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

Abstract

The importance of designing proteins, such as high affinity antibodies, has become ever more apparent. Computational Protein Design can cast such design problems as optimization tasks with the objective of maximizing K^*, an approximation of binding affinity. Here we lay out a graphical model framework for K^* optimization that enables use of compact AND/OR search spaces. We introduce two distinct graphical model formulations, a new K^* heuristic, AOBB-K^* - an efficient depth-first branch-and-bound algorithm, and modifications that improve performance with theoretical guarantees. As AOBB-K^* is inspired by algorithms from the well studied task of Marginal MAP, this work provides a foundation for adaptation of state-of-the-art mixed inference schemes to protein design.

1 INTRODUCTION

Graphical models provide a powerful framework for reasoning about conditional dependencies over many variables. The 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 marginals of conditionally disjoint subsets of the remaining variables. This ratio is known as K^*. In computational protein design (**CPD**), K^* estimates affinity between interacting subunits. Maximizing K^* corresponds to maximizing the likelihood that the subunits will associate Hallen and Donald [2019].

Like MMAP, K^*MAP distinguishes between maximization (MAP) variables and summation (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 sum and max operations do not commute. This forces constrained variable orderings that may have significantly higher induced widths Dechter [1999, 2019]. This also implies larger search spaces when using search algorithms or larger messages when using message-passing schemes. Furthermore, bounding K^* requires both upper and lower bounding of marginals, producing an additional challenge.

Nevertheless, over the last several years, there have been several advances in algorithms for solving the MMAP task Marinescu et al. [2019, 2018a,b], 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 four main contributions:

1. Two formulations of K^*MAP as a graphical model
2. wMBE-K^*, a weighted mini-buckets algorithm for K^*MAP, enhanced with a domain partitioning scheme
3. AOBB-K^*, a depth-first branch-and-bound algorithm over AND/OR search spaces for solving K^*MAP
4. A thresholding scheme to exploit determinism accompanied with correctness guarantees
5. Extensive empirical analysis comparing these schemes to state-of-the-art BBK^* illustrating their potential

2 BACKGROUND

Here we provide a brief background about protein design and graphical models.

2.1 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 Dahiyat and Mayo [1996]. In CPD, certain amino acid positions (or **residues**) of a protein-of-interest are deemed as mutable - these are amino acid

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

positions where different amino acid mutations will be considered - and through a computational process, a preferred sequence is determined Donald [2011].

Throughout the process, sets of mutations are explored, each comprising a specific amino acid sequence. Given a sequence (or in some methods, even partial sequence) an estimate of the resulting protein's goodness can be estimated. This goodness is determined by considering the possible conformations of the protein, 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.

To remedy, simplifications can be made: (i) consider only a subset of residues as mutable, (ii) discretize side-chain conformations into rotamers, and (iii) assume 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 et al. [2016].

2.2 K* AND K*MAP

The affinity between two protein subunits P and L relates to the rate at which they bind into a complexed state PL and dissociate reforming P and L (as indicated by the chemical equation: $P + L \rightleftharpoons PL$). This said equilibrium is associated with a constant, K_a, and can be determined in vivo by computing the ratio of persisting concentrations of each species by $K_a = \frac{[PL]}{[P][L]}$. However, in order to compare K_a values of various designs in vivo, it is necessary to synthesize the protein subunits through molecular processes that are both timely and costly.

K_a can also be approximated as $K^{\int} = \frac{Z^{\int}_{PL}}{Z^{\int}_{P} Z^{\int}_{L}}$, where $Z^{\int}_{\gamma} = \int_{\mathcal{C}} e^{-\frac{E_{\gamma}(c)}{\mathscr{R}T}} dc$ and $Z^{\int}_{PL}$, $Z^{\int}_{P}$, and $Z^{\int}_{L}$ are approximations of partition functions of the bound and unbound states that capture the entropic contributions of their various conformations $\mathcal{C}$ Hill [1987], McQuarrie [2000]. $E_{\gamma}(c)$ represents the energy of a particular conformation c of state $\gamma \in \varphi$ where $\varphi = \{P, L, PL\}$, $\mathscr{R}$ 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* Lilien et al. [2004]:

$$K^* = \frac{Z_{PL}}{Z_P Z_L}, \quad Z_{\gamma} = \sum_{c \in \boldsymbol{D}(\boldsymbol{C})} e^{-\frac{E_{\gamma}(c)}{\mathscr{R}T}} \tag{1}$$

Due to independence between residues across the dissociate subunits, we can generalize further $K^* = \frac{Z_B}{Z_U}$, where B represents the bound (complexed) state(s) and U represents the unbound (dissociate) states. (For a two-subunit system, $B = \{PL\}$ and $U = \{P\} \cup \{L\}$). This more generalized representation, can be used directly for K* computations involving more than two subunits.

A common goal in protein design is to maximize protein-ligand interaction. Previously, this was done by minimizing an objective called the GMEC (global minimum energy conformation) over only the complexed protein state PL Ruffini et al. [2021], Hallen and Donald [2019], Zhou et al. [2016]. The GMEC, being a pure minimum over the energies of the complex's conformations, ignores the realization that protein structures are dynamic. Furthermore, by the GMEC focusing only on the protein's complexed state, it ignores the dynamicity of the subunit interactions. However, since minimizing the GMEC results in a pure optimization task - a task much easier than that of mixed inference, many solvers use this objective. On the other hand, the stronger K* objective captures both the dynamicity of protein conformations and subunit interactions. K*MAP is the formalization of computational protein design as a task to maximize K*,

$$K^*\text{MAP} = \underset{\boldsymbol{R}}{\operatorname{argmax}}\, K^*(r) \tag{2}$$

where we look for amino acid assignments $\boldsymbol{R} = r$ that maximize K*. The goal of recent work, and that presented here, is to develop efficient algorithms for computing K*MAP, from which one can predict a small set of promising sequences to experiment on in vivo, saving great time and cost.

2.3 GRAPHICAL MODELS AND THEIR AND/OR SEARCH SPACES

Our work taps into algorithms developed for the MMAP task defined on graphical models. A **graphical model**, such as a Bayesian or 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 $\mathbf{D} = \{D_i : i \in D\}$ 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 $f_{\alpha} : D_{\alpha} \rightarrow R^{\geq 0}$. **Primal graph** $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ of $\mathcal{M}$ 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 function. $\mathcal{M}$ defines a global function, often a factorized probability distribution on $\mathbf{X}$, $P(\mathbf{X}) = \frac{1}{Z} \prod_{\alpha} f_{\alpha}(X_{\alpha})$. The partition function, Z, normalizes the probability.

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. [2018b].

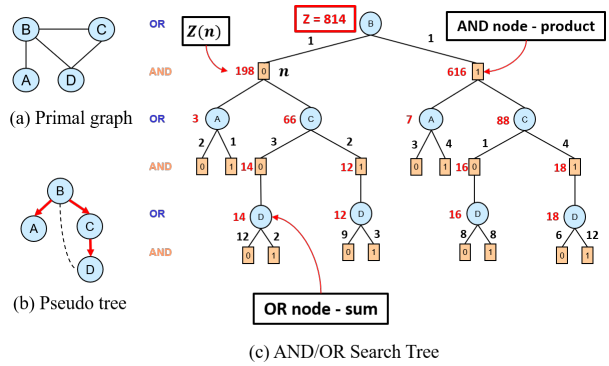


Figure 1: A full AND/OR tree representing all 16 solutions.

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 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 (see also Dechter and Mateescu [2007]).

The size of the search space is exponential in the depth of the pseudo tree (rather than the number of variables) and therefore mixed inference tasks such as MMAP can be solved efficiently by traversing the AND/OR search space.

3 GRAPHICAL MODEL FOR K*MAP

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

3.1 FORMULATION 1 (F1)

Variables and Domains: Let $\boldsymbol{R} = \{R_i \,|\, i \in \{1, 2, ..., N\}\}$ be the set of **residue variables** representing N different residues (i.e., positions) of the protein. Each R_i has a corresponding domain $D_{R_i} = \{aa \mid aa$ 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.

$\boldsymbol{C_\gamma} = \{C_{\gamma(i)} \mid i \in \{1, 2, ..., N\}\}$ are **conformation variables**, each indexing a discrete spacial conformation (rotamer) of the amino acid at residue R_i when its subunit is in state γ. Each $C_{\gamma(i)}$ has corresponding domain $D_{C_{\gamma(i)}} = \{1, 2, ..., M_{\gamma(i)}\}$, where $M_{\gamma(i)}$ is the maximum number of rotamers for any amino acid assignment to R_i in state γ. The assignment to $C_{\gamma(i)}$ acts as an index to the possible side chain conformations of the amino acid assigned to R_i. The $\boldsymbol{C_\gamma}$ are the SUM variables marginalized over.

Functions: We consider two sets of functions: $E^{sb}_\gamma = \{E^{sb}_{\gamma(i)}(R_i, C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}\}$ that capture the energies of interaction of the amino acid at each residue i with itself and the surrounding backbone, and $E^{pw}_\gamma = \{E^{pw}_{\gamma(ij)}(R_i, C_{\gamma(i)}, R_j, C_{\gamma(j)}) \mid$ for i, j s.t. R_i and R_j interact$\}$ that capture pair-wise energies of interaction between amino acids in close proximity. For any assignment to $C_{\gamma(i)}$ (which corresponds to a rotamer-index) that is out of range of R_i's assigned amino acid's possible rotamers, an infinite energy value is assigned as an implicit constraint.

Objective: The K* objective can thus 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 $\mathscr{R}$ and where:

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

3.2 FORMULATION 2 (F2)

Formulation 2, inspired by the works of Viricel et al. [2018] and Vucinic et al. [2019], uses explicit constraints to restrict invalid amino acid - rotamer combinations.

Variables and Domains: As in F1, a set of residue variables $\boldsymbol{R} = \{R_i \mid i \in \{1, 2, ..., N\}\}$ and conformation variables $\boldsymbol{C_\gamma} = \{C_{\gamma(i)} | i \in \{1, 2, ..., N\}\}$ are considered. Here, each $C_{\gamma(i)}$ represents a specific amino acid and conformation of the N different residues. Namely, each $C_{\gamma(i)}$ has a domain $D_{C_{\gamma(i)}} = \{c \mid c$ is a rotamer for one of the possible amino acids of $R_i\}$. The amino acid assignment to R_i acts as a selector into the possible assignments to $C_{\gamma(i)}$.

Functions: $\mathscr{C} = \{\mathscr{C}_{\gamma(i)}(R_i, C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}, \gamma \in \varphi\}$ is a set of constraints ensuring that the assigned rotamer to $C_{\gamma(i)}$ belongs to the amino acid assigned to R_i. In addition, $E^{sb}_\gamma = \{E^{sb}_{\gamma(i)}(C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}\}$ captures the energies of interaction of the amino acid at each residue i with itself and the surrounding backbone and $E^{pw}_\gamma = \{E^{pw}_{\gamma(ij)}(C_{\gamma(i)}, C_{\gamma(j)}) \mid$ for i, j s.t. R_i and R_j interact$\}$ captures the pair-wise energies of interaction between amino acids in close proximity.

Objective: 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

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

Algorithm 1: wMBE-K*

input : Graphical model $\boldsymbol{\mathcal{M}} = \{\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}\}$; constrained variable order $o = [X_1, ..., X_n]$ (where $\boldsymbol{X} = \boldsymbol{R} \cup \boldsymbol{C_B} \cup \boldsymbol{C_U}$), i-bound i

output : upper bound on the K*MAP value

1 **begin**

2 Partition the functions $f \in \boldsymbol{F}$ into buckets $B_n, ..., B_1$ s.t. each function is placed in the bucket corresponding to the highest-index variable in its scope.

3 **foreach** $k = n...1$ **do**

4 Generate a mini-bucket partitioning of the bucket functions $\boldsymbol{MB_k} = \{MB_k^1, ..., MB_k^T\}$ s.t. $vars(MB_k^t) \leq i$, for all $MB_k^t \in \boldsymbol{MB_k}$

5 **if** $X_k \in \boldsymbol{MAP}$ **then**

6 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**

7 $\lambda_k^t \leftarrow \max_{X_k} \prod_{f \in MB_k^t} f$

8 **else**

9 **if** $X_k \in \boldsymbol{C_B}$ **then**

10 Select positive weights $\boldsymbol{w} = \{w_1, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$

11 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**

12 $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$

13 **else if** $X_k \in \boldsymbol{C_U}$ **then**

14 Select a negative weight w_1 and positive weights $\boldsymbol{w} = \{w_2, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$

15 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**

16 $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$

17 **if** $scope(\lambda_k^t) \cap \boldsymbol{C_U} = \emptyset$ **then**

18 $\lambda_k^t \leftarrow 1/\lambda_k^t$

19 Add each λ_k^t to the bucket of the highest-index variable in its scope.

20 **return** λ_1

With a graphical model framework in place, next we describe a heuristic that can be used to bound the K*MAP value and that can be used to guide search.

4 WMBE-K*

We present next a weighted mini-bucket based approximation scheme for K^*, which adapts the weighted mini-buckets for MMAP Dechter and Rish [2002], Marinescu et al. [2014] to the new K^* objective. Since the constrained order mini-bucket bounds are compatible with AND/OR search we will leverage these bounds in the subsequent section to guide search as part of a branch-and-bound scheme.

The algorithm, called wMBE-K*, is described in Algorithm 1 and operates similarly to wMBE-MMAP Marinescu et al. [2014]. Two key similarities are that (1) it takes a variable ordering that constrains buckets of MAP variables to be processed last (line 3) 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 4) and taking the product of their power-sums over the bucket variable (lines 9-12, 13-16), leveraging Holder's Inequality Hardy et al. [1988], namely $\sum_x^w f(x) = (\sum_x f(x)^{\frac{1}{w}})^w$ Marinescu et al. [2014].

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 13-16), 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 18). Although details are omitted here, wMBE-K* can also employ cost shifting to tighten its bounds (see Liu and Ihler [2011]). In our empirical evaluation cost-shifting is implemented as well. Finally, the complexity of the algorithm, which is parametrized by the i-bound i, is time and space exponential in i only.

As can be expected, bounding a ratio of functions (as in the case for K*) is particularly challenging, relying on both upper and lower bounds. Lower bounding of functions is particularly challenging. For larger problems and low i-bounds, this can often yield relatively weak bounds. We provide an improvement to help remedy this next.

Domain-Partitioned MBE Until now, we used a mini-bucket scheme blind to explicit hard constraints and consistency issues. This can be handicapping when lower-bounding since constraints are represented as zero's in functions and can cause premature deflation of lower bounds. In particular, in mini-bucket elimination Dechter and Rish [2003] where lower bounds are created via minimizing over function values, zeros in the functions being minimized will cause the resulting lower bound itself to drop to zero. However, in the CPD domain where functions represent protein energetics, satisfiable configurations corresponds to positive function values and thus can guarantee a positive lower bound by using the following simple remedy:

Let X, Y, and Z be three variables and let $obj = \sum_X f(x,y) \cdot g(x,z)$, let $X' = \{x \in X | g(x,z) \neq 0\}$ be a set such that $\epsilon_{X'} = min_{x \in X'} g(x,z)$. Clearly, $\epsilon_{X'} > 0$ and therefore we can derive: $obj = \sum_{x \in X'} f(x,y) \cdot g(x,z) + \sum_{x \in X \setminus X'} f(x,y) \cdot g(x,z) = \sum_{x \in X'} f(x,y) \cdot g(x,z) \geq \epsilon_{X'} \cdot \sum_{x \in X'} f(x,y) > 0$, assuming $f(x,z)$ is not identically zero over X'. Mini-buckets can then be computed according to such domain-partitions to improve their bounds.

5 AOBB-K*

We present the key algorithmic contribution of this work: **AOBB-K*** (Algorithm 2), a depth-first AND/OR branch-and-bound scheme for solving the K*MAP task. With state-

Algorithm 2: AOBB-K*

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

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

1 **begin**
2 Encode deterministic relations in $\mathcal{M}$ into CNF
3 $\pi \leftarrow$ root OR node s
4 $ub_{K^*}(s) \leftarrow h_{K^*}(s)$
5 $lb_{K^*}(s) \leftarrow -inf$
6 $g(s) \leftarrow 1$
7 **foreach** $\gamma \in \varphi$ **do**
8 $UB_{Z_\gamma}(s) \leftarrow \prod_{m \in ch_{T_\gamma}(s)} h_{Z_\gamma}(m)$
9 **while** $n_X \leftarrow EXPAND(\pi)$ **do**
10 **if** $ConstraintPropagation(\pi) = false$ **then**
11 $PRUNE(\pi)$
12 **else if** $\exists \gamma \in \varphi$ *s.t.* $UB_{Z_\gamma}(n_X) < S_\gamma$ **then**
13 $PRUNE(\pi)$
14 **else if** $X \in \boldsymbol{R}$ **then**
15 **if** $\exists a \in anc^{OR}(n)$ *s.t.* $ub_{K^*}(a, \pi) < lb_{K^*}(a)$ **then**
16 $PRUNE(\pi)$
17 **else if** $ch_T^{unexp}(n) = \emptyset$ **then**
18 $BACKTRACK(\pi)$
19 **return** $ub_{K^*}(s) = lb_{K^*}(s) = K^*MAP(\mathcal{M})$

of-the-art K* optimizers employing memory intensive best-first search Ojewole et al. [2018], Hallen et al. [2018], depth-first algorithms provide a search methodology linear in space allowing for solving problems unable to be solved by best-first methodologies due to memory Zhou et al. [2016].

AOBB-K* traverses the underlying AND/OR search tree guided by the provided pseudo tree $\mathcal{T}$, expanding nodes in a depth-first manner (line 9), and pruning whenever any of three conditions are triggered: (1) the resulting variable assignments violate any of the hard constraints induced by the infinite energy function values present in $\mathcal{M}$ (line 10), (2) a subunit-stability constraint (SSC – a constraint which enforce the partition function of each protein subunit, Z_γ, to be greater than an inputted biologically-relevant threshold S_γ Ojewole et al. [2018]) is violated (line 12), or (3) it can be asserted that the current amino acid configuration cannot produce a K* better than any previously found (line 15). Backtracking occurs when all of the node's children have been explored and returned from (line 17), at which point the K* value of the sub problem the node roots is known exactly and the bounds of its parents are tightened accordingly. The infinite energy tuples in $\mathcal{M}$'s functions model inconsistent amino acid - rotamer pairs and correspond to hard constraints which we encode into a CNF formula using the encoding of Allen and Darwiche [2003]. During search, unit-propagation (e.g., Eén and Sörensson [2004]) is employed to propagate these constraints and detect infeasible variable configurations.

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

Throughout search, each node n maintains a progressive upper bound $ub_{K^*}(n)$ on the K*MAP of the sub problem it roots. When a node is first expanded, this value is initialized based on upper-bounding heuristic function $h^{ub}_{K^*}(.)$ (line 4). As search progresses, $ub_{K^*}(n)$ decreases, converging towards the K*MAP of the sub problem rooted at n. Furthermore, each node n also maintains a progressively improved upper bound on the partition function of each subunit γ consistent with the path to n, $UB_{Z_\gamma}(n)$ (line 8). At each step in the search, $UB_{Z_\gamma}(n)$ is recomputed to ensure that it is greater than the given S_γ, thus satisfying the SSC's and enforcing consideration of only biologically relevant solutions Ojewole et al. [2018]. Note that the SSC's are not encoded into the problem, and thus add another layer of complexity on the search not present in tasks such as MMAP.

Theorem 5.1 (correctness, completeness)**.** *AOBB-K* is sound and complete, returning the optimal K* value of all amino-acid configurations that do not violate the inputted subunit-stability constraints.*

Clearly, the algorithm is time exponential in the depth of the guiding pseudo tree and it can operate in linear space.

5.1 WEIGHTED SEARCH FOR K*

Weighted best-first search (e.g., WA* Pohl [1970], WAO* Desarkar et al. [1987]) is a well known principle for converting best-first search into an anytime scheme by multiplying the heuristic function $h(n)$ of a node n in the search space by a weight $\omega > 1$. The solution is guaranteed to be ω-optimal (i.e., within a factor ω from the optimal one).

Therefore, AOBB-K* can easily be relaxed to an ω-approximation scheme (for $\omega \in [0, 1)$) by multiplying $h_{K^*}(n)$ at each node n in the AND/OR search tree by a factor of $(1 - \omega)$. Note that we use $\omega < 1$ because K*MAP is a maximization task. It can be shown that the resulting solution will be at worst $(1-\omega) \cdot K^*MAP$ Pohl [1970]. We explore the performance of applying such approximations to a class of difficult CPD problem instances in Section 6.

5.2 INFUSING DETERMINISM VIA THRESHOLDED UNDERFLOWS

Exploiting determinism via constraint propagation (**CP**) can be a powerful tool Dechter [2019] helping to prune invalid configurations corresponding to inconsistent amino acid - rotamer pairs or configurations not contributing to a subunit partition function. CP can be accelerated by introducing additional determinism such that the solution is unchanged.

Namely, extremely small function values (corresponding to extremely unfavorable side-chain interactions that would not occur in feasible solutions) can be underflowed (ie. replaced by zeros) and treated as hard constraints.

Let f be a non-negative function and $\tau \in \mathbb{R}^+$. The τ-underflow of f is $f^\tau(x) = f(x)$ if $f(x) \geq \tau$ and 0, otherwise. For $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, the τ-underflow of $\mathcal{M}$ is $\mathcal{M}^\tau = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}^\tau \rangle$, where $\mathbf{F}^\tau = \{f^\tau \mid f \in \boldsymbol{F}\}$.

We denote: f_γ^{max} - the maximum value in all the functions associated with subunit γ, $|\boldsymbol{f_\gamma}|$ - the number of functions included in the partition function computation for subunit γ, excluding explicit constraints, $|\boldsymbol{C_\gamma}|^{\Uparrow}$ - the cardinality of the Cartesian product of greatest number of assignments to each variable in $\boldsymbol{C_\gamma}$ that are individually consistent with any assignment $\boldsymbol{R} = r$, Z_B^{min} - the smallest Z_B that could possibly lead to a valid $K^* > K^{*(wt)}$, $prec(x)$ - the smallest decimal place for which the value of x is recorded.

We give two sufficient conditions that, when upheld, allow us to assert $K^*MAP(\mathcal{M}^\tau) = K^*MAP(\mathcal{M})$: (1) τ-underflows do not alter subunit partition functions that satisfy SSC's, i.e., $\forall r \in \boldsymbol{R}$ s.t. $Z_\gamma^{\mathcal{M}}(r) \geq S_\gamma,\ Z_\gamma^{\mathcal{M}}(r) = Z_\gamma^{\mathcal{M}^\tau}(r)$ and (2) τ-underflows do not alter any $Z_B \geq Z_B^{min}$, i.e., $\forall r \in \boldsymbol{R}$ s.t. $Z_B^{\mathcal{M}}(r) \geq Z_B^{min},\ Z_B^{\mathcal{M}}(r) = Z_B^{\mathcal{M}^\tau}(r)$. These conditions can be verified in linear time.

Condition 1: A problem created with underflows using an underflow threshold τ such that $\forall \gamma \in \varphi$: (i) if $\tau \leq 1$, $|\boldsymbol{C_\gamma}|^{\Uparrow} \cdot (f_\gamma^{max})^{|\boldsymbol{f_\gamma}|} \cdot (\tau) < 0.5 \cdot prec(S_\gamma)$, or (ii) if $\tau > 1$, $|\boldsymbol{C_\gamma}|^{\Uparrow} \cdot (f_\gamma^{max})^{|\boldsymbol{f_\gamma}|} \cdot (\tau)^{|\boldsymbol{f_\gamma}|} < 0.5 \cdot prec(S_\gamma)$ will not violate any subunit stability constraints.

Condition 2: A problem created with underflows using an underflow threshold τ such that τ does not violate the SSC's and such that: (i) if $\tau \leq 1$, $|\boldsymbol{C_B}|^{\Uparrow} \cdot (f_B^{max})^{|\boldsymbol{f_B}|} \cdot (\tau) < 0.5 \cdot prec(Z_B^{min})$, or (ii) if $\tau > 1$, $|\boldsymbol{C_B}|^{\Uparrow} \cdot (f_B^{max})^{|\boldsymbol{f_B}|} \cdot (\tau)^{|\boldsymbol{f_B}|} < 0.5 \cdot prec(Z_B^{min})$, will also not alter the K*MAP solution.

Theorem 5.2. *If a chosen τ satisfies Conditions 1 and 2, then $K^*MAP(\mathcal{M}^\tau) = K^*MAP(\mathcal{M})$.*

6 EMPIRICAL EVALUATION

We now present the empirical evaluation of our schemes on real protein benchmarks provided to us by *ANONYMOUS*.

We experimented on 48 protein design benchmarks, 30 with two mutable residues (denoted "original"), 12 made harder expanding to three mutable residues ("expanded"), and six to four mutable residues ("hard"). Problems were generated using OSPREY 3.0 Hallen et al. [2018] to compute interaction energies and then formulated into F1 and F2.

AOBB-K* vs BBK*. We run AOBB-K* and its derived ω-approximate algorithm scheme ω-AOBB-K* implemented in C++ built on top of AOBB-MMAPMarinescu et al. [2014]. For comparison, we also run BBK* Ojewole et al. [2018], a state-of-the-art best-first search algorithm available in OSPREY 3.0. BBK* (implemented in Java, set to use rigid rotamers and with a bound-tightness of 1×10^{-2001}). Experiments were allotted 1hr on a 2.66 GHz processor with 4 GB of memory and using subunit-stability thresholds $S_\gamma = Z_\gamma^{(wt)} \cdot e^{-\frac{5}{\mathscr{R}T}}$ where $Z_\gamma^{(wt)}$ is the partition function of the wild-type sequence. As BBK* can take advantage of parallelism, it was run with access to 4 CPU cores.

Table 1: Aggregated statistics comparing the solution and time of AOBB-K* (on both F1 and F2, as an exact and anytime scheme) with BBK*. **K*≥** counts the the number of times AOBB-K*'s K* solution was equal or greater; **K*>** counts the number of AOBB-K*'s solution was strictly better; **time<** counts the number of times AOBB-K* found the optimal solution faster.

	AOBB-K*			**any-AOBB-K***	
Dataset (#instances)	**K*≥** (F1,F2)	**K*>** (F1,F2)	**time<** (F1,F2)	**K*≥** (F1,F2)	**K*>** (F1,F2)
Orig. (30)	30,30	2,2	23,28	30,30	2,2
Expand. (12)	6,11	0,4	2,4	11,11	1,4

AOBB-K* was run using wMBE-K* with moment matching Liu and Ihler [2011]) for guiding and bounding search. For hard problems, a derivation of wMBE-K*, wMBE+-K*, was tested that avoided consideration of zeros during lower-bounding approximations (motivated by Section 4. For all experiments, wMBE-MMAP was used to upper-bound the partition function of each subunit. According to Ojewole et al. [2018], BBK* uses a dynamic greedy heuristic based on optimistic values for all variables not yet instantiated.

In all problems, each mutable residues considers 21 different amino acid assignments. Conformation variables of non-mutable residues have a domain size of 2-14 rotamers (most having domain sizes 4-9). Conformation variables of mutable residues have a domain size of 34-35 when formulated as F1 and 203-205 when formulated as F2.

We use the following notation: **F**: formulation type, $\boldsymbol{\omega}$: weight applied to the heuristic, $\boldsymbol{\tau}$: underflow-threshold, **H**: heuristic, **iB**: i-bound used, **w***: induced width, **d**: pseudo tree depth, |**R**|: number of MAP variables, |**X**|: total number of variables, $Dmax$: maximum domain size, **UB**: heuristic upper bound at the root (empty cells representing no finite bound), **pre-t**: pre-processing time (ex. compiling heuristics), **search**: search time, **time**: total time, **K***: the returned K* solution (in log_{10}), and **Any-K***: best valid K* value found in 1hr. The best performance points are highlighted.

Formulation 1 vs. Formulation 2: We can see from the aggregated statistics in Table 1that F2 is generally superior to F1, outperforming BBK* more frequently. F1 uses an indexing scheme between amino-acids and their rotamers,

[1] BBK*'s bound tightness parameter does not correlate directly with an ω-approximation. See Ojewole et al. [2018].

Table 2: F1 vs F2 on original and Expanded(*) problems.

		AOBB-K*								
benchmark	F	iB	|R|	|X|	D^{max}	w*	d	UB	pre-t	time
1a0r_00031	F1	3	2	16	34	8	8		0.9	226.9
	F2	3	2	16	203	6	8		0.7	93.0
1gwc_00021	F1	6	2	12	34	6	6	10.28	69.2	101.2
	F2	4	2	12	203	4	6	10.29	6.8	15.7
1gwc_00033	F1	3	2	18	35	9	9		1.0	30.2
	F2	3	2	18	203	7	9		1.3	12.3
2hnu_00026	F1	6	2	14	34	7	7	15.18	14.6	25.0
	F2	4	2	14	203	5	7	15.08	1.7	7.3
2rfe_00012*	F1	3	3	15	34	8	8		2.9	24.5
	F2	4	3	15	205	5	8	14.80	58.3	60.9
2xgy_00020*	F1	6	3	15	35	8	8	12.28	122.0	775.2
	F2	5	3	15	203	5	8	11.39	81.8	360.7
3u7y_00011*	F1	5	3	13	34	7	7		12.9	45.1
	F2	4	3	13	203	4	7	12.29	74.0	76.1
4wwi_00019*	F1	6	3	15	34	8	8	16.93	119.6	1494.9
	F2	5	3	15	203	5	8	16.05	169.2	181.3

which implies that all functions need to include *both* residue and conformation variables thus leading to densely connected graphs compared with F2. A strength of F1 vs. F2 is its smaller domain sizes in the presence of MAP variables which facilitates use of higher i-bounds compared with F2.

AOBB-K*vs BBK* (solution quality): The summary in Table 1show that AOBB-K* and BBK* find the same K* solution for the majority of the original problems. As problems are expanded, AOBB-K* begins to find better solutions more frequently. Concrete examples are displayed in Tables 3 and 4 where greater K* solutions by AOBB-K* are highlighted in the K*MAP columns.

AOBB-K*vs BBK* (running time): The results in Table 3 show that AOBB-K* solved nearly all original benchmark problems faster than BBK* (see the **time** column). As problems were expanded (Table 4), we see AOBB-K*'s time-performance begin to drop more rapidly than BBK*'s; AOBB-K* surpasses BBK* on four problems, but not on the other eight. No hard problems was solved exactly without modifications (presented shortly). However, it is important to note that: (1) AOBB-K* finds solutions greater than that of BBK* (which may account for increased time searching) and (2) AOBB-K* returns intermediate anytime solutions, some of which exceeded the K* value returned by BBK*.

Weighted Search: Each pair of rows in Table 5 compares AOBB-K*(denoted $\omega = 1$) and ω-AOBB-K*($\omega = 0.001$) on all Expanded problems. Moving to approximate search reduces the MAP search space and improves search time (sometimes by more than a factor of ten). With other advances, this was instrumental to solving the hard problems.

Infusing Determinism: We assessed the impact of infusing determinism by applying thresholded-underflow with $\tau = 1 \times 10^{-5}$ and comparing to base problems ($\tau = 0$). Table 6 shows the results obtained on expanded problems. In all cases, we see that the τ-underflows improved search times (see "search" column), sometimes so much so that the best time corresponded to allowing for a more crudely computed heuristic to enter search more quickly. Even when

Table 3: AOBB-K* vs BBK* on Original F2 problems.

	(203 ≤ Dmax ≤ 206)					AOBB-K*		BBK*	
benchmark	iB	w*	|X|	UB	pre-t	time	K*	time	K*
1a0r_00031	3	6	16		0.7	92.99	7.88	109.1	7.88
1gwc_00021	4	4	12	10.29	6.8	15.68	9.79	152.3	9.79
1gwc_00033	3	7	18		1.3	12.33	10.48	512.5	10.48
2hnu_00026	4	5	14	15.08	1.7	7.28	13.18	436.9	13.18
2hnv_00025	4	6	16	15.04	1.7	16.56	13.65	962.1	13.65
2rf9_00007	6	7	18	14.52	4.1	4.67	14.08	45.5	14.08
2rf9_00013	5	6	16	14.12	1.4	1.93	13.25	11.8	13.25
2rf9_00018	6	7	18	16.68	8.4	15.13	15.79	187.2	15.79
2rf9_00042	6	9	22		15.8	148.78	22.65	897.1	22.65
2rfd_00035	6	6	16	17.70	80.1	379.82	17.27	1242.3	16.77
2rfe_00012	4	5	14	14.80	0.8	1.49	13.93	11.1	13.93
2rfe_00014	4	5	14	15.23	0.8	1.72	14.36	31.4	14.36
2rfe_00017	5	5	14	10.96	1.7	7.02	10.52	29.2	10.52
2rfe_00030	4	5	14	11.53	6.9	18.94	10.50	181.5	10.50
2rfe_00041	5	7	18		48.1	401.68	22.73	1181.5	22.73
2rfe_00043	6	6	16	18.48	75.5	79.4	18.04	50.5	18.04
2rfe_00044	6	6	16	18.62	75.2	85.6	18.19	74.5	18.19
2rfe_00047	3	7	18		0.7	88.27	22.70	348.9	22.70
2rfe_00048	4	8	20		2.8	158.65	22.81	387.7	22.81
2rl0_00008	4	3	10	11.16	2.6	2.62	11.16	261.9	9.46
2xgy_00020	4	5	14	11.47	1.9	16.36	10.60	887.5	10.60
3cal_00032	6	6	16	13.38	59.5	125.26	11.62	1428.9	11.62
3ma2_00016	4	5	14	13.39	0.7	3.51	8.38	9.6	8.38
3u7y_00009	6	4	12	4.51	5.7	5.72	4.51	190.8	4.51
3u7y_00011	3	4	12	12.72	0.5	1.95	11.85	27.8	11.85
4hem_00027	3	6	16		1.0	2.01	15.48	39.9	15.48
4hem_00028	3	6	16		0.9	1.84	15.27	34.5	15.27
4kt6_00023	4	6	16	14.80	2.1	7.30	12.69	136.5	12.69
4kt6_00024	4	6	16	14.87	2.1	5.07	12.93	120.7	12.93
4wwi_00019	5	5	14	15.43	5.6	7.85	14.99	26.3	14.99

Table 4: AOBB-K* vs BBK* on Expanded F2 problems.

	(203 ≤ Dmax ≤ 206)					AOBB-K*		BBK*	
benchmark	iB	w*	|X|	UB	pre-t	time	K*	time	K*
1gwc_00021*	4	4	13	12.51	123.8	205.1	11.92	551.3	11.72
2hnv_00025*	4	6	17	18.38	109.8	153.8	16.18	880.5	13.65
2rf9_00007*	-	-	-	-	-	-	-	369.4	14.73
2rf9_00013*	4	6	17	16.36	83.0	100.8	15.03	39.2	15.03
2rfe_00012*	4	5	15	14.80	58.3	60.9	13.93	11.8	13.93
2rfe_00014*	4	5	15	15.23	58.2	60.6	14.36	44.9	14.36
2rfe_00017*	5	5	15	11.46	166.8	334.6	10.86	78.0	10.80
2rfe_00030*	4	5	15	13.61	115.2	276.6	11.12	275.4	10.97
2xgy_00020*	5	5	15	11.39	81.8	360.7	10.96	1388.1	10.96
3u7y_00009*	4	4	13	4.95	62.6	99.5	4.51	215.8	4.51
3u7y_00011*	4	4	13	12.29	74.0	76.1	11.85	26.6	11.85
4wwi_00019*	5	5	15	16.05	169.2	181.3	14.99	34.0	14.99

the same iB is used, interestingly, the speedup due to underflows does not always correspond to a reduced search of the MAP space. In fact, the heuristic can be adversely affected by underflows (as described in Section 4) resulting in less efficient pruning of the MAP space. However, this can potentially be remedied, which we explore next.

wMBE$^+$-K*: Bounding a ratio of functions such as K* is especially difficult - particularly because it relies on lower-bounding, which is especially problematic in the presence of determinism. To empirically test the intuition presented in Section 4, and also in the attempt to solve problems that could not be solved by vanilla AOBB-K*, we used wMBE$^+$-K* (which avoids considering zeros when lower bounding) with ω-AOBB-K* and τ-underflows ($\omega = 0.001, \tau = 1 \times 10^{-5}$, respectively) on hard F1 problems, for which wMBE-

Table 5: AOBB-K* ($\omega = 1$) vs. Weighted ω-AOBB-K* ($\omega = 0.001$ on Expanded F2 problems.

				(203 ≤ Dmax ≤ 206)					ω-AOBB-K*	
benchmark	ω	iB	w*	d	|X|	UB	pre-t	search	time	K*
1gwc_00021*	1	4	4	7	13	28.80	123.8	81.3	205.1	11.92
	0.001	4	4	7	13	28.80	124.3	12.1	136.4	
2hnv_00025*	1	4	6	9	17	42.32	109.8	44.0	153.8	16.18
	0.001	4	6	9	17	42.32	109.3	12.2	121.5	
2rf9_00013*	1	4	6	9	17	37.68	83.0	17.8	100.8	15.03
	0.001	4	6	9	17	37.68	82.8	1.6	84.4	
2rfe_00012*	1	4	5	8	15	34.07	58.3	2.6	60.9	13.93
	0.001	4	5	8	15	34.07	58.6	0.3	58.9	
2rfe_00014*	1	4	5	8	15	35.07	58.2	2.4	60.6	14.36
	0.001	4	5	8	15	35.07	58.2	1.3	59.4	
2rfe_00017*	1	5	5	8	15	26.39	166.8	167.8	334.6	10.86
	0.001	4	5	8	15	27.39	89.1	5.0	94.1	
2rfe_00030*	1	4	5	8	15	31.34	115.2	161.5	276.6	11.1
	0.001	4	5	8	15	31.34	101.4	0.5	101.9	10.9
2xgy_00020*	1	5	5	8	15	26.24	81.8	278.9	360.7	10.96
	0.001	4	5	8	15	27.24	60.4	8.2	68.6	
3u7y_00009*	1	4	4	7	13	11.41	62.6	36.8	99.5	4.51
	0.001	4	4	7	13	11.41	62.5	2.1	64.7	
3u7y_00011*	1	4	4	7	13	28.29	74.0	2.1	76.1	11.85
	0.001	4	4	7	13	28.29	83.4	0.1	83.5	
4wwi_00019*	1	5	5	8	15	36.96	169.2	12.1	181.3	14.99
	0.001	4	5	8	15	37.96	62.0	7.2	69.2	

K* was unbounded (Table 7). In every case, the modified heuristic was able to provide a bounded estimate, in four of the six problems enabling AOBB-K* to find a solution better than BBK*, and terminating, finding an ω-optimal solution, in two of the problems.

Table 6: Impact of τ-underflows on Expanded F2 problems.

				(203 ≤ Dmax ≤ 206)					ω-AOBB-K*	
benchmark	τ	iB	w*	d	|X|	UB	pre-t	search	time	K*
1gwc_00021*	0	4	4	7	13	28.80	123.8	81.3	205.1	11.92
	1E-05	4	4	7	13	28.80	117.1	3.5	120.7	
2hnv_00025*	0	4	6	9	17	42.32	109.8	44.0	153.8	16.18
	1E-05	4	6	9	17	42.32	100.8	1.7	102.4	
2rf9_00013*	0	4	6	9	17	37.68	83.0	17.8	100.8	15.03
	1E-05	4	6	9	17	37.68	71.4	0.4	71.8	
2rfe_00012*	0	4	5	8	15	34.07	58.3	2.6	60.9	13.93
	1E-05	3	5	8	15		5.5	14.9	20.3	
2rfe_00014*	0	4	5	8	15	35.07	58.2	2.4	60.6	14.36
	1E-05	3	5	8	15		4.9	15.2	20.0	
2rfe_00017*	0	5	5	8	15	26.39	166.8	167.8	334.6	10.86
	1E-05	4	5	8	15	27.39	85.7	7.1	92.8	
2rfe_00030*	0	4	5	8	15	31.34	115.2	161.5	276.6	11.12
	1E-05	4	5	8	15	30.43	105.4	3.5	108.9	
2xgy_00020*	0	5	5	8	15	26.24	81.8	278.9	360.7	10.96
	1E-05	4	5	8	15	27.24	60.1	23.8	83.9	
3u7y_00009*	0	4	4	7	13	11.41	62.6	36.8	99.5	4.51
	1E-05	4	4	7	13		61.3	5.1	66.4	
3u7y_00011*	0	4	4	7	13	28.29	74.0	2.1	76.1	11.85
	1E-05	4	4	7	13	28.29	80.7	0.5	81.2	
4wwi_00019*	0	5	5	8	15	36.96	169.2	12.1	181.3	14.99
	1E-05	4	5	8	15		54.2	8.4	62.7	

In summary, AOBB-K* had strong performance on the original protein problems, especially with formulation F2, outperforming state-of-the-art BBK* both in time and by finding better solutions. AOBB-K* also performed admirably on expanded problems, again finding solutions faster for several problems and finding better solutions. Additional innovations of ω-approximate search, infusing determinism, and omitting zeros from lower bounding enabled the algorithm to find solutions to the hard problems, some of which were better than BBK*'s. Nevertheless, hard problems

Table 7: Observing the effects on wMBE approximations on Hard problems when omitting zeros while computing lower bounds.

						ω-AOBB-K* (ω=0.001)		BBK*	
benchmark (τ = 1e-5)	H	iB	w*	|X|	UB (34 ≤ Dmax ≤ 35)	time	K*	time	K*
1gwc_00021**	wMBE	6	8	14		timeout	11.72	625.4	11.72
	wMBE$^+$	6	8	14	19.92	1511.9	11.92		
2hnv_00025**	wMBE	6	10	18		timeout	11.52	1013.2	13.65
	wMBE$^+$	6	10	18	21.81	2085.6	16.18		
2rf9_00007**	wMBE	8	11	20		timeout	12.02	399.6	14.73
	wMBE$^+$	8	11	20	515.99	timeout	13.41		
2rf9_00013**	wMBE	6	10	18		1686.5	15.03	43.8	15.03
	wMBE$^+$	5	10	18	17.68	62.3	15.03		
2rfe_00017**	wMBE	5	9	16		timeout	10.33	91.2	10.80
	wMBE$^+$	5	9	16	23.00	timeout	10.86		
2rfe_00030**	wMBE	6	9	16		timeout	10.27	248.3	10.97
	wMBE$^+$	6	9	16	515.23	timeout	10.29		

were a challenge for AOBB-K* without modifications. The increase in domain size as MAP variables increase proved problematic memory-wise when compiling the heuristic, restricting to low iB's that produced less helpful bounds. This challenge, and the realization that bounds can be improved by domain partitioning (namely ignoring some domain values), motivates future work into new representations.

7 CONCLUSION

Conclusion. We leave readers with **1.** two distinct graphical model formulations to address the K*MAP query, and analysis showing formulation F2 is generally superior; **2.** a new guiding wMBE-K* heuristic and demonstration of the potential to improve its bounds by domain-partitioning; **3.** AOBB-K*, a depth-first AND/OR branch-and-bound algorithm for optimizing K* (and an accompanying approximate ω-AOBB-K*) that proved competitive against state-of-the-art BBK*; **4.** a scheme to exploit determinism into problems by introducing underflows, with theoretical guarantees; and **5.** extensive empirical analysis on over forty benchmarks providing the above-mentioned analysis. As a part of our results, we note that AOBB-K* finds a K* greater than that of BBK* on several occasions, leaving an open question as to the cause and how it impacts time-performance.

As a foundation. This work can now be tuned to the protein domain by: **(1)** evaluating new CPD's with independencies exploitable by our AND/OR scheme, **(2)** integrating optimizations present in developed protein software such as BBK*'s OSPREY 3.0; and **(3)** considering backbone ensembles. Some research directions illuminated by our results are: **(1)** how to incorporate compact representations able to exploit determinism such as by Mateescu and Dechter [2008], Larkin and Dechter [2003]; **(2)** exploration of new K* heuristics (by incorporating constraints, using alternative frameworks such as Deep Bucket Elimination Razeghi et al. [2021]), or employing dynamic heuristics); and **(3)** adaptation of state-of-the-art mixed-inference schemes to solving K*MAP Marinescu et al. [2019, 2018a,b].

References

D. Allen and A. Darwiche. New advances in inference by recursive conditioning. In *Proceedings of the 19th Conference on uncertainty in Artificial Intelligence (UAI03)*, pages 2–10, 2003.

Bassil I. Dahiyat and Stephen L. Mayo. Protein design automation. *Protein Science*, 5(5):895–903, 1996. doi: https://doi.org/10.1002/pro.5560050511. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.5560050511.

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

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

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

Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. doi: 10.2200/S00529ED1V01Y201308AIM023. URL http://dx.doi.org/10.2200/S00529ED1V01Y201308AIM023.

Rina Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms, second edition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13:1–199, 02 2019. doi: 10.2200/S00893ED2V01Y201901AIM041.

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

Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *J. ACM*, 50(2):107–153, 2003. doi: 10.1145/636865.636866. URL http://doi.acm.org/10.1145/636865.636866.

S. Desarkar, P. Chakrabarti, and S. Ghose. Admissibility of AO* when heuristics overestimate. *Artificial Intelligence*, 34(1):97–113, 1987.

Bruce Donald. *Algorithms in structural molecular biology*. MIT Press, Cambridge, Mass, 2011. ISBN 978-0262015592.

Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, pages 502–518, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Mark Hallen, Jeffrey Martin, Adegoke Ojewole, Jonathan Jou, Anna Lowegard, Marcel Frenkel, Pablo Gainza, Hunter Nisonoff, Aditya Mukund, Siyu Wang, Graham Holt, David Zhou, Elizabeth Dowd, and Bruce Donald. Osprey 3.0: Open-source protein redesign for you, with powerful new features. *Journal of Computational Chemistry*, 39, 10 2018. doi: 10.1002/jcc.25522.

Mark A. Hallen and Bruce R. Donald. Protein design by provable algorithms. *Commun. ACM*, 62(10):76–84, sep 2019. ISSN 0001-0782. doi: 10.1145/3338124. URL https://doi.org/10.1145/3338124.

G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities*. Cambridge Mathematical Library. Cambridge University Press, 1988. ISBN 9781107647398. URL https://books.google.com/books?id=EfvZAQAAQBAJ.

Terrell Hill. *Statistical mechanics : principles and selected applications*. Dover Publications, New York, 1987. ISBN 978-0486653907.

D. Larkin and R. Dechter. Bayesian inference in the presence of determinism. *AI and Statistics(AISTAT03)*, 2003.

Junkyu Lee, Radu Marinescu, Rina Dechter, and Alexander Ihler. From exact to anytime solutions for marginal map. AAAI'16, page 3255–3262. AAAI Press, 2016.

Ryan H. Lilien, Brian W. Stevens, Amy C. Anderson, and Bruce R. Donald. A novel ensemble-based scoring and search algorithm for protein redesign, and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme. In *Proceedings of the Eighth Annual International Conference on Resaerch in Computational Molecular Biology*, RECOMB '04, page 46–57, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581137559. doi: 10.1145/974614.974622. URL https://doi.org/10.1145/974614.974622.

Qiang Liu and Alexander Ihler. Bounding the partition function using Hölder's inequality. In *International Conference on Machine Learning (ICML)*, pages 849–856, New York, NY, USA, June 2011. ACM.

Radu Marinescu, Rina Dechter, and Alexander Ihler. And/or search for marginal map. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 563–572, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

Radu Marinescu, Rina Dechter, and Alexander T Ihler. Stochastic anytime search for bounding marginal map. In *IJCAI*, pages 5074–5081, 2018a.

Radu Marinescu, Junkyu Lee, Rina Dechter, and Alexander Ihler. And/or search for marginal map. *J. Artif. Int. Res.*, 63(1):875–921, sep 2018b. ISSN 1076-9757. doi:

10.1613/jair.1.11265. URL https://doi.org/10.1613/jair.1.11265.

Radu Marinescu, Akihiro Kishimoto, Adi Botea, Rina Dechter, and Alexander Ihler. Anytime recursive best-first search for bounding marginal map. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7924–7932, Jul. 2019. doi: 10.1609/aaai.v33i01.33017924. URL https://ojs.aaai.org/index.php/AAAI/article/view/4792.

Robert Mateescu and Rina Dechter. Mixed deterministic and probabilistic networks. *Annals of mathematics and artificial intelligence*, 54(1):3–51, 2008.

Donald McQuarrie. *Statistical mechanics*. University Science Books, Sausalito, Calif, 2000. ISBN 978-1891389153.

Adegoke Ojewole, Jonathan D. Jou, Vance G. Fowler, and Bruce Randall Donald. *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, 2018. doi: 10.1089/cmb.2017.0267. URL https://doi.org/10.1089/cmb.2017.0267.

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

I. Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4):193–204, 1970.

Yasaman Razeghi, Kalev Kask, Yadong Lu, Pierre Baldi, Sakshi Agarwal, and Rina Dechter. Deep bucket elimination. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4235–4242. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/582. URL https://doi.org/10.24963/ijcai.2021/582. Main Track.

Manon Ruffini, Jelena Vucinic, Simon de Givry, George Katsirelos, Sophie Barbe, and Thomas Schiex. Guaranteed diversity and optimality in cost function network based computational protein design methods. *Algorithms*, 14(6), 2021. ISSN 1999-4893. URL https://www.mdpi.com/1999-4893/14/6/168.

Clement Viricel, Simon de Givry, Thomas Schiex, and Sophie Barbe. Cost function network-based design of protein-protein interactions: predicting changes in binding affinity. *Bioinformatics (Oxford, England)*, 34, 02 2018. doi: 10.1093/bioinformatics/bty092.

Jelena Vucinic, David Simoncini, Manon Ruffini, Sophie Barbe, and Thomas Schiex. Positive multistate protein design. *Bioinformatics (Oxford, England)*, 36, 06 2019. doi: 10.1093/bioinformatics/btz497.

Yichao Zhou, Yuexin Wu, and Jianyang Zeng. Computational protein design using and/or branch-and-bound search. *Journal of computational biology : a journal of computational molecular cell biology*, 23, 05 2016. doi: 10.1089/cmb.2015.0212.

AND/OR Branch-and-Bound for Computational Protein Design Optimizing K*
(SUPPLEMENTAL MATERIALS)

1 NOTATION AND DEFINITIONS

$\mathcal{M}$**:** A CPD graphical model for computing the K*MAP. More formally, $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$ where:

- $\boldsymbol{X}$ is the set of all variables in the model
- $\boldsymbol{D}$ is the set of domains for all respective variables in the model
- $\boldsymbol{F}$ is the set of all functions over the variables in the model

$\boldsymbol{X}$**:**

- the set of all variables of the model
- $\boldsymbol{X} = \boldsymbol{R} \cup \boldsymbol{C}$
  - $\boldsymbol{R}$
    - "residue variables" - the set of variables corresponding to the protein residues
    - the set of variables that will be maximized over (ie. the MAP variables)
  - $\boldsymbol{C}$
    - "conformation variables" - each $C \in \boldsymbol{C}$ indexes the rotamer conformation of a particular $R \in \boldsymbol{R}$
    - the set of variables that will be summed over (ie. the SUM variables)
    - there can be several $C \in \boldsymbol{C}$ that correspond to the same $R \in \boldsymbol{R}$. These different C's capture the rotamer conformations of their particular R when the protein is in different structural states...
    - $\boldsymbol{C} = \cup_{\gamma \in \varphi} \boldsymbol{C_\gamma}$
      - φ represents the set of all different substructures the protein's subunits can exist as
      - $\varphi = B \cup U$
        - B is the set of substructures corresponding to bound (ie. complexed) subunits corresponding to the numerator of the K^* ratio
        - U is the set of substructures corresponding to the unbound (ie. dissociate) subunits corresponding to the denominator of the K^* ratio
      - thus $\boldsymbol{C_\gamma}$ is the set of conformation variables corresponding to residues of substructure $\gamma \in \varphi$
        - $C_{\gamma(i)}$ is the conformation variable for residue i when residue i's subunit is in substructure γ
- $\boldsymbol{X_\gamma} = \boldsymbol{R} \cup \boldsymbol{C_\gamma}$

$\boldsymbol{D}$**:**

- the set of domains for each variable in $\boldsymbol{X}$
- $\boldsymbol{D_\gamma}$ is the set of domains for the variables in $\boldsymbol{X_\gamma}$

- for all $R_i \in \boldsymbol{R}$, the respective $D_i = \{$ ALA, VAL, LEU, ILE, PHE, TYR, TRP, CYS, MET, SER, THR, LYS, ARG, HIP, HIE, HID, ASP, GLU, ASN, GLN, GLY$\}$

- for all $C_{\gamma(i)} \in \boldsymbol{C}$,
  - Formulation 1
    - $D_{C_{\gamma(i)}} = \{1, 2, ..., M_i\}$, where M_i is the maximum number of rotamers for any possible amino acid assignment to R_i in state $\gamma \in \varphi$.
    - the assignment to $C_{\gamma(i)}$ acts as an index to the possible side chain conformations of the amino acid assigned to R_i.
  - Formulation 2
    - $D_{C_{\gamma(i)}} = \{c \mid c$ is a rotamer in substructure γ for one of the possible amino acids of residue $R_i\}$.

$\boldsymbol{F}$**:**

- the set of all functions over the model
- $\boldsymbol{F} = \cup_{\gamma \in \varphi} F_\gamma$
  - Formulation 1
    - $F_\gamma = \boldsymbol{E}^{\boldsymbol{sb}}_{\boldsymbol{\gamma}} \cup \boldsymbol{E}^{\boldsymbol{pw}}_{\boldsymbol{\gamma}}$
      - $\boldsymbol{E}^{\boldsymbol{sb}}_{\boldsymbol{\gamma}}$ are the set of all single bodied energies over the residues and their conformations for substructure γ
      - $\boldsymbol{E}^{\boldsymbol{pw}}_{\boldsymbol{\gamma}}$ are the set of all pair-wise energies for all pairs of residues and their conformations that interact in substructure γ
  - Formulation 2
    - $F_\gamma = \boldsymbol{E}^{\boldsymbol{sb}}_{\boldsymbol{\gamma}} \cup \boldsymbol{E}^{\boldsymbol{pw}}_{\boldsymbol{\gamma}} \cup \mathscr{C}_{\boldsymbol{\gamma}}$
      - $\mathscr{C}_{\boldsymbol{\gamma}}$ are constraints ensuring that the assigned rotamer to $C_{\gamma(i)}$ belongs to the amino acid assigned to R_i.

$\mathcal{M}_\gamma$**:** The CPD graphical model $\mathcal{M}$ modified to include only components corresponding to substructure γ. Namely, $\mathcal{M}_\gamma = \langle \boldsymbol{X_\gamma}, \boldsymbol{D_\gamma}, \boldsymbol{F_\gamma} \rangle$

$\mathcal{T}$**:** Pseudo tree for $\mathcal{M}$ constrained for K*MAP computation and providing decomposition of the various substructures $\gamma \in \varphi$.

$\mathcal{T}_\gamma$**:** Pseudo tree for $\mathcal{M}_\gamma$ based on a modified $\mathcal{T}$ such that nodes corresponding to $C_{\gamma'}$, where $\gamma' \neq \gamma$, have been removed.

T**:** Full AND/OR search tree based on $\mathcal{T}$.

T_γ**:** Full AND/OR search tree of $\mathcal{M}_\gamma$ based on $\mathcal{T}_\gamma$.

π**:** Currently expanded path in T.

π_n**:** Path from the root to n in T.

$tip(\pi)$**:** The last node in π that was expanded to.

$OR^R, OR^C, etc.$**:** The set of OR nodes whose corresponding variables belong to the variable set denoted by the superscript. (Absence of superscript corresponds to all OR nodes).

$AND^R, AND^C, etc.$**:** The set of AND nodes of T whose corresponding variables belong to the variable set denoted by the superscript. (Absence of superscript corresponds to all AND nodes).

$LEAF_T$**:** The set of AND nodes that are leaves in T.

n_X**:** Search tree node n corresponding to a particular variable $X \in \boldsymbol{X}$.

$ch_T(n)$**:** Children nodes of n in the search tree indicated by its subscript.

$ch_\pi(n)$**:** Child of n along the π.

$ch_T^{unexp}(n)$**:** Children nodes of n in the search tree indicated by its subscript that have yet to be expanded to (ie. explored) by the algorithm.

$ch_T^{solved}(n)$**:** Children nodes of n in the search tree indicated by its subscript who have been returned to after exploration of all of their children and provably with $lb_{K^*}(n) = ub_{K^*}(n) = v^*(n)$, $v^*(n)$ being the exact K$*$MAP value for the subproblem rooted at n (described futher below).

$ch_T^{unsolved}(n)$**:** Children nodes of n in the search tree indicated by its subscript whose $lb_{K^*}(n)$ and $ub_{K^*}(n)$ values are not yet known to be exact because not all their children have been expanded to and returned from. $ch_T^{unsolved}(n) = ch_T(n) \setminus ch_T^{solved}(n)$.

$anc^{OR}(n), anc^{AND}(n), etc.$**:** Ordered ancestors of n in T, from most recent to eldest, that also belong to the set described by the superscript.

$c(n)$**:** Edge cost into n in T.

$g(n)$**:** Path cost from the root into n in T. Namely, $c(n) \cdot \prod_{m \in anc^{AND}(n)} c(m)$.

$v^*(n_X)$**:** For $X \in \boldsymbol{R}$, $v^*(n_X)$ is the K*MAP value (ie. the optimal K* value) for the problem rooted at n. For $X \in \boldsymbol{C}$, $v^*(n_X)$ is the partition function value of the problem rooted at n. Namely,

$$v^*(n) = \begin{cases} \max_{m \in ch_T(n)} v^*(m), & n \in OR^{\boldsymbol{R}} \\ \frac{(\prod_{m \in ch_T^R(n)} v^*(m))(\prod_{\gamma \in B} \prod_{m' \in ch_T^{C_\gamma}(n)} v^*(m'))}{(\prod_{\gamma \in U} \prod_{m" \in ch_T^{C_\gamma}(n)} v^*(m"))}, & n \in AND^{\boldsymbol{R}} \\ \sum_{m \in ch_T(n)} v^*(m) \cdot c(m), & n \in OR^{\boldsymbol{C}} \\ \prod_{m \in ch_T(n)} v^*(m), & n \in AND^{\boldsymbol{C}} \setminus LEAF_T \\ 1, & n \in LEAF_T \end{cases}$$

$v(n)$**:** A progressively accumulated quantity based on processing of fully solved and returned children fo a node n. $v(n)$ converges to the exact K$*$MAP value of the subproblem rooted at n once all of its children have been expanded to, solved, and returned from. Namely, if $ch_T^{solved}(n) = ch_T(n)$, then $v(n) = v^*(n)$. Formally,

$$v(n) = \begin{cases} \max_{m \in ch_T^{solved}(n)} v(m), & n \in OR^{\boldsymbol{R}} \\ \frac{(\prod_{m \in ch_T^{solved,R}(n)} v(m))(\prod_{\gamma \in B} \prod_{m' \in ch_T^{solved,C_\gamma}(n)} v(m'))}{(\prod_{\gamma \in U} \prod_{m" \in ch_T^{solved,C_\gamma}(n)} v(m"))}, & n \in AND^{\boldsymbol{R}} \\ \sum_{m \in ch_T^{solved}(n)} v(m) \cdot c(m), & n \in OR^{\boldsymbol{C}} \\ \prod_{m \in ch_T^{solved}(n)} v(m), & n \in AND^{\boldsymbol{C}} \setminus LEAF_T \\ 1, & n \in LEAF_T \end{cases}$$

$\mu_\gamma^*(n_X)$**:** For $X \in \boldsymbol{R}$, $\mu_\gamma^*(n_X)$ is the MMAP value for the problem rooted at n in T_γ. For $X \in \boldsymbol{C_\gamma}$, $v^*(n_X)$ is the partition function value of the problem rooted at n in T_γ.

$$\mu_\gamma^*(n) = \begin{cases} \max_{m \in ch_{T_\gamma}(n)} \mu_\gamma^*(m), & n \in OR^{\boldsymbol{R}} \\ \sum_{m \in ch_{T_\gamma}(n)} \mu_\gamma^*(m) \cdot c(m), & n \in OR^{\boldsymbol{C}} \\ \prod_{m \in ch_{T_\gamma}(n)} \mu_\gamma^*(m), & n \in AND \setminus LEAF_T \\ 1, & n \in LEAF_T \end{cases}$$

$\mu_\gamma(n)$**:** A progressively accumulated quantity based on the fully solved and returned children of node n in T_γ. $\mu_\gamma(n)$ converges to the exact MMAP value of the γ-subproblem rooted at n once all of its children in T_γ have been expanded to, solved, and returned from. Namely, if $ch^{solved}_{T_\gamma}(n) = ch_{T_\gamma}(n)$, then $\mu_\gamma(n) = \mu^*_\gamma(n)$. Formally,

$$\mu_\gamma(n) = \begin{cases} \max_{m \in ch_{T^{solved}_\gamma}(n)} \mu_\gamma(m), & n \in OR^{\boldsymbol{R}} \\ \sum_{m \in ch_{T^{solved}_\gamma}(n)} \mu_\gamma(m) \cdot c(m), & n \in OR^{\boldsymbol{C}} \\ \prod_{m \in ch_{T^{solved}_\gamma}(n)} \mu_\gamma(m), & n \in AND \setminus LEAF_T \\ 1, & n \in LEAF_T \end{cases}$$

$h_{K^*}(n)$**:** Precompiled WMBE-K*MAP heuristic for the problem rooted at n.

$h_{Z_\gamma}(n)$**:** Precompiled WMBE-MMAP heuristic for the problem rooted at n considering only $X \in \boldsymbol{X_\gamma} = \boldsymbol{R} \cup \boldsymbol{C_\gamma}$.

$ub_{K^*}(n)$**:** Progressively updated upper bound heuristic of the K*MAP problem rooted at n. Formally,

$$ub_{K^*}(n) = \begin{cases} max(v(n), max_{m \in ch^{unsolved}_T(n)} h_{K^*}(m)), & n \in OR^{\boldsymbol{R}} \\ v(n) \cdot \frac{(\prod_{m' \in ch^{unsolved,R}_T(n)} h_{K^*}(m'))(\prod_{\gamma \in B} \prod_{m" \in ch^{unsolved,C_\gamma}_T(n)} h_{Z_\gamma}(m"))}{(\prod_{\gamma \in U} \prod_{m'" \in ch^{unsolved,C_\gamma}_T(n)} h_{Z_\gamma}(m'"))}, & n \in AND^{\boldsymbol{R}} \\ v(n) + \sum_{m \in ch^{unsolved}_T(n)} h_{K^*}(m) \cdot c(m), & n \in OR^{\boldsymbol{C}} \\ v(n) \cdot \prod_{m \in ch^{unsolved}_T(n)} h_{K^*}(m), & n \in AND^{\boldsymbol{C}} \setminus LEAF_T \\ 1, & n \in LEAF_T \end{cases}$$

$ub_{K^*}(n, \pi)$**:** Progressively updated upper bound heuristic of the K*MAP problem rooted at n consistent with the partial search tree π. Formally,

$$ub_{K^*}(n, \pi) = \begin{cases} ub_{K^*}(n), & n \in OR^{\boldsymbol{R}} \cap tip(\pi) \\ ub_{K^*}(ch_\pi(n), \pi), & n \in OR^{\boldsymbol{R}} \cap \pi \setminus tip(\pi) \\ ub_{K^*}(n) \cdot \prod_{m \in ch_\pi(n)} \frac{ub_{K^*}(m,\pi)}{h_{K^*}(m)} & n \in AND^{\boldsymbol{R}} \cap \pi \end{cases}$$

$ub_{Z_\gamma}(n)$**:** Progressively updated upper bound of the MMAP problem for substructure γ rooted at n. Formally,

$$ub_{Z_\gamma}(n) = \begin{cases} max(\mu_\gamma(n), max_{m \in ch^{unsolved}_{T_\gamma}(n)} h_{Z_\gamma}(m)), & n \in OR^{\boldsymbol{R}} \\ \mu_\gamma(n) + \sum_{m \in ch^{unsolved}_{T_\gamma}(n)} c(m) \cdot h_{Z_\gamma}(m)), & n \in OR^{\boldsymbol{C_\gamma}} \\ \mu_\gamma(n) \cdot \prod_{ch^{unsolved}_{T_\gamma}(n)} h_{Z_\gamma}(m), & n \in AND \end{cases}$$

$MMAP_\gamma(n)$**:** The marginal map value of subunit γ conditioned on residue assignments consistent with the path from the root to n. Formally,

$$MMAP_\gamma(n) = \begin{cases} \mu^*_\gamma(n) \cdot \prod_{m \in anc^{AND}(n)} \prod_{m' \in ch_{T_\gamma(m)} \setminus \pi_n} \mu^*_\gamma(m'), & n \in \boldsymbol{R} \\ \mu^*_\gamma(n) \cdot \prod_{m \in anc^{AND^{\boldsymbol{R}}}(n)} \prod_{m' \in ch_{T_\gamma(m)} \setminus \pi_n} \mu^*_\gamma(m'), & n \in \boldsymbol{C_\gamma} \end{cases}$$

$A^{\times *}_{Z_\gamma}(n)$**:** Called the multiplicative ancestral branching mass, $A^{\times *}_{Z_\gamma}(n)$ captures the portion of $MMAP_\gamma(n)$ due to OR branchings off of n's AND ancestors. The product of $A^{\times *}_{Z_\gamma}$ and the partition function of the subtree consisting of π_n and the subtree of T_γ rooted at n is the contribution to Z_γ from all full configurations consistent with π_n. Formally,

$$A^{\times *}_{Z_\gamma}(n) = \prod_{m \in anc^{AND}(n)} \prod_{m' \in ch_{T_\gamma(m)} \setminus \pi_n} v^*_{Z_\gamma}(m')$$

$A^{\times}_{Z_\gamma}(n)$**:** Upper bound of $A^{\times *}_{Z_\gamma}$ for node n.

$$A^{\times}_{Z_\gamma}(n) = \prod_{m \in anc^{AND}(n)} v_{Z_\gamma}(m) \cdot \prod_{m' \in ch^{unexp}_{T_\gamma}(m)} h_{Z_\gamma}(m')$$

$S^*_{Z\gamma}(n)$**:** The contribution to Z_γ from all configurations not consistent with π_n.

$$S^*_{Z\gamma}(n) = \sum_{m \in anc^{OR}(n)} R^*_{Z\gamma}(m) \cdot g(m) \cdot \sum_{m' \in ch_{T_\gamma(m)} \setminus \pi_n} c(m') \cdot \mu^*_\gamma(m)$$

$S_{Z\gamma}(n)$**:** Upper bound of $S^*_{Z_\gamma}$.

$$S_{Z\gamma}(n) = \sum_{m \in anc^{OR}(n)} A^\times_{Z_\gamma}(m) \cdot g(m) \cdot (\mu_\gamma(m) + \sum_{m' \in ch^{unexp}_{T_\gamma}(m)} c(m') \cdot h_{Z_\gamma}(m'))$$

$UB_{Z_\gamma}(n)$**:** Progressively updated upper bound heuristic on the *entire* partition function of the substructure $\gamma \in \Phi$. Formally,

$$UB_{Z_\gamma}(n) = \Big\{ A^\times_{Z_\gamma}(n) \cdot g(n) \cdot ub_{Z_\gamma}(n) + S_{Z\gamma}(n), \quad n \in tip(\pi)$$

2 AOBB-K*MAP ALGORITHM DETAILS

Algorithm 1: AOBB-K*MAP

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

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

```
begin
    Initialize MiniSat with constraints from M                    // MiniSat initialization
    and generate literals via constraint propagation
    π ← dummy AND node n_D                                        // initialize DFS to start from dummy root node, n_D
    ub_K*(n_D) ← ∏_{m∈ch_T(n_D)} h_K*(m)                          // initialize n_D with global UB on K*
    lb_K*(n_D) ← −inf                                             // no solution yet found as a lower bound
    g(n_D) ← 1
    foreach γ ∈ φ do                                              // initialize n_D with subunit-specific UB values
        A^×_{Z_γ}(n_D) ← 1
        A^+_{Z_γ}(n_D) ← 0
        ub_{Z_γ}(n_D) ← ∏_{m∈ch_{T_γ}(n_D)} h_{Z_γ}(m)            // initialize ub_{Z_γ}(n_D) to the MMAP_γ global UB value
    end
    while n_X ← EXPAND(π) do                                      // DFS Branch-and-Bound
        if MiniSat(π) = false then                                // Constraint-Propagation Pruning (CPP)
            PRUNE(π)
        else if ∃γ ∈ φ s.t. UB_{Z_γ}(n_X) < threshold(γ) then     // Subunit-Stability Pruning (SSP)
            PRUNE(π)
        else if X ∈ R then
            if ∃a ∈ anc^OR(n) s.t. ub_K*(a, π) < lb_K*(a) then    // K*MAP Upper-Bound Pruning (UBP)
                PRUNE(π)
            end
        else if ch^unexp_T(n) = ∅ then                            // DFS Backtracking Step
            BACKTRACK(π)
    end
    return ub_K*(n_D) = lb_K*(n_D) = K*MAP(M)
end
```

Algorithm 2: AOBB-K*MAP subroutine, EXPAND

input : partial search tree π
output : newly expanded node n_X of π

begin
  if $\pi = \emptyset$ **then** // signals end of DFS search
    return $null$
  else
    $n_W \leftarrow tip(\pi)$ // n_W is the node to be expanded
    $n_X \leftarrow$ next unexplored child of n_W // n_X is the next node in the DFS
    $ch_T^{unexp}(n_W) \leftarrow ch_T^{unexp}(n_W) \setminus n_X$
    if $n_X \in OR$ **then**
      foreach $\gamma \in \varphi$ *associated with* X **do**
        $A^{\times}_{Z_\gamma}(n_X) \leftarrow A^{\times}_{Z_\gamma}(n_W) \cdot ub_{Z_\gamma}(n_W)/h_{Z_\gamma}(n_X)$ // update R_{Z_γ} to include siblings of n_X
        $A^{+}_{Z_\gamma}(n_X) \leftarrow A^{+}_{Z_\gamma}(n_W)$ // no new additive ancestral branching
      end
      if $X \in \boldsymbol{R}$ **then** // for OR MAP nodes...
        $ub_{K^*}(n_X) \leftarrow \max_{m \in ch_T(n_X)} h_{K^*}(m)$
        foreach $\gamma \in \varphi$ *associated with* X **do**
          $ub_{Z_\gamma}(n_X) \leftarrow \max_{m \in ch_{T_\gamma}(n_X)} h_{Z_\gamma}(m)$
        end
      else if $X \in \boldsymbol{C_\gamma}$ **then** // for OR SUM nodes of subunit γ...
        $ub_{K^*}(n_X) \leftarrow \sum_{m \in ch_T(n_X)} c(m) \cdot h_{K^*}(m)$
        $ub_{Z_\gamma}(n_X) \leftarrow \sum_{m \in ch_T(n_X)} c(m) \cdot h_{Z_\gamma}(m)$
      end
    else if $n_X \in AND$ **then** // for SUM or MAP AND nodes...
      foreach $\gamma \in \varphi$ **do**
        $A^{\times}_{Z_\gamma}(n_X) \leftarrow A^{\times}_{Z_\gamma}(n_W)$
        $ub_{Z_\gamma}(n_X) \leftarrow \prod_{m \in ch_{T_\gamma}(n_X)} h_{Z_\gamma}(m)$
      end
      if $X \in \boldsymbol{R}$ **then**
        $ub_{K^*}(n_X) \leftarrow \frac{(\prod_{m \in ch_T^R(n)} h_{K^*}(m))(\prod_{\gamma \in B} \prod_{m' \in ch_T^{C_\gamma}(n)} h_{Z_\gamma}(m'))}{(\prod_{\gamma \in U} \prod_{m" \in ch_T^{C_\gamma}(n)} h_{Z_\gamma}(m"))}$
        // $ub_{K^*}(n_X)$ initialized by combining UB heuristic values of its
        // children, dividing values corresponding to dissociate subunits
        foreach $\gamma \in \varphi$ *associated with* X **do**
          $A^{+}_{Z_\gamma}(n_X) \leftarrow A^{+}_{Z_\gamma}(n_W)$ // no new additive ancestral branching
        end
      else if $X \in \boldsymbol{C_\gamma}$ **then**
        $A^{+}_{Z_\gamma}(n_X) \leftarrow A^{+}_{Z_\gamma}(n_W) + A^{\times}_{Z_\gamma}(n_W) \cdot g(n_W) \cdot (ub_{Z_\gamma}(n_W) - c(n_X) \cdot h_{Z_\gamma}(n_X))$
        // update $A^{+}_{Z_\gamma}$ to include siblings of n_X
      end
    end
    $ch_T^{unexp}(n_X) \leftarrow ch_T(n_X)$
    $\pi \leftarrow \pi \cup n_X$
    return n_X;
  end
end

Algorithm 3: AOBB-K*MAP subroutine, BACKTRACK

input : partial search tree π
output : None

```
begin
    if π = ∅ then                                          // backtracked all the way through root
        return
    else
        n_X ← tip(π)                                       // n_X is the node we're backtracking from
        n_W ← par_T(n_X)                                   // n_W is the node we're backtracking to
        π ← π \ n_X
        if n_W ∈ AND then                                  // backtracking from OR node n_X to AND node n_W
            if X ∈ C_γ s.t. γ ∈ U and W ∈ R then
                ub_K*(n_W) ← ub_K*(n_W) · h_K*(n_X)/ub_K*(n_X)   // tighten ub_K*(n_W) via update of
                                                                 // denominator term n_X contributes to
            else
                ub_K*(n_W) ← ub_K*(n_W)/h_K*(n_X) · ub_K*(n_X)   // tighten ub_K*(n_W) via update of
                                                                 // numerator term n_X contributes to
            end
            foreach γ ∈ φ associated with X do
                ub_Zγ(n_W) ← ub_Zγ(n_W)/h_Zγ(n_X) · ub_Zγ(n_X)   // update upper-bound bound on Z_γ at n_W
            end
        else if n_W ∈ OR then                              // backtracking from AND node n_X to OR node n_W
            if W ∈ R then
                ub_K*(n_W) ← max_{m∈ch_T(n_W)} ub_K*(m)      // tighten ub_K*(n_W) via reevaluation of its children
                foreach γ ∈ φ associated with W do
                    ub_Zγ(n_W) ← max_{ch_T(n_W)} ub_Zγ(m)    // update upper-bound bound on Z_γ at n_W
                end
            else if W ∈ C_γ then
                ub_K*(n_W) ← ub_K*(n_W) − c(n_X) · (h_K*(n_X) − ub_K*(n_X))   // tighten ub_K*(n_W) via update
                                                                              // of ub_K*(n_X) of summed AND child n_X
                ub_Zγ(n_W) ← ub_Zγ(n_W) − c(n_X) · (h_Zγ(n_X) − ub_Zγ(n_X))   // tighten ub_Zγ(n_W) via update
                                                                              // of ub_Zγ(n_X) of summed AND child n_X
            end
        end
        if ch_T^unexp(n_W) = ∅ then                        // Continue Backtracking
            BACKTRACK(π)
        else if ∃γ ∈ φ s.t. UB_Zγ(n_W) < threshold(γ) then // Subunit-Stability Pruning (SSP)
            PRUNE(π)
        else if W ∈ R then
            if ∃a ∈ anc^OR(n_W) s.t. ub_K*(a, π) < lb_K*(a) then   // K*MAP Upper-Bound Pruning (UBP)
                PRUNE(π)
            end
        end
end
```

Algorithm 4: AOBB-K*MAP subroutine, PRUNE

input : partial search tree π
output : None

```
begin
    if π = ∅ then                                          // pruned all the way through root
        return
    else
        n_X ← tip(π)                                       // n_X is the node we're pruning
        n_W ← par_T(n_X)                                   // n_W is the node we're backtracking to
        π ← π \ n_X                                        // explicitly prunes n_X from π
        if n_W ∈ AND then                                  // also prune AND parent n_W which will be missing pruned OR child n_X
            PRUNE(π)
        else if n_W ∈ OR then
            if W ∈ R then
                ub_K*(n_X) ← −inf                          // implicitly marks n_X as having been pruned
                ub_K*(n_W) ← max_{m∈ch_T(n_W)} ub_K*(m)    // recompute ub_K*(n_W) excluding pruned child
                foreach γ ∈ φ associated with W do
                    ub_Zγ(n_X) ← −inf                      // implicitly marks n_X as having been pruned
                    ub_Zγ(n_W) ← max_{ch_T(n_W)} ub_Zγ(m)  // recompute ub_Zγ(n_W) excluding pruned child
                end
            else if W ∈ C_γ then
                PRUNE(π)                                   // invalidity of a portion of the SUM search space implies
                                                           // invalidity of the entire corresponding SUM search space
            end
            if ch_T^unexp(n_W) = ∅ then                    // Continue Backtracking
                BACKTRACK(π)
            else if ∃γ ∈ φ s.t. UB_Zγ(n_W) < threshold(γ) then   // Subunit-Stability Pruning
                PRUNE(π)
            else if W ∈ R then
                if ∃a ∈ anc^OR(n_W) s.t. ub_K*(a, π) < lb_K*(a) then   // K*MAP Pruning
                    PRUNE(π)
                end
            end
        end
    end
end
```

1 INTRODUCTION

Graphical models provide a powerful framework for reasoning about conditional dependency structures over many variables. The Marginal MAP (MMAP) query asks for the optimal configuration of a subset of variables, called 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 is known as K^*. In the context of computational protein design (**CPD**), K^* estimates binding affinity between interacting 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 (MAP) variables and summation (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 [1999, 2019]. This in turn also implies larger search spaces when using search algorithms or larger messages when using message-passing schemes. Even the simpler 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 four main contributions:

1. **Two formulations of K^*MAP as a graphical model**
2. **A Weighted Mini-Bucket Elimination K^*MAP heuristic, wMBE-K^***
3. **Proof-of-concept for Domain-Partitioned Weighted Mini-Bucket Elimination K^*MAP heuristic**
4. **AOBB-K^*, a depth-first branch-and-bound algorithm over AND/OR search spaces for solving CPD formulated as a K^*MAP problem**
5. **A thresholding scheme to exploit determinism accompanied with performance guarantees**

2 BACKGROUND

2.1 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 amino acid positions (or **residues**) of a protein-of-interest are deemed as mutable - these are amino acid positions where different amino acid mutations will be considered - and through a computational process, a preferred sequence is determined.

Typically, 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 for a more tractable problem:

- **Consider a subset of 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 et al. [2016].

2.2 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 defined by

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

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

$$K^{\int} = \frac{Z_{PL}^{\int}}{Z_P^{\int} Z_L^{\int}}, \quad Z_\gamma^{\int} = \int_{\mathcal{C}} e^{-\frac{E_\gamma(c)}{\mathscr{R}T}} dc \tag{3}$$

where $Z_{PL}^{\int}$, $Z_P^{\int}$, and $Z_L^{\int}$ are partition functions of the bound and unbound states that capture the entropic contributions of their various conformations $\mathcal{C}$. ($E_\gamma(c)$ represents the energy of a particular conformation c of state $\gamma \in \varphi$ where $\varphi = \{P, L, PL\}$, $\mathscr{R}$ 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]:

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

Due to the independence between the interaction of residues with each other 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(s) and U represents the unbound (dissociate) states. (For the two-subunit system in our example, $B = \{PL\}$ and $U = \{P\} \cup \{L\}$). This more generalized representation, can be used directly for K* computations involving more than two subunits.

A common goal in protein design is to maximize protein-ligand interaction. Previously, this was done by minimizing an objective called the GMEC (global minimum energy conformation) over only the complexed protein state PL Ruffini et al. [2021], Hallen and Donald [2019], Zhou et al. [2016]. The GMEC, being a pure minimum over the energies of the complex's conformations, ignores the realization that protein structures are dynamic. Furthermore, by the GMEC focusing only on the protein's complexed state, it ignores the dynamicity of the subunit interactions. However, since minimizing the GMEC results in a pure optimization task - a task much easier than that of mixed inference, many solvers use this objective. On the other hand, the stronger K* objective captures both the dynamicity of protein conformations and subunit interactions. K*MAP is the formalization of computational protein design as a task to maximize K*,

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

where we look for amino acid assignments $\boldsymbol{R}=r$ that maximize K*. Thus, the goal of recent work and the work presented here is to develop efficient algorithms for computing K*MAP, from which one can predict a small set of promising sequences to experiment on in vivo, saving great time and cost. Our work taps into recent algorithms developed for the marginal map task defined over graphical models.

2.3 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 $Pr(X) \propto \prod_\alpha f_\alpha(X_\alpha)$.

2.4 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].

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 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].

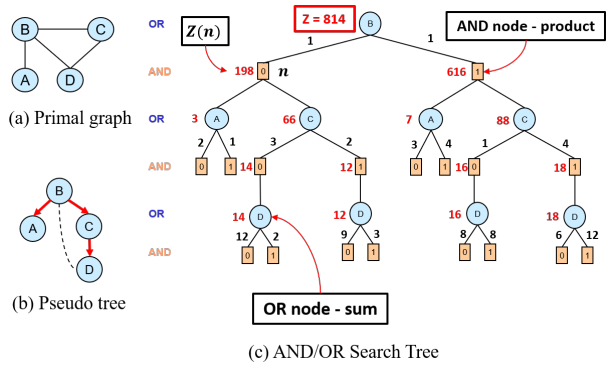


Figure 1: A full AND/OR tree representing all 16 solutions.

2.5 BUCKET ELIMINATION

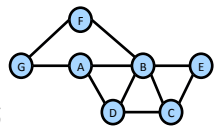
Given a variable ordering d, *Bucket Elimination* Dechter [1999], or BE, is an inference scheme that processes variables one by one with respect to the reverse of d. For any next variable X_p, all the functions in bucket B_p - namely the original functions in the graphical model and any messages passed 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 2** 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 $F = \{f(A), f(A,B), f(A,D), f(A,G), f(B,C), f(B,D), f(B,E), f(B,F), f(C,D), f(C,E), f(F,G)\}$.

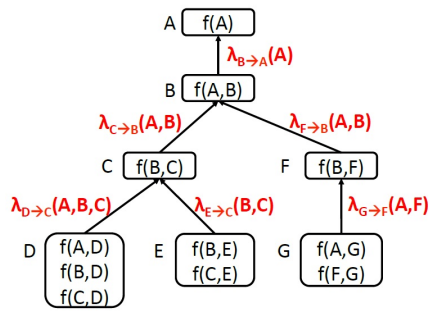
Bucket-Elimination can be viewed as a 1-iteration message-passing algorithm along its *bucket-tree* (bottom-up). The nodes of the tree are the different buckets. Each bucket of a variable contains a set of the model's functions depending on the given order of processing. There is an arc from bucket B_p 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 3**.

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].



0.25
(a) b

Figure 2: A primal graph.



0.4
(a)
b

Figure 3: Bucket elimination example

Figure 4: (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.

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 et al. [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**, $\boldsymbol{R} = \{R_i \mid i \in \{1, 2, ..., N\}\}$, representing the N different residues (ie. positions) of the proteins. Each R_i has corresponding domain $D_{R_i} = \{aa \mid aa$ 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 $\sim$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**, $\boldsymbol{C_\gamma} = \{C_{\gamma(i)} \mid i \in \{1, 2, ..., N\}\}$, each indexing discretized spacial conformations (ie. rotamers) of the amino acid at residue R_i when the protein is in state $\gamma \in \varphi$, where $\varphi = B \cup U$. 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 γ. 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. The $\boldsymbol{C_\gamma}$ are the SUM variables which we marginalize over.

Functions There are two sets of functions in F1.

$E^{sb}_\gamma = \{E^{sb}_{\gamma(i)}(R_i, C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}\}$ 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. 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.

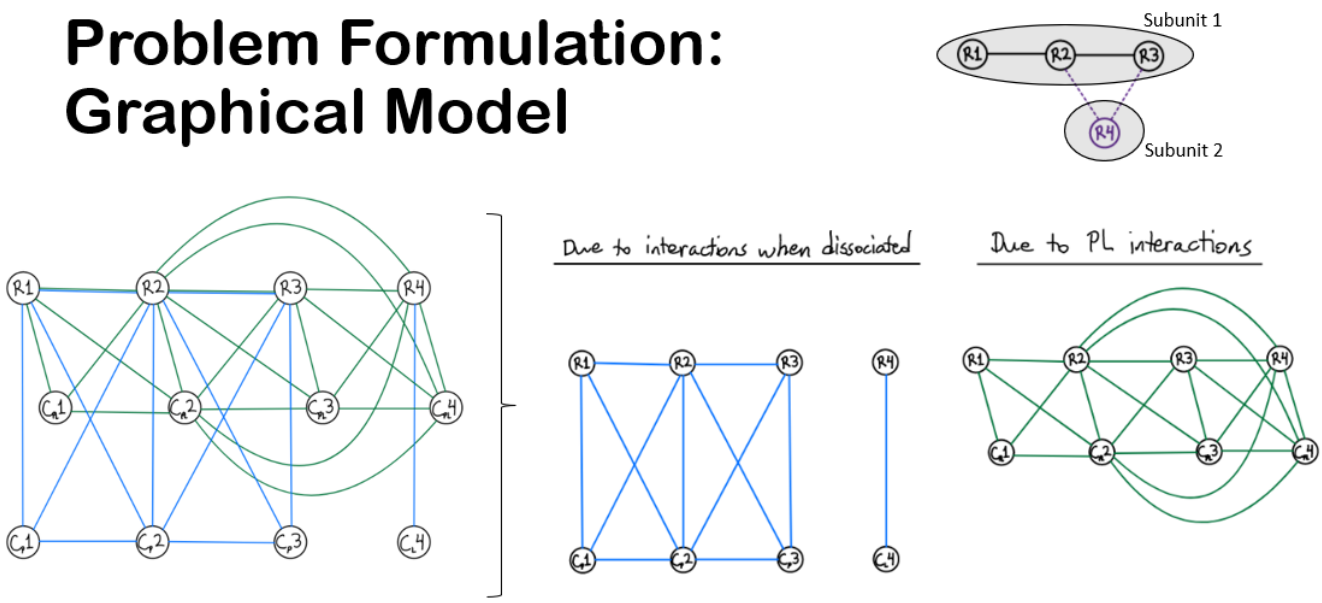
$E^{pw}_\gamma = \{E^{pw}_{\gamma(ij)}(R_i, C_{\gamma(i)}, R_j, C_{\gamma(j)}) \mid$ for i, j s.t. R_i and R_j interact$\}$ is a set of functions that captures the pair-wise energies of interaction between the amino acids of residues that are in close spacial proximity. For any assignment to $C_{\gamma(i)}$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

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

F1's graphical formulation can be seen in Figure 5.

Figure 5: Illustration of F1.



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, $R = \{R_i \mid i \in \{1, 2, ..., N\}\}$, representing the N different residues (ie. positions) of the proteins. Each R_i has corresponding domain $D_{R_i} = \{aa \mid aa$ 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, $\boldsymbol{C_\gamma} = \{C_{\gamma(i)} \mid i \in \{1, 2, ..., N\}\}$, this time each representing the specific amino acid and conformation of the N different residues. Namely, each each $C_{\gamma(i)}$ has corresponding domain $D_{C_{\gamma(i)}} = \{c \mid c$ 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.

$\mathscr{C} = \{\mathscr{C}_{\gamma(i)}(R_i, C_{\gamma(i)}) \mid i \in \{1, 2, ..., N\}, \gamma \in \varphi\}$ 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.

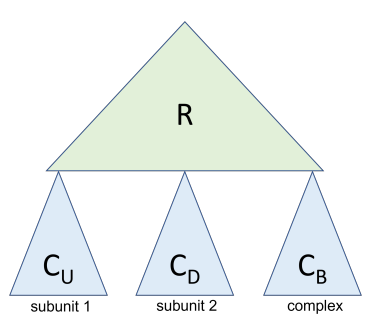
$E^{pw}_{\gamma} = \{E^{pw}_{\gamma(ij)}(C_{\gamma(i)}, C_{\gamma(j)}) \mid$ for i, j s.t. R_i and R_j interact$\}$ is a set of functions that captures the pair-wise energies of interaction between the amino acids of residues that are in close spacial proximity.

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

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

A general schematic for a guiding pseudo tree that results from both F1 and F2 formulations showing the decomposition that results can be seen in Figure 6.

Figure 6: Schematic of resulting pseudo tree for CPD formulated as F1 or F2.



4 WMBE-K*

We present a weighted mini-bucket scheme for K^*, which is an adaptation of the mini-bucket scheme to the new K^* objective. The algorithm, called wMBE-K*, is described in Algorithm 5.

Algorithm 5: wMBE-K*

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 $\boldsymbol{X}$. an i-bound i

output : upper bound on the K*MAP value

1 **begin**
2 Condition each $f \in \boldsymbol{F}$ according to the provided evidence e and remove the corresponding variables from the scopes of the functions.
3 Partition the conditioned fs into buckets $B_n, ..., B_1$ s.t. each function is placed in the greatest bucket corresponding to a variable in its scope.
4 **foreach** $k = n...1$ **do**
5 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 $\leq i$
6 **if** $X_k \in \boldsymbol{MAP}$ **then**
7 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**
8 $\lambda_k^t \leftarrow \max_{X_k} \prod_{f \in MB_k^t} f$
9 **end**
10 **else**
11 **if** $X_k \in \boldsymbol{C_B}$ **then**
12 Select a set of positive weights $\boldsymbol{w} = \{w_1, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$
13 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**
14 $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$
15 **end**
16 **else if** $X_k \in \boldsymbol{C_U}$ **then**
17 Select a negative weight for w_1
18 Select a set of positive weights $\boldsymbol{w} = \{w_2, ..., w_T\}$ s.t. $\sum_{w_t \in \boldsymbol{w}} w_t = 1$
19 **foreach** $MB_k^t \in \boldsymbol{MB_k}$ **do**
20 $\lambda_k^t \leftarrow (\sum_{X_k} \prod_{f \in MB_k^t} f^{w_t})^{1/w_t}$
21 **if** $scope(\lambda_k^t) \cap \boldsymbol{C_U} = \emptyset$ **then**
22 $\lambda_k^t \leftarrow 1/\lambda_k^t$
23 **end**
24 **end**
25 **end**
26 **end**
27 Add each λ_k^t to the bucket of the highest-index variable in its scope.
28 **end**
29 **return** λ_1
30 **end**

wMBE-K* operates similarly to wMBE-MMAP Dechter and Rish [2002], Ping et al. [2015], Marinescu et al. [2014]. 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 et al. [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* can also employ cost shifting to tighten its bounds (see Liu and Ihler [2011]). In our empirical evaluation cost-shifting is implemented as well.

Complexity. Like wMBE-MMAP Dechter and Rish [2002], Ping et al. [2015], Marinescu et al. [2014], wMBE-K* is exponential in the i-bound parameter both in time and memory.

Challenges in the Quality of the Bound. As can be expected, bounding a ratio of functions (as in the case for K*) is particularly challenging, relying on both upper and lower bounds. Lower bounding of functions is particularly challenging. For larger problems and low i-bounds, this can often yield relatively weak bounds. We provide an improvement to help remedy this next.

5 DOMAIN-PARTITIONED MBE

Until now, we have been using a [weighted] mini-bucket heuristic that is blind to explicit hard constraints and any consistency issues. This can be potentially handicapping when lower-bounding given that constraints are represented as zero's in functions and can cause premature deflation of lower bounds. In particular, in mini-bucket elimination Dechter and Rish [2003] where lower bounds are created via minimizing over function values any zeros in the functions being minimized will cause the resulting lower bound itself to drop to zero. However in the CPD domain, where functions represent protein energetics, we can guarantee that the partition function for every subunit is positive (ie. every configuration is "satisfiable"), we can guarantee a positive lower bound by using the following simple remedy.

Given variables X, Y, and Z, and objective

$$obj = \sum_{X} f(x,y) \cdot g(x,z)$$

.

Lets identify a set

$$X' = \{x \in X | g(x,z) \neq 0\}$$

and

$$\epsilon_{X'} = min_{x \in X'} g(x,z)$$

Clearly $\epsilon_{X'} > 0$. Therefore we can derive:

$$obj = \sum_{x \in X'} f(x,y) \cdot g(x,z) + \sum_{x \in X \setminus X'} f(x,y) \cdot g(x,z) =$$

$$\sum_{X \in X'} f(x,y) \cdot g(x,z) >=$$

$$\sum_{X \in X'} f(x,y) \cdot min_{x \in X'} g(x,z) = \epsilon_{X'} \cdot \sum_{X \in X'} f(x,y) > 0.$$

The last quantity is greater than zero unless f(x,z) is identically zero over X'.

6 AOBB-K*

We now present the key algorithmic contribution of this work: **AOBB-K*** (Algorithm 6), an depth-first AND/OR branch-and-bound (**BnB**) 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], depth-first algorithms provide a search methodology linear in space allowing for solving problems unable to be solved by best-first methodologies due to memory Zhou et al. [2016].

At a high level, AOBB-K* adapts AOBB-MMAP Marinescu et al. [2014] for the K*MAP task for CPD by (1) guiding search using a K* upper-bounding heuristic such as wMBE-K*, (2) adjusting computations to be appropriate for computing K* values, (3) incorporating subunit-stability constraints (**SSC**'s) which enforce the partition function of each protein subunit, Z_γ, to be greater than an inputted biologically-relevant threshold S_γ Ojewole et al. [2018], and (4) by using the SAT-solver $MiniSat$ Eén and Sörensson [2004] to identify, and avoid searching, provably invalid configurations such as those with inconsistent amino-acid - rotamer pairs.

AOBB-K* traverses the underlying AND/OR search tree guided by the provided pseudo tree $\mathcal{T}$, expanding nodes in a depth-first manner (line 11), and pruning whenever any of three conditions are triggered: (1) the resulting variable assignments violate a constraint established by $MiniSat$ (constraint-propagation pruning, **CPP**) (line 12), (2) a subunit-stability constraint is violated (subunit-stability pruning, **SSP**) (line 14), or (3) it can be asserted that the current amino acid configuration cannot produce a K* better than any previously found (upper-bound pruning, **UBP**) (line 17). Backtracking occurs when all of the node's children have been explored and returned from (line 20), at which point the K* value of the sub problem the node roots is known exactly and the bounds of its parents are tightened accordingly.

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.

More specifically, the algorithm begins with a two-step initialization. First, constraint literals are generated by $MiniSat$ through applying full constraint propagation to the problem (line 3). Second, the depth-first 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[1]. When a node is first expanded, this value is initialized based on upper-bounding heuristic function $h^{ub}_{K^*}(.)$ (line 5). As search progresses, $ub_{K^*}(n)$ decreases, converging towards the K*MAP of the sub problem rooted at n.

Each node n also maintains a progressively improved upper bound on the partition function of each subunit γ consistent with the path to n, $UB_{Z_\gamma}(n)$[1], (line 9). At each step in the search, $UB_{Z_\gamma}(n)$ is recomputed to ensure that it is greater than the given S_γ, thus satisfying the SSC's) and enforcing consideration of only biologically relevant solutions Ojewole et al. [2018].

Theorem 6.1 (AOBB-K* Correctness and Completeness)**.** *AOBB-K* is sound and complete, returning the optimal K* value of all amino-acid configurations that do not violate the inputted subunit-stability constraints.*

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 early UBP, reducing time greatly in practice).

Exploiting Depth-First Branch-and-Bound Properties. As the core of AOBB-K* is a depth-first BnB algorithm, well known properties can also be exploited such as (1) the ability to extract **anytime**-solutions (ie. intermediate solutions that satisfy the subunit-stability constraints) and (2) the ability to apply bounded approximations.

Although best-first search algorithms, such as those used in current state-of-the-art software OSPREY 3.0 Hallen et al. [2018], can begin exploring potential solutions quickly, they may take a long time completing the exploration of any one solution as search is spread among a breadth of optimistic contenders. On the other hand, AOBB-K* is sure to compute the K* of its most optimistic solutions immediately, yielding K* values for potential designs early on.

Furthermore, the search can easily be relaxed to an ϵ-approximation (for $\epsilon \in [0, 1)$) by multiplying h_{K^*} by a factor of $(1 - \epsilon)$. It can be shown that the resulting solution will be at worst $(1 - \epsilon) \cdot K^*MAP$. We explore the performance of applying such approximations in Section 8.

It is worthy to note that this ϵ-approximation framework can also be extended to the summation space and also to the estimation of Z_γ, though this is beyond the scope of this work and left for future exploration.

[1] Additional details provided in the supplemental materials

Algorithm 6: AOBB-K*

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

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

```
begin
    Initialize MiniSat with constraints from M
    and generate literals via constraint propagation
    π ← dummy AND node n_D
    ub_{K*}(n_D) ← ∏_{m∈ch_T(n_D)} h_{K*}(m)
    lb_{K*}(n_D) ← −inf
    g(n_D) ← 1
    foreach γ ∈ φ do
        UB_{Z_γ}(n_D) ← ∏_{m∈ch_{T_γ}(n_D)} h_{Z_γ}(m)
    end
    while n_X ← EXPAND(π) do
        if MiniSat(π) = false then
            PRUNE(π)
        else if ∃γ ∈ φ s.t. UB_{Z_γ}(n_X) < S_γ then
            PRUNE(π)
        else if X ∈ R then
            if ∃a ∈ anc^{OR}(n) s.t. ub_{K*}(a, π) < lb_{K*}(a) then
                PRUNE(π)
            end
        else if ch_T^{unexp}(n) = ∅ then
            BACKTRACK(π)
    end
    return ub_{K*}(n_D) = lb_{K*}(n_D) = K*MAP(M)
end
```

7 INFUSING DETERMINISM VIA THRESHOLDED UNDERFLOWS

During search in the presence of determinism, constraint propagation (CP) can be a powerful tool by helping to prune invalid configurations. In the context of protein design, such invalid configurations correspond to inconsistent amino acid - rotamer pairs or configurations that will have no contribution to the a subunit's partition function. The more determinism that is present, the more CP can be leveraged to speed up search. Until now, problems were formulated as accurately as possible using 64-bit floating point values. However, some function values can be extremely small, corresponding to very unfavorable side-chain conformations that would not appear in feasible solutions. By underflowing these values, namely replacing them with zero, we can allow CP to exclude configurations containing those tuples and, thus, speed up search.

Definition 7.1 (Thresholded-underflow of a function)**.** *Given a non-negative function f, and a non-negative real number τ, we define the τ-underflow of a function f as*

$$f^{\tau}(x) = \begin{cases} f(x), & f(x) >= \tau \\ 0, & \textit{otherwise} \end{cases}$$

Definition 7.2 (Thresholded-underflow of a problem)**.** *Given a graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, and a non-negative real number τ, we define the τ-underflow of a $\mathcal{M}$ as*

$$\mathcal{M}^{\tau} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}^{\tau} \rangle, \quad \textit{where } \mathbf{F}^{\tau} = \{ f^{\tau} \mid f \in \boldsymbol{F} \}$$

In the next subsections, we describe conditions such that $K^*MAP(\mathcal{M}^{\tau}) = K^*MAP(\mathcal{M})$. In Section 8 we demonstrate the speed up that results empirically. Furthermore, these conditions can be trivially extended to the tasks of MMAP task and computing the partition. We now elaborate.

7.1 VALIDITY OF THRESHOLDED UNDERFLOWS

Definitions.

$tuples_{\gamma}(\tau, r)$: the set of all tuples of assignments [to all variables in $\boldsymbol{C_{\gamma}}$] that are consistent with $\boldsymbol{R} = r$ and that will be affected by a τ-underflow

$f_{\gamma}^{max}(r)$: the maximum value in the functions associated with subunit γ consistent with configuration $\boldsymbol{R} = r$

$|\boldsymbol{f_{\gamma}}|$: denotes the number of functions included in the partition function computation for the protein subunit γ, excluding explicit constraints

$|\boldsymbol{\tau_{\gamma}}(c_{\gamma}, r)|$: the number of functions associated with subunit γ who's function values are less than τ given the assignment $\boldsymbol{C_{\gamma}} = c_{\gamma}$ and $\boldsymbol{R} = r$

$|\boldsymbol{C_{\gamma}}(r)|$: the cardinality of the Cartesian product of assignments to the variables in $\boldsymbol{C_{\gamma}}$ that are consistent with the assignment $\boldsymbol{R} = r$

$|\boldsymbol{C_{\gamma}}|^{\Uparrow}$: the cardinality of the Cartesian product of greatest number of assignments to each variable in $\boldsymbol{C_{\gamma}}$ that are individually consistent with any assignment $\boldsymbol{R} = r'$

Z_B^{min}: the smallest Z_B that can lead to a valid $K^* > K^{*(wt)}$. Formally, $Z_B^{min} = K^{*(wt)} \cdot \prod_{\gamma \in U} S_{\gamma})$[1]

$precison(x)$: the smallest decimal place for which the value of x is recorded

Noting that underflows can only potentially decrease partition function values (and never increase them), intuitively there are two sufficient conditions that, when upheld, allow us to assert $K^*MAP(\mathcal{M}^{\tau}) = K^*MAP(\mathcal{M})$:

1. τ-underflows do not alter subunit partition functions that satisfy SSC's. Namely, $\forall r \in \boldsymbol{R}$ s.t. $Z_{\gamma}^{\mathcal{M}}(r) \geq S_{\gamma},\ Z_{\gamma}^{\mathcal{M}}(r) = Z_{\gamma}^{\mathcal{M}^{\tau}}(r)$
2. τ-underflows do not alter any $Z_B > Z_B^{min}$. More formally, $\forall r \in \boldsymbol{R}$ s.t. $Z_B^{\mathcal{M}}(r) \geq Z_B^{min},\ Z_B^{\mathcal{M}}(r) = Z_B^{\mathcal{M}^{\tau}}(r)$

Verifying τ satisfies condition 1

Theorem 7.1 (τ that will not violate SSC's)**.** *A problem created via τ-underflows such that, for every $\boldsymbol{R} = r$ and $\forall \gamma \in \varphi$,*

$$\sum_{c_\gamma \in tuples_\gamma(\tau)} [\ (f_\gamma^{max}(r))^{(|\boldsymbol{f_\gamma}| - |\boldsymbol{\tau_\gamma}(c_\gamma, r)|}\ ] \cdot [\ (\tau)^{|\boldsymbol{\tau_\gamma}(c_\gamma, r)|}\ ] < 0.5 \cdot precision(S_\gamma)$$

will not violate any subunit stability constraints.

Complexity. Testing the condition in Theorem 7.1 is exponential in the number of variables in $\boldsymbol{R} \cup \boldsymbol{C_B}$.

Corollary 7.2. *A problem created with underflows using an underflow threshold τ such that for every $\boldsymbol{R} = r$ and $\forall \gamma \in \varphi$*

$$\left.\begin{array}{ll} \tau \leq 1, & |\boldsymbol{C_\gamma}(r)| \cdot (f_\gamma^{max}(r))^{|\boldsymbol{f_\gamma}|} \cdot (\tau) \\ \tau > 1, & |\boldsymbol{C_\gamma}(r)| \cdot (f_\gamma^{max}(r))^{|\boldsymbol{f_\gamma}|} \cdot (\tau)^{|\boldsymbol{f_\gamma}|} \end{array}\right\} < 0.5 \cdot precision(S_\gamma)$$

will not violate any subunit stability constraints.

Complexity. Testing the condition in Corollary 7.2 is exponential in the number of variables in $\boldsymbol{R}$.

Corollary 7.3. *A problem created with underflows using an underflow threshold τ and $\forall \gamma \in \varphi$*

$$\left.\begin{array}{ll} \tau \leq 1, & |\boldsymbol{C_\gamma}|^{\Uparrow} \cdot (f_\gamma^{max})^{|\boldsymbol{f_\gamma}|} \cdot (\tau) \\ \tau > 1, & |\boldsymbol{C_\gamma}|^{\Uparrow} \cdot (f_\gamma^{max})^{|\boldsymbol{f_\gamma}|} \cdot (\tau)^{|\boldsymbol{f_\gamma}|} \end{array}\right\} < 0.5 \cdot precision(S_\gamma)$$

will not violate any subunit stability constraints.

Complexity. Testing the condition in Corollary 7.3 is can be done in linear time.

Verifying τ satisfies condition 2

Theorem 7.4 (τ that will not alter K*MAP)**.** *A problem created with underflows using an underflow threshold τ such that τ is known not to violate the SSC's and such that for every $\boldsymbol{R} = r$*

$$\sum_{c_B \in tuples_B(\tau, r)} [\ (f_B^{max}(r))^{|\boldsymbol{f_B}| - |\boldsymbol{\tau_B}(c_B, r)|}\ ] \cdot [\ (\tau)^{|\boldsymbol{\tau_B}(c_B, r)|}\ ] < 0.5 \cdot precision(Z_B^{min})$$

will also not alter the K*MAP solution.

Complexity. Testing the condition in Theorem 7.4 is exponential in the number of variables in $\boldsymbol{R} \cup \boldsymbol{C_B}$.

Corollary 7.5. *A problem created with underflows using an underflow threshold τ such that τ is known not to violate the SSC's and such that for every $\boldsymbol{R} = r$*

$$\left.\begin{array}{ll} \tau \leq 1, & |\boldsymbol{C_B}(r)| \cdot (f_B^{max}(r))^{|\boldsymbol{f_B}|} \cdot (\tau) \\ \tau > 1, & |\boldsymbol{C_B}(r)| \cdot (f_B^{max}(r))^{|\boldsymbol{f_B}|} \cdot (\tau)^{|\boldsymbol{f_B}|} \end{array}\right\} < 0.5 \cdot precision(Z_B^{min})$$

will also not alter the K*MAP solution.

Complexity. Testing the condition in Corollary 7.5 is exponential in the number of variables in $\boldsymbol{R}$.

Corollary 7.6. *A problem created with underflows using an underflow threshold τ such that τ is known not to violate the SSC's and such that*

$$\left.\begin{array}{ll} \tau \leq 1, & |\boldsymbol{C_B}|^{\Uparrow} \cdot (f_B^{max})^{|\boldsymbol{f_B}|} \cdot (\tau) \\ \tau > 1, & |\boldsymbol{C_B}|^{\Uparrow} \cdot (f_B^{max})^{|\boldsymbol{f_B}|} \cdot (\tau)^{|\boldsymbol{f_B}|} \end{array}\right\} < 0.5 \cdot precision(Z_B^{min})$$

*will also not alter the K*MAP solution.*

Complexity. Testing the condition in Corollary 7.6 is can be done in linear time.

Conclusion.

If a chosen τ satisfies Theorem 7.1 and Theorem 7.4 (or any of their corollaries), then it is a valid underflow threshold and AOBB-K* is guaranteed to find the optimal K*MAP given the τ-underflowed problem.

8 EMPIRICAL EVALUATION

8.1 METHODS

Benchmarks. We experimented on a total of 48 protein design benchmark problems, 30 of which encoded two mutable residues (provided by the Donald lab at Duke University and denoted "original") and 12 of which were made harder by expanding to three mutable residues (denoted "expanded"), and six modified to have four mutable residues (denoted "hard"). CPD problems were then generated using OSPREY 3.0 Hallen et al. [2018] to compute interaction energies and then formulated into both F1 and F2 in UAI format for AOBB-K* to be run on. Each UAI problem was also formulated with underflows using a threshold of 1×10^{-5}.

Algorithms. Experiments were run using AOBB-K* (implemented in C++) using a BnB-factor of 1 and 0.001 (corresponding to ϵ 0 and 0.999, respectively). For comparison, problems were also run using BBK* Ojewole et al. [2018], a state-of-the-art best-first search algorithm as part of the comprehensive protein design software package OSPREY 3.0 Hallen et al. [2018] that has been developed for over a decade. BBK* (implemented in Java) was used with rigid side chains and a bound-tightness parameter of 1×10^{-200}. Being native to OSPREY, BBK* used OSPREY's native problem formulation. Experiments were run for a maximum of 1hr on a 2.66 GHz processor with 4 GB of memory with the same subunit-stability threshold as BBK* of $S_\gamma = Z_\gamma^{(wt)} \cdot e^{-\frac{5}{\mathscr{R}T}}$ where $Z_\gamma^{(wt)}$ is the partition function given the wild-type amino acid sequence. As BBK* can take advantage of parallelism, it was also run with access to 4 CPU cores.

Heuristic. AOBB-K* was run using wMBE-K* with moment matching Liu and Ihler [2011]) for guiding and bounding search on all problems. For hard problems, versions of MBE-K* and wMBE-K* were tested that avoided consideration of zeros during lower-bounding approximations (MBE$^+$-K* and wMBE$^+$-K*, respectively). For all experiments, wMBE-MMAP was used to upper-bound the partition function of each subunit. BBK* uses a dynamic greedy heuristic based on the most optimistic values for all variables not yet instantiatedOjewole et al. [2018].

8.2 RESULTS

Data Tables. Table 1 displays aggregated results from experiments across all benchmarks using AOBB-K*[1] (with wMBE-K*, run on F1 and F2) compared to BBK*. $K^* \geq$ counts the number of times the K* value found by AOBB-K* was greater than that of BBK*, $K^*>$ counts the number of times the K* value found by AOBB-K* was *strictly greater* than that of BBK*, and $t^{total}<$ counts the number of times AOBB-K* found its solution faster than BBK*.

Table 2 shows select results[2] on original benchmarks and Table 3 on the expanded benchmarks. F denotes the UAI formulation type, ω is the weight multiplied to the K*MAP heuristic, τ currently $f^<$ is the underflow-threshold used, *iB* is the i-bound used, $w*$ is the induced width due to the generated constrained variable ordering, d is the depth of the resulting pseudo tree, $|X|$ is the total number of variables, *UB* is the wMBE-K* bound (with empty cells representing no finite bound), OR^R and AND^R display the number of each type of each type of node visited in the MAP search space,OR^{all} and AND^{all} display the total number of each type of node visited, *CPP* are the number of nodes pruned due being deemed a deadend by $MiniSat$, *UBP* are nodes pruned due to K* bounding, *SSP* are the nodes pruned due to subunit-stability constraint violations, *EH* counts the number of times an exact heuristic was used instead of search, t^{pre} is the pre-processing time of the algorithm (in seconds) - the majority of which is for compiling the heuristic, t^{search} is the time spend during search, t^{total} is the total elapsed algorithm time, K^*MAP is the returned K*MAP solution (in log_{10}), *BBK* t* is BBK*'s runtime (in seconds), and *BBK* sln* is BBK*'s highest valid K* value found (in log_{10}). Missing rows indicate the inability to solve the problem within the hour limit. Highlighted *UB's* indicate when the fastest way to solve the problem was by using the exactly computed K*MAP via BE. Highlighted t^{total} indicate when AOBB-K*'s runtime was faster than BBK*'s. Highlighted K*MAP values indicate when AOBB-K*'s reported K*MAP value is greater than the solution reported by BBK*.

Table 11 show sample results[2] on a hard benchmarks using modified [w]MBE-BBK* heuristics. The columns are labeled as before with the addition of H denoting the type of K* heuristic used, and *Anytime-K** denoting the best valid K* value found.

Domain Sizes. Each mutable residues considers 21 different amino acid assignments. Conformation variables corresponding to non-mutable residues had a domain size of 2-14 rotamers (with most having domain sizes 4-9). Conformation

[1] AOBB-K* without use of a weighted heuristic and without added determinism through thresholded-underflows

[2] Additional results available in the supplementary materials.

variables corresponding to mutable residues had a domain size of 34-35 when formulated as F1 and 203-205 when formulated as F2.

Table 1: Aggregated Statistics on F1,F2 (exact) vs. BBK*.

Dataset	$\mathbf{K^*}\geq$	$\mathbf{K^*}>$	$\mathbf{t^{total}}<$
Original (30)	30,30	2,2	23,28
Expanded (12)	11,12	1,4	2,4
Hard (6)	2,3	0,2	0,0

Table 2: Original problems with two MAP variables.

benchmark	F	ω	τ	iB	w*	|X|	UB	OR^R	AND^R	OR^{all}	AND^{all}	CPP	UBP	SSP	EH	t^{pre}	t^{search}	t^{total}	K*MAP	BBK* t	BBK* sln
1a0r_00031 P=5, L=2, PL=7	F1	1	0	3	8	16		12	23	300184	1784305	1243743	0	151	1	0.9	226.0	226.9	7.88	109.1	7.88
		1	1E-05	3	8	16		12	23	8886	18057	105692	0	44	1	1.1	4.9	5.9			
		0.001	0	7	8	16	10.17	2	3	12379	72187	57871	22	4	2	146.2	10.4	156.5			
		0.001	1E-05	3	8	16		12	23	8886	18057	105692	0	44	1	1.3	4.9	6.2			
	F2	1	0	3	6	16		2	12	304309	1798801	937451	0	52	1	0.7	92.3	93.0			
		1	1E-05	3	6	16		2	12	9066	18310	78238	0	19	1	0.7	0.8	1.5			
		0.001	0	5	6	16	9.95	2	3	11777	71585	33739	21	5	2	23.8	4.6	28.4			
		0.001	1E-05	3	6	16		2	12	9066	18310	78238	0	19	1	0.6	0.7	1.3			
1gwc_00021 P=4, L=1, PL=5	F1	1	0	6	6	12	10.28	4	9	41049	158841	378716	76	0	6	69.2	32.0	101.2	9.79	152.3	9.79
		1	1E-05	6	6	12	10.28	4	9	2511	7875	39530	66	0	6	68.0	2.0	70.1			
		0.001	0	6	6	12	10.28	2	3	3282	9492	52507	40	0	2	69.3	1.9	71.2	9.76		
		0.001	1E-05	6	6	12	10.28	2	3	255	579	5700	34	0	2	67.5	0.2	67.7			
	F2	1	0	4	4	12	10.29	3	7	28766	134930	77823	55	2	5	6.8	8.9	15.7	9.79		
		1	1E-05	4	4	12	10.23	3	7	1259	5687	7234	46	1	5	6.2	0.3	6.5			
		0.001	0	4	4	12	10.29	2	3	7289	34775	19392	39	1	2	6.8	2.2	8.9			
		0.001	1E-05	4	4	12	10.23	2	3	279	1251	1862	34	0	2	6.2	0.1	6.3			
2rl0_00008 P=2, L=2, PL=4	F1	1	0	5	5	10	11.60	2	3	2385	43767	1978	40	0	2	2.5	3.3	5.9	11.16	261.9	9.46
		1	1E-05	4	5	10	12.03	4	7	232	2077	2532	71	0	4	1.0	0.3	1.2			
		0.001	0	4	5	10	12.03	2	3	2385	43767	1978	40	0	2	1.0	3.3	4.3			
		0.001	1E-05	4	5	10	12.03	2	3	48	393	574	37	0	2	1.3	0.1	1.4			
	F2	1	0	4	3	10	11.16	2	3	2	3	0	40	0	3	2.6	0.0	2.6			
		1	1E-05	3	3	10	11.59	2	3	47	392	1387	37	0	2	0.7	0.0	0.7			
		0.001	0	5	3	10	11.16	2	3	2	3	0	40	0	3	2.6	0.0	2.6			
		0.001	1E-05	3	3	10	11.59	2	3	47	392	1387	37	0	2	1.0	0.0	1.0			

Table 3: Expanded problems with three MAP variables.

benchmark	F	ω	τ	iB	w*	|X|	UB	OR^R	AND^R	OR^{all}	AND^{all}	CPP	UBP	SSP	EH	t^{pre}	t^{search}	t^{total}	K*MAP	BBK* t	BBK* sln
1gwc_00021* P=4, L=1, PL=5	F2	1	0	4	4	13	12.51	20	27	33881	590621	473189	388	6	8	123.8	81.3	205.1	11.92	551.3	11.72
		1	1E-05	4	4	13	12.51	20	27	3194	25973	110351	362	6	8	117.1	3.5	120.7			
		0.001	0	4	4	13	12.51	3	4	4882	84496	75745	60	0	2	124.3	12.1	136.4			
		0.001	1E-05	4	4	13	12.51	3	4	349	2698	15569	54	0	2	116.8	0.4	117.2			
2xgy_00020* P=3, L=3, PL=6	F1	1	0	6	8	15	12.28	35	64	425512	2552281	7259396	652	226	30	122.0	653.2	775.2	10.96	1388.1	10.96
		1	1E-05	6	8	15	12.28	35	64	66457	186650	1324450	652	109	30	121.1	62.0	183.1			
		0.001	0	6	8	15	12.28	3	4	13639	81575	233365	60	0	2	121.3	14.4	135.7			
		0.001	1E-05	6	8	15	12.28	3	4	2351	6575	46989	60	0	2	121.2	1.5	122.7			
	F2	1	0	5	5	15	11.39	3	22	398102	2383318	7422285	42	0	20	81.8	278.9	360.7			
		1	1E-05	4	5	15	11.83	3	24	66355	185683	1579935	40	0	22	60.1	23.8	83.9			
		0.001	0	4	5	15	11.83	3	4	13627	81563	254094	60	0	2	60.4	8.2	68.6			
		0.001	1E-05	4	5	15	11.83	3	4	2352	6576	55979	60	0	2	61.1	0.8	61.9			
4wwi_00019* P=3, L=3, PL=6	F1	1	0	6	8	15	16.93	152	303	1257591	4864928	2446418	2672	865	152	119.6	1375.4	1494.9	14.99	34.0	14.99
		1	1E-05	6	8	15		219	530	90584	188534	480735	3232	2035	312	118.8	74.7	193.5			
		0.001	0	6	8	15	16.93	3	4	8047	30727	16189	59	1	2	119.4	5.2	124.6			
		0.001	1E-05	6	8	15		87	92	859	1686	4419	1547	35	6	119.1	1.3	120.4			
	F2	1	0	5	5	15	16.05	11	15	22945	87485	91677	176	75	5	169.2	12.1	181.3			
		1	1E-05	4	5	15		25	56	17779	35282	396450	397	84	32	54.2	8.4	62.7			
		0.001	0	4	5	15	16.48	5	8	14938	57354	54826	59	105	4	62.0	7.2	69.2			
		0.001	1E-05	4	5	15		5	8	893	1774	19843	54	61	4	61.9	0.5	62.4			

Table 4: Modified [w]MBE run on hard problems with 4 MAP variables.

benchmark	H	F	iB	ω	τ	w*	d	|X|	UB	t^{total}	Anytime-K*	K*MAP	BBK* t	BBK* sln
1gwc_00021** P=4,L=1, PL=5	wMBE	F1	6	0.001	1E-05	8	8	14		timeout	11.72		625.4	11.72
	MBE*	F1	6	0.001	1E-05	8	8	14	20.53	2090.5	11.92	11.92		
	wMBE*	F1	6	0.001	1E-05	8	8	14	19.92	1511.9	11.92	11.92		
	wMBE	F2	3	0.001	1E-05	4	8	14		timeout	6.24			

benchmark	F	bnb	f<	iB	w*	d	|X|	UB	OR*	AND*	OR-all	AND-all	CPP	UBP	SSP	EH	pre-t	Time	K*MAP	BBK* t	BBK* sln
1a0r_00031	F1	1	0	3	8	8	16		12	23	300184	1784305	1243743	0	151	1	0.9	226.9	7.88	109.1	7.88
		1	1E-05	3	8	8	16		12	23	8886	18057	105692	0	44	1	1.1	5.9			
		0.001	0	7	8	8	16	10.17	2	3	12379	72187	57871	22	4	2	146.2	156.5			
		0.001	1E-05	3	8	8	16		12	23	8886	18057	105692	0	44	1	1.3	6.2			
	F2	1	0	3	6	8	16		2	12	304309	1798801	937451	0	52	1	0.7	93.0			
		1	1E-05	3	6	8	16		2	12	9066	18310	78238	0	19	1	0.7	1.5			
		0.001	0	5	6	8	16	9.95	2	3	11777	71585	33739	21	5	2	23.8	28.4			
		0.001	1E-05	3	6	8	16		2	12	9066	18310	78238	0	19	1	0.6	1.3			
1gwc_00021	F1	1	0	6	6	6	12	10.28	4	9	41049	158841	378716	76	0	6	69.2	101.2	9.79	152.3	9.79
		1	1E-05	6	6	6	12	10.28	4	9	2511	7875	39530	66	0	6	68.0	70.1			
		0.001	0	6	6	6	12	10.28	2	3	3282	9492	52507	40	0	2	69.3	71.2	9.76		
		0.001	1E-05	6	6	6	12	10.28	2	3	255	579	5700	34	0	2	67.5	67.7			
	F2	1	0	4	4	6	12	10.29	3	7	28766	134930	77823	55	2	5	6.8	15.7			
		1	1E-05	4	4	6	12	10.23	3	7	1259	5687	7234	46	1	5	6.2	6.5	9.79		
		0.001	0	4	4	6	12	10.29	2	3	7289	34775	19392	39	1	2	6.8	8.9			
		0.001	1E-05	4	4	6	12	10.23	2	3	279	1251	1862	34	0	2	6.2	6.3			
1gwc_00033	F1	1	0	3	9	9	18		4	7	46660	183409	321999	0	47	1	1.0	30.2	10.48	512.5	10.48
		1	1E-05	4	9	9	18		4	7	14176	26101	140121	0	0	1	1.4	5.6			
		0.001	0	3	9	9	18		4	7	46660	183409	321999	0	47	1	1.1	34.5			
		0.001	1E-05	3	9	9	18		4	7	14176	26101	140121	0	0	1	1.1	5.4			
	F2	1	0	3	7	9	18		2	5	56498	193247	178597	0	19	1	1.3	12.3			
		1	1E-05	3	7	9	18		2	5	18176	30101	100362	0	0	1	1.5	2.8			
		0.001	0	3	7	9	18		2	5	56498	193247	178597	0	19	1	1.3	12.4			
		0.001	1E-05	3	7	9	18		2	5	18176	30101	100362	0	0	1	1.2	2.5			
2hnu_00026	F1	1	0	6	7	7	14	15.18	4	8	20896	104681	72450	77	37	5	14.6	25.0	13.18	436.9	13.18
		1	1E-05	6	7	7	14	15.18	4	8	4702	15353	25692	48	13	5	14.7	16.5			
		0.001	0	6	7	7	14	15.18	2	3	8686	42886	30172	40	0	2	14.7	16.5			
		0.001	1E-05	6	7	7	14	15.18	2	3	1592	5204	8640	30	0	2	14.4	14.7			
	F2	1	0	4	5	7	14	15.08	2	5	22010	105458	76657	38	0	4	1.7	7.3			
		1	1E-05	4	5	7	14	15.08	2	5	4972	15584	29869	30	0	4	1.8	2.6			
		0.001	0	4	5	7	14	15.08	2	3	9250	43450	33091	40	0	2	1.7	3.8			
		0.001	1E-05	4	5	7	14	15.08	2	3	1696	5308	10143	32	0	2	1.6	1.8			
2hnv_00025	F1	1	0	6	8	8	16	14.97	4	7	128010	332754	87628	78	0	4	15.6	45.8	13.65	962.1	13.65
		1	1E-05	6	8	8	16	14.97	4	7	11242	26138	22473	53	0	4	15.3	18.1			
		0.001	0	6	8	8	16	14.97	2	3	59796	155100	41420	40	0	2	15.6	22.8			
		0.001	1E-05	6	8	8	16	14.97	2	3	4140	9572	8415	30	0	2	15.3	15.9			
	F2	1	0	4	6	8	16	15.04	2	4	115194	297138	84882	39	0	3	1.7	16.6			
		1	1E-05	4	6	8	16	15.04	2	4	9383	21423	22246	33	0	3	1.6	2.6			
		0.001	0	4	6	8	16	15.04	2	3	60606	155910	46451	40	0	2	1.7	8.8			
		0.001	1E-05	4	6	8	16	15.04	2	3	4278	9710	10479	34	0	2	1.6	2.0			
2rf9_00007	F1	1	0	7	9	9	18		17	33	111674	380720	453549	0	410	1	3.0	74.6	14.08	45.5	14.08
		1	1E-05	5	9	9	18		18	35	45551	103794	273804	0	32	1	1.3	21.9			
		0.001	0	7	9	9	18		17	33	111674	380720	453549	0	410	1	3.0	74.6			
		0.001	1E-05	7	9	9	18		17	33	43248	99153	254219	0	39	1	2.6	19.6			
	F2	1	0	6	7	9	18	14.52	2	3	4264	11559	9772	36	5	1	4.1	4.7			
		1	1E-05	3	7	9	18		2	19	52616	109733	123399	0	31	1	0.5	4.1			
		0.001	0	6	7	9	18	14.52	2	3	4264	11559	9772	41	0	1	4.1	4.6			
		0.001	1E-05	3	7	9	18		2	19	52616	109733	123399	0	31	1	0.5	4.1			
2rf9_00013	F1	1	0	6	8	8	16	14.57	3	5	4074	19034	15979	61	0	3	6.6	9.0	13.25	11.8	13.25
		1	1E-05	3	8	8	16		22	43	7668	12012	47855	0	20	1	1.1	3.2			
		0.001	0	6	8	8	16	14.57	2	3	1691	7851	6697	41	0	2	5.9	6.5			
		0.001	1E-05	3	8	8	16		22	43	7668	12012	47855	0	20	1	1.0	3.2			
	F2	1	0	5	6	8	16	14.12	2	3	1691	7851	7275	41	0	2	1.4	1.9			
		1	1E-05	4	6	8	16	14.57	2	4	619	931	5204	19	0	3	0.8	0.8			
		0.001	0	4	6	8	16	14.57	2	3	1691	7851	7275	41	0	2	0.8	1.4			
		0.001	1E-05	4	6	8	16	14.57	2	3	153	225	1411	20	0	2	0.8	0.8			
2rf9_00018	F1	1	0	8	9	9	18	16.68	2	5	19927	84823	98124	39	0	4	66.5	76.1	15.79	187.2	15.79
		1	1E-05	4	9	9	18		2	20	32834	78293	278613	0	68	1	1.3	15.5			
		0.001	0	8	9	9	18	16.68	2	3	7241	28745	37652	41	0	2	66.5	69.6			
		0.001	1E-05	3	9	9	18		3	24	34358	82814	293945	0	104	1	0.9	14.0			
	F2	1	0	6	7	9	18	16.68	4	7	20137	85033	87306	78	0	4	8.4	15.1			
		1	1E-05	3	7	9	18		18	35	45584	99045	218730	0	269	1	0.7	6.8			
		0.001	0	6	7	9	18	16.68	2	3	7323	28827	33772	40	0	2	8.4	10.5			
		0.001	1E-05	3	7	9	18		18	35	45584	99045	218730	0	269	1	0.7	6.8			
2rf9_00042	F1	1	0	7	11	11	22		2	8	459950	2342453	1816775	0	77	1	4.3	307.5	22.65	897.1	22.65
		1	1E-05	5	11	11	22		2	9	31661	53664	241548	0	35	1	1.6	8.9			
		0.001	0	7	11	11	22		2	8	459950	2342453	1816775	0	77	1	4.9	352.0			
		0.001	1E-05	5	11	11	22		2	9	31661	53664	241548	0	35	1	1.7	9.0			
	F2	1	0	6	9	11	22		6	11	366672	1835478	1320278	0	162	1	15.8	148.8			
		1	1E-05	3	9	11	22		8	15	32902	55385	195925	0	22	1	1.0	3.0			

Table 5: Small problems (two mutable residues, ie. MAP variables).

		0.001	0	6	9	11	22		6	11	366672	1835478	1320278	0	162	1	15.7	151.0			
		0.001	1E-05	4	9	11	22		7	13	26784	44980	160623	0	30	1	1.3	2.7			
2rfd_00035	F1	1	0	7	8	8	16	18.14	7	13	1136806	5390146	4229817	142	0	7	188.1	818.6	17.27	1242.3	16.77
		1	1E-05	7	8	8	16	18.14	7	13	275481	1126929	1163732	136	0	7	182.5	311.1			
		0.001	0	6	8	8	16	18.57	2	3	94839	448619	364423	42	0	2	38.3	81.4			
		0.001	1E-05	6	8	8	16	18.57	2	3	23511	95635	103770	41	0	2	33.7	41.8			
	F2	1	0	6	6	8	16	17.70	2	5	896239	4253159	3273123	40	0	4	80.1	379.8			
		1	1E-05	5	6	8	16	18.17	2	8	275669	1127117	1152311	36	0	7	17.7	93.1			
		0.001	0	4	6	8	16	18.61	2	3	94321	448101	344311	42	0	2	3.9	31.1			
		0.001	1E-05	4	6	8	16	18.61	2	3	23372	95496	97779	41	0	2	3.6	9.3			
2rfe_00012	F1	1	0	5	7	7	14	15.23	4	7	4930	15982	26937	72	0	4	3.5	5.2	13.93	11.1	13.93
		1	1E-05	3	7	7	14		21	41	5621	11767	41873	0	33	1	1.0	3.0			
		0.001	0	5	7	7	14	15.23	2	3	636	2040	3675	38	0	2	3.5	3.7			
		0.001	1E-05	4	7	7	14		19	37	5591	11688	41054	0	29	1	1.1	3.0			
	F2	1	0	4	5	7	14	14.80	2	4	3126	10002	21164	37	0	3	0.8	1.5			
		1	1E-05	3	5	7	14	15.23	2	5	1816	3496	15378	18	0	4	0.5	0.7			
		0.001	0	3	5	7	14	15.23	2	3	640	2044	4382	38	0	2	0.5	0.7			
		0.001	1E-05	3	5	7	14	15.23	2	3	204	384	1922	20	0	2	0.5	0.5			
2rfe_00014	F1	1	0	5	7	7	14	15.66	4	7	6821	22109	37197	72	0	4	3.5	5.4	14.36	31.4	14.36
		1	1E-05	4	7	7	14		22	43	4656	10472	36297	0	27	1	1.1	2.9			
		0.001	0	5	7	7	14	15.66	2	3	3250	10546	17526	38	0	2	3.5	4.0			
		0.001	1E-05	3	7	7	14		22	43	4666	10493	36353	0	25	1	1.0	2.9			
	F2	1	0	4	5	7	14	15.23	2	4	4086	13086	26801	37	0	3	0.8	1.7			
		1	1E-05	3	5	7	14	15.66	2	5	1479	3063	12951	18	0	4	0.5	0.6			
		0.001	0	3	5	7	14	15.66	2	3	3304	10600	21476	38	0	2	0.5	1.3			
		0.001	1E-05	3	5	7	14	15.66	2	3	740	1532	6374	20	0	2	0.5	0.6			
2rfe_00017	F1	1	0	7	7	7	14	10.96	5	10	13061	54335	292185	82	2	6	16.3	22.4	10.52	29.2	10.52
		1	1E-05	6	7	7	14	11.39	9	27	1871	3862	37844	102	5	19	10.3	11.0			
		0.001	0	6	7	7	14	11.39	2	3	5303	22475	118890	37	0	2	10.4	12.4			
		0.001	1E-05	6	7	7	14	11.39	2	3	305	641	6090	30	0	2	10.3	10.4			
	F2	1	0	5	5	7	14	10.96	3	8	13148	54422	300675	49	4	6	1.7	7.0			
		1	1E-05	4	5	7	14	11.39	6	24	1917	3908	51904	80	4	19	0.7	1.1			
		0.001	0	4	5	7	14	11.39	2	3	5366	22538	122621	37	0	2	0.8	2.8			
		0.001	1E-05	4	5	7	14	11.39	2	3	318	654	7583	30	0	2	0.8	0.9			
2rfe_00030	F1	1	0	6	7	7	14	11.56	6	12	19405	160026	391901	101	37	7	28.9	60.1	10.50	181.5	10.50
		1	1E-05	5	7	7	14		18	68	8074	56199	171959	0	409	1	6.3	19.9			
		0.001	0	6	7	7	14	11.56	2	3	2499	18711	52966	38	0	2	28.8	32.1			
		0.001	1E-05	5	7	7	14		18	68	8074	56199	171959	0	409	1	6.3	19.9			
	F2	1	0	4	5	7	14	11.53	5	11	20393	164126	359007	87	40	7	6.9	18.9			
		1	1E-05	3	5	7	14		12	72	9466	65490	252486	0	378	1	0.7	5.7			
		0.001	0	4	5	7	14	11.53	2	3	2344	18556	41009	38	0	2	6.8	8.2			
		0.001	1E-05	3	5	7	14		12	72	9466	65490	252486	0	378	1	0.7	5.7			
2rfe_00041	F1	1	0	8	9	9	18		6	19	1784746	5246734	29278970	0	552	1	212.4	2477.2	22.73	1181.5	22.73
		1	1E-05	7	9	9	18		7	23	217800	511839	3687164	0	419	1	113.5	331.3			
		0.001	0	8	9	9	18		6	19	1784746	5246734	29278970	0	552	1	213.5	2481.5			
		0.001	1E-05	7	9	9	18		7	23	217800	511839	3687164	0	419	1	113.4	331.1			
	F2	1	0	5	7	9	18		13	29	1909509	5483503	3336192	0	533	1	48.1	401.7			
		1	1E-05	4	7	9	18		15	38	262067	579851	579770	0	441	1	4.3	34.4			
		0.001	0	5	7	9	18		13	29	1909509	5483503	3336192	0	533	1	46.3	403.1			
		0.001	1E-05	4	7	9	18		15	38	262067	579851	579770	0	441	1	4.2	34.3			
2rfe_00043	F1	1	0	5	8	8	16		13	44	308739	1146880	8281137	0	614	1	4.2	231.6	18.04	50.5	18.04
		1	1E-05	4	8	8	16		13	51	36691	168262	863934	0	402	1	1.1	34.7			
		0.001	0	7	8	8	16	18.91	2	3	15123	40023	419424	37	0	2	188.9	193.6			
		0.001	1E-05	4	8	8	16		13	51	36691	168262	863934	0	402	1	1.1	35.0			
	F2	1	0	6	6	8	16	18.48	2	5	15390	40297	422357	34	43	4	75.5	79.4			
		1	1E-05	4	6	8	16		7	32	41371	165735	989647	0	225	1	2.9	15.7			
		0.001	0	5	6	8	16	18.91	2	3	15378	40278	421535	37	0	2	52.7	56.5			
		0.001	1E-05	4	6	8	16		7	32	41371	165735	989647	0	225	1	2.9	15.7			
2rfe_00044	F1	1	0	7	8	8	16	19.06	14	29	236159	625979	6568520	222	5	16	211.4	319.6	18.19	74.5	18.19
		1	1E-05	6	8	8	16		14	70	65114	310015	1571020	0	407	1	14.5	75.7			
		0.001	0	7	8	8	16	19.06	2	3	15258	40338	422759	37	0	2	211.8	217.3			
		0.001	1E-05	6	8	8	16		14	70	65114	310015	1571020	0	407	1	14.5	75.8			
	F2	1	0	6	6	8	16	18.62	2	7	37887	99927	1047107	30	3	6	75.2	85.6			
		1	1E-05	4	6	8	16		7	63	77546	334512	1890564	0	338	1	2.9	26.2			
		0.001	0	5	6	8	16	19.06	2	3	15483	40563	424391	37	0	2	52.6	56.8			
		0.001	1E-05	4	6	8	16		7	63	77546	334512	1890564	0	338	1	2.9	26.1			
2rfe_00047	F1	1	0	5	9	9	18		2	7	265870	1106761	1516836	0	37	1	3.5	187.9	22.70	348.9	22.70
		1	1E-05	4	9	9	18		2	9	29098	66231	259146	0	31	1	1.2	9.9			
		0.001	0	5	9	9	18		2	7	265870	1106761	1516836	0	37	1	3.5	187.9			
		0.001	1E-05	4	9	9	18		2	9	29098	66231	259146	0	31	1	1.2	9.9			
	F2	1	0	3	7	9	18		6	11	258709	1099600	1056935	0	121	1	0.7	88.3			

Table 6: Small problems (two mutable residues, ie. MAP variables).

	F2	1	1E-05	3	7	9	18		6	11	25081	55130	131962	0	13	1	0.7	2.6			
		0.001	0	6	7	9	18	23.57	2	3	46488	171288	230173	41	0	2	60.6	73.2			
		0.001	1E-05	3	7	9	18		6	11	25081	55130	131962	0	13	1	0.7	2.6			
2rfe_00048	F1	1	0	6	10	10	20		2	9	410749	1668173	2542598	0	102	1	4.8	297.6	22.81	387.7	22.81
		1	1E-05	5	10	10	20		2	11	44592	109012	408214	0	76	1	2.0	16.8			
		0.001	0	6	10	10	20		2	9	410749	1668173	2542598	0	102	1	4.9	298.2			
		0.001	1E-05	5	10	10	20		2	11	44592	109012	408214	0	76	1	2.3	17.1			
	F2	1	0	4	8	10	20		8	15	477523	1933600	1813547	0	219	1	2.8	158.7			
		1	1E-05	3	8	10	20		9	17	49248	116219	204465	0	59	1	1.0	4.7			
		0.001	0	3	8	10	20		9	17	506753	2046164	1841727	0	259	1	0.9	146.8			
		0.001	1E-05	3	8	10	20		9	17	49248	116219	204465	0	59	1	1.0	4.7			
2rl0_00008	F1	1	0	5	5	5	10	11.60	2	3	2385	43767	1978	40	0	2	2.5	5.9	11.16	261.9	9.46
		1	1E-05	4	5	5	10	12.03	4	7	232	2077	2532	71	0	4	1.0	1.2			
		0.001	0	4	5	5	10	12.03	2	3	2385	43767	1978	40	0	2	1.0	4.3			
		0.001	1E-05	4	5	5	10	12.03	2	3	48	393	574	37	0	2	1.3	1.4			
	F2	1	0	4	3	5	10	11.16	2	3	2	3	0	40	0	3	2.6	2.6			
		1	1E-05	3	3	5	10	11.59	2	3	47	392	1387	37	0	2	0.7	0.7			
		0.001	0	5	3	5	10	11.16	2	3	2	3	0	40	0	3	2.6	2.6			
		0.001	1E-05	3	3	5	10	11.59	2	3	47	392	1387	37	0	2	1.0	1.0			
2xgy_00020	F1	1	0	5	7	7	14	12.34	3	7	58141	350035	986465	55	111	5	8.3	35.4	10.60	887.5	10.60
		1	1E-05	5	7	7	14	12.34	3	7	9977	28095	195848	55	59	5	8.2	10.9			
		0.001	0	5	7	7	14	12.34	2	3	43630	262510	739882	40	0	2	8.3	27.6			
		0.001	1E-05	5	7	7	14	12.34	2	3	7516	21244	148161	40	0	2	7.3	8.9			
	F2	1	0	4	5	7	14	11.47	2	3	43643	262523	743860	40	0	2	1.9	16.4			
		1	1E-05	4	5	7	14	11.47	2	3	7523	21251	150589	40	0	2	1.8	3.2			
		0.001	0	4	5	7	14	11.47	2	3	43643	262523	743860	40	0	2	1.9	16.3			
		0.001	1E-05	3	5	7	14	12.40	2	3	7523	21251	150589	38	2	2	1.1	2.5			
3cal_00032	F1	1	0	8	8	8	16	47.77	6	9	286785	1220353	8408660	71	47	4	221.2	751.7	11.62	1428.9	11.62
		1	1E-05	5	8	8	16		17	40	12053	45040	343587	0	670	1	1.7	16.5			
		0.001	0	7	8	8	16	49.01	4	5	95597	373997	2835676	35	45	2	83.9	265.7			
		0.001	1E-05	5	8	8	16		17	40	12053	45040	343587	0	670	1	1.7	16.6			
	F2	1	0	6	6	8	16	13.38	2	5	133851	1067419	531976	32	6	4	59.5	125.3			
		1	1E-05	3	6	8	16		3	17	8452	40467	74780	0	143	1	0.8	4.0			
		0.001	0	5	6	8	16	14.36	3	4	45209	369081	177421	33	27	2	20.2	46.6			
		0.001	1E-05	3	6	8	16		3	17	8452	40467	74780	0	143	1	0.9	4.0			
3ma2_00016	F1	1	0	5	7	7	14	13.82	14	23	21915	64035	91516	62	223	10	2.3	10.8	8.38	9.6	8.38
		1	1E-05	3	7	7	14		14	27	5520	9931	39870	0	6	1	1.0	2.8			
		0.001	0	5	7	7	14	13.82	14	15	4459	12875	18348	215	78	2	2.3	3.5	7.37		
		0.001	1E-05	5	7	7	14	9.69	2	3	259	403	1873	14	0	2	2.3	2.3			
	F2	1	0	4	5	7	14	13.39	2	7	12471	37071	57964	16	21	6	0.7	3.5			
		1	1E-05	3	5	7	14	9.69	2	10	4684	7396	35187	7	0	9	0.4	0.9	8.38		
		0.001	0	4	5	7	14	13.39	2	3	611	1811	2981	21	20	2	0.7	0.9			
		0.001	1E-05	3	5	7	14	9.69	2	3	251	395	1949	14	0	2	0.4	0.5			
3u7y_00009	F1	1	0	6	6	6	12	4.96	13	36	33137	212545	134394	232	6	24	8.1	37.7	4.51	190.8	4.51
		1	1E-05	6	6	6	12	4.96	13	36	5984	28798	40072	232	3	24	8.0	12.6			
		0.001	0	5	6	6	12	5.39	2	3	2586	16256	11107	40	0	2	1.8	3.1			
		0.001	1E-05	5	6	6	12	5.39	2	3	521	2457	3426	37	0	2	2.0	2.3			
	F2	1	0	6	4	6	12	4.51	2	3	2	3	0	40	0	3	5.7	5.7			
		1	1E-05	4	4	6	12	4.95	7	32	6832	31470	55724	88	3	26	0.9	3.1			
		0.001	0	3	4	6	12	5.39	2	3	2647	16317	10988	40	0	2	0.5	1.2			
		0.001	1E-05	3	4	6	12	5.39	2	3	553	2489	3744	36	0	2	0.5	0.6			
3u7y_00011	F1	1	0	5	6	6	12	12.72	5	9	5714	16064	34855	96	0	5	2.5	4.8	11.85	27.8	11.85
		1	1E-05	5	6	6	12	12.72	5	9	1075	1837	10834	89	0	5	2.4	2.7			
		0.001	0	5	6	6	12	12.72	2	3	248	644	2299	39	0	2	2.5	2.6			
		0.001	1E-05	5	6	6	12	12.72	2	3	55	85	766	33	0	2	2.4	2.5			
	F2	1	0	3	4	6	12	12.72	2	6	5757	16107	37892	36	0	5	0.5	2.0			
		1	1E-05	3	4	6	12	12.72	2	6	1172	1934	15745	28	0	5	0.5	0.7			
		0.001	0	3	4	6	12	12.72	2	3	220	616	1593	39	0	2	0.5	0.6			
		0.001	1E-05	3	4	6	12	12.72	2	3	49	79	810	31	0	2	0.5	0.5			
4hem_00027	F1	1	0	3	8	8	16		2	4	5499	12643	49235	0	42	1	1.1	3.3	15.48	39.9	15.48
		1	1E-05	3	8	8	16		2	4	2075	4259	19973	0	3	1	1.1	1.9			
		0.001	0	3	8	8	16		2	4	5499	12643	49235	0	42	1	1.1	3.2			
		0.001	1E-05	3	8	8	16		2	4	2075	4259	19973	0	3	1	1.1	1.9			
	F2	1	0	3	6	8	16		3	5	4467	11460	18378	0	45	1	1.0	2.0			
		1	1E-05	3	6	8	16		3	5	1639	3720	8771	0	4	1	0.9	1.0			
		0.001	0	3	6	8	16		3	5	4467	11460	18378	0	45	1	1.3	2.4			
		0.001	1E-05	3	6	8	16		3	5	1639	3720	8771	0	4	1	0.9	1.0			
	F1	1	0	3	8	8	16		2	4	3587	12104	24106	0	56	1	1.1	2.5			
		1	1E-05	3	8	8	16		2	4	1809	4206	15090	0	13	1	1.1	1.5			
		0.001	0	3	8	8	16		2	4	3587	12104	24106	0	56	1	1.1	2.5			

Table 7: Small problems (two mutable residues, ie. MAP variables).

4hem_00028		0.001	1E-05	3	8	8	16		2	4	1809	4206	15090	0	13	1	1.1	1.5	15.27	34.5	15.27
		1	0	3	6	8	16		3	5	3177	11537	13051	0	41	1	0.9	1.8			
	F2	1	1E-05	3	6	8	16		3	5	1589	3981	9282	0	2	1	0.9	1.0			
		0.001	0	3	6	8	16		3	5	3177	11537	13051	0	41	1	0.9	1.8			
		0.001	1E-05	3	6	8	16		3	5	1589	3981	9282	0	2	1	0.9	1.0			
4kt6_00023		1	0	7	8	8	16	18.24	10	12	17300	58100	363924	168	28	3	20.7	40.7	12.69	136.5	12.69
	F1	1	1E-05	6	8	8	16	14.80	10	13	5251	10171	117045	41	31	4	11.1	16.6			
		0.001	0	6	8	8	16	18.68	2	3	8647	22567	188442	33	4	2	11.3	19.5			
		0.001	1E-05	6	8	8	16	14.80	2	3	1749	2709	39669	16	2	2	11.0	12.5			
		1	0	4	6	8	16	14.80	2	5	38186	101546	23877	16	19	4	2.1	7.3			
	F2	1	1E-05	4	6	8	16	14.80	2	5	3163	8083	4889	9	7	4	2.0	2.3			
		0.001	0	4	6	8	16	14.80	2	3	8388	22308	5231	35	2	2	2.1	2.9			
		0.001	1E-05	4	6	8	16	14.80	2	3	618	1578	999	16	2	2	2.0	2.1			
4kt6_00024		1	0	7	8	8	16	19.87	4	5	8649	22569	188442	52	25	2	20.8	29.7	12.93	120.7	12.93
	F1	1	1E-05	6	8	8	16	16.94	8	10	4174	8782	92775	18	13	3	11.1	15.1			
		0.001	0	6	8	8	16	20.30	2	3	8647	22567	188442	27	10	2	11.3	20.8			
		0.001	1E-05	6	8	8	16	16.94	2	3	2085	3237	47497	12	2	2	11.0	12.7			
		1	0	4	6	8	16	14.87	2	4	21980	58460	13622	17	19	3	2.1	5.1			
	F2	1	1E-05	4	6	8	16	14.87	2	4	2848	7456	3193	10	3	3	2.0	2.3			
		0.001	0	4	6	8	16	14.87	2	3	8388	22308	5231	34	3	2	2.1	3.0			
		0.001	1E-05	4	6	8	16	14.87	2	3	714	1866	902	13	1	2	2.0	2.1			
4wwi_00019		1	0	7	7	7	14	15.43	2	3	8046	30726	16164	40	0	2	58.4	61.0	14.99	26.3	14.99
	F1	1	1E-05	5	7	7	14	16.30	21	87	25831	53574	133265	273	19	67	3.2	11.9			
		0.001	0	5	7	7	14	16.30	2	3	8046	30726	16164	40	0	2	3.0	5.6			
		0.001	1E-05	5	7	7	14	16.30	2	3	475	964	2493	36	0	2	3.2	3.4			
		1	0	5	5	7	14	15.43	2	3	8094	30774	17888	40	0	2	5.6	7.9			
	F2	1	1E-05	4	5	7	14	15.86	6	26	10670	20946	100937	97	0	21	1.0	3.3			
		0.001	0	3	5	7	14	16.30	2	3	8094	30774	17888	40	0	2	0.5	2.8			
		0.001	1E-05	3	5	7	14	16.30	2	3	521	1010	4649	36	0	2	0.5	0.6			

Table 8: Small problems (two mutable residues, ie. MAP variables).

benchmark	F	bnb	f<	iB	w*	d	|X|	UB	OR*	AND*	OR-all	AND-all	CPP	UBP	SSP	EH	pre-t	Time	K*MAP	BBK* t	BBK* sln
1gwc_00021_1_0_4_3	F2	1	0	4	4	7	13	12.51	20	27	33881	590621	473189	388	6	8	123.8	205.1	11.92	551.3	11.72
		1	1E-05	4	4	7	13	12.51	20	27	3194	25973	110351	362	6	8	117.1	120.7			
		0.001	0	4	4	7	13	12.51	3	4	4882	84496	75745	60	0	2	124.3	136.4			
		0.001	1E-05	4	4	7	13	12.51	3	4	349	2698	15569	54	0	2	116.8	117.2			
2hnv_00025_3_0_4_3	F1	1	1E-05	5	9	9	17		247	1029	1001206	2687713	6118892	0	7839	1	11.9	1433.6	16.18	880.5	13.65
		0.001	1E-05	5	9	9	17		247	1029	1001206	2687713	6118892	0	7839	1	11.9	1430.7			
	F2	1	0	4	6	9	17	18.38	4	7	215171	550559	220825	77	0	4	109.8	153.8			
		1	1E-05	4	6	9	17	18.38	4	7	11158	25046	44795	63	0	4	100.8	102.4			
		0.001	0	4	6	9	17	18.38	3	4	72135	184463	74753	59	0	2	109.3	121.5			
		0.001	1E-05	4	6	9	17	18.38	3	4	3539	7907	14703	46	0	2	100.7	101.2			
2rf9_00007_1_0_7_3	F1	0.001	1E-05	8	10	10	19		379	738	1694883	6536479	24664824	0	529	1	57.9	3611.5	14.73	369.4	14.73
	F2	1	1E-05	3	7	10	19		22	442	4353727	9503674	15149348	0	659	1	6.0	725.8			
		0.001	1E-05	3	7	10	19		22	442	4353727	9503674	15149348	0	659	1	6.0	733.9			
2rf9_00013_3_0_4_3	F1	1	1E-05	3	9	9	17		441	881	427301	783693	3444343	21	317	1	3.0	408.5	15.03	39.2	15.03
		0.001	1E-05	3	9	9	17		441	881	427301	783693	3444343	21	317	1	3.0	407.1			
	F2	1	0	4	6	9	17	16.36	3	10	36574	172094	184989	55	0	8	83.0	100.8			
		1	1E-05	4	6	9	17	16.36	3	10	2638	3934	38141	34	0	8	71.4	71.8			
		0.001	0	4	6	9	17	16.36	3	4	3328	15648	16891	61	0	2	82.8	84.4			
		0.001	1E-05	4	6	9	17	16.36	3	4	296	440	4283	40	0	2	80.3	80.4			
2rfe_00012_3_0_3_3	F1	1	0	3	8	8	15		21	77	17726	60980	112686	1	518	1	2.9	24.5	13.93	11.8	13.93
		1	1E-05	3	8	8	15		21	77	5402	12005	48249	1	397	1	2.9	8.8			
		0.001	0	3	8	8	15		21	77	17726	60980	112686	1	518	1	2.9	24.4			
		0.001	1E-05	3	8	8	15		21	77	5402	12005	48249	1	397	1	2.9	8.8			
	F2	1	0	4	5	8	15	14.80	5	9	5239	18227	58358	96	0	5	58.3	60.9			
		1	1E-05	3	5	8	15		35	321	18964	44087	1487077	0	1180	1	5.5	20.3			
		0.001	0	4	5	8	15	14.80	3	4	641	2045	6710	58	0	2	58.6	58.9			
		0.001	1E-05	3	5	8	15		35	321	18964	44087	1487077	0	1180	1	5.4	21.0			
2rfe_00014_3_0_3_3	F1	1	0	3	8	8	15		22	85	27154	90364	170428	1	532	1	3.0	34.3	13.96	44.9	14.36
		1	1E-05	3	8	8	15		22	85	5243	12262	48900	1	449	1	2.9	9.2			
		0.001	0	3	8	8	15		22	85	27154	90364	170428	1	532	1	2.9	34.3			
		0.001	1E-05	3	8	8	15		22	85	5243	12262	48900	1	449	1	2.9	9.3			
	F2	1	0	4	5	8	15	15.23	5	8	5882	18770	62846	94	0	4	58.2	60.6	14.36		
		1	1E-05	3	5	8	15		41	381	19834	46560	1681191	0	1413	1	4.9	20.0			
		0.001	0	4	5	8	15	15.23	3	4	3305	10601	31370	58	0	2	58.2	59.4			
		0.001	1E-05	3	5	8	15		41	381	19834	46560	1681191	0	1413	1	5.5	22.9			
2rfe_00017_2_0_4_3	F1	1	1E-05	6	8	8	15		292	1951	372617	1038706	10337795	0	6797	1	37.6	1041.4	10.86	78.0	10.80
		0.001	1E-05	6	8	8	15		292	1951	372617	1038706	10337795	0	6797	1	38.0	1063.2			
	F2	1	0	5	5	8	15	11.46	16	58	245894	1063198	6389737	227	25	43	166.8	334.6			
		1	1E-05	4	5	8	15	11.89	39	189	15748	32709	606708	411	141	151	85.7	92.8			
		0.001	0	4	5	8	15	11.89	4	17	7328	31941	194059	57	151	14	89.1	94.1			
		0.001	1E-05	4	5	8	15	11.89	4	17	480	871	24031	50	51	14	85.7	86.0			
2rfe_00030_4_3_2_0	F1	1	1E-05	6	8	8	15		245	1002	134024	432910	3963332	0	5708	1	117.2	613.3	11.12	275.4	10.97
		0.001	1E-05	6	8	8	15		245	1002	134024	432910	3963332	0	5708	1	117.0	617.5			
	F2	1	0	4	5	8	15	13.61	48	124	256957	1327425	2816050	726	83	77	115.2	276.6			
		1	1E-05	4	5	8	15	13.21	30	106	5948	14903	245173	355	27	77	105.4	108.9			
		0.001	0	4	5	8	15	13.61	3	4	890	4082	11901	54	2	2	101.4	101.9	10.95		
		0.001	1E-05	4	5	8	15	13.21	3	4	18	36	1128	45	1	2	106.0	106.0	11.03		
2xgy_00020_3_0_3_3	F1	1	0	6	8	8	15	12.28	35	64	425512	2552281	7259396	652	226	30	122.0	775.2	10.96	1388.1	10.96
		1	1E-05	6	8	8	15	12.28	35	64	66457	186650	1324450	652	109	30	121.1	183.1			
		0.001	0	6	8	8	15	12.28	3	4	13639	81575	233365	60	0	2	121.3	135.7			
		0.001	1E-05	6	8	8	15	12.28	3	4	2351	6575	46989	60	0	2	121.2	122.7			
	F2	1	0	5	5	8	15	11.39	3	22	398102	2383318	7422285	42	0	20	81.8	360.7			
		1	1E-05	4	5	8	15	11.83	3	24	66355	185683	1579935	40	0	22	60.1	83.9			
		0.001	0	4	5	8	15	11.83	3	4	13627	81563	254094	60	0	2	60.4	68.6			
		0.001	1E-05	4	5	8	15	11.83	3	4	2352	6576	55979	60	0	2	61.1	61.9			
3u7y_00009_2_0_3_3	F1	1	0	6	7	7	13	5.39	118	325	221684	1419747	921627	1792	175	208	118.6	574.4	4.51	215.8	4.51
		1	1E-05	6	7	7	13		129	382	45828	231276	302283	1322	252	254	118.1	191.0			
		0.001	0	6	7	7	13	5.39	3	4	2587	16257	11133	57	0	2	119.5	122.2			
		0.001	1E-05	6	7	7	13		3	4	40	232	254	48	1	2	118.1	118.2	3.07		
	F2	1	0	4	4	7	13	4.95	12	39	36760	228568	564654	204	7	28	62.6	99.5	4.51		
		1	1E-05	4	4	7	13		12	39	7147	32753	222131	169	4	28	61.3	66.4			
		0.001	0	4	4	7	13	4.95	3	4	2648	16318	37313	57	0	2	62.5	64.7			
		0.001	1E-05	4	4	7	13		3	4	554	2490	15834	47	1	2	61.2	61.5			
3u7y_00011_2_0_3_3	F1	1	0	5	7	7	13		41	77	23893	81707	183381	0	939	1	12.9	45.1	11.85	26.6	11.85
		1	1E-05	4	7	7	13		58	111	8367	30329	124844	0	904	1	2.9	21.3			
		0.001	0	5	7	7	13		41	77	23893	81707	183381	0	939	1	12.8	45.1			
		0.001	1E-05	4	7	7	13		58	111	8367	30329	124844	0	904	1	2.9	21.4			
	F2	1	0	4	4	7	13	12.29	3	7	5758	16108	68579	50	0	5	74.0	76.1			
		1	1E-05	4	4	7	13	12.29	3	7	1173	1935	40256	39	0	5	80.7	81.2			
		0.001	0	4	4	7	13	12.29	3	4	221	617	2751	53	0	2	83.4	83.5			
		0.001	1E-05	4	4	7	13	12.29	3	4	50	80	1775	42	0	2	80.8	80.8			
4wwi_00019_3_0_3_3	F1	1	0	6	8	8	15	16.93	152	303	1257591	4864928	2446418	2672	865	152	119.6	1494.9	14.99	34.0	14.99
		1	1E-05	6	8	8	15		219	530	90584	188534	480735	3232	2035	312	118.8	193.5			
		0.001	0	6	8	8	15	16.93	3	4	8047	30727	16189	59	1	2	119.4	124.6			
		0.001	1E-05	6	8	8	15		87	92	859	1686	4419	1547	35	6	119.1	120.4			
	F2	1	0	5	5	8	15	16.05	11	15	22945	87485	91677	176	75	5	169.2	181.3			
		1	1E-05	4	5	8	15		25	56	17779	35282	396450	397	84	32	54.2	62.7			
		0.001	0	4	5	8	15	16.48	5	8	14938	57354	54826	59	105	4	62.0	69.2			

Table 9: Expanded problems (three mutable residues, ie. MAP variables).

		0.001	1E-05	4	5	8	15		5	8	893	1774	19843	54	61	4	61.9	62.4

Table 10: Expanded problems (three mutable residues, ie. MAP variables).

benchmark	H	F	iB	bnb	f<	w*	d	|X|	UB	Time	Anytime-K*	K*MAP	BBK* t	BBK* sln
1gwc_00021_1_0_4_4	wMBE	F1	6	0.001	1E-05	8	8	14		timeout	11.72		625.4	11.72
	MBE*	F1	6	0.001	1E-05	8	8	14	20.53	2090.5	11.92	11.92		
	wMBE*	F1	6	0.001	1E-05	8	8	14	19.92	1511.9	11.92	11.92		
	wMBE	F2	3	0.001	1E-05	4	8	14		timeout	6.24			
2hnv_00025_3_0_4_4	wMBE	F1	6	0.001	1E-05	10	10	18		timeout	11.52		1013.2	13.65
	MBE*	F1	6	0.001	1E-05	10	10	18	21.91	2038.1	16.18	16.18		
	wMBE*	F1	6	0.001	1E-05	10	10	18	21.81	2085.6	16.18	16.18		
	wMBE	F2	4	0.001	1E-05	6	10	18		3384.4	16.18	16.18		
	MBE*	F2	4	0.001	0	6	10	18	1017.56	timeout	14.32			
	wMBE*	F2	4	0.001	0	6	10	18	1016.46	timeout	14.32			
2rf9_00007_1_0_7_4	wMBE	F1	8	0.001	0	11	11	20		timeout	12.02		399.6	14.73
	MBE*	F1	8	0.001	0	11	11	20	516.51	timeout	13.41			
	wMBE*	F1	8	0.001	0	11	11	20	515.99	timeout	13.41			
	wMBE	F2	3	0.001	0	7	11	20		timeout				
2rf9_00013_3_0_4_4	wMBE	F1	6	0.001	1E-05	10	10	18		1686.5	15.03	15.03	43.8	15.03
	MBE*	F1	5	0.001	1E-05	10	10	18	17.78	105.1	15.03	15.03		
	wMBE*	F1	5	0.001	1E-05	10	10	18	17.68	62.3	15.03	15.03		
	wMBE	F2	4	0.001	1E-05	6	10	18		timeout	15.03			
2rfe_00017_2_0_4_4	wMBE	F1	5	0.001	1E-05	9	9	16		timeout	10.33		91.2	10.80
	MBE*	F1	5	0.001	1E-05	9	9	16	22.70	timeout	10.86			
	wMBE*	F1	5	0.001	1E-05	9	9	16	23.00	timeout	10.86			
	wMBE	F2	4	0.001	1E-05	5	9	16		timeout	10.86			
2rfe_00030_4_4_2_0	wMBE	F1	6	0.001	0	9	9	16		timeout	10.27		248.3	10.97
	MBE*	F1	6	0.001	0	9	9	16	515.44	timeout	10.29			
	wMBE*	F1	6	0.001	0	9	9	16	515.23	timeout	10.29			
	wMBE	F2	3	0.001	0	5	9	16		timeout				

Table 11: Adjusted [w]MBE on Large problems (4 mutable residues, ie. MAP variables).

8.3 ANALYSIS

Formulation 1 vs. Formulation 2. The aggregated statistics in Table 1 show that F2 is generally superior to F1. One aspect of F1 that is particularly handicapping is that, due to it using an indexing scheme between amino-acids and their rotamers, all interaction functions need to include *both* residue and conformation variables, thus leeding to a more densely connected graph (as can seen by the greater w* for F1 in Tables 2-4). That being said, a strength of F1 vs. F2 is its smaller domain sizes in the presence of MAP variables, which comes into play during the heuristic evaluation. When looking at the highest iB able to be used for each formulation (not explicitly shown) across the different expansions of a benchmark, the highest iB possible by F2 drops more quickly as MAP variables are added (due to the great increase in domain sizes incurred by adding additional MAP variables). An open question is how to combine the strengths of both formulations, perhaps in conjunction with other potential innovations (as will be highlighted).

AOBB-K^{*1} vs BBK*: K* solution. AOBB-K* and BBK* find the same K* solution for all but two of the original problems solved. However, as problems are expanded, the frequency that the two algorithms find different solutions increases. In each exception, AOBB-K* outputted a K* solution was greater than that outputted by BBK*. These results are captured in Table 1 and concrete examples can be seen in Tables 2 and 3 where cases where the K* solution differ are highlighted in the K*MAP columns. Through further analysis, AOBB-K*'s solutions were verified to be valid based on the problem formulations.

AOBB-K^{*1} vs BBK*: Speed. AOBB-K* showed powerful performance on the original benchmarks solving nearly every problem faster than BBK*. As the problems were expanded, however, we see AOBB-K*'s performance begin to drop more rapidly than BBK*'s. In the expanded problems, AOBB-K* was able to surpass BBK* on four problems, but not the other eight. For the hard problems, AOBB-K* was only able to confirm a K*MAP value on one problem in the allotted hour (not explicitly shown). These results are captured in Table 1. Concrete examples can be seen in Tables 2, 3, and 4 where we can also see AOBB-K* taking advantage of wMBE-K*, using the heuristic value when it is known to be exact rather than searching the underlying subspace (see column EH). That being said, it is important to note that: (1) AOBB-K* finds solutions greater than that of BBK* (which may partially account for the increased time) and (2) AOBB-K* returns intermediate anytime solutions, some of which exceeded the K* value returned by BBK*. Nevertheless, improving scalability is an important future direction, and potential directions illuminated by this work will be discussed in the final section.

Weighted Heuristic Search. As an initial venture into an approximate version of AOBB-K* we modified the algorithm to employ a weighted heuristic (see Tables 2, 3, $\omega = 0.001$ vs. $\omega = 1$). In nearly all cases, moving to approximate search reduced the MAP search space and thus improved the time. As would be expected however, applying a weighted heuristic did not help as much when the heuristic was particularly weak or few MAP nodes were already being searched. In some cases, it was more beneficial to spend longer computing a more accurate heuristic so that it could be taken advantage of during search when using a weighted heuristic (notice the increase in iB used for benchmark 2rl0_00008 when using $\omega = 0.001$).

Infusing Determinism. To observe the effects of infusing determinism, we applied thresholded-underflow using a $\tau = 1 \times 10^{-5}$ and compared to base problems ($\tau = 0$). In all cases, we see that the thresholded-underflows improved search times (see columns t^{search} in Tables 2 and 3), sometimes so much so that the best time corresponded to allowing for a more crudely computed heuristic to enter search more quickly (notice the iB drop, yet shorter t^{search}, on benchmarks 2rl0_00008, 2xyg_00020*, and 4wwi_00019*). Even when the same iB is used, interestingly, the speedup due to underflows does not always correspond to a reduced search of the MAP space (see OR 4wwi_00019*). Here we see that the heuristic is adversely affected by the underflow (see Section 5) resulting in less efficient pruning of the MAP space. Next we present initial exploration into a potential remedy.

MBE$^+$-K* and wMBE$^+$-K*. As described in Section 5, bounding a ratio of functions such as K* is especially difficult - particularly because it relies on lower-bounding, which can be especially problematic in the presence of determinism. However, in domains such as CPD where we are guaranteed non-zero solutions for the lower bounded portions, it can be safe to ignore certain tuples that correspond to zeros in computations. To empirically test the effects of such a modification, we applied MBE$^+$-K* and wMBE$^+$-K* to particularly large F1 problems for which wMBE-K* was unbounded (Table 4). In every case, the modified heuristics were able to provide a bounded estimate, in four of the six problems enabling AOBB-K* to find an anytime solution better than BBK*, and in two of the five problem that previously could not be solved exactly, allowing the algorithm to find a solution within the hour. As we will outline further in the conclusion, these results, motivate incorporating constraints into the heuristic evaluation and exploring new internal representations.

9 CONCLUSION

Conclusion. In summary, this work provides **(1)** two distinct graphical model formulations (with strengths and weaknesses explored) for use with K*MAP-adapted algorithms over AND/OR search spaces. **(2)** a new wMBE-K* heuristic and exploration into innovations to improve the quality of K*MAP heuristic bounds by considering constraints. **(3)** AOBB-K*, a depth-first AND/OR branch-and-bound algorithm for optimizing K* (and an accompanying approximate ω-AOBB-K*), that shows great promise - outperforming the state-of-the-art BBK* on many problems - yet leaves room for advancement to address scaling to larger problems. And **(4)** a scheme to exploit determinism safely introducing underflows into problem formulations with theoretical guarantees, which provided significant speed-up.

As a foundation. Although it is rewarding that our algorithms and modifications performed well on modest problems, the purpose of this work is foundational and lays an initial framework from which to explore new innovations. From this initial exploration, three directions that have immediately illuminated are: **(1)** research into methodologies and problem representations that exploiting determinism applicable to the K*MAP task such as the work of Larkin and Dechter [2003], **(2)** generation of new schemes to incorporate constraints - both local and global - into the K* heuristic or new heuristic schemes in general (such as using an alternate bucket elimination methods such as Deep Bucket Elimination Razeghi et al. [2021]), and **(3)** adaptation of advanced state-of-the-art mixed-inference algorithms to solving the K*MAP query, including sophisticated exact algorithms such as Recursive Best-First AND/OR Search Marinescu et al. [2018], 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 et al. [2019] or Abstraction Sampling Kask et al. [2020].

More applicably, this framework can now be tuned more specifically for the protein domain by: **(1)** forming designs containing independencies that can be exploited by an AND/OR scheme, **(2)** integrating the many optimizations present in well established optimized CPD software, such as BBK* via OSPREY, and extending the complexity of problems addressed, such as to include backbone ensembles.

10 CONCLUSION

Conclusion. In summary, this work provides **(1)** two distinct graphical model formulations (with strengths and weaknesses explored) for use with K*MAP-adapted algorithms over AND/OR search spaces. **(2)** a new wMBE-K* heuristic and exploration into innovations to improve the quality of K*MAP heuristic bounds by considering constraints. **(3)** AOBB-K*, a depth-first AND/OR branch-and-bound algorithm for optimizing K* (and an accompanying approximate ω-AOBB-K*), that shows great promise - outperforming the state-of-the-art BBK* on many problems - yet leaves room for advancement to address scaling to larger problems. And **(4)** a scheme to exploit determinism safely introducing underflows into problem formulations with theoretical guarantees, which provided significant speed-up.

As a foundation. Although it is rewarding that our algorithms and modifications performed well on modest problems, the purpose of this work is foundational and lays an initial framework from which to explore new innovations. From this initial exploration, three directions that have immediately illuminated are: **(1)** research into methodologies and problem representations that exploiting determinism applicable to the K*MAP task such as the work of Larkin and Dechter [2003], **(2)** generation of new schemes to incorporate constraints - both local and global - into the K* heuristic or new heuristic schemes in general (such as using an alternate bucket elimination methods such as Deep Bucket Elimination Razeghi et al. [2021]), and **(3)** adaptation of advanced state-of-the-art mixed-inference algorithms to solving the K*MAP query, including sophisticated exact algorithms such as Recursive Best-First AND/OR Search Marinescu et al. [2018], 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 et al. [2019] or Abstraction Sampling Kask et al. [2020].

More applicably, this framework can now be tuned more specifically for the protein domain by: **(1)** forming designs containing independencies that can be exploited by an AND/OR scheme, **(2)** integrating the many optimizations present in well established optimized CPD software, such as BBK* via OSPREY, and extending the complexity of problems addressed, such as to include backbone ensembles.

References

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

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

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

Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. doi: 10.2200/S00529ED1V01Y201308AIM023. URL http://dx.doi.org/10.2200/S00529ED1V01Y201308AIM023.

Rina Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms, second edition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13:1–199, 02 2019. doi: 10.2200/S00893ED2V01Y201901AIM041.

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

Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *J. ACM*, 50(2):107–153, 2003. doi: 10.1145/636865.636866. URL http://doi.acm.org/10.1145/636865.636866.

Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, pages 502–518, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Mark Hallen, Jeffrey Martin, Adegoke Ojewole, Jonathan Jou, Anna Lowegard, Marcel Frenkel, Pablo Gainza, Hunter Nisonoff, Aditya Mukund, Siyu Wang, Graham Holt, David Zhou, Elizabeth Dowd, and Bruce Donald. Osprey 3.0: Open-source protein redesign for you, with powerful new features. *Journal of Computational Chemistry*, 39, 10 2018. doi: 10.1002/jcc.25522.

Mark A. Hallen and Bruce R. Donald. Protein design by provable algorithms. *Commun. ACM*, 62(10):76–84, sep 2019. ISSN 0001-0782. doi: 10.1145/3338124. URL https://doi.org/10.1145/3338124.

G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities*. Cambridge Mathematical Library. Cambridge University Press, 1988. ISBN 9781107647398. URL https://books.google.com/books?id=EfvZAQAAQBAJ.

Kalev Kask, Bobak Pezeshki, Filjor Broka, Alexander Ihler, and Rina Dechter. Scaling up and/or abstraction sampling. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4266–4274. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/589. URL https://doi.org/10.24963/ijcai.2020/589. Main track.

D. Larkin and R. Dechter. Bayesian inference in the presence of determinism. *AI and Statistics(AISTAT03)*, 2003.

Junkyu Lee, Radu Marinescu, Rina Dechter, and Alexander Ihler. From exact to anytime solutions for marginal map. AAAI'16, page 3255–3262. AAAI Press, 2016.

Qiang Liu and Alexander Ihler. Bounding the partition function using Hölder's inequality. In *International Conference on Machine Learning (ICML)*, pages 849–856, New York, NY, USA, June 2011. ACM.

Qi Lou, Rina Dechter, and Alexander Ihler. Interleave variational optimization with monte carlo sampling: A tale of two approximate inference paradigms. 2019.

Radu Marinescu, Rina Dechter, and Alexander Ihler. And/or search for marginal map. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 563–572, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

Radu Marinescu, Junkyu Lee, Rina Dechter, and Alexander Ihler. And/or search for marginal map. *J. Artif. Int. Res.*, 63(1): 875–921, sep 2018. ISSN 1076-9757. doi: 10.1613/jair.1.11265. URL https://doi.org/10.1613/jair.1.11265.

Adegoke Ojewole, Jonathan D. Jou, Vance G. Fowler, and Bruce Randall Donald. *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, 2018. doi: 10.1089/cmb.2017.0267. URL https://doi.org/10.1089/cmb.2017.0267.

James D. Park. Map complexity results and approximation methods. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, page 388–396, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608974.

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

Wei Ping, Qiang Liu, and Alexander T Ihler. Decomposition bounds for marginal MAP. In *Advances in Neural Information Processing Systems 28*, pages 3267–3275. Curran Associates, Inc., 2015.

Yasaman Razeghi, Kalev Kask, Yadong Lu, Pierre Baldi, Sakshi Agarwal, and Rina Dechter. Deep bucket elimination. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4235–4242. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/582. URL https://doi.org/10.24963/ijcai.2021/582. Main Track.

Manon Ruffini, Jelena Vucinic, Simon de Givry, George Katsirelos, Sophie Barbe, and Thomas Schiex. Guaranteed diversity and optimality in cost function network based computational protein design methods. *Algorithms*, 14(6), 2021. ISSN 1999-4893. URL https://www.mdpi.com/1999-4893/14/6/168.

Clement Viricel, Simon de Givry, Thomas Schiex, and Sophie Barbe. Cost function network-based design of protein-protein interactions: predicting changes in binding affinity. *Bioinformatics (Oxford, England)*, 34, 02 2018. doi: 10.1093/bioinformatics/bty092.

Jelena Vucinic, David Simoncini, Manon Ruffini, Sophie Barbe, and Thomas Schiex. Positive multistate protein design. *Bioinformatics (Oxford, England)*, 36, 06 2019. doi: 10.1093/bioinformatics/btz497.

Yichao Zhou, Yuexin Wu, and Jianyang Zeng. Computational protein design using and/or branch-and-bound search. *Journal of computational biology : a journal of computational molecular cell biology*, 23, 05 2016. doi: 10.1089/cmb.2015.0212.