Guaranteed Weighted Counting for Affinity Computation: Beyond Determinism and Structure

Clément Viricel[1−2], David Simoncini[1], Sophie Barbe[2], and Thomas Schiex[1]

[1] MIAT, Université de Toulouse, INRA UR 875, Castanet-Tolosan, France
firstname.lastname@toulouse.inra.fr
[2] LISBP, Université de Toulouse, CNRS, INRA, INSA, Toulouse, France
firstname.lastname@insa-toulouse.fr

Abstract. Computing the constant Z that normalizes an arbitrary distribution into a probability distribution is a difficult problem that has applications in statistics, biophysics and probabilistic reasoning. In biophysics, it is a prerequisite for the computation of the binding affinity between two molecules, a central question for protein design.
In the case of a discrete stochastic Graphical Model, the problem of computing Z is equivalent to weighted model counting in SAT or CSP, known to be #P-complete [38]. SAT solvers have been used to accelerate guaranteed normalizing constant computation, leading to exact tools such as `cachet` [33], `ace` [8] or `minic2d` [28]. They exploit determinism in the stochastic model to prune during counting and the dependency structure of the model (partially captured by tree-width) to cache intermediary counts, trading time for space. When determinism or structure are not sufficient, we consider the idea of discarding sufficiently negligible contributions to Z to speedup counting. We test and compare this approach with other solvers providing deterministic guarantees on various benchmarks, including protein binding affinity computations, and show that it can provide important speedups.

1 Introduction

Graphical models [12] are sparse representations of highly dimensional multivariate distributions that rely on a factorization of the distribution in small factors. When variables are discrete, graphical models cover a variety of mathematical models that represent joint discrete distributions (or functions) that can be either Boolean functions (*e.g.,* in propositional satisfiability SAT and constraint satisfaction CSP), cost functions (as in partial weighted MaxSAT and Cost Function Networks) or probability distributions (in stochastic models such as Markov Random Fields and Bayesian networks).

Typical queries on such graphical models are either optimization or counting queries (or a mixture of these). In optimization queries, we look for an assignment that maximizes the joint function, *i.e.,* a model in SAT, a solution in CSP or a Maximum a posteriori assignment (MAP) in a Markov Random Field (MRF). All these problems have an associated NP-complete decision problem.

Counting problems are central in stochastic graphical models because they capture the computation of marginal probabilities on subsets of variables and the computation of

the normalizing constant Z that is required to define a probability distribution from the non-normalized distribution of Markov Random Fields. This difficult problem requires a summation over an exponential number of elementary terms and is #P-complete [38]. As shown by [37], one call to a #-P oracle suffices to solve any problem in the Meyer-Stockmeyer polynomial hierarchy in deterministic polynomial time, an indication that it could be outside of the PH.

Computing Z is a central problem in statistics (*e.g.,* for parameter estimation in MRFs), for Bayesian network processing (to account for evidence) and is also crucial in statistical physics where it is called the partition function. A typical domain where partition function computation can be extremely useful is computational protein design. Indeed, the affinity of a protein for a specific target molecule can be estimated by modeling both molecules as MRFs representing physics force fields and by computing the two partition functions: one for the bound protein and target and another for the same molecules in unbound state [35].

For these reasons, a large number of approaches have been designed to tackle this problem. The Mean-Field algorithm [17], Tree-reweighted Belief Propagation [40] as well as more recent proposals [25] have been proposed, but they do not offer any formal guarantee on the quality of the approximation they produce, except in very special cases. Monte-Carlo methods and more specifically Markov Chain Monte Carlo methods [16] offer asymptotic convergence proofs, but convergence is impractically slow. Indeed, there are recent significant examples showing that the time needed for Monte Carlo methods to converge can be easily under-estimated [36]. Practical MCMC based tools also rely on heuristics that destroy these theoretical guarantees. More recent stochastic methods exploiting universal hashing functions offer "Probably Approximately Correct" (PAC) estimators [14, 7]. Here, a bound δ on the probability that the estimation does not lie within a $(1+\varepsilon)$ ratio of the true value is set and a corresponding estimation produced.

Finally, different methods, mostly based on SAT-solvers, have been defined that can perform exact weighted model counting (#SAT) with deterministic guarantees, a problem to which the problem of computing Z can be easily reduced. To avoid the exponential blowup in the number of terms to add, solvers providing deterministic guarantees rely on two independent ideas: exploiting determinism (zero weights) to prune regions of the space that do not contribute to the sum, and exploiting independence which may be detected at the graphical model structure level, as captured by its tree-width, but also at a finer level as context-sensitive independence [33][3]. Independence enables caching of intermediate counts that can be factored out and lead to exponential time savings at the cost of memory. The very same ideas are also exploited in knowledge compilers that may compile graphical models or SAT formulas to languages on which counting becomes easy [8, 28].

In this paper, we explore the possibility of preserving the *deterministic* guarantees of exact solvers and explore a new source of pruning that may be present even when determinism or independence are too limited to allow for exact counting: detecting and pruning regions for which it is possible to prove, at limited cost, that they contain an

[3] They may also exploit the fact that counting the number of models of a valid formula is easy. This requires to check for validity, something that modern CDCL solvers do not do anymore.

amount of weight which is too small to significantly change the computed value of Z. Instead of providing a PAC guarantee, our algorithm provides an approximation of the normalizing constant that is guaranteed to lie within a ratio of $1 + \varepsilon$ of the true value (with probability 1), a guarantee that none of the PAC randomized algorithms above can provide in finite time.

Our initial motivation for computing Z lies in Computational Protein Design (CPD). The aim of CPD is to design new proteins that have desirable properties which are not available in the existing catalog of known proteins. One of these properties is the *affinity* between a protein and another molecule (such as another protein, a peptide, an amino-acid, a small organic molecule, etc...). The binding affinity gives an indication of the likelihood that two molecules will prefer to bind together rather than remain dissociated and thus that a protein will be likely to bind to another molecule of interest. Proteins can be described as a set of bound atoms subjected to a number of atom scale forces captured by a pairwise force field defining a Markov Random Field [29]. From this MRF, the binding affinity can be estimated by computing the ratio of the partition functions of the molecules in bound and unbound states [35, 15].

In the rest of the paper, after introducing our notations and the binding affinity computation problem, we present the Z^*_ε algorithm, a variant of Branch and Bound targeted at counting instead of optimizing. Z^*_ε relies on the availability of a local upper bound on Z. We then consider different simple, fast, safe and incremental upper bounds on Z, integrate them in Z^*_ε and compare them to exact counting tools on two categories of benchmarks: general benchmarks extracted from the UAI and Probabilistic Inference (PIC'2011) challenges and partition function computation problems appearing as sub-problems of binding affinity computation on real proteins. Surprisingly, despite a very limited caching strategy, the resulting algorithm is able to outperform exact solvers on a variety of problems and is especially efficient on CPD-derived problems.

2 Background

A Markov Random Field defines a joint probability distribution over a set of variables as a factorized product of local functions, usually denoted as potential functions.

Definition 21 *A discrete Markov Random Field (MRF) is a pair (X, Φ) where $X = \{1, \ldots, n\}$ is a set of n random variables, and Φ is a set of potential functions. Each variable $i \in X$ has a finite domain D^i of values that can be assigned to it. A potential function $\phi_S \in \Phi$, with scope $S \subseteq X$, is a function $\phi_S : D^S \mapsto \mathbb{R} \cup \{\infty\}$ where D^S denotes the Cartesian product of all D^i for $i \in S$.*

The energy or potential of an assignment $t \in D^X$ is denoted as $E(t) = \sum_{\phi_S \in \Phi} \phi_S(t[S])$ where $t[S]$ is the projection (or restriction) of t to the variables in S. Notice that this definition shows that an MRF is essentially equivalent to a Cost Function Network (or WCSP).

The probability of a tuple $t \in D^X$ is then defined as:

$$P(t) = \frac{\exp(-E(t))}{\sum_{t' \in D^X} \exp(-E(t'))}$$

The normalizing constant below the fraction is usually denoted as Z. The potential ϕ_S are called energies, in relation with statistical physics. An assignment with minimum energy has therefore maximum probability. With pairwise potentials ($|S| \leq 2$), an MRF defines a graph with variables as vertices and potential scopes S as edges. In the rest of this paper, for the mere sake of simplicity and w.l.o.g., we assume pairwise MRFs including also unary potential functions and a constant $\phi_\varnothing$ potential function. We denote by d the maximum domain size and e the number of pairwise potential functions. Using table representations, a pairwise MRF requires $O(ed^2)$ space to be represented.

Note that Bayesian networks can be seen as specific MRFs enforcing a local normalization condition of potentials and a specific DAG-base graph structure, that together guarantee that $Z = 1$. As soon as evidence (observations) change the domain of the variables however, Bayesian networks become unnormalized and computing Z becomes #P-complete in general.

2.1 Computational Protein Design and Binding Affinity

Proteins are linear chains of small molecules called "amino-acids". There are 20 natural different amino-acids. All amino-acids share a common core and the cores of all successive amino-acids in a proteins are linked together to form a linear chain, called the protein backbone. Each amino-acid also has a variable side-chain which chemical nature defined the precise amino-acid used. This lateral chain is highly flexible. The structure of a protein in 3D-space is therefore characterized by the shape of the linear chain itself (the backbone), and the specific spatial orientation of all side-chains, at each position of the chain. Proteins are universally present in the cells of all living organisms and perform a vast array of functions including catalyze, signaling, recognition, transporting, repair... Proteins differ from one another primarily in their sequence of amino-acids which usually results in protein folding into a specific 3D structure that determines its function. The characteristic of proteins that also allows their diverse set of functions is their ability to bind other molecules, with high affinity and specificity.

Proteins have a relatively stable general shape. The relative stability of a molecule in a given conformation can be evaluated by computing its energy, lower energy states being more stable. This energy is derived from various molecular forces including bond angles, electrostatic forces, molecular clashes and distances. It can be computed using existing force fields such as Amber [29], the one used in our experiments. Notice that molecular clashes – interpenetrating atoms – may generate infinite energies *i.e.,* determinism.

Despite a plethora of functionalities of proteins, there is still an ever-increasing demand for proteins endowed with specific properties of interest for many applications (in biotechnology, synthetic biology, green chemistry and nanotechnology) which either do not exist in nature or have yet not been found in the biodiversity. To this end, Computational structure-based Protein Design (CPD) has become a key technology. By combining physico-chemical models governing relations between protein amino-acid composition and protein 3D structure with advanced computational algorithms, CPD seeks to identify one or a set of amino-acid sequences that fold into a given 3D structure and possess the targeted properties. This *in silico* search for the best sequence candidates opens up new possibilities to better guide protein engineering by focusing exper-

imentation on the relevant sequence space for the desired protein function and thereby reducing the size of mutant libraries that need to be built and screened. In recent years, CPD has experienced important success, especially in the design of therapeutic proteins [27], novel enzymes [31], protein-protein interfaces [18, 32], and large oligomeric ensembles [19]. Nevertheless, the computational design of proteins with defined affinity for a given molecule (such as a small organic, a peptide, another protein...) which is essential for large range of applications, continues to present challenges.

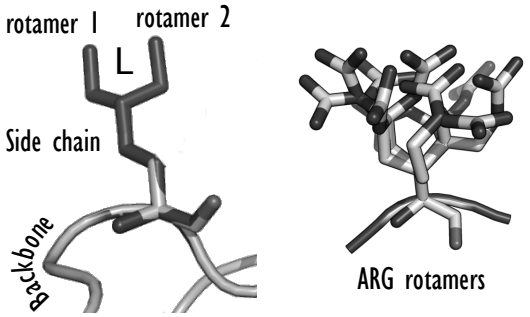

Fig. 1. A local view of a protein with a backbone and two acid side-chain reorientations (rotamers) for a given amino-acid (L = Leucine). A typical rotamer library for another amino-acid is shown on the right (ARG = Arginine).

A traditional approach to model proteins in CPD is to assume that their backbone is totally rigid and that only side-chains move, each side-chain being able to adopt a discrete set of most likely conformations defined in a so-called "rotamer" library (see Figure 1). We use the Penultimate rotamer library [26].

With one variable per side-chain, each with a domain equal to the set of available rotamers for this side-chain and a pairwise decomposable energy function such as Amber force field, a protein naturally defines a pairwise MRF with a rather dense graph. The partition function Z of this MRF captures important properties of the protein. Specifically, the association constant (or binding constant) is used to describe the affinity between a protein and a ligand (a protein or another molecule of interest). This association constant can be estimated by computing the partition function of the two molecules in bound and unbound states. The ratio of these two partition functions being proportional to their affinity.

From a computational point of view, an important property of proteins of interest is that their general shape is stable which means that the proportion of low energy (or high probability) states among the exponential number of possible states is likely to be very small. On the opposite side of the energy scale, the infinite energies created by molecular clashes means that there will be states with 0 probability. This is favorable for exact solvers that can exploit determinism to speedup Z computation. It however means that CPD instances will exhibit unbounded *tilt* (defined in [7] as the

ratio $\tau = \frac{\max_{t \in D^X} P(t)}{\min_{t \in D^X} P(t)}$). This situation is not ideal for the WeightMC PAC algorithm which requires a finite upper-bound on τ to run in finite time.

3 Guaranteed counting

Because it is rarely (if ever) needed to compute a probability or a partition function with an absolute precision (which is also inherently limited by finite representations), we consider the general problem of computing an ε- approximation $\hat{Z}$ of Z, *i.e.,* such that:

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

Such approximation allows us to compute an estimate $\hat{P}(t) = \frac{\exp(-E(t))}{\hat{Z}}$ such that $P(t) \leq \hat{P(t)} \leq (1+\varepsilon)P(t)$. In the context of #-SAT, it has been shown that providing such relative approximations remains intractable for most of the known SAT polynomial classes [30]. As we will see, it can however be exploited to prune during polynomial space depth-first tree-search based counting and sometimes provide important speedups.

Assuming that for any MRF, and any assignment t of *some* of its variables, we can compute an upper bound $Ub(t)$ of the partition function of the MRF where variables are assigned as in t, the Depth First Branch and Bound schema used for exactly solving optimization problems on cost function networks [9, 1] can be adapted to compute Z [39].

```
   Function Z*_ε(t, V)
1      if V = ∅ then
2          Ẑ ← Ẑ + exp(−E(t));
       else
3          Choose i ∈ V;
           for a ∈ D^i do
               t' ← t ∪ {(i, a)};
4              if (U + Ub(t') + Ẑ ≤ (1 + ε)Ẑ) then
5                  U ← U + Ub(t');
               else
                   Z*_ε(t', V − {i});
```

Algorithm 1: Guaranteed approximate counting. Initial call: $Z^*_\varepsilon(\varnothing, X)$. U and $\hat{Z}$ are global variables initialized to 0.

The algorithm simply explores the tree of all possible assignments of the MRF, starting with the whole set of unassigned variables (in V), choosing an unassigned variable (line 3), trying all possible values. When all variables are assigned (line 1),

the contribution of the complete assignment t is accumulated in a running count which will eventually define the approximation $\hat{Z}$ (line 2). However, branches which provide a sufficiently small mass of probability (as estimated by $Ub(t')$) are pruned and this overestimation of the neglected mass is accumulated in U (line 5). Because pruning may occur, eventually, $\hat{Z}$ will be a lower bound of Z.

Theorem 1. *Z^*_ε terminates and returns an ε-approximation of Z.*

Proof. The termination follows from the fact that Z^*_ε explores a finite tree. We now show that the algorithm always provides a ε-approximation. When the algorithm finishes, all the assignments have either been explored (line 2) and counted or pruned (line 5). Since U is the sum of all the upper bounds on the mass of probability in all pruned branches, we have that $\hat{Z} + U \geq Z$. Initially, $\hat{Z} = U = 0$ and the invariant $\hat{Z} \geq \frac{\hat{Z}+U}{1+\varepsilon}$ holds. The test at line 4 guarantees that this invariant still holds at the end of the algorithm. Therefore $\hat{Z} \geq \frac{Z}{(1+\varepsilon)}$. □

While inspired by Depth First Branch and Bound (DFBB) that provides polynomial space complexity, this algorithm behaves differently from it. In DFBB, for a fixed order of exploration, when the local bound used for pruning (here $Ub(t)$) is tighter, less nodes are explored. This property is lost in Z^*_ε. Indeed, it is easy to imagine a scenario where a tight bound $Ub(t)$ will lead to more nodes being explored than using a weaker $Ub'(t)$: imagine that search has started and collected a mass $\hat{Z} = 1$ and $U = 0$ for either bounds. Then comes a subtree of small size for which $Ub(t) = \varepsilon$ while $Ub'(t) \gg \varepsilon$. This subtree will be pruned by $Ub(t)$ leading to $U = \varepsilon$ but instead will be enumerated with $Ub'(t)$ preserving $U = 0$. In this context, the algorithm using the tight $Ub(t)$ is not allowed to prune anymore in the immediate future: if the forthcoming leaves all have very small probability mass, it will be forced to visit all of them while the algorithm using $Ub'(t)$ preserved some margin and may be able to skip a significant fraction of them.

Indeed, similarly to what happens with the α-β algorithm [20], the order in which leaves are explored may have a major effect on the algorithm efficiency. Let us assume that we have a perfect $Ub(t)$ and that the leaves of the tree have exponentially decreasing mass of probability, the i^{th} visited leaf having a mass of $\varepsilon^{i-1}, \varepsilon < 1$ (such an extreme distribution of probability mass may seem unlikely, but corresponds to linearly increasing energies). In this case, the first leaf bears more mass than all the rest of the tree and the Z^*_ε algorithm would visit just one leaf. If the inverse ordering of leaves is assumed, the algorithm will have to explore all leaves. It seems therefore important to collect highest masses first. The polynomial space complexity of DFBB comes however with strict constraints on the order of exploration of leaves and best-first algorithms that could overcome this restriction would lead to worst-case exponential space complexity. Interesting future work would be to use the recent highly flexible any-space Branch and Bound algorithm HBFS [2] to improve the leaf ordering within bounded space.

However, contrary to what happens with optimization, even an exact upper bound and a perfect ordering does not guarantee that only one leaf needs to be explored. If we instead assume a totally flat energy landscape, with all leaves having the same energy, Z^*_ε will have to explore a $\frac{1}{1+\varepsilon}$ fraction of the leaves just to accumulate enough mass in $\hat{Z}$ to prune.

Overall, it is important to realize that Z^*_ε needs to achieve two goals:

1. collect probability masses on a potentially very large number of complete assignments to compute a suitable approximation
2. exploit its upper bound to prune the largest possible part of the tree

The first goal could be achieved by existing algorithms producing an exhaustive list of the m-best assignments [13] or all assignment within a threshold of the optimum (a service that any DFBB-based optimization system provides for free). These algorithms use bounds on the *maximum* probability instead of the total probability mass which leads to stronger pruning and potentially higher efficiency than Z^*_ε but do not provide any guarantee since the number m of assignments that would need to be enumerated to provide ε-approximation is unknown.

Because a potentially very large number of probability masses need to be collected, a very fast search is required. To accelerate it, we equip Z^*_ε with a very simple form of "on the fly" caching: at any node during the search, we eliminate any variable which is either assigned or of bounded degree as proposed initially for optimization [22], but using sum-product variable elimination [11]. This caches all the influence of the eliminated variable in a temporary (trailed) potential function. This means that the leaves of the search tree will be sub-problems with bounded tree-width that may represent an exponential number of assignments. This naturally makes Z^*_ε related to the `vec` weighted counting algorithm, an anytime MRF counter based on w-cutsets (vertex cutset which if assigned leave a w-tree) and variable elimination over w-trees [11].

The second goal is to prune the largest possible part of the tree search. However, since the first goal requires a very fast search algorithm, using a powerful but computationally expensive bound is probably doomed to fail. For this reason, we have considered simple fast incrementally updated upper bounds by borrowing recent optimization bounds [9] which are known to work well in conjunction with Depth First Search.

3.1 Bounds for guaranteed counting

For any MRF, we define a first upper bound on Z denoted by Ub_1.

$$Z \leq Ub_1 = \left(\prod_{\phi_S, |S|<2} \sum_{t \in D^S} \exp\left(-\phi_S(t)\right) \right) \cdot \left(\prod_{\phi_S, |S|\geq 2} \exp\left(- \min_{t \in D^S} \phi_S(t) \right) \right)$$

Proof. By definition, we have that

$$Z = \sum_{t \in D^X} \left(\prod_{\phi_S, |S|<2} \exp\left(-\phi_S(t)\right) \cdot \prod_{\phi_S, |S|\geq 2} \exp\left(-\phi_S(t)\right) \right)$$

Trivially, $\exp\left(-\phi_S(t)\right) \leq \max_{t \in D^S} \left(\exp\left(-\phi_S(t)\right)\right) = \exp\left(-\min_{t \in D^S} \phi_S(t)\right)$ (by monotonicity). Applying this to the right term above, and exploiting the fact that this term now does not depend on t, we get that:

$$Z \leq \left(\sum_{t \in D^X} \prod_{\phi_S, |S|<2} \exp\left(-\phi_S(t)\right) \right) \cdot \left(\prod_{\phi_S, |S| \geq 2} \exp\left(- \min_{t \in D^S} \phi_S(t) \right) \right)$$

Since the set $\{\phi_S, |S| < 2\}$ contains only unary or constant functions, distributivity allows to swap sum and product and the result follows. Notice that this bound can be computed in linear time. □

This bound can be strengthened by selecting a subset of all pairwise potentials in Φ defining a partial spanning k-tree $T \subset \Phi$. By applying a sum-product non serial dynamic programming [11] on $T' = T \cup \{\phi_S \in \Phi : |S| < 2\}$, we can obtain the exact $Z_{T'}$ for this sub-MRF in polynomial time. We can multiply $Z_{T'}$ by $(\prod_{\phi_S \in \Phi \setminus T'} \exp(-\min_{t \in D^S} \phi_S(t)))$ and get a tighter upper bound on Z which we denote Ub_T:

$$Z \leq Ub_T = \underbrace{\left(\sum_{t \in D^X} \prod_{\phi_S \in T'} \exp\left(-\phi_S(t)\right) \right)}_{\text{Computed using non serial dynamic programming}} \cdot \left(\prod_{\phi_S \in \Phi \setminus T'} \exp\left(- \min_{t \in D^S} \phi_S(t) \right) \right)$$

Proof. The proof is essentially similar to the previous one, and obtained by just replacing the set $\{\phi_S, |S| < 2\}$ and its complement set $\{\phi_S, |S| \geq 2\}$, defining the ranges of the products by the sets $\{\phi_S \in T'\}$ (and its complement respectively). The first item can be simply computed in $O(nd^2)$ time using non serial dynamic programming. □

These bounds alone are very weak. To further strengthen them, we reformulate the MRF using soft arc-consistencies [9] on its energy representation [1]. Soft arc consistencies essentially shift energy from pairwise potential functions to unary potential functions and eventually to the constant potential function $\phi_\varnothing$ while preserving equivalence. The result of this is an equivalent MRF (defining the same distribution) with increased unary and constant $\phi_\varnothing$ potential functions and pairwise potential functions that satisfy $\min_{t \in D^S} \phi_S(t) = 0$. Besides strengthening the bounds, it removes the need to compute the right term which is always equal to 1. Ub_1 and Ub_T can then be computed in $O(nd)$ and $O(nd^2)$ instead of $O(ed^2)$.

In the rest of the paper we consider spanning trees and try both Existential Directional Arc Consistency (EDAC) and Virtual Arc Consistency (VAC) [9] as possible ways of strengthening Ub_1 and Ub_T.

4 Experimental evaluation and comparison

To evaluate the ability of the Z^*_ε algorithm to provide guaranteed deterministic approximations to Z, we implemented it on the top of the open source `toulbar2` solver[4].

[4] `http://www.inra.fr/mia/T/toulbar2`

The variable and value ordering used are the default weighted-degree and last conflict variable ordering and the existential support value-ordering [9]. We enforce EDAC at the root node and during search as usual for optimization. When VAC is used, it is only enforced at the root node because of its computational cost. Instead of the k-way branching described in Algorithm 1, we use a binary branching that either includes or reject a chosen value a at each branching decision. At each node, all variables of degree ≤ 2 are eliminated. The upper bound Ub_T uses a fixed maximum spanning tree (a 1-tree) with maximum sum of mean cost after enforcing arc consistencies at the root node. Our implementation is limited to pairwise potentials.

We compared it to different exact weighted counting approaches in terms of efficiency and quality of our guaranteed approximation. Four different exact counters have been considered. The first one is the already described `vec` exact counter [11]. The second one is the exact SAT based weighted counting tool `cachet` [33][5]. `cachet` relies internally on the Zchaff SAT solver to enumerate models with non zero weight and uses context-sensitive independence to cache intermediate counts. We also used the `ace` 3.0 compiler [8], using the UAI competition executable provided in the `ace` distribution (always using a pseudo-random generator seed of 0). `ace` computes a tree decomposition and based on the obtained width may either perform tabular variable elimination or encode to CNF and compile in d-DNNF using `c2d`. We also tested the recent `minic2d` Sentential Decision Diagram (SDD) compilation package [28]. SDD are more constrained than d-DNNF and may therefore lead to larger compiled forms than d-DNNF, but since we do not need a compiled form and just the value of Z, we used the `-W` option of `minic2d` that performs weighted counting without compilation hoping to trade space for time. `minic2d` relies on its own internal SAT solver which is provided as a compiled binary in the distributed `minic2d` package. Because some of the compared solvers (`vec`, `cachet`) provide only a double floating representation of Z (or its logarithm), all software has been used in double floating point mode.

All executions have been performed on one core of an Intel® Xeon® CPU E5-2680 v3 @ 2.50GHz (a Q4 2014 cpu) with a limit of 60 GB on RAM usage.

4.1 MRF to #SAT encoding

If `ace` uses its own internal optimized MRF to SAT encoding, both `cachet` and `minic2d` require specific SAT encoding. Exact #SAT weighted counters use weighted literals and define the weight of a model as the product of the weights of all literals which are true in the model. They therefore rely on multiplicative potentials $\exp(-\phi_S(t))$. To transform an MRF into a literal-weighted CNF formula with a weighted count equal to the partition function, we use the ENC1 encoding of [8], originally described in [10]. This encoding is the CNF version of the so-called *local polytope*-based ILP encoding introduced in [34] for MRFs and [21] for weighted CSPs [9]. For each variable $i \in X$, we use one proposition $d_{i,r}$ for each value $r \in D^i$. This proposition is true iff variable i is assigned the value r. We encode *At Most One* (AMO) with hard clauses $(\neg d_{i,r} \vee \neg d_{i,s})$ for all $i \in X$ and all $r < s$, $r, s \in D^i$, as well as *At Least One* (f) with one hard

[5] We thank Jean-Marie Lagniez, CRIL, France for providing us with a patched version of cachet that can be compiled and run without any issue on recent systems.

clause ($\bigvee_r d_{i,r}$) for each i. These clauses ensure that the propositional encoding allows exactly one value for each variable in each model. For each potential ϕ_S, and each tuple $t \in D^S$, we have a propositional variable $p_{S,t}$. For non-zero energies $\phi_S(t)$, we have the literal $p_{S,t}$ with weight $\exp(-\phi_S(t))$. This represents the multiplicative potential to use if the tuple t is used. $\neg p_{S,t}$ is instead weighted by 1, the identity for multiplication. For every variable $i \in S$, we have a hard clause $(d_{i,t[i]} \vee \neg p_{S,t})$. These clauses enforce that if tuple t is used, its values $t[i]$ must be used. Then, for each variable $i \in S$ and each value $r \in D^i$, we have hard clauses $(\neg d_{i,r} \vee \bigvee_{t \in D^S, t[i]=r} p_{S,t})$ that enforces that if a value $r \in D^i$ is used, one of the allowed tuples $t \in D^S$ such that $t[i] = r, w_S(t) < k$ must be used.

It is interesting to notice that for pure Constraint Satisfaction Problems (MRFs having only $0/\infty$ potentials), it is known that Unit Propagation (UP) on this encoding enforces arc consistency in the original CSP [3].

We apply obvious optimization steps, explicitly forbidding local assignments with zero mass (sources of determinism). This encoding can be directly fed into `minic2d`. Large problems however could not be encoded because `minic2d` only allows to express weights in a one-line list of maximum 100,000 chars in length[6].

In `cachet`, weighted literals l are either such that l and $\bar{l}$ receive a mass of 1 that has no effect on final mass, or such that the weights of a variable and its negation sum to 1. This is sufficient and convenient to express Bayesian nets because of their local normalization constraint. For arbitrary MRFs, for every $p_{S,t}$ corresponding to a mass $m = \exp(-\phi_S(t))$ we introduce another propositional variable $n_{S,t}$ with weights m (positive) and $1 - m$ (negative) and a simple implication clause $p_{S,t} \rightarrow n_{S,t}$. This extra variable is connected to the rest of the problem only through this clause and can therefore easily be eliminated, leading to a multiplicative factor m in models where $p_{S,t}$ is true and $1 = m + (1 - m)$ in models where $p_{S,t}$ is false, as required.

4.2 Benchmarks

Two types of benchmarks have been used. The first type of benchmark is made of instances of partition function computation appearing as sub-problems of binding affinity computations on molecular systems defined by a protein interacting with a peptide or an amino-acid. 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. Missing heavy atoms in crystal structures as well as hydrogen atoms were added using the *tleap* module of the *Amber 14* software package [6]. The molecular all-atom *ff14SB* force field was used for the proteins and the ligands (peptides and amino-acids). The molecular systems were then subjected to 1000 steps of energy minimization with the *Sander* module of *Amber 14*. Next, a portion of the proteins including amino-acids at the interface between the protein and the ligand as well as surrounding amino-acids with at least one atom within 8 to 12 Å (according to the molecular system) of the interface was selected.[7]

[6] This parameter could not be changed, being in the non open-source part on `minic2d`.

[7] Each of these systems requires extensive molecular modeling expertise to be properly defined. We intend to make this benchmark together with the Z_ε^* implementation available.

To evaluate the effect of the strength of the upper bound on the algorithm efficiency, we applied Z^*_ε with $\varepsilon = 10^{-3}$ on a series of 349 systems using the four different bounds. For each system, the most complex partition function, defined on the compound system, is computed. The largest problem has 22 variables and the largest domain size is 34. The gap between our two bounds and the guaranteed approximation of Z determined by Z^*_ε is shown in Figure 2. The bound Ub_T is clearly stronger than Ub_1, as expected.

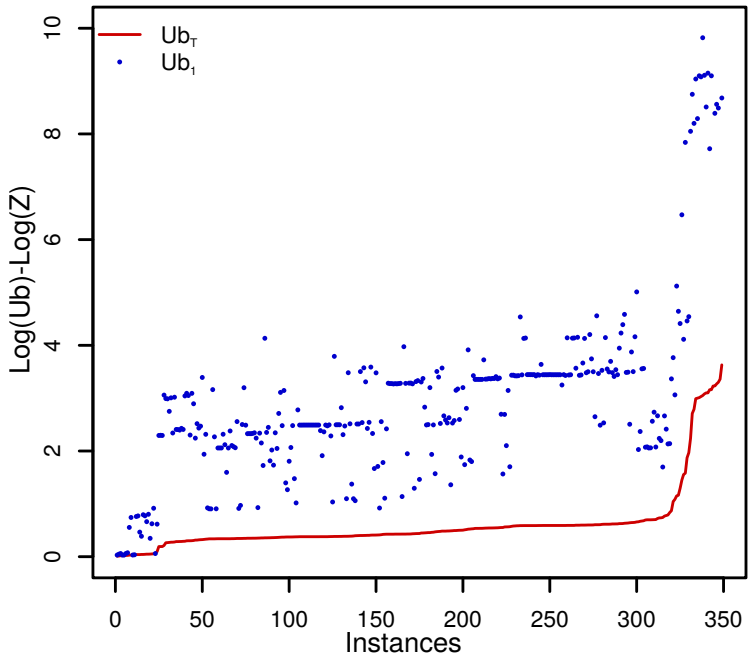

Fig. 2. Gap to Z: we represent $\log(Ub) - \log(\hat{Z})$ at the root node for Ub_1 and Ub_T using EDAC tightening (only negligible difference with VAC on these problems). The instances on the x axis are sorted in increasing gap size for the strongest Ub_T bound.

We then compare the running times of Z^*_ε using these 4 bounds in a cactus plot in Figure 3. The best bound in terms of run-time is the lightest Ub_1+EDAC bound confirming that stronger, thus more expensive, bounds may quickly become counter productive.

We represent the same information with the fastest Ub_1+EDAC and two of the three exact counting tools in Figure 4. We omit `ace` and `minic2d`. Indeed, `ace` was able to solve only 17 problems within the time limit and failed on all remaining problems with a memory exception (despite the explicit allocation of 60GB to the JAVA machine). `minic2d` was instead unable to model 294 systems out of the 349 because of its previously mentioned limitation on the length of the weight line. On the remaining 55 problems, `minic2d` solved 7 problems in less than one hour.

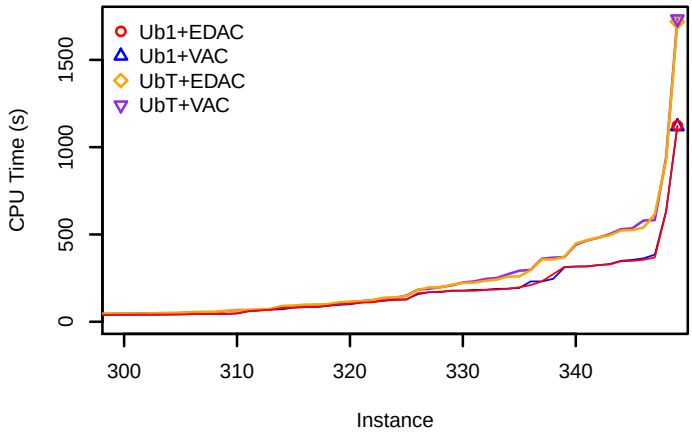

Fig. 3. Cactus plot of the running times of Z_ε^* with our four bounds. A point at (x, y) indicates that the number of solved problems is x if a deadline of y seconds is imposed on each resolution.

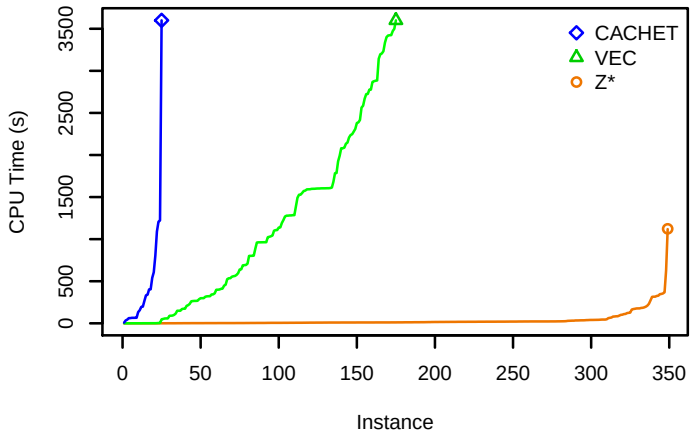

Fig. 4. Cactus plot of the running times of Z_ε^* with the Ub_1+EDAC bound together with `cachet` and `vec`. A point at (x, y) indicates that the number of solved problems is x if a deadline of y seconds is imposed on each resolution.

In the rest of the experiments we therefore use the Ub_1+EDAC upper bound which seems the most efficient bound. To see how the Z_ε^* algorithm performs on other types of problems, we used instances extracted from UAI and PIC'2011 challenge instances (PR task)[8] that use only pairwise potentials as a second set of benchmark. Using the same value of $\varepsilon = 10^{-3}$, we again compared Z_ε^* with `vec`, `cachet`, `ace` and `minic2d`.

The results clearly show there is no single winner: except for `cachet` which is always dominated by one of the other solvers, each algorithm may outperform others. Specifically, the Z_ε^* algorithm, despite its lack of sophisticated caching technology, is able to outperform its competitors in various cases. Nevertheless, `minic2d` outperformed Z_ε^* on the Grid category (probably because of the combination of Boolean variables and relatively small treewidth), itself outperformed by `vec` and further outperformed by `ace`.

Instance	Z_ε^*	`minic2d`	`ace`	`vec`	`cachet`
smokers_10	$\mathbf{< 0.01}$	0.663	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	0.264
smokers_20	$\mathbf{< 0.01}$	312.825	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	332.168
rbm_20	7.46	T	**3.376**	16.17	941.14
rbm_ferro_20	$\mathbf{< 0.01}$	T	3.24	16.18	893.84
rbm_21	13.75	T	**6.854**	34.35	2041.64
rbm_ferro_21	$\mathbf{< 0.01}$	T	6.868	34.17	1975.78
rbm_22	33.75	T	**14.411**	72.78	T
rbm_ferro_22	$\mathbf{< 0.01}$	T	14.418	72.43	T
grid10x10.f10	66.32	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	T
grid20x20.f10	T	T	**13.316**	2104.57	T
grid20x20.f15	T	T	**13.665**	2099.24	T
grid20x20.f2	T	T	**13.603**	2107.92	T
grid20x20.f5	T	T	**13.609**	2102.32	T
GEOM30a_3	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	2.668
GEOM30a_4	7.66	78.604	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	43.77
GEOM30a_5	67.48	368.361	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	405.38
GEOM40_2	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$
GEOM40_3	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	3.62
GEOM40_4	1.42	0.58	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	616.59
GEOM40_5	12.67	12.801	$\mathbf{< 0.01}$	$\mathbf{< 0.01}$	828.50
myciel5g_3	**37.2**	T	M	T	T
myciel5g_4	M	T	M	T	T
myciel5g_5	M	T	M	T	T
queen5_5_3	367.2	T	**83.004**	945.20	T
queen5_5_4	**2423.67**	T	M	T	T

Table 1. Time results for UAI/PIC'2011 instances. Three different categories are represented: Boltzmann machines (rbm) with attractive (ferro) and non attractive coupling, Grids, and graph problems. Running-times are given in seconds. M: Memory Out ($60GB$), T: Time out ($1h$). Bold is best.

[8] http://www.cs.huji.ac.il/project/PASCAL

Conclusion

Existing solvers providing deterministic guarantees for partition function computation exploit two sources of efficiency. This first one is caching of local counts based on context-sensitive independence [33], related to tree-decomposition. The other one is determinism *i.e.,* , the existence of zero probability assignments allowing to prune zero probability mass sub-trees during search. This second source of efficiency will provide significant speedups only when a significant fraction of the search space has 0 probability. Such distributions have very low entropy.

In this paper, motivated by the computation of statistical estimate of affinity between bio-molecules, we have proposed to build upon existing optimization technology to provide a new source of pruning for partition function computation with deterministic guarantees: a branch and bound-based schema equipped with upper bounds derived from soft local consistencies. As existing SAT-based exact approaches, our algorithm exploits determinism and a much simpler and less powerful form of caching than those based on tree-decomposition. It is however able to prune regions of proven negligible mass of probability and is therefore able to exploit relatively low entropy distributions having a much wider support, including those with no determinism. The resulting algorithm offers an adjustable deterministic guarantee on the quality of the computed partition function and, despite its limited caching strategy, may already offer interesting speedups compared to exact solvers.

Z^* includes two crucial ingredients to quickly gather large number of probability masses: pruning based on very fast incremental upper bounds derived from optimization bounds and on-the-fly sum-prod elimination. An important point is that these ingredients can be easily injected into existing SAT-based counters, including knowledge-compilation based counters using SAT-solver traces. This could be achieved by defining counting upper bounds from existing Max-SAT bounds. These bounds have already been related to soft arc-consistency bounds [23, 4, 24]. This should extend their range of application to guaranteed approximate probabilistic inference on problems with limited or no determinism.

From an affinity computation point of view, the next step is now to evaluate the actual empirical quality of the association constant estimation provided by the computed ratio of partition functions. Beyond algorithmic approximations, the modeling may also have important effects on the estimated value based on different rotamer discretizations, relative positions of molecules in the complex or weights of different contributions in the energy function. To pursue this target, we intend to use available databases that provide experimental values of the association constant of various protein-ligand complexes following various mutations on one of the partners. To keep the modeling to a reasonable level of complexity, this will be preferably achieved on protein-protein complexes.

Acknowledgments : We would like to thank Simon de Givry for his help with `toulbar2`. We thank the Computing Center of Region Midi-Pyrénées (CALMIP, Toulouse, France) and the Genotoul Bioinformatics Platform of INRA-Toulouse for providing computing resources and support. C. Viricel was supported by a grant from INRA and Region Midi-Pyrénées.

References

1. 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. Artif. Intell. 212, 59–79 (2014)
2. Allouche, D., De Givry, S., Katsirelos, G., Schiex, T., Zytnicki, M.: Anytime hybrid best-first search with tree decomposition for weighted csp. In: Principles and Practice of Constraint Programming. pp. 12–29. Springer (2015)
3. Bacchus, F.: GAC via unit propagation. In: Proc. of CP. pp. 133–147 (2007)
4. Bonet, M.L., Levy, J., Manyà, F.: Resolution for max-sat. Artificial Intelligence 171(8), 606–618 (2007)
5. Campeotto, F., Dal Palù, A., Dovier, A., Fioretto, F., Pontelli, E.: A constraint solver for flexible protein models. Science 253(5016), 164–170 (1991)
6. 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)
7. Chakraborty, S., Fremont, D.J., Meel, K.S., Seshia, S.A., Vardi, M.Y.: Distribution-aware sampling and weighted model counting for sat. In: Proc. 28^{th} Conference on Artificial Intelligence. p. 1722–1730 (2014)
8. Chavira, M., Darwiche, A.: On probabilistic inference by weighted model counting. Artificial Intelligence 172(6), 772–799 (2008)
9. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence 174, 449–478 (2010)
10. Darwiche, A.: A logical approach to factoring belief networks. KR 2, 409–420 (2002)
11. Dechter, R.: Bucket elimination: A unifying framework for reasoning. Artificial Intelligence 113(1–2), 41–85 (1999)
12. Dechter, R.: Reasoning with probabilistic and deterministic graphical models: Exact algorithms. Synthesis Lectures on Artificial Intelligence and Machine Learning 7(3), 1–191 (2013)
13. Dechter, R., Flerova, N., Marinescu, R.: Search algorithms for m best solutions for graphical models. In: AAAI. Citeseer (2012)
14. Ermon, S., Gomes, C.P., Sabharwal, A., Selman, B.: Taming the curse of dimensionality: Discrete integration by hashing and optimization. arXiv preprint arXiv:1302.6677 (2013)
15. 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. J. Comput. Chem. 29(10), 1527–42 (Jul 2008)
16. Gilks, W.R.: Markov chain monte carlo. Wiley Online Library (2005)
17. Jaakkola, T.S.: Tutorial on variational approximation methods. Advanced mean field methods: theory and practice p. 129 (2001)
18. Karanicolas, J., Kuhlman, B.: Computational design of affinity and specificity at protein–protein interfaces. Curr. Opin. Struct. Biol. 19(4), 458–463 (2009)
19. King, N.P., Bale, J.B., Sheffler, W., McNamara, D.E., Gonen, S., Gonen, T., Yeates, T.O., Baker, D.: Accurate design of co-assembling multi-component protein nanomaterials. Nature 510(7503), 103–108 (2014)
20. Knuth, D.E., Moore, R.W.: An analysis of alpha-beta pruning. Artificial intelligence 6(4), 293–326 (1976)
21. Koster, A.: Frequency assignment: Models and Algorithms. Ph.D. thesis (1999)
22. Larrosa, J.: Boosting search with variable elimination. In: Principles and Practice of Constraint Programming - CP 2000. LNCS, vol. 1894, pp. 291–305. Singapore (Sep 2000)
23. Larrosa, J., Heras, F.: Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In: Proc. of the 19^{th} IJCAI. pp. 193–198. Edinburgh, Scotland (2005)

24. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient max-sat solving. Artif. Intell. 172(2-3), 204–233 (2008)
25. Liu, Q., Ihler, A.T.: Bounding the partition function using holder's inequality. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 849–856 (2011)
26. Lovell, S.C., Word, J.M., Richardson, J.S., Richardson, D.C.: The penultimate rotamer library. Proteins 40(3), 389–408 (Aug 2000), http://www.ncbi.nlm.nih.gov/pubmed/10861930
27. 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), 449–455 (2012)
28. Oztok, U., Darwiche, A.: A top-down compiler for sentential decision diagrams. In: Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press (2015)
29. Pearlman, D.A., Case, D.A., Caldwell, J.W., Ross, W.S., Cheatham, T.E., DeBolt, S., Ferguson, D., Seibel, G., Kollman, P.: Amber, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. Computer Physics Communications 91(1), 1–41 (1995)
30. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(1), 273–302 (1996)
31. Röthlisberger, D., Khersonsky, O., Wollacott, A.M., Jiang, L., DeChancie, J., Betker, J., Gallaher, J.L., Althoff, E.a., Zanghellini, A., Dym, O., Albeck, S., Houk, K.N., Tawfik, D.S., Baker, D.: Kemp elimination catalysts by computational enzyme design. Nature 453(7192), 190–5 (May 2008), http://www.ncbi.nlm.nih.gov/pubmed/18354394
32. Sammond, D.W., Eletr, Z.M., Purbeck, C., Kuhlman, B.: Computational design of second-site suppressor mutations at protein–protein interfaces. Proteins: Structure, Function, and Bioinformatics 78(4), 1055–1065 (2010)
33. Sang, T., Beame, P., Kautz, H.: Solving bayesian networks by weighted model counting. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05). vol. 1, pp. 475–482 (2005)
34. Schlesinger, M.: Syntactic analysis of two-dimensional visual signals in noisy conditions. Kibernetika 4, 113–130 (1976)
35. Silver, N.W., King, B.M., Nalam, M.N., Cao, H., Ali, A., Kiran Kumar Reddy, G., Rana, T.M., Schiffer, C.A., Tidor, B.: Efficient computation of small-molecule configurational binding entropy and free energy changes by ensemble enumeration. J. Chem. Theory Comput. 9(11), 5098–5115 (2013)
36. Simoncini, D., Allouche, D., de Givry, S., Delmas, C., Barbe, S., Schiex, T.: Guaranteed discrete energy optimization on large protein design problems. Journal of chemical theory and computation 11(12), 5980–5989 (2015)
37. Toda, S.: On the computational power of pp and $\oplus$p. In: Foundations of Computer Science, 1989., 30th Annual Symposium on. pp. 514–519. IEEE (1989)
38. Valiant, L.G.: The complexity of computing the permanent. Theoretical computer science 8(2), 189–201 (1979)
39. 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. pp. 165–176. Springer (2015)
40. Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: A new class of upper bounds on the log partition function. Information Theory, IEEE Transactions on 51(7), 2313–2335 (2005)