4 | 14 | 6 |
| | 50 | I | – | I | 26 | – | 12 | 36 | 15 |
| | 60 | I | – | I | 62 | – | 36 | 87 | 37 |
| | 70 | F | – | F | 116 | – | 98 | 170 | 112 |
| 2end | 15 | I | – | I | 30 | – | 35 | 34 | 41 |
| | 25 | I | – | I | 286 | – | 214 | 343 | 240 |

solver that exploits the special structure of the GMEC problem. As an effort toward this direction, in Section B.4, we investigated the possible use of the branch-and-price algorithm based on F1 formulation. However, our preliminary results in this direction are not yet competitive with the standard IP solvers and nowhere near as effective as DEE/A* or MP.

## ACKNOWLEDGEMENT

This work was supported in part by the Singapore-MIT Alliance. The authors would like to thank Bruce Tidor for suggesting the problem and for helpful advice. Also, various members of the Tidor group, especially Michael Altman and Alessandro Senes, gave freely of their time and provided us the use of their software.

## REFERENCES

[1] M. De Maeyer, J. Desmet, and I. Lasters, "All in one: a highly detailed rotamer library improves both accuracy and speed in the modeling of side-chains by dead-end elimination," *Folding & Design*, vol. 2, pp. 53–66, 1997.

[2] B. I. Dahiyat and S. L. Mayo, "Protein design automation," *Protein Science*, vol. 5, pp. 895–903, 1996.

[3] J. W. Ponders and F. M. Richards, "Tertiary templates for proteins," *Journal of Molecular Biology*, vol. 193, no. 4, pp. 775–791, 1987.

[4] J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters, "The dead-end elimination theorem and its use in protein side-chain positioning," *Nature*, vol. 356, pp. 539–542, 1992.

[5] A. R. Leach and A. P. Lemon, "Exploring the conformational space of protein side chains using dead-end elimination and the a* algorithm," *PROTEINS: Structure, Function, and Genetics*, vol. 33, pp. 227–239, 1998.

[6] 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*, vol. 13, pp. 1505–1514, 1998.

[7] 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*, vol. 21, pp. 999–1009, 2000.

[8] D. B. Gordon and S. L. Mayo, "Branch-and-terminate: a combinatorial optimization algorithm for protein design," *Structure with Folding and Design*, vol. 7, no. 9, pp. 1089–1098, 1999.

[9] W. T. Freeman and Y. Weiss, "On the fixed points of the max-product algorithm," MERL, Tech. Rep. TR-99-39, 2000.

[10] M. I. Jordan, *An introduction to probabilistic graphical models*, University of California, Berkeley, 2003.

[11] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: column generation for solving huge integer programs," *Operations Research*, vol. 46, pp. 316–329, 1998.

[12] C. A. Voigt, D. B. Gordon, and S. L. Mayo, "Protein design automation," *Protein Science*, vol. 5, pp. 895–903, 1996.

[13] P. Koehl and M. Delarue, "Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy," *Journal of Molecular Biology*, vol. 239, pp. 249–275, 1994.

[14] C. Yanover and Y. Weiss, "Approximate inference and protein-folding," in *Proceedings of Neural Information Processing Systems*, 2002.

[15] J. S. Yedida, W. T. Freeman, and W. Yair, "Bethe free energy, kikuchi approximations and belief propagation algorithms," MERL, Tech. Rep. TR2001-16, 2001.

[16] A. A. S. A. A. Canutescu and J. R. L. Dunbrack, "A graph theory algorithm for protein side-chain prediction." *Protein Science*, vol. 12, pp. 2001–2014, 2003.

[17] E. Althaus, O. Kohlbacher, H.-P. Lenhof, and P. Müller, "A combinatorial approach to protein docking with flexible side-chains," in *Proceedings of the 4th Annual International Conference on Computational Molecular Biology*, R. Shamir, S. Miyano, S. Istrail, P. Pevzner, and M. Waterman, Eds. ACM Press, 2000, pp. 15–24.

[18] O. Eriksson, Y. Zhou, and A. Elofsson, "Side chain-positioning as an integer programming problem," in *WABI*, ser. Lecture Notes in Computer Science, vol. 2149. Springer, 2001, pp. 128–141.

[19] O. Hunting, U. Faigle, and W. Kern, "A lagrangian relaxation approach to the edge-weighted clique problem," *European Journal of Operations Research*, vol. 131, no. 1, pp. 119–131, May 2001.

[20] E. M. Macambira and C. C. de Souza, "The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations," *European Journal of Operations Research*, vol. 123, no. 2, pp. 346–371, June 2000.

[21] A. M. Koster, S. P. van Hoesel, and A. W. Kolen, "Solving frequency assignment problems via tree-decomposition," Maastricht University, Tech. Rep. RM 99/011, 1999.

[22] A. M. Koster, "Frequency assignment: Models and algorithms," Ph.D. dissertation, Maastricht University, November 1999.

[23] A. M. Koster, S. P. van Hoesel, and A. W. Kolen, "The partial constraint satisfaction problem: Facets and lifting theorems," *Operations research letters*, vol. 23, no. 3-5, pp. 89–97, 1998.

[24] ——, "Lower bounds for minimum interference frequency assignment problems," Maastricht University, Tech. Rep. RM 99/026, October 1999.

[25] M. D. Altman, "DEE/A* implementation." the Tidor group, AI Lab, MIT, 2003.