Approximate Counting with Deterministic Guarantees for Affinity Computation*

Clément Viricel[1,2], David Simoncini[1], David Allouche[1], Simon de Givry[1], Sophie Barbe[2], and Thomas Schiex[1]

[1] Unité de Mathématiques et Informatique Appliquées UR 875, INRA, F-31320 Castanet Tolosan, France,
[2] Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés, INSA, UMR INRA 792/CNRS 5504, F-31400 Toulouse, France

Abstract. Computational Protein Design aims at rationally designing amino-acid sequences that fold into a given three-dimensional structure and that will bestow the designed protein with desirable properties/functions. Usual criteria for design include stability of the designed protein and affinity between it and a ligand of interest. However, estimating the affinity between two molecules requires to compute the partition function, a #P-complete problem.
Because of its extreme computational cost, bio-physicists have designed the K^* algorithm, which combines Best-First A^* search with dominance analysis to provide an estimate of the partition function with deterministic guarantees of quality. In this paper, we show that it is possible to speed up search and keep reasonable memory requirement using a Cost Function Network approach combining Depth First Search with arc consistency based lower bounds. We describe our algorithm and compare our first results to the Computational Protein Design (CPD)-dedicated software *Osprey 2.0*.

Keywords: computational protein design, protein-ligand affinity, constraint Programming, cost function network, soft arc consistency, counting problems, weighted #CSP

Introduction

Proteins are polymer chains composed of amino-acids. Natural evolutionary processes have fashioned by means of amino-acid sequence variations (mutations, recombinations and duplications) an array of proteins endowed with functions ranging from catalysis, signaling to recognition and repair [2]. These functions are made possible by the ability of proteins to self-assemble into well-defined

* This workshop paper is an updated version of reference [1]. Compared to this paper, the lower bound used has been simply improved by accounting for unary costs, leading to the speedups shown in Table 1.

three-dimensional (3D) structures specified by their amino-acid sequences and hence, to interact with various types of molecular partners with high affinity and selectivity. Despite a plethora of functionalities, there is still an ever-increasing demand for proteins endowed with specific properties/functions which are not known to exist in nature. To this end, protein engineering has become a key technology to generate the proteins with the targeted properties/functions [3]. However, despite significant advances, protein engineering still suffers from a major drawback related to the limited diversity of protein sequences that can be explored in regard to the huge potential sequence space. If one considers that each position of a small protein of 100 amino-acid can be replaced by any of the 20 possible natural amino-acids, a theoretical space size of 20^{100} sequences is reached. One can thus easily imagine that sampling such large spaces is out of reach of wet methods. Consequently, approaches aiming at rationalizing protein evolution and favoring exploration of sequence regions of particular relevance for the targeted property/function are crucially needed to increase the odds of success to tailor the desired proteins while decreasing experimental efforts. In this context, computational methods have gained a prominent place in protein engineering strategies. Structure-based Computational Protein Design (CPD) has emerged with the increasing number of protein structures available in databases and the advents in computational structural biology and chemistry to understand fundamental forces between atoms and (macro)molecules. CPD uses the tight relationships existing between the function and the structure of a protein to design novel functions by searching the amino-acid sequences compatible with a known 3D scaffold, that should be able to carry out the desired function. CPD can thus be seen as the inverse folding problem [4] which can be formulated as an optimization problem, solvable computationally. In recent years, CPD has experienced important success, especially in the design of therapeutic proteins [5], novel enzymes [6]... Nevertheless, the computational design of proteins with defined affinity for a given partner (such as a small molecule, a substrate or a peptide) which is essential for large range of applications, continues to present challenges.

Estimating the binding affinity between two molecular partners requires to compute the so-called partition function of each of the molecules and of the system formed by the two bound molecules. The partition function of a molecule that can have states $\ell \in \Lambda$ each with energy E_ℓ is defined through the Boltzmann distribution as $\sum_{\ell \in \Lambda} \exp(-\frac{E_\ell}{k_B \cdot T})$ where T is the temperature and k_B is the Boltzmann constant. Computing the partition function of a system at constant temperature therefore requires to sum a simple function of the energy over a large set of possible states Λ. This computation is easy for systems with a small number of states. It becomes extremely complex for systems that have a number of states which is exponential in the size of their description. Even with the usual simplifying assumptions adopted in the field of CPD, the problem is actually #P-complete, a computational complexity class of problems with extreme complexity.

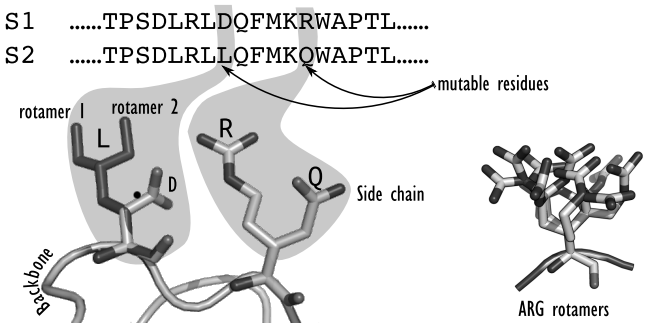

Fig. 1. A local view of a protein backbone and side-chains. Changes can be caused by amino acid identity substitutions (*for example D/L*) or by side-chain reorientations (rotamers). A typical rotamer library for one amino acid is shown on the right (ARG=Arginine).

A protein is defined by a supporting skeleton (or backbone) which is built by polymerization. Each amino-acid in the protein is composed of two parts: one part participates in the construction of the linear backbone of the protein and the remaining part is called the side-chain of the amino-acid. At each position, one may choose among 20 different possible side-chains, each defining a different amino-acid type, associated with a specific letter in the alphabet. A protein can therefore be described as a sequence of letters.

To perform their function, proteins need to be folded into a specific 3D shape which is defined by the amino-acid composition and the effect of various atomic forces (torsion angles, electrostatic, van der Waals, solvation effects). The precise conformation of a protein is therefore defined by the conformation of the backbone and the relative conformation of all its side-chains. A usual simplification assumption in CPD is to assume that the backbone is rigid (it has only one conformation) and that the side-chains can only adopt a specific set of discrete low-energy conformations, called rotamers. Several rotamer libraries have been defined over the last two decades. For a protein with a sequence S of length n, with a known composition (side-chains), the set of reachable conformations Λ^S is then defined as the cross-product of the set of the rotamers of every amino-acid. A conformation ℓ is just a choice of rotamer for each side-chain of the protein.

The computation of the partition function Z_A of a molecule A requires the computation of the energy of every conformation. A second usual assumption of CPD states that the energy of a molecule, defined as a function of the conformation ℓ, can be decomposed as a sum of a constant term (describing the energy of the backbone), a variable term for each side-chain (describing the interaction between the side-chain and the backbone and side-chain internal interactions) and a variable term for every pair of side-chains that interact (describing the interaction between the two corresponding side-chains). Stated otherwise, the energy can be pairwise decomposed. Different such pairwise decompositions of the energy have been built over the last decades.

The affinity of 2 molecules A and B is then proportional to the ratio of the partition function of the complex formed by the two molecules denoted Z_{AB} to the product of their partition functions $Z_A \times Z_B$. One affinity estimation therefore requires to compute three partition functions. The estimated affinity being just the criteria used for molecule design, i.e. to fix the amino-acid sequence, it must be repeatedly evaluated for each possible sequence considered for design. With a choice among 20 naturally occurring amino-acids at every position, the size of the sequence space is also exponential. This is why most affinity based designs or analysis are usually done on very small or drastically simplified systems.

In this paper, we focus on the problem of computing a lower estimate of the partition function of a protein with deterministic guarantees, enabling to approximate the affinity. The algorithm we define can be seen as an evolution of a traditional approximate partition function computation algorithm developed for affinity approximation called K^* [7]. The K^* algorithm combines dominance analysis with A^* Best First Search to produce a sequence of conformations of increasing energy. This sequence can be used to compute a running sum of contributions to the partition function and prune conformations that cannot contribute sufficiently to the partition function (because of their high energy).

Our method follows the same idea but relies on the use of specific lower bounding techniques that have been developed for exact combinatorial optimization in the last decades in the field of weighted Constraint Programming [8] and more precisely Cost Function Network and local consistencies [9]. While the use of such bounds is usual for exact optimization, this is, to our knowledge, the first use of soft local consistency to prune the search space of approximate partition function computation. The second new ingredient is a shift from space-intensive best-first search to depth-first search, completed by on-the-fly variable elimination [10].

We compare our algorithm to the K^* algorithm implemented in *Osprey* [11], a CPD-dedicated platform.

1 Background and Notations

We denote by A_A the set of all 20 amino-acid types. We consider an initial protein defined by sequence of amino acids $S = s_1, \ldots, s_n$, $s_i \in A_A$. The protein S has a 3D scaffold which is selected amongst high resolution 3D structures determined either from X-ray crystallography or Nuclear Magnetic Resonance (and deposited in the Protein Data Bank). This defines an initial fixed backbone.

Among all positions of S, some positions $M \subset \{1, \ldots, n\}$ are considered as mutable and can be replaced by any amino acid in a subset A_{A_i} of A_A. For each amino-acid type t in A_A, we have a discrete library of possible conformations (or rotamers) denoted as Λ_t. Given the sequence S, the atomic coordinates of the associated backbone and a choice $\ell_i \in \Lambda_{S_i}$ of a given conformation for each side-chain, the energy of the global conformation can be written as a function

of the vector of rotamers used, denoted as $\ell \in \Lambda^S = \prod_i \Lambda_{S_i}$:

$$E^S(\ell) = E^S_\varnothing + \sum_i E^S(\ell_i) + \sum_i \sum_{j>i} E^S(\ell_i, \ell_j)$$

where E^S is the potential energy of the protein S, $E^S_\varnothing$ is a constant energy contribution capturing interactions between fixed parts of the model, $E^S(\ell_i)$ is the energy contribution of rotamer ℓ_i at position i capturing internal interactions or interactions with the backbone, and $E^S(\ell_i, \ell_j)$ is the pairwise interaction energy between rotamer ℓ_i at position i and rotamer ℓ_j at position j [12]. Thanks to this decomposition, each energy terms, in $kcal \cdot mol^{-1}$, can be pre-computed and cached, allowing to quickly compute the energy of a conformation once the rotamer used at each position is fixed.

Assuming for the sake of simplicity that no symmetry or state degeneracy exists in the molecule considered, the partition function of a protein S for a fixed position, rotation and backbone over all side-chain conformations is defined as:

$$Z^S = \sum_{\ell \in \Lambda^S} e^{-\frac{E^S(\ell)}{k_B T}}$$

This sum contains an exponential number of terms. The exponential distribution leads to terms with sharp changes in magnitude. Most significant terms correspond to low energies. Our aim is to provide a deterministic algorithm that computes a lower approximation $\hat{Z}^S$ such that:

$$\frac{Z^S}{1+\varepsilon} \leq \hat{Z}^S \leq Z^S \tag{1}$$

The problem of exactly computing Z^S in this case is known to be #P-complete. The "simple" problem of finding a rotamer vector ℓ with energy below a value k is NP-hard [13]. Usual methods for estimating Z^S are stochastic methods using Monte-Carlo (Markov-chain) approaches with no non-asymptotic guarantees [14]. In the context of protein design, the K^* algorithm is a simple algorithm that provides an approximation to Z^S satisfying (1). We now quickly present it.

1.1 The K^* Algorithm

The K^* algorithm is an approximate *counting* algorithm that exploits optimization techniques. The first component is a dominance analysis called "Dead End Elimination" (DEE [12]). This process compares a worst case energy for a rotamer $\ell_i \in \Lambda_{S_i}$ with a best case cost for a different rotamer $\ell'_i \in \Lambda_{S_i}$ and prunes $\ell'_i \in \Lambda_{S_i}$ if its best case energy is larger than the other energy (replacing ℓ'_i by ℓ_i in any solution can only improve the energy) i.e., iff:

$$\left[E^S(\ell'_i) + \sum_{j \neq i} \min_{\ell_j \in \Lambda_{S_j}} E^S(\ell'_i, \ell_j)\right] - \left[E^S(\ell_i) + \sum_{j \neq i} \max_{\ell_j \in \Lambda_{S_j}} E^S(\ell_i, \ell_j)\right] > 0 \tag{2}$$

This is done iteratively on all positions i and all possible pairs of rotamers in Λ_i, pruning the space of conformations.

The second component is a best-first branch and bound A^* optimization algorithm exploring a tree where each node corresponds to a partially determined conformation where a subset of all positions have fixed rotamers. The sons of a node are defined by choosing a position i with a yet unfixed rotamer and creating one node per possible rotamer $\ell_i \in \Lambda_{S_i}$. Leaves correspond to completely determined conformations ℓ. At a given node at depth d with fixed rotamers $\ell = (\ell_1, \ldots, \ell_d)$, it is possible to compute a lower bound $Lb(\ell)$ on the energy of any complete conformation below this node. Following the A^* terminology, this lower bound defines an "admissible heuristics" and is defined as $Lb(\ell) =$

$$\underbrace{\sum_{i=1}^{d} E(\ell_i) + \sum_{j=i+1}^{d} E(\ell_i, \ell_j)}_{\text{Fixed}} + \sum_{j=d+1}^{n} \Big[\underbrace{\min_{\ell_j \in \Lambda_{S_j}} (E(\ell_j) + \sum_{i=1}^{d} E(\ell_i, \ell_j)}_{\text{Fixed-Unfixed}} + \underbrace{\sum_{k=j+1}^{n} \min_{\ell_k \in \Lambda_{S_k}} E(\ell_j, \ell_k))}_{\text{Unfixed}} \Big]$$

Starting from the root, if the node with the best lower bound is always developed first, the sequence of *leaves* explored by A^* will produce a stream of complete conformations, starting with the optimal conformation with minimum energy (and thus maximal contribution to the partition function) and followed by sub-optimal conformations sorted by increasing energy.

For optimization alone, it is amazing to see that the simple algorithm combining DEE and A^* (DEE/A^*) is only slightly dominated by 01-Linear programming models solved using CPLEX [15] and outperforms very concise quadratic programming formulations solved using CPLEX or BiqMac [16].

For counting, the DEE preprocessing may remove sub-optimal conformations whose energy is sufficiently low to significantly contribute to the partition function. It is therefore weakened by using an energy threshold $E_w > 0$, replacing the rhs zero in equation 2. Then DEE can only prune conformations with energy at least E_w away from the optimum energy.

As A^* produces conformations in increasing order of energy, the space of all conformations Λ^S is split in 3 subsets: a set P of conformations that have been pruned by $DEE(E_w)$, a set V of conformations already visited by A^* and a set U of yet unexplored conformations. This splits the partition function in 3 terms: $Z^S = Z_P^S + Z_V^S + Z_U^S$, each summed on the corresponding conformation space. The value of the second term is known. It is possible to bound the value of the first term: we compute a lower bound $Lb_P = \min_{\ell_i \text{ pruned}} Lb(\ell_i)$ on the energy of every conformations pruned by $DEE(E_w)$ and conclude that $Z_P^S \leq |P| \cdot \exp(\frac{-Lb_P}{k_B T})$. Finally, if E_{A^*} is the energy of the last conformation produced by A^*, since the sequence of conformation is decreasing, we have $Z_U^S \leq |V| \cdot \exp(\frac{-E_{A^*}}{k_B T})$. Thus:

$$Z^S \leq Z_V^S + |P| \cdot e^{\frac{-Lb_P}{k_B T}} + |V| \cdot e^{\frac{-E_{A^*}}{k_B T}}$$

.

Hence, if $E_{A^*} \geq -k_B T \cdot [\log(Z_V^S \cdot \frac{\varepsilon}{1+\varepsilon} - |P| \cdot e^{\frac{-Lb_P}{k_B T}}) - \log(|U|))$, we know that Z_V^S is an ε-approximation of Z_S and we can stop A^* search, possibly pruning a

lot of conformations that do not contribute enough to Z^S. However, it is also possible that too many conformations have been pruned by $DEE(E_w)$ and this threshold is never reached. In this case, the A^* must exhaust all conformations and a sufficient (easy to determine) number of conformations in P must be restored and the A^* search redone. Remind that A^* is a worst-case exponential space and time algorithm. The dominance analysis of DEE is a double-edged sword: it may prune a lot of conformations but if too many conformations are pruned, a full search is required and another search needs to be done on a larger set of conformations.

1.2 Cost Function Network and Local Consistency

Following our previous work on optimization for CPD presented in [17, 15], we model the distribution of energies as a Cost Function Network (CFN) or Weighted CSP. In a CFN (X, D, C, k), X is a finite set of variables, each variable $i \in X$ has a finite domain $D_i \in D$, each cost function $c_Y \in C$ is a function from $\prod_{i \in Y} D_i$ to $\{0, \ldots, k\}$ and k is an upper bound defining an intolerable cost. For a given assignment ℓ of X, the cost of ℓ is the sum of all cost functions (if it is less than k) or k otherwise. A complete assignment is a solution of the CFN if its cost is strictly less than k. Notice that all costs being non negative, $c_\varnothing \in C$ is a lower bound on the cost of any assignment.

The energy distribution of S can be obviously modeled in a CFN with one variable $i \in X$ for every position, the domain of i is the set of rotamers Λ_{S_i} and the cost functions include one zero-ary, unary and binary cost functions representing respectively $E^S_\varnothing, E^S(\ell_i)$ and $E^S(\ell_i, \ell_j)$.

If K^* relies on DEE and A^*, our new algorithm combines soft local consistencies (instead of DEE) and a Depth First Branch and Bound (DFBB) algorithm (instead of A^*). The DFBB algorithm performs counting instead of minimization and uses a dedicated dynamic pruning condition that guarantees to compute an ε-approximation of Z^S.

We quickly present the fundamental properties of soft local consistencies, without details. The reader is invited to read [9] for a precise description of existing arc consistencies for CFN. Enforcing a given soft local consistency transforms a CFN (X, D, C, k) into an equivalent CFN (X, D', C', k) that satisfies the corresponding local consistency. By equivalence, we mean that the two CFNs have the same set of solutions with the same cost. The fact that (X, D', C', k) satisfies the local consistency usually means that $c_\varnothing$ has increased (it cannot decrease in any case), providing a stronger lower bound on the optimum and that domains have been reduced (some values that cannot participate in a solution are deleted). Naturally, the lower the k is, the stronger is the pruning.

2 DFBB+Arc Consistency to Compute Z with Pruning

Traditional DFBB algorithms for CFN maintain a given level of soft arc consistency at each node (producing a non naive lower bound $c_\varnothing$ at each node) and

use the cost of the current best known solution as the current upper bound k. This allows to prune efficiently as k decreases rapidly as search proceeds.

To transform this algorithm in a counting algorithm:

- each time a leaf ℓ with cost (energy) E_ℓ is encountered we contribute $\exp(\frac{-E_\ell}{k_B T})$ to a running lower estimate $\hat{Z}^S$ of Z^S (initially set to 0).
- each time we prune a node with partial assignment ℓ', we compute an upper bound $Ub(\ell')$ on the contribution to the partition function of all the leaves below the pruned node and contribute this upper bound to a running upper bound Z^S_+ of the partition function over pruned conformations
- to decide when to prune, we enforce the invariant $\hat{Z}^S \geq \frac{\hat{Z}^S + Z^S_+}{1+\varepsilon}$. We prune if and only if adding $Ub(\ell')$ to Z^S_+ does not break this invariant.

Theorem 1. *The algorithm is correct: it provides a final estimate $\hat{Z}^S$ such that $Z^S \geq \hat{Z}^S \geq \frac{Z^S}{1+\varepsilon}$*

Proof. Initially, $\hat{Z}^S = Z^S_+ = 0$ and the invariant is satisfied. When the search finishes, all conformations have either been explored or pruned and therefore $(\hat{Z}^S + Z^S_+) \geq Z^S$. Using the invariant, we conclude that ultimately $\hat{Z}^S \geq \frac{Z^S}{1+\varepsilon}$. By construction, $Z^S \geq \hat{Z}^S$. □

The remaining ingredient is the upper bound $Ub(\ell)$. We know that all conformations below the current node have a cost (energy) larger than $c_\varnothing$, thus a contribution to Z^S smaller than $\exp(\frac{-c_\varnothing}{k_B T})$. Denoting by $N(\ell)$ the product of the domain sizes of all unassigned variables, we can conclude that $Ub_0(\ell) = N(\ell) \cdot \exp(\frac{-c_\varnothing}{k_B T})$ is an upper bound on the contribution of all leaves below the current node. Instead of only taking the size of the different domains, we can easily tighten this bound by taking into account unary energies in the reformulated model as $Ub_1(\ell) = \exp(\frac{-c_\varnothing}{k_B T}) \times \prod(\sum_{a \in \Lambda_{S_i}} e^{-E_i(a)/k_b T})$ (Note that if you assume that unary costs are equal to 1 , you will find the domain sizes).

This general schema is improved using on-the-fly variable elimination [10]. If a variable i has only few neighbors (typically 2 or 3) at the current node, we eliminate this variable using a simple sum-product algorithm. We call the resulting algorithms Z^*.

Z^* can be easily improved by providing it with either a set of conformations of low energies or with any available lower bound on the partition function Z^S. These can be used to strengthen the pruning condition in the beginning of the search. Double counting of identical conformations can be easily avoided using efficient direct comparison of conformations by discrimination trees or hash tables or simply by using the maximum of this initial upper bound and the running upper bound $\hat{Z}^S$ in the pruning condition. Our two first versions of Z^* are called Z^*_0 and Z^*_1 (to leave possible improvement in the pruning algorithm).

3 Experimental Comparison

To compare K^* to Z^*_0/Z^*_1, we examined the binding affinity of different protein/ligand complexes. The 3D model of these molecular systems were derived

from crystallographic structures of the proteins in complex with their ligands, deposited in the protein data bank (Table 1). Missing heavy atoms in crystal structures as well as hydrogen atoms were added using the *tleap* module of the *Amber 14* software package [18]. The molecular all-atom *ff14SB* was used for the proteins and the ligands while the *gaff* force field was used for the cofactor AMP (present in one studied system, PDB: 1AMU). The molecular systems were then subjected to 1000 steps of energy minimizations with the *Sander* module of *Amber 14*. Next, we have selected a portion of the proteins including residues at the interface between the protein and the ligand as well as a shell surrounding residues with at least one atom within 8 to 12 Å (according to the molecular system) of the interface.

PDB ID	Ligand	Variable res.	Seq. Number
1AMU	F	(10,2/9)	1584
1TP5	KKETWV	(12,2/3)	1121
1B74	Q	(10,2/9)	1809
2Q2A	R	(13,2/12)	4716

Table 1. For each ID, we give the ligand, the number of variable residues (flexible, mutations/mutable residues) and the number of sequences represented.

The residues at the interface protein/ligand and the ligand were considered flexible and represented by rotamers from the Lovell's penultimate library [19]. At most 2 of the flexible residues of the protein were allowed to simultaneously mutate, while the remaining flexible residues were allowed to change their side-chain conformation. In addition to the wild-type identity for the positions allowed to mutate, a selective subset of amino-acid types (ranging from 7 to 19 amino-acids) was allowed in the redesign. All energy matrices were generated using *Osprey 2.0* [11]. The energy function is the sum of the *Amber* electrostatic, van der Waals and dihedral terms.

Floating point energies were translated, multiplied by a large constant K and truncated to the nearest integer in the CFN. We implemented our algorithm in our open source solver `toulbar2`[3]. Our DFBB algorithm maintains Existential Directional Arc Consistency (EDAC) [20], on-the-fly elimination [10], a value heuristics using the existential support and a variable ordering combining weighted degree [21] and last conflict [22] heuristics.

To compare Z^* as implemented in our C++ solver `toulbar2` to the algorithm K^* implemented in *Osprey 2.0* in Java, we set ε to 10^{-3} and `toulbar2` and asked to estimate the affinity of each mutated proteins sequence with the substrate. For *Osprey 2.0*, we used the options `doMinimize = false`, `gamma = 0.0`. This last option guarantees that all mutations will be evaluated. The default values `initEw = 6.0`, `pruningE = 100.0`, `stericE = 30.0` were also used. For

[3] `mulcyber.toulouse.inra.fr/projects/toulbar2/`

`toulbar2 0.9.7`, we used options `-logz -zub=0` and `-logz -zub=1` that selects the Z_0^* and Z_1^* algorithm respectively and the upper bounding $Ub_{0,1}(\ell)$ described before and turns DEE processing off. For the 1AMU model, this represents a total of around 3,168 counting problems, each with different energy landscapes. The proxy to the affinity defined by the ratio of the partition functions is only approximated by the two methods, and these approximations may differ and rank sequences with close values differently. We compared the rank of the first 40 sequences and the two sets were identical and identically ordered.

For the 1AMU model, K^* explored 6,451,997 nodes in 76,654 seconds (CPU user time) while `toulbar2` explored 84,993 nodes in 30 seconds (CPU user time) for the same value of ε. So, Z_0^* is around 2,500 times faster and offers stronger pruning, moreover Z_1^* take 23% less time and explored 30% less nodes than our first version. While the difference in efficiency between C++ and Java may contribute to this difference, it can only explain a small part of this ratio. For the other models, K^* took over the time limit (250 hours CPU user time) that we set and it is therefore impossible to compare precisely. Moreover the improvement provide by $Ub_1(\ell)$ is approximatively twice better in consuming time and explored nodes (Table 2). We also studied the influence of ε on the computing

	K^*		Z_0^*		Z_1^*	
PDB ID	nodes	time	nodes	time	nodes	time
1AMU	6.45	1278	0.085	0.5	−23%	−30%
1TP5	∞	∞	3.19	31	−51%	−47%
1B74	∞	∞	5.64	35	−41%	−35%
2Q2A	∞	∞	39.9	596	−56%	−43%

Table 2. For each system, we give the number of nodes ($\times 10^6$) and user cpu-time in minutes. We used $\varepsilon = 10^{-3}$. ∞ means that the computational time is over 250h cpu-time

time and pruning. Even if the influence is not really drastic, as expected the higher ε is, the worst the approximation $\hat{Z}$ is. Notice that a value of $\varepsilon = 1$ means that our estimate is guaranteed to be only withing a factor 2 of the true value of Z^S.

4 Conclusion

In statistical physics in general and specifically in Computational Protein Design, computing or estimating the partition function is a central but difficult problem. The computation of the simple pure conformational partition function of an amino-acid based system with fixed backbone and flexible side-chains represented as rotamer sets with a pairwise decomposed energy already defines a $\#P$-complete problem.

In general, this problem has a very dense graph which makes exact graph-structure based approaches inadequate. It also has a very sharp energy landscape

	$\varepsilon = 1$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-6}$	
PDB ID	nodes	time	nodes	time	nodes	time
1AMU	6.45	27.123	84,993	30.209	91,046	30.741
1TP5	2,290,910	1,254.456	3,190,349	1,845.352	3,278,235	2,138.06
1B74	3,725,217	3,072.183	5,635,612	5,113.682	6,464,729	5,682.806
2Q2A	27,003,446	24,012.182	39,939,197	35,383.914	56,956,558	50,752.495

Table 3. For each system, and for different values of ε we give the number of nodes and user cpu-time in seconds. The value $\varepsilon = 0$ was also tried for 1AMU. In this case, steric clashes are the only possible cause of pruning (infinite energy) and more than 340,000 nodes were explored in 72 seconds. In tight interfaces, this is an important source of pruning.

that allows for efficient optimization but does not facilitate a stable probabilistic (Monte Carlo-based) estimation of the partition function. Such estimates also offer no deterministic guarantee on the quality of the estimation.

The K^* approach developed for CPD and our new Z_0^* algorithm instead exploits the fact that in some cases, a very small fraction of the conformational space may contribute enough to essentially define the partition function. In our knowledge, our algorithm is the first deterministic algorithm for weighted counting that exploits soft local consistency reformulation based lower bounds for counting with deterministic guarantees. Naturally, these results are preliminary. Several directions can be pursued to reinforce these results:

- consider larger molecular systems for comparison. The combinatorial advantage of Z_0^* compared to K^* that results from the stronger lower bound of local consistencies, the absence of DEE, together with the space effectiveness of DFBB compared to A^* should lead to even stronger speedups.
- shift to higher resolution rotamer libraries. The penultimate library has a lower resolution than alternative libraries such as [23] or [24]. In *Osprey*, another approach using continuous pre and post minimization is used instead. We could then compare the two approaches in terms of consistency of the predicted affinity ranking.
- compare Z_0^* to exact counting algorithms such as Cachet [25], Sentential Decision Diagrams [26] both on CPD problems (with sharp changes in contributions to Z^S) and other usual probabilistic systems such as classical instances of Markov Random Fields and Bayesian Nets.
- it is known that our lower bounds have corresponding lower bounding mechanisms in Weighted Satisfiability [27]. They could therefore be used to improve Cachet or SDD solvers for approximate computations with deterministic guarantees.
- improve Z_0^* with stronger bounds and alternative search strategies that favors the early discovery of low energy conformations and possibly exploits the interaction graph structure through a tree-decomposition.

These preliminary results also show the interest of using these bounds instead of DEE dominance analysis for pruning the conformational space. Local

consistency provides good lower bounds and its pruning can be easily controlled to avoid pruning sub-optimal solutions that can participate significantly to the partition function. This is a major advantage compared to DEE.

References

1. Viricel, C., Simoncini, D., Allouche, D., de Givry, S., Barbe, S., Schiex, T.: Approximate counting with deterministic guarantees for affinity computation. In: Modelling, Computation and Optimization in Information Systems and Management Sciences. Springer (2015) 165–176
2. Fersht, A.: Structure and mechanism in protein science: a guide to enzyme catalysis and protein folding. WH. Freemean and Co., New York (1999)
3. Peisajovich, S.G., Tawfik, D.S.: Protein engineers turned evolutionists. Nature methods **4**(12) (December 2007) 991–4
4. Pabo, C.: Molecular technology. Designing proteins and peptides. Nature **301**(5897) (January 1983) 200
5. Miklos, A.E., Kluwe, C., Der, B.S., Pai, S., Sircar, A., Hughes, R.A., Berrondo, M., Xu, J., Codrea, V., Buckley, P.E., et al.: Structure-based design of supercharged, highly thermoresistant antibodies. Chemistry & biology **19**(4) (2012) 449–455
6. Siegel, J.B., Zanghellini, A., Lovick, H.M., Kiss, G., Lambert, A.R., St Clair, J.L., Gallaher, J.L., Hilvert, D., Gelb, M.H., Stoddard, B.L., Houk, K.N., Michael, F.E., Baker, D.: Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction. Science (New York, N.Y.) **329**(5989) (July 2010) 309–13
7. Georgiev, I., Lilien, R.H., Donald, B.R.: The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles. Journal of computational chemistry **29**(10) (July 2008) 1527–42
8. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Elsevier (2006)
9. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence **174** (2010) 449–478
10. Larrosa, J.: Boosting search with variable elimination. In: Principles and Practice of Constraint Programming - CP 2000. Volume 1894 of LNCS., Singapore (September 2000) 291–305
11. Gainza, P., Roberts, K.E., Georgiev, I., Lilien, R.H., Keedy, D.A., Chen, C.Y., Reza, F., Anderson, A.C., Richardson, D.C., Richardson, J.S., et al.: Osprey: Protein design with ensembles, flexibility, and provable algorithms. Methods Enzymol (2012)
12. Desmet, J., De Maeyer, M., Hazes, B., Lasters, I.: The dead-end elimination theorem and its use in protein side-chain positioning. Nature **356**(6369) (April 1992) 539–42
13. Pierce, N.A., Winfree, E.: Protein design is NP-hard. Protein engineering **15**(10) (October 2002) 779–82
14. Rubinstein, R.Y., Ridder, A., Vaisman, R.: Fast sequential Monte Carlo methods for counting and optimization. John Wiley & Sons (2013)
15. Allouche, D., André, I., Barbe, S., Davies, J., de Givry, S., Katsirelos, G., O'Sullivan, B., Prestwich, S., Schiex, T., Traoré, S.: Computational protein design as an optimization problem. Artificial Intelligence **212** (2014) 59–79

16. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Programming **121**(2) (2010) 307
17. Traoré, S., Allouche, D., André, I., de Givry, S., Katsirelos, G., Schiex, T., Barbe, S.: A new framework for computational protein design through cost function network optimization. Bioinformatics **29**(17) (2013) 2129–2136
18. Case, D., Babin, V., Berryman, J., Betz, R., Cai, Q., Cerutti, D., Cheatham Iii, T., Darden, T., Duke, R., Gohlke, H., et al.: Amber 14. (2014)
19. Lovell, S.C., Word, J.M., Richardson, J.S., Richardson, D.C.: The penultimate rotamer library. Proteins **40**(3) (August 2000) 389–408
20. Larrosa, J., de Givry, S., Heras, F., Zytnicki, M.: Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In: Proc. of the 19^{th} IJCAI, Edinburgh, Scotland (August 2005) 84–89
21. Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: ECAI. Volume 16. (2004) 146
22. Lecoutre, C., Saïs, L., Tabary, S., Vidal, V.: Reasoning from last conflict(s) in constraint programming. Artificial Intelligence **173** (2009) 1592,1614
23. Shapovalov, M.V., Dunbrack, R.L.: A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. Structure **19**(6) (2011) 844–858
24. Subramaniam, S., Senes, A.: Backbone dependency further improves side chain prediction efficiency in the energy-based conformer library (bebl). Proteins: Structure, Function, and Bioinformatics **82**(11) (2014) 3177–3187
25. Sang, T., Bacchus, F., Beame, P., Kautz, H.A., Pitassi, T.: Combining component caching and clause learning for effective model counting. SAT **4** (2004) 7th
26. Choi, A., Kisa, D., Darwiche, A.: Compiling probabilistic graphical models using sentential decision diagrams. In: Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Springer (2013) 121–132
27. Larrosa, J., Heras, F.: Resolution in max-sat and its relation to local consistency in weighted csps. In: IJCAI. (2005) 193–198