AND/OR Branch-and-Bound for Computational Protein Design Optimizing K*

Bobak Pezeshki[1], **Radu Marinescu**[2], **Alex Ihler**[1], **Rina Dechter**[1]

[1] University of California, Irvine, [2] IBM Research
{pezeshkb, ihler, dechter}@uci.edu, radu.marinescu@ie.ibm.com

Abstract

The importance of designing proteins to improve their interactions, such as designing high affinity monoclonal antibodies, has become ever more apparent as of late. Computational Protein Design, or CPD, can cast such design problems as an optimization problem with the objective of maximizing K*, an approximation of binding affinity based on a computational protein model. We introduce AOBB-K*MAP, a new branch-and-bound algorithm over AND/OR search spaces for solving the K*MAP problem. In addition to formulating CPD as a graphical model for K* optimization and providing an new efficient algorithm, we also introduce a statically compiled heuristics for K*MAP not previously used in CPD. As AOBB-K*MAP is inspired by algorithms from the well studied task of Marginal MAP, in addition to the algorithm itself, this work provides a framework for continued adaptation of existing state-of-the-art mixed inference schemes over AND/OR search spaces to the problem of protein design.

1 Introduction

Graphical models provide a powerful framework for reasoning about conditional dependency structures over many variables. The well known Marginal MAP (MMAP) query asks for the optimal configuration of a subset of variables (MAP variables) that have the highest marginal probability. We define a new related task, K*MAP, which instead asks for the configuration of MAP variables that maximizes a quotient of the marginalization of conditionally disjoint subsets of the remaining variables, this ratio known as K*. In context of computational protein design (**CPD**), K* estimates binding affinity between subunits. Thus, maximizing K* corresponds to maximizing the likelihood that the subunits will associate (Hallen and Donald 2019).

Like MMAP, K*MAP distinguishes between maximization variables (**MAP variables**) and summation variables (**SUM variables**). Moreover, the SUM variables are further partitioned into a subset whose marginal corresponds to the numerator of the K* ratio and a subset corresponding to the denominator. Like MMAP, the K*MAP problem is a mixed inference task and more difficult than either pure max- or sum- inference tasks as its summation and maximization operations do not commute. In terms of processing of variables for inference, this forces constrained variable orderings that may have significantly higher induced widths (Dechter and Rish 2002; Dechter 2019). This in turn also implies larger search spaces when using search algorithms or larger messages when using message-passing schemes. Even the easier case of MMAP is NP^{PP}-complete and it can be NP-hard even on tree structured models (Park 2002). In terms of bounded approximations, bounding the K* ratio requires both upper and lower bounding of marginals, producing an additional challenge over bounding of a MMAP value.

Nevertheless, over the last several years, there have been several advances in algorithms for solving the MMAP task (Marinescu et al. 2018), many of which have potential for being adapted for the K*MAP query. In order to set the framework for leveraging these advances for K*MAP, this work presents three main contributions:

1. **Two formulations of K*MAP as a graphical model**
2. **A statically compiled Weighted Mini-Bucket Elimination K*MAP heuristic, wMBE-K*MAP**
3. **A branch-and-bound algorithm, AOBB-K*MAP, for solving CPD formulated as a K*MAP problem**

2 Background

Computational Protein Design. Computational Protein Design (CPD) is the task of mutating a known protein's amino acid sequence in hopes of achieving a desired objective such as improving the protein's energetics, improving protein-ligand interactions, or reducing interactions of a protein with inhibitors. In CPD, certain residues of a protein-of-interest are deemed as mutable - these are amino acid positions (or residues) where different amino acid mutations will be considered - and through a computational process, a preferred sequence is determined.

Throughout the computational process, various sets of mutations are explored, each comprising a particular amino acid sequence. Given a particular sequence (or in some methods, even partial sequence) an estimate of the resulting protein's goodness can be determined. This goodness is determined by considering the possible conformations of the resulting protein, namely considering possible positioning of its backbone and side-chains. The state space for these conformations is continuous (and even when discretized, is extremely large) leading to an intractable problem.

As such, many simplifications can be made to allow a more tractable problem:

- **Select Mutable Residues:** consideration of only a subset of the residues involved in the interactions as mutable.
- **Predetermined Side-Chain Rotamers:** discretization of side-chain conformations as rotamers.
- **Fixed Backbone Structure:** assumption of a fixed protein backbone conformation.

With these simplifying assumptions, many algorithms have been designed to find mutations that can potentially result in improved protein functionality (Hallen and Donald 2019; Zhou, Wu, and Zeng 2016).

K* and K*MAP. The affinity between two interacting protein subunits P and L is correlated to an equilibrium of the chemical reaction forming their complexed state PL:

$$P + L \rightleftharpoons PL \tag{1}$$

This said equilibrium is associated with a constant, K_a, and can be determined in vivo by observing the persisting concentrations of each species as follows (Freiser 1962),

$$K_a = \frac{[PL]}{[P][L]} \tag{2}$$

However, in order to compare K_a values of various designs in vivo, it is necessary to synthesize the interacting subunits through molecular processes that are both timely and costly.

K_a can also be approximated as

$$K^{\int} = \frac{Z^{\int}_{PL}}{Z^{\int}_{P} Z^{\int}_{L}}, \quad Z^{\int}_{\gamma} = \int_{\boldsymbol{c}} e^{-\frac{E_\gamma(c)}{\mathscr{R}T}} dc \tag{3}$$

where $Z^{\int}_{PL}$, $Z^{\int}_{P}$, and $Z^{\int}_{L}$ are partition functions of the bound and unbound states that capture the entropic contributions of their various conformations $\boldsymbol{\mathcal{C}}$. ($E_\gamma(c)$ represents the energy of a particular conformation c of state $\gamma \in \{PL, P, L\} = \varphi$, $\mathscr{R}$ is the universal gas constant, and T is temperature (in Kelvin). We can further use a model that discretizes the conformation space. This computed estimate is denoted as K* (Ojewole et al. 2018):

$$K^* = \frac{Z_{PL}}{Z_P Z_L}, \quad Z_\gamma = \sum_{\boldsymbol{c}} e^{-\frac{E_\gamma(c)}{\mathscr{R}T}} \tag{4}$$

Due to the independence between residues across the dissociate subunits, we can generalize further expressing K* as:

$$K^* = \frac{Z_B}{Z_U}, \tag{5}$$

where B respresents the bound (complexed) state and U represents the unbound (dissociate) states. (For the two-subunit system described earlier, $B = PL$ and $U = P \cup L$). This more generalized representation, can be used directly for K* computations involving more than two subunits.

A common task in protein design is to maximize protein-ligand interaction. K*MAP is a formalization of this task using the K* objective,

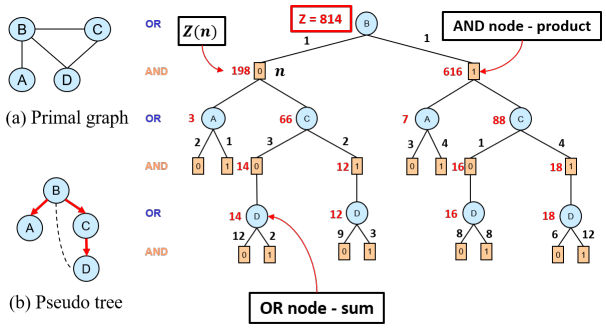
$$K^*\text{MAP} = \underset{\boldsymbol{R}}{\operatorname{argmax}}\, K^*(r) \tag{6}$$

where we look for amino acid assignments $R = r$ that maximize K*. Thus, using efficient algorithms for computing K*MAP, one can predict a small set of promising sequences to experiment on in vivo, saving great time and cost.

Graphical Models. A **graphical model**, such as a Bayesian or a Markov network (Pearl 1988; Darwiche 2009; Dechter 2013), can be defined by a 3-tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by a set V and $\mathbf{U} = \{D_i : i \in D\}$ is the set of finite domains of values for each X_i. Each function $f_\alpha \in \mathbf{F}$ is defined over a subset of the variables called its scope, X_α, where $\alpha \subseteq V$ are the indices of variables in its scope and D_α denotes the Cartesian product of their domains, so that $f_\alpha : D_\alpha \rightarrow R^{\geq 0}$. The **primal graph** $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ of a graphical model associates each variable with a node ($\mathbf{V} = \mathbf{X}$), while arcs $e \in \mathbf{E}$ connect nodes whose variables appear in the scope of the same local function. Graphical models can be used to represent a global function, often a probability distribution, defined by $Pr(X) \propto \prod_\alpha f_\alpha(X_\alpha)$.

AND/OR Search Space for Mixed Inference A graphical model can be transformed into a weighted state space graph. In an OR search space, which is constructed layer-by-layer relative to a variable ordering, paths from the root to the leaves represent full configurations - or assignments to all variables - where each successive level corresponds to an assignment of the next variable in the ordering. A more compact AND/OR search space can also be constructed by capturing conditional independencies, thus facilitating more effective algorithms (Dechter and Mateescu 2007).

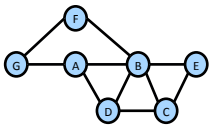
An AND/OR search space is defined relative to a *pseudo tree* of a primal graph which can capture conditional independences. A **pseudo tree** $\mathcal{T} = (\mathbf{V}, \mathbf{E}')$ of a primal graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a directed rooted tree that spans $\mathcal{G}$ such that every arc of $\mathcal{G}$ not in $\mathbf{E}'$ is a back-arc in $\mathcal{T}$ connecting a node to one of its ancestors (Figure 1(a),(b)). For mixed inference problems where a subset of variables are to be maximized (MAP variables) and the remaining variables (SUM variables) marginalized, the pseudo tree must be constrained such that the MAP variables precede SUM variables in the variable ordering (Lee et al. 2016; Marinescu et al. 2018).



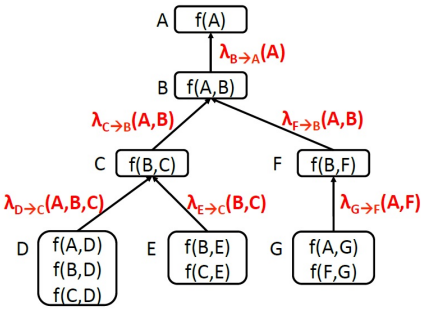
(c) AND/OR Search Tree

Figure 1: A full AND/OR tree representing all 16 solutions.

Given a pseudo tree $\mathcal{T}$ of a primal graph $\mathcal{G}$, the AND/OR search tree $T_{\mathcal{T}}$ guided by $\mathcal{T}$ has alternating levels of OR



(a) A primal graph.



(b) Bucket elimination example

Figure 2: (a) A primal graph of a graphical model with 7 variables. (b) Illustration of *BE* with an ordering A B C E D F G.

nodes corresponding to variables, and AND nodes corresponding to assignments from its domain with edge costs extracted from the original functions (Dechter and Mateescu 2007). Each arc into an AND node n has a cost $c(n)$ defined to be the product of all factors f_α in $\mathcal{M}$ that are instantiated at n but not before.

A **solution tree** is a subtree of $T_\mathcal{T}$ satisfying: (1) it contains the root of $T_\mathcal{T}$; (2) if an OR node is in the solution tree, exactly one of its AND child nodes is in the solution tree; (3) if an AND node is in the tree then all of its OR children are in the solution tree. (Dechter and Mateescu 2007).

Bucket Elimination. Given a variable ordering d, *Bucket Elimination* (Dechter 1999), or BE, is an inference scheme that processes variables one by one with respect to the reverse of d. For any next variable X_p, all the functions in bucket B_p - namely the original functions in the graphical model and any messages passed to B_p from previous buckets - are processed by marginalizing X_p from the product of the functions. This generates a new *bucket function* or message, denoted $\lambda_{p\to a}$, or λ_p for short.

$$\lambda_{p\to a} = \sum_{X_p} \prod_{f_\alpha \in B_i} f_\alpha \tag{7}$$

where X_a is the latest variable in λ's scope along d. The λ function is placed in the bucket of X_a, B_a. Once all the variables are processed, *BE* outputs all the messages and the exact value of Z by taking the product of all the functions present in the bucket of the first variable. **Figure 2a** shows a primal graph of a graphical model with variables indexed from A to G with functions over pairs of variables that are connected by an edge. In this particular example $F = \{f(A), f(A,B), f(A,D), f(A,G), f(B,C), f(B,D), f(B,E), f(B,F), f(C,D), f(C,E), f(F,G)\}$.

Bucket-Elimination can be viewed as a 1-iteration message-passing algorithm along its *bucket-tree* (bottom-up). The nodes of the tree are the different buckets. Each bucket of a variable contains a set of the model's functions depending on the given order of processing. There is an arc from bucket B_p to a parent bucket B_a, if the function created at bucket B_p is placed in bucket B_a. We illustrate *BE* message flow on our example problem in **Figure 2b**.

Complexity. Both the time and space complexity of *BE* are exponential in the **induced-width**, which can be computed as a graph parameter based on the ordered primal graph (Dechter 2019). The induced width is the size of the largest number of variables, in the scope of any message. *BE* becomes impractical if the induced-width is large and approximation schemes have been developed to address this (Dechter and Rish 2002; Liu and Ihler 2011).

3 Graphical Model for K*MAP Computation

As the first main contribution of this work, we describe two formulations of CPD problems as graphical models for use in computing K*MAP. These build upon previous work from MMAP (see Marinescu et al. (2018)) and CPD graphical model formulations for optimizing a weaker objective called the GMEC (Zhou, Wu, and Zeng 2016).

3.1 Formulation 1 (F1)

Formulation 1 distinguishes itself by using an indexing scheme for identifying residue rotamers. For any amino acid assignment to a residue i, $R_i = aa$, the assignment to its associated conformation variable, $C_{\gamma(i)} = c$, indexes the particular rotamer of amino acid aa that is being considered. We elaborate below.

Variables and Domains We introduce a set of **residue variables**, $\boldsymbol{R} = \{R_i \,|\, i \in \{1, 2, ..., N\}\}$, representing the N different residues (ie. positions) of the proteins. Each R_i has corresponding domain $D_{R_i} = \{aa \,|\, aa$ is a possible amino acid assignment to residue $i\}$. For residues that are being considered for mutation (**mutable residues**), each R_i considers one of ∼20 possible amino acid assignments. These are the MAP variables maximized over in the K*MAP task.

We also introduce a set of **conformation variables**, $\boldsymbol{C_\gamma} = \{C_{\gamma(i)} \,|\, i \in \{1, 2, ..., N\}\}$, each indexing discretized spacial conformations (ie. rotamers) of the amino acid at residue R_i when the protein is in state $\gamma \in \{B, U\}$. Each $C_{\gamma(i)}$ has corresponding domain $D_{C_{\gamma(i)}} = \{1, 2, ..., M_i\}$, where M_i is the maximum number of rotamers for any possible amino acid assignment to R_i in state $\gamma \in \{B, U\}$. Since each amino acid assignment to R_i has a different side chain with different possible rotamers, the assignment to $C_{\gamma(i)}$ acts as an index to the possible side chain conformations of the amino acid assigned to R_i. These are the SUM variables which we marginalize over.

Functions There are two sets of functions in F1.

$E^{sb}_\gamma = \{E^{sb}_{\gamma(i)}(R_i, C_{\gamma(i)}) \,|\, i \in \{1, 2, ..., N\}\}$ is a set of functions that captures the energies of interaction of the amino acid at each residue i with itself and the surrounding backbone. For any assignment to $C_{\gamma(i)}$ (which corresponds to an index for the rotamers of the amino acid assigned to R_i) that is out of range of the assigned amino acid's possible

rotamers, an infinite energy value is assigned as an implicit constraint.

$E^{pw}_{\gamma} = \{E^{pw}_{\gamma(ij)}(R_i, C_{\gamma(i)}, R_j, C_{\gamma(j)}) \mid \text{for } i,j \text{ s.t. } R_i \text{ and } R_j \text{ interact}\}$ is a set of functions that captures the pair-wise energies of interaction between the amino acids of residues that are in close spacial proximity. For any assignment to $C_{\gamma(i)}$ (which corresponds to an index for the rotamers of the amino acid assigned to R_i) that is out of the range of its residue's assigned amino acid's possible rotamers, an infinite energy value is assigned as an implicit constraint.

Objective Function The K* objective can be expressed as $K^*(R_1...R_N) = \frac{Z_B(R_1...R_N)}{Z_U(R_1...R_N)}$, where we assume temperature T in Kelvin and Universal Gas Constant $\mathscr{R}$ where

$$Z_{\gamma}(R_1...R_N) = \sum_{C_{\gamma(1)},...,C_{\gamma(N)}} \prod_{E^{sb}_{\gamma(i)} \in \boldsymbol{E^{sb}_{\gamma}}} e^{-\frac{E^{sb}_{\gamma(i)}(R_i, C_{\gamma(i)})}{\mathscr{R}T}} \cdot \prod_{E^{pw}_{\gamma(ij)} \in \boldsymbol{E^{pw}_{\gamma}}} e^{-\frac{E^{pw}_{\gamma(ij)}(R_i, C_{\gamma(i)}, R_j, C_{\gamma(j)})}{\mathscr{R}T}} \tag{8}$$

3.2 Formulation 2 (F2)

Formulation 2 was inspired by the works of Viricel et al. (2018) and Vucinic et al. (2019) and distinguishes itself by using explicit constraints to restrict invalid amino acid - rotamer combinations. For each corresponding residue - conformation variable pair, there exists a constraint to ensure the assignment to the residue variable matches the rotamer assignment of its conformation variable. We elaborate below.

Variables and Domains As in F1, we introduce a set of residue variables, $R = \{R_i \,|\, i \in \{1, 2, ..., N\}\}$, representing the N different residues (ie. positions) of the proteins. Each R_i has corresponding domain $D_{R_i} = \{aa \mid aa \text{ is a possible amino acid assignment to residue } i\}$. For mutable residues, each R_i considers one of $\sim$20 possible amino acid assignments. As before, these are the MAP variables maximized over in the K*MAP task.

We also introduce a set of conformation variables, $C = \{C_{\gamma(i)} \mid i \in \{1, 2, ..., N\}\}$, this time each representing the specific amino acid and conformation of the N different residues. Namely, each each $C_{\gamma(i)}$ has corresponding domain $D_{C_{\gamma(i)}} = \{c \mid c \text{ is a rotamer for one of the possible amino acids of residue } R_i\}$. Since each amino acid (ie. assignment to R_i) has a different side chain with different possible rotamers, the amino acid assignment to R_i will act as a selector into the possible assignments to $C_{\gamma(i)}$. These are the SUM variables which we marginalize over.

Functions There are three sets of functions in F2.

$\mathscr{C} = \{\mathscr{C}_{\gamma(i)}(R_i, C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}, \gamma \in B \cup U\}$ is a set of constraints ensuring that the assigned rotamer to $C_{\gamma(i)}$ belongs to the amino acid assigned to R_i.

$E^{sb}_{\gamma} = \{E^{sb}_{\gamma(i)}(C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}\}$ is a set of functions that captures the energies of interaction of the amino acid at each residue i with itself and the surrounding backbone.

$E^{pw}_{\gamma} = \{E^{pw}_{\gamma(ij)}(C_{\gamma(i)}, C_{\gamma(j)}) \mid \text{for } i,j \text{ s.t. } R_i \text{ and } R_j \text{ interact}\}$ is a set of functions that captures the pair-wise energies of interaction between the amino acids of residues that are in close spacial proximity.

Objective Function As before, the K* objective can be expressed as $K^*(R_1...R_N) = \frac{Z_B(R_1...R_N)}{Z_U(R_1...R_N)}$, where we assume temperature T and Universal Gas Constant $\mathscr{R}$ where

$$Z_{\gamma}(R_1...R_N) = \sum_{C_1,...,C_N} \prod_{\mathscr{C}_{\gamma(i)} \in \mathscr{C}} \mathscr{C}_{\gamma(i)}(R_i, C_{\gamma(i)}) \cdot \prod_{E^{sb}_{\gamma(i)} \in \boldsymbol{E^{sb}_{\gamma}}} e^{-\frac{E^{sb}_{\gamma(i)}(C_{\gamma(i)})}{\mathscr{R}T}} \cdot \prod_{E^{pw}_{\gamma(ij)} \in \boldsymbol{E^{pw}_{\gamma}}} e^{-\frac{E^{pw}_{\gamma(ij)}(C_{\gamma(i)}, C_{\gamma(j)})}{\mathscr{R}T}} \tag{9}$$

4 wMBE-K*MAP

As the next contribution of this work, we describe a new statically compiled weighted mini-bucket heuristic for bounding the K*MAP value, wMBE-K*MAP (Algorithm 1).

wMBE-K*MAP operates similarly to wMBE-MMAP (Ping, Liu, and Ihler 2015). Two key similarities are that (1) it takes a variable ordering that constrains buckets of MAP variables to be processed last (line 4) for which maximization (instead of summation) occurs, and (2) for any bucket that has a width larger than a provided i-bound, a bounded approximation is made by partitioning the bucket functions into mini-buckets (line 5) and taking the product of their power-sums over the bucket variable (lines 11-15, 16-20), leveraging Holder's Inequality (Hardy, Littlewood, and Pólya 1988). The power sum is defined as follows:

$$\sum_{x}^{w} f(x) = (\sum_{x} f(x)^{\frac{1}{w}})^w \tag{10}$$

The power sum reduces to a standard summation when $w = 1$ and approaches max when $w \to 0^+$.

Proposition 4.1 (Holder inequality) *Let $f_i(x)$, $i = 1..r$ be a set of functions and $w_1, ..., w_r$ be a set of positive weights, s.t., $w = \sum_{i=1}^{r} w_i$ then,*

$$\sum_{x}^{w} \prod_{i=1}^{r} f_i(x) \leq \prod_{i=1}^{r} \sum_{x}^{w_i} f_i(x) \tag{11}$$

In order to adapt wMBE-MMAP for K*MAP, two key innovations are required: (1) buckets corresponding to variables in $\boldsymbol{C_U}$, whose marginal belongs to the denominator of the K* expression, are lower-bounded (to lead to an upper bound on K*) by using a modification to Holder's inequality that incorporates negative weights (Liu and Ihler 2011) (lines 16-20), and (2) when messages are passed from buckets corresponding to variables in $\boldsymbol{C_U}$ to that of $\boldsymbol{R}$, the messages are inverted to accommodate being part of the denominator (line 22).

Although details are omitted here, wMBE-K*MAP can also employ cost shifting to tighten its bounds (see Liu and Ihler (2011)).

Complexity. Like wMBE-MMAP, wMBE-K*MAP is exponential in both space and time in the provided i-bound parameter or induced width of the constrained ordering - whichever is smaller.

5 AOBB-K*MAP

We present the final contribution of this work: AOBB-K*MAP (Algorithm 2), an AND/OR branch-and-bound scheme for solving the K^*MAP task. With state-of-the-art K* optimizers employing memory intensive best-first search (Ojewole et al. 2018; Hallen et al. 2018), branch-and-bound algorithms provide a search methodology linear in space allowing for solving problems unable to be solved by best-first methodologies due to memory (Zhou, Wu, and Zeng 2016).

At a high level, AOBB-K*MAP adapts AOBB-MMAP (Marinescu, Dechter, and Ihler 2014) for the K^*MAP task for CPD by (1) guiding search via wMBE-K*MAP, (2) computing ratios corresponding to K* computation, (3) incorporating a biologically relevant constraint that lower-bounds the partition function of each protein subunit Z_γ, and (4) using MiniSAT (Eén and Sörensson 2004) to avoid searching provably invalid amino acid - rotamer configurations.

The algorithm begins with a two-step initialization. First, constraint literals are generated by MiniSat after full constraint propagation of the constraints in $\mathcal{M}$ (line 3). Second, DFS search is initialized to start at a dummy AND node that roots the AND/OR search space corresponding to $\mathcal{T}$ (line 4).

Throughout search, each node n maintains a progressive upper bound $ub_{K^*}(n)$ on the K*MAP of the sub problem it roots. For the dummy node, this value is initialized to the global upper bound on K*MAP based on the K* upper-bounding heuristic function $h^{ub}_{K^*}(.)$ (line 5). As search progresses, $ub_{K^*}(n)$ converges to the K*MAP of the sub problem rooted at n.

Each node n maintains four quantities per protein subunit $\gamma \in \varphi$ - $A^{\times}_{Z_\gamma}(n)$, $g(n)$, $ub_{Z_\gamma}(n)$, and $A^{+}_{Z_\gamma}(n)$ (lines 8-12) - that are instrumental in computing an upper bound on the partition functions of each subunit $Z_\gamma(n)$ consistent with the path to n via $UB_{Z_\gamma}(n) = A^{\times}_{Z_\gamma}(n) \cdot g(n) \cdot ub_{Z_\gamma}(n) + A^{+}_{Z_\gamma}(n)$. $A^{\times}_{Z_\gamma}(n)$ captures an upper bound on the multiplicative portion of $Z_\gamma(n)$ due to OR branchings off of n's AND ancestors, $g(n)$ is the path cost to n, $ub_{Z_\gamma}(n)$ is a progressive upper bound on the $Z_\gamma(n)$ sub problem rooted at n, and $A^{+}_{Z_\gamma}(n)$ is an upper bound on the summation contribution from AND branchings off of n's OR ancestors. At each point in the algorithm, $UB_{Z_\gamma}(n)$ is computed to ensure it is greater than the inputted $threshold(\gamma)$ in order to enforce biologically relevant solutions (Ojewole et al. 2018).

The algorithm traverses the underlying AND/OR search tree guided by $\mathcal{T}$ expanding nodes in a depth-first manner (line 13), pruning whenever one of three conditions occurs: (1) the current assignment violates a constraint established by $MiniSat$ (ie. constraint-propagation pruning, or CPP) (line 14), (2) the subunit-stability threshold is provably violated (ie. subunit-stability pruning, or SSP) (line 16), or it (3) can be asserted that an amino acid configuration cannot produce a K* better than one previously found (ie. upper-bound pruning, or UBP) (line 19). When the algorithm backtracks from a node (line 22), the stored bounded values of the parent node are tightened accordingly. (Details are omitted for brevity, but can be found in the supplemental materials[1]).

The algorithm progresses in this manner until it finally returns to, and updates, the dummy root of the tree with the maximal K* value corresponding to an amino acid configuration that also satisfies the subunit-stability thresholds.

Complexity. The algorithm is linear in space and exponential in time with respect to the height of $\mathcal{T}$. (However a powerful guiding heuristic can lead to UBP, potentially reducing time greatly in practice).

6 Empirical Evaluation

6.1 Methods

Algorithms. Experiments were run using AOBB-K*MAP with constraint propagation, implemented in C++. For comparison, problems were also tested against state-of-the-art BBK* (implemented in Java) (Ojewole et al. 2018) with rigid side chains and an epsilon of 1×10^{-12}. Experiments were run for a maximum of 12hrs on a 2.66 GHz processor with 4 GB of memory with the same subunit-stability threshold as BBK* of $threshold(\gamma) = Z^*_\gamma \cdot e^{-\frac{5}{RT}}$ where Z^*_γ is the partition function given the wild-type amino acid sequence.

Heuristics. For AOBB-K*MAP, the statically compiled wMBE-K*MAP heuristic was used for guiding and bounding search, as well as wMBE-MMAP heuristics for upper-bounding the partition function of each subunit.

Benchmarks. We experimented on 41 protein design benchmark problems, 29 of which encoded two mutable residues (denoted "small") and 12 of which were harder with three mutable residues (denoted "expanded"). The benchmarks were generated using OSPREY 3.0 (Hallen et al. 2018) and formulated into both F1 and F2 in UAI format for AOBB-K*MAP to be run on. (BBK* was run via OSPREY and used its native formulation).

Performance Measure. We verified AOBB-K*MAP's correctness by comparing the returned K*MAP value to that of BBK* and its performance by comparing elapsed time. For diagnostic purposes, we also report the initial heuristic bound for the K*MAP value, the number of nodes traversed, and number of nodes pruned.

6.2 Analysis

Data Tables. Figures 3-6 show results for AOBB-K*MAP on both sets of benchmarks for both formulation F1 and F2. *iB* denotes the best performing i-bound used, $|X|$ are the number of mutable residues and conformation variables, *Dmax* is the maximum domain size, *w** is the induced width due to the generated constrained variable ordering, *UB* is the wMBE-K*MAP bound, *OR* and *AND* display the number of each type of node visited, *CPP* are the number of nodes pruned due to MiniSat constraint propagation, *UBP* are

[1] https://www.ics.uci.edu/~dechter/publications/r268-supplemental.pdf

Problem	iB	|X|	Dmax	w*	d	UB	OR	AND	CPP	UBP	SSP	EH	time	K*MAP	BBK* t	BBK* sln
1a0r_00031	5	16	34	8	8	inf	3E+05	1782040	1242234	0	122	1	262	7.88	13	7.88
1gwc_00021	6	12	34	6	6	10.28	41049	158841	378716	76	0	6	89	9.79	35	9.79
1gwc_00033	5	18	35	9	9	inf	46660	183409	321999	0	47	1	39	10.48	33	10.48
2hnu_00026	6	14	34	7	7	15.18	20896	104681	72450	77	37	5	**25**	13.18	63	13.18
2hnv_00025	6	16	34	8	8	14.97	1E+05	332754	87628	78	0	4	**46**	13.65	69	13.65
2rf9_00007	7	18	34	9	9	inf	1E+05	380720	453549	0	410	1	75	14.08	23	14.08
2rf9_00013	6	16	34	8	8	14.57	4074	19034	15979	61	0	3	9	13.25	4	13.25
2rf9_00018	8	18	34	9	9	16.68	19927	84823	98124	39	0	4	76	15.79	18	15.79
2rf9_00042	7	22	34	11	11	inf	5E+05	2342453	1816775	0	77	1	353	22.65	41	22.65
2rfd_00035	7	16	34	8	8	18.14	1E+06	5390146	4229817	142	0	7	817	**17.27**	199	16.77
2rfe_00012	5	14	34	7	7	15.23	4930	15982	26937	72	0	4	**5**	13.93	6	13.93
2rfe_00014	5	14	34	7	7	15.66	6821	22109	37197	72	0	4	**6**	14.36	11	14.36
2rfe_00017	7	14	35	7	7	10.96	13061	54335	292185	82	2	6	25	10.52	9	10.52
2rfe_00030	6	14	35	7	7	11.56	19405	160026	391901	101	37	7	60	10.50	40	10.50
2rfe_00041	8	18	34	9	9	inf	2E+06	5246734	2.9E+07	0	552	1	2473	22.73	245	22.73
2rfe_00043	5	16	35	8	8	inf	3E+05	1146880	8281137	0	614	1	232	18.04	9	18.04
2rfe_00044	7	16	35	8	8	19.06	2E+05	625979	6568520	222	5	16	283	18.19	8	18.19
2rfe_00047	5	18	34	9	9	inf	3E+05	1106761	1516836	0	37	1	189	22.70	42	22.70
2rfe_00048	6	20	34	10	10	inf	4E+05	1668173	2542598	0	102	1	299	22.81	71	22.81
2rl0_00008	5	10	34	5	5	11.60	2385	43767	1978	40	0	2	**6**	**11.16**	28	9.46
2xgy_00020	5	14	35	7	7	12.34	58141	350035	986465	55	111	5	35	10.60	33	10.60
3cal_00032	8	16	34	8	8	47.77	3E+05	1220353	8408660	71	47	4	757	11.62	17	11.62
3ma2_00016	5	14	34	7	7	13.82	21915	64035	91516	62	223	10	11	8.38	6	8.38
3u7y_00009	6	12	34	6	6	4.96	33137	212545	134394	232	6	24	38	4.51	32	4.51
3u7y_00011	5	12	34	6	6	12.72	5714	16064	34855	96	0	5	**5**	11.85	13	11.85
4hem_00027	5	16	34	8	8	inf	5347	12346	47544	0	61	1	**5**	15.48	17	15.48
4hem_00028	5	16	34	8	8	inf	3266	10885	22106	0	54	1	**4**	15.27	18	15.27
4kt6_00023	7	16	34	8	8	18.24	17300	58100	363924	168	28	3	40	12.69	31	12.69
4wwi_00019	7	14	34	7	7	15.43	8046	30726	16164	40	0	2	60	14.99	8	14.99

Figure 3: F1 on problems with two mutable residue (ie. MAP) variables.

Problem	iB	|X|	Dmax	w*	d	UB	OR	AND	CPP	UBP	SSP	EH	time	K*MAP	BBK* t	BBK* sln
1a0r_00031	3	16	203	6	8	inf	3E+05	1798801	937451	0	52	1	94	7.88	13	7.88
1gwc_00021	4	12	203	4	6	10.29	28766	134930	77823	55	2	5	**16**	9.79	35	9.79
1gwc_00033	3	18	203	7	9	inf	56498	193247	178597	0	19	1	**12**	10.48	33	10.48
2hnu_00026	4	14	203	5	7	15.08	22010	105458	76657	38	0	4	**7**	13.18	63	13.18
2hnv_00025	4	16	203	6	8	15.04	1E+05	297138	84882	39	0	3	**16**	13.65	69	13.65
2rf9_00007	6	18	205	7	9	14.52	4264	11559	9772	36	5	1	**5**	14.08	23	14.08
2rf9_00013	5	16	205	6	8	14.12	1691	7851	7275	41	0	2	**2**	13.25	4	13.25
2rf9_00018	6	18	205	7	9	16.68	20137	85033	87306	78	0	4	**15**	15.79	18	15.79
2rf9_00042	6	22	205	9	11	inf	4E+05	1835478	1320278	0	162	1	151	22.65	41	22.65
2rfd_00035	6	16	205	6	8	17.70	9E+05	4253159	3273123	40	0	4	381	**17.27**	199	16.77
2rfe_00012	4	14	205	5	7	14.80	3126	10002	21164	37	0	3	**1**	13.93	6	13.93
2rfe_00014	4	14	205	5	7	15.23	4086	13086	26801	37	0	3	**2**	14.36	11	14.36
2rfe_00017	5	14	203	5	7	10.96	13148	54422	300675	49	4	6	**7**	10.52	9	10.52
2rfe_00030	4	14	203	5	7	11.53	20393	164126	359007	87	40	7	**19**	10.50	40	10.50
2rfe_00041	5	18	205	7	9	inf	2E+06	5483503	3336192	0	533	1	402	22.73	245	22.73
2rfe_00043	6	16	203	6	8	18.48	15390	40297	422357	34	43	4	80	18.04	9	18.04
2rfe_00044	6	16	203	6	8	18.62	37887	99927	1047107	30	3	6	86	18.19	8	18.19
2rfe_00047	3	18	205	7	9	inf	3E+05	1099600	1056935	0	121	1	88	22.70	42	22.70
2rfe_00048	4	20	205	8	10	inf	5E+05	1933600	1813547	0	219	1	159	22.81	71	22.81
2rl0_00008	4	10	203	3	5	**11.16**	2	3	0	40	0	3	**3**	**11.16**	28	9.46
2xgy_00020	4	14	203	5	7	11.47	43643	262523	743860	40	0	2	**14**	10.60	33	10.60
3cal_00032	6	16	203	6	8	13.38	1E+05	1067419	531976	32	6	4	125	11.62	17	11.62
3ma2_00016	4	14	205	5	7	13.39	12471	37071	57964	16	21	6	**4**	8.38	6	8.38
3u7y_00009	5	12	203	4	6	**4.51**	2	3	0	40	0	3	**6**	4.51	32	4.51
3u7y_00011	3	12	203	4	6	12.72	5757	16107	37892	36	0	5	**2**	11.85	13	11.85
4hem_00027	3	16	205	6	8	inf	4467	11460	18378	0	45	1	**2**	15.48	17	15.48
4hem_00028	3	16	205	6	8	inf	3177	11537	13051	0	41	1	**2**	15.27	18	15.27
4kt6_00023	4	16	203	6	8	14.80	38186	101546	23877	16	19	4	**7**	12.69	31	12.69
4wwi_00019	5	14	203	5	7	15.43	8094	30774	17888	40	0	2	**7**	14.99	8	14.99

Figure 4: F2 on problems with two mutable residue (ie. MAP) variables.

Problem	iB	|X|	Dmax	w*	d	UB	OR	AND	CCP	UBP	SSP	EH	time	K*MAP	BBK* t	BBK* sln
1gwc_00021	6	13	34	7	7	inf	15545642	81923965	446455833	0	34816	1	69088	**11.92**	113	11.72
2hnv_00025	6	17	34	9	9	inf	15597590	45361875	34618109	0	23995	1	17373	**16.18**	96	13.65
2rf9_00007	9	19	34	10	10	inf	6037876	35285773	56583854	0	8124	1	13287	14.73	137	14.73
2rf9_00013	6	17	34	9	9	inf	4804048	23029900	20157331	0	9614	1	7389	15.03	11	15.03
2rfe_00012	4	15	34	8	8	inf	22301	77935	133262	0	579	1	29	13.93	8	13.93
2rfe_00014	4	15	34	8	8	inf	33721	112331	202496	0	620	1	40	14.36	15	14.36
2rfe_00017	6	15	35	8	8	inf	4339838	20401558	110555327	0	18750	1	10810	**10.86**	22	10.80
2rfe_00030	6	15	35	8	8	inf	1632268	7396607	47354556	0	12412	1	6224	**11.12**	41	10.97
2xgy_00020	6	15	35	8	8	12.28	425512	2552281	7259396	652	226	30	774	10.96	42	10.96
3u7y_00009	6	13	34	7	7	5.39	221684	1419747	921627	1792	175	208	577	4.51	40	4.51
3u7y_00011	5	13	34	7	7	inf	23893	81707	183381	0	939	1	45	11.85	18	11.85
4wwi_00019	6	15	34	8	8	16.93	1257591	4864928	2446418	2672	865	152	1493	14.99	13	14.99

Figure 5: F1 on problems with three mutable resiude (ie. MAP) variables.

Problem	iB	|X|	Dmax	w*	d	UB	OR	AND	CPP	UBP	SSP	EH	time	K*MAP	BBK* t	BBK* sln
1gwc_00021	4	13	203	4	7	12.51	33881	590621	473189	388	6	8	205	**11.92**	113	11.72
2hnv_00025	4	17	203	6	9	18.38	215171	550559	220825	77	0	4	153	**16.18**	96	13.65
2rf9_00007	4	19	205	7	10	inf									137	14.73
2rf9_00013	6	17	205	6	9	15.47									11	15.03
2rfe_00012	5	15	205	5	8	14.36	3127	10003	32610	57	0	3	85	13.93	8	13.93
2rfe_00014	5	15	205	5	8	14.79	4087	13087	39411	57	0	3	85	14.36	15	14.36
2rfe_00017	5	15	203	5	8	11.46	245894	1063198	6389737	227	25	43	333	**10.86**	22	10.80
2rfe_00030	4	15	203	5	8	13.61	256957	1327425	2816050	726	83	77	274	**11.12**	41	10.97
2xgy_00020	5	15	203	5	8	11.39	398102	2383318	7422285	42	0	20	360	10.96	42	10.96
3u7y_00009	4	13	203	4	7	4.95	36760	228568	564654	204	7	28	99	4.51	40	4.51
3u7y_00011	4	13	203	4	7	12.29	5758	16108	68579	50	0	5	86	11.85	18	11.85
4wwi_00019	5	15	203	5	8	16.05	22945	87485	91677	176	75	5	180	14.99	13	14.99

Figure 6: F2 on problems with three mutable resiude (ie. MAP) variables.

nodes pruned due to bounding, *SSP* are the nodes pruned due to subunit-stability constraint violation, *EH* counts the number of times an exact heuristic was used instead of search, *time* is the elapsed algorithm time (in seconds), *K*MAP* is the highest K* value found that does not violate subunit stability constriants (in log_{10}), *BBK* t* is BBK*'s runtime (in seconds), *BBK* sln* is BBK*'s highest valid K* value found (in log_{10}). Highlighted in red are values that are significantly suboptimal, and highlighted in bright green under UB, time, and K*MAP are when an exact wMBE-K*MAP bound is used, when AOBB-K*MAP performs faster than BBK*, and when AOBB-K*MAP's reported K*MAP value is greater than that of BBK*, respectively.

K*MAP Value. AOBB-K*MAP outputted the same K*MAP value as BBK* for the majority of the problems (both F1 and F2, small and expanded) with the exception of two problems in the small set (2rfd_00035 and 2rl0_00008) and 4 problems (1gwc_00021, 2hnv_00025, 2rfe_00017, and 2rfe_00030) in the expanded set. Each exception resulted in a greater K*MAP value than that outputted by BBK*. Through further analysis, the solutions were verified to be valid based on the F1 and F2 problem formulations, however BBK* was unable to be used to obtain the K* value for the corresponding configurations as confirmation.

wMBE-K*MAP Heuristic. For small problems F1, wMBE-K*MAP was able to compute an exact K*MAP solution for five problems (not explicitly shown), however, the elapsed time was faster when a lower i-bound was used. For six of the problems, a bounded heuristic could not be computed even with the highest i-bound (not explicitly shown). For expanded problems, the heuristic on F1 was unable to produce bounds on half of the problems. Analysis of the heuristic showed that in the cases that it was unable to produce global bounds, the heuristic was also universally uninformative, namely unbounded for all MAP variables. This was found to be to due to lower bounding of the $\boldsymbol{C_U}$ buckets to zero values likely due to the implicit constraints found in the functions. In contrast, the heuristic applied to small F2 was able to determine finite bounds on the K*MAP value for all but two problems (not explicitly shown). Furthermore, an exact heuristic was able to be computed for 17 of the 29 small problems. The better performance of the heuristic on F2 is believed to be due to the reduced induced width of the F2 as compared to F1 (compare w^* across Figures 3 and 4 and Figures 5 and 6) and explicit, rather than implicit, constraints.

Time. Under *time* in Figures 3-6, highlighted in bright green, we see when AOBB-K*MAP finds a solution faster than BBK*, and, in red, when its runtime is more than ten fold slower than BBK*. We see that for small problems (Figure 4), AOBB-K*MAP on F2 did particularly well outperforming BBK* 20 out of 29 times. Under *EH* we can see that, in many of these cases, an exact value computed by the heuristic was used instead of search showing the value of the statically compiled heuristic. However, AOBB-K*MAP performed worse on the expanded problems, believed to be due to weaknesses in the heuristic (to be addressed as described in Future Work).

7 Conclusion and Future Work

Conclusion. This work provides three contributions to the advancement of CPD algorithms. First, a new graphical model framework of two distinct problem formulations is presented for which state-of-the-art algorithms over AND/OR search spaces can be adapted to address CPD as a K*MAP task. Second, a new statically compiled wMBE-K*MAP heuristic is presented which can guide K*MAP search over AND/OR search spaces. And finally, an AND/OR branch-and-bound algorithm for optimizing K*, AOBB-K*MAP, was presented with experiments showing promise outperforming state-of-the-art BBK* on small benchmark problems, yet leaves room for advancement to address shortcomings on larger benchmarks.

Future Work. Future directions include: (1) adaptation of advanced state-of-the-art MMAP algorithms such as Recursive Best-First AND/OR Search (Marinescu et al. 2018) to solving the K*MAP query, including anytime approximation schemes such as Learning Depth-First or Stochastic Best-First AND/OR Search (Marinescu et al. 2018), and incorporating state-of-the-art sampling methods such as Dynamic Importance Sampling (Lou, Dechter, and Ihler 2019) or Abstraction Sampling (Kask et al. 2020); (2) addressing challenges of wMBE-K*MAP heuristic, particularly during lower bounding, such as by incorporating constraints into the heuristic generation, using an alternate bucket elimination methods such as Deep Bucket Elimination (Razeghi et al. 2021), or using a mix of statically compiled and dynamic heuristics; (3) analysis of the trade-offs between the two problem formulations presented and exploration of how to exploit their strengths and mitigate their weaknesses.

Algorithm 1: wMBE-K*MAP

input : Graphical model $\boldsymbol{\mathcal{M}} = \{\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}\}$; evidence e; constrained variable order $o = [X_1, ..., X_n]$ with MAP variables first; a partition of $\boldsymbol{X}$ into $\boldsymbol{R}$, $\boldsymbol{C_B}$, and $\boldsymbol{C_U}$; an i-bound i

output: upper bound on the K*MAP value

begin
  Condition each $f \in \boldsymbol{F}$ according to the provided evidence e and remove the corresponding variables from the scopes of the functions.
  Partition each conditioned f into buckets $B_n, ..., B_1$ s.t. each F is placed in the greatest bucket corresponding to a variable in its scope.
  foreach $k = n...1$ **do**
    Generate a mini-bucket partitioning of the bucket functions $\boldsymbol{MB_k} = \{MB_k^1, ..., MB_k^T\}$ s.t. the number of variables in the scopes of the functions of any mini bucket $MB_k^t \in \boldsymbol{MB_k}$ is $\leq i$
    if $X_k \in \boldsymbol{MAP}$ **then**
      foreach $MB_k^t \in \boldsymbol{MB_k}$ **do**
        $\lambda_k^t \leftarrow \max_{X_k} \prod_{f \in MB_k^t} f$
      end
    else
      if $X_k \in \boldsymbol{C_B}$ **then**
        Select a set of positive weights $\boldsymbol{w} = \{w_1, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$
        foreach $MB_k^t \in \boldsymbol{MB_k}$ **do**
          $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$
        end
      else if $X_k \in \boldsymbol{C_U}$ **then**
        Select a negative weight for w_1
        Select a set of positive weights $\boldsymbol{w} = \{w_2, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$
        foreach $MB_k^t \in \boldsymbol{MB_k}$ **do**
          $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$
          if $scope(\lambda_k^t) \cap \boldsymbol{C_U} = \emptyset$ **then**
            $\lambda_k^t \leftarrow 1/\lambda_k^t$
          end
        end
      end
    end
    Add each λ_k^t to the bucket of the highest-index variable in its scope.
  end
  return λ_1
end

Algorithm 2: AOBB-K*MAP

input : CPD graphical model $\mathcal{M}$; pseudo-tree $\mathcal{T}$; K^* upper-bounding heuristic function $h_{K^*}^{ub}(.)$; Z_γ upper-bounding heuristic function $h_{Z_\gamma}^{ub}(.)$; and subunit stability threshold $threshold(\gamma)$ for each subunit $\gamma \in \varphi$

output: $K^*MAP(\mathcal{M})$

begin
  Initialize $MiniSat$ with constraints from $\mathcal{M}$
  and generate literals via constraint propagation
  $\pi \leftarrow$ dummy AND node n_D
  $ub_{K^*}(n_D) \leftarrow \prod_{m \in ch_T(n_D)} h_{K^*}(m)$
  $lb_{K^*}(n_D) \leftarrow -inf$
  $g(n_D) \leftarrow 1$
  foreach $\gamma \in \varphi$ **do**
    $A_{Z_\gamma}^{\times}(n_D) \leftarrow 1$
    $A_{Z_\gamma}^{+}(n_D) \leftarrow 0$
    $ub_{Z_\gamma}(n_D) \leftarrow \prod_{m \in ch_{T_\gamma}(n_D)} h_{Z_\gamma}(m)$
  end
  while $n_X \leftarrow EXPAND(\pi)$ **do**
    if $MiniSat(\pi) = false$ **then**
      $PRUNE(\pi)$
    else if $\exists \gamma \in \varphi$ *s.t.* $UB_{Z_\gamma}(n_X) < threshold(\gamma)$ **then**
      $PRUNE(\pi)$
    else if $X \in \boldsymbol{R}$ **then**
      if $\exists a \in anc^{OR}(n)$ *s.t.* $ub_{K^*}(a, \pi) < lb_{K^*}(a)$ **then**
        $PRUNE(\pi)$
      end
    else if $ch_T^{unexp}(n) = \emptyset$ **then**
      $BACKTRACK(\pi)$
  end
  return $ub_{K^*}(n_D) = lb_{K^*}(n_D) = K^*MAP(\mathcal{M})$
end

Acknowledgements Special thanks to Bruce Donald, Graham Holt, Jonathan Jou, and Thomas Schiex for their support and guidance in our delving into the computational protein design domain. A warm thanks also to the reviewer for their time and comments.

References

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113: 41–85.

Dechter, R. 2013. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Dechter, R. 2019. Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms, Second Edition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13: 1–199.

Dechter, R.; and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3): 73–106.

Dechter, R.; and Rish, I. 2002. Mini-buckets: A general scheme for approximating inference. *Journal of the ACM*, 107–153.

Eén, N.; and Sörensson, N. 2004. An Extensible SAT-solver. In Giunchiglia, E.; and Tacchella, A., eds., *Theory and Applications of Satisfiability Testing*, 502–518. Berlin, Heidelberg: Springer Berlin Heidelberg.

Freiser, H. 1962. The determination of stability constants and other equilibrium constants in solution (Rossotti, Francis J. C.; Rossotti, Hazel). *Journal of Chemical Education*, 39(7): 379.

Hallen, M.; Martin, J.; Ojewole, A.; Jou, J.; Lowegard, A.; Frenkel, M.; Gainza, P.; Nisonoff, H.; Mukund, A.; Wang, S.; Holt, G.; Zhou, D.; Dowd, E.; and Donald, B. 2018. OSPREY 3.0: Open-source protein redesign for you, with powerful new features. *Journal of Computational Chemistry*, 39.

Hallen, M. A.; and Donald, B. R. 2019. Protein Design by Provable Algorithms. *Commun. ACM*, 62(10): 76–84.

Hardy, G.; Littlewood, J.; and Pólya, G. 1988. *Inequalities*. Cambridge Mathematical Library. Cambridge University Press. ISBN 9781107647398.

Kask, K.; Pezeshki, B.; Broka, F.; Ihler, A.; and Dechter, R. 2020. Scaling Up AND/OR Abstraction Sampling. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 4266–4274. International Joint Conferences on Artificial Intelligence Organization. Main track.

Lee, J.; Marinescu, R.; Dechter, R.; and Ihler, A. 2016. From Exact to Anytime Solutions for Marginal MAP. AAAI'16, 3255–3262. AAAI Press.

Liu, Q.; and Ihler, A. 2011. Bounding the Partition Function using Hölder's Inequality. In *International Conference on Machine Learning (ICML)*, 849–856. New York, NY, USA: ACM.

Lou, Q.; Dechter, R.; and Ihler, A. 2019. Interleave variational optimization with Monte Carlo sampling: A tale of two approximate inference paradigms.

Marinescu, R.; Dechter, R.; and Ihler, A. 2014. AND/OR Search for Marginal MAP. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, 563–572. Arlington, Virginia, USA: AUAI Press. ISBN 9780974903910.

Marinescu, R.; Lee, J.; Dechter, R.; and Ihler, A. 2018. AND/OR Search for Marginal MAP. *J. Artif. Int. Res.*, 63(1): 875–921.

Ojewole, A.; Jou, J. D.; Fowler, V. G.; and Donald, B. R. 2018. *BBK* (Branch and Bound Over K*):* A Provable and Efficient Ensemble-Based Protein Design Algorithm to Optimize Stability and Binding Affinity Over Large Sequence Spaces. *J. Comput. Biol.*, 25(7): 726–739.

Park, J. D. 2002. MAP Complexity Results and Approximation Methods. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, 388–396. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558608974.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Ping, W.; Liu, Q.; and Ihler, A. T. 2015. Decomposition Bounds for Marginal MAP. In *Advances in Neural Information Processing Systems 28*, 3267–3275. Curran Associates, Inc.

Razeghi, Y.; Kask, K.; Lu, Y.; Baldi, P.; Agarwal, S.; and Dechter, R. 2021. Deep Bucket Elimination. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 4235–4242. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Viricel, C.; de Givry, S.; Schiex, T.; and Barbe, S. 2018. Cost Function Network-based Design of Protein-Protein Interactions: predicting changes in binding affinity. *Bioinformatics (Oxford, England)*, 34.

Vucinic, J.; Simoncini, D.; Ruffini, M.; Barbe, S.; and Schiex, T. 2019. POsitive Multistate Protein design. *Bioinformatics (Oxford, England)*, 36.

Zhou, Y.; Wu, Y.; and Zeng, J. 2016. Computational Protein Design Using AND/OR Branch-and-Bound Search. *Journal of computational biology : a journal of computational molecular cell biology*, 23.