Approximation via Correlation Decay When Strong Spatial Mixing Fails*†‡§

Ivona Bezáková[1], Andreas Galanis[2], Leslie Ann Goldberg[3], Heng Guo[4], and Daniel Štefankovič[5]

1 **Rochester Institute of Technology, Rochester, NY, USA**
2 **University of Oxford, Oxford, UK**
3 **University of Oxford, Oxford, UK**
4 **Queen Mary University of London, London, UK**
5 **University of Rochester, Rochester, NY, USA**

Abstract

Approximate counting via correlation decay is the core algorithmic technique used in the sharp delineation of the computational phase transition that arises in the approximation of the partition function of anti-ferromagnetic two-spin models.

Previous analyses of correlation-decay algorithms implicitly depended on the occurrence of *strong spatial mixing*. This, roughly, means that one uses worst-case analysis of the recursive procedure that creates the sub-instances. In this paper, we develop a new analysis method that is more refined than the worst-case analysis. We take the shape of instances in the computation tree into consideration and we amortise against certain "bad" instances that are created as the recursion proceeds. This enables us to show correlation decay and to obtain an FPTAS even when strong spatial mixing fails.

We apply our technique to the problem of approximately counting independent sets in hypergraphs with degree upper-bound Δ and with a lower bound k on the arity of hyperedges. Liu and Lin gave an FPTAS for $k \geq 2$ and $\Delta \leq 5$ (lack of strong spatial mixing was the obstacle preventing this algorithm from being generalised to $\Delta = 6$). Our technique gives a tight result for $\Delta = 6$, showing that there is an FPTAS for $k \geq 3$ and $\Delta \leq 6$. The best previously-known approximation scheme for $\Delta = 6$ is the Markov-chain simulation based FPRAS of Bordewich, Dyer and Karpinski, which only works for $k \geq 8$.

Our technique also applies for larger values of k, giving an FPTAS for $k \geq 1.66\Delta$. This bound is not as strong as existing randomised results, for technical reasons that are discussed in the paper. Nevertheless, it gives the first deterministic approximation schemes in this regime. We further demonstrate that in the hypergraph independent set model, approximating the partition function is NP-hard even within the uniqueness regime.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases approximate counting, independent sets in hypergraphs, correlation decay

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.45

* The full version of the paper is available at `arxiv.org/abs/1510.09193`. The theorem numbering here matches the full version.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

‡ Research supported by NSF grants CCF-0910415 and CCF-1319987.

§ This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

We develop a new method for analysing correlation decays in spin systems. In particular, we take the shape of instances in the computation tree into consideration and we amortise against certain "bad" instances that are created as the recursion proceeds. This enables us to show correlation decay and to obtain an FPTAS even when strong spatial mixing fails. To the best of our knowledge, strong spatial mixing is a requirement for all previous correlation-decay based algorithms. To illustrate our technique, we focus on the computational complexity of approximately counting independent sets in *hypergraphs*, or equivalently on counting the satisfying assignments of monotone CNF formulas.

The problem of counting independent sets in *graphs* (denoted #IS) is extensively studied. A beautiful connection has been established, showing that approximately counting independent sets in graphs of maximum degree Δ undergoes a computational transition which coincides with the uniqueness phase transition from statistical physics on the infinite Δ-regular tree. The computational transition can be described as follows. Weitz [17] designed an FPTAS for counting independent sets on graphs with maximum degree at most $\Delta = 5$. On the other hand, Sly [15] proved that there is no FPRAS for approximately counting independent sets on graphs with maximum degree at most $\Delta = 6$ (unless NP = RP). The same connection has been established in the more general context of approximating the partition function of the hard-core model [17, 12, 15, 4, 5, 16] and in the even broader context of approximating the partition functions of generic antiferromagnetic 2-spin models [14, 5, 16, 8] (which includes, for example, the antiferromagnetic Ising model). As a consequence, the boundary for the existence of efficient approximation algorithms for these models has been mapped out.

Approximate counting via correlation decay is the core technique in the algorithmic developments which enabled the sharp delineation of the computational phase transition. Another standard approach for approximate counting, namely Markov Chains Monte Carlo (MCMC) simulation, is also conjectured to work up to the uniqueness threshold, but the current analysis tools that we have do not seem to be powerful enough to show that. For example, sampling independent sets via MCMC simulation is known to have fast mixing only for graphs with degree at most 4 [11, 3], rather than obtaining the true threshold of 5.

In this work, we consider counting independent sets in hypergraphs with upper-bounded vertex degree, and lower-bounded hyperedge size. A hypergraph $H = (V, \mathcal{F})$ consists of a vertex set V and a set $\mathcal{F}$ of hyperedges, each of which is a subset of V. A hypergraph is said to be *k-uniform* if every hyperedge contains exactly k vertices. Thus, a 2-uniform hypergraph is the same as a graph. We will consider the more general case where each hyperedge has arity at least k, rather than exactly k.

An independent set in a hypergraph H is a subset of vertices that does not contain a hyperedge as a subset. We will be interested in computing Z_H, which is the total number of independent sets in H (also referred to as the partition function of H). Formally, the problem of counting independent sets has two parameters – a degree upper bound Δ and a lower bound k on the arity of hyperedges. The problem is defined as follows (see also Section 2 for an equivalent formulation in terms of monotone CNF formulas).

Name #HyperIndSet(k, Δ).
Instance A hypergraph H with maximum degree at most Δ where each hyperedge has cardinality (arity) at least k.
Output The number Z_H of independent sets in H.

Previously, #HyperIndSet(k, Δ) has been studied using the MCMC technique by Borderwich, Dyer, and Karpinski [1, 2] (see also [3]). They give an FPRAS for all $k \geq \Delta + 2 \geq 5$

and for $k \geq 2$ and $\Delta = 3$. Despite equipping path coupling with optimized metrics obtained using linear programming, these bounds are not tight for small k. Liu and Lu [9] showed that there exists an FPTAS for all $k \geq 2$ and $\Delta \leq 5$ using the correlation decay technique.

Thus, the situation seems to be similar to the graph case – given the analysis tools that we have, correlation-decay brings us closer to the truth than the best-tuned analysis of MCMC simulation algorithms. On the other hand, the technique of Liu and Lu [9] does not extend beyond $\Delta = 5$. To explain why, we need to briefly describe the correlation-decay-based algorithm framework introduced by Weitz [17]. The main idea is to build a recursive procedure for computing the marginal probability that any given vertex is in the independent set. The recursion works by examining sub-instances with "boundary conditions" which require certain vertices to be in, or out, of the independent set. The recursion structure is called a "computation tree". Nodes of the tree correspond to intermediate instances, and boundary conditions are different in different branches. The computation tree allows one to compute the marginal probability exactly but the time needed to do so may be exponentially large since, in general, the tree is exponentially large. Typically, an approximate marginal probability is obtained by truncating the computation tree to logarithmic depth so that the (approximation) algorithm runs in polynomial time. If the correlation between boundary conditions at the leaves of the (truncated) computation tree and the marginal probability at the root decays exponentially with respect to the depth, then the error incurred from the truncation is small and the algorithm succeeds in obtaining a close approximation.

All previous instantiations under this framework require a property called *strong spatial mixing* (SSM), which roughly states that, conditioned on *any* boundary condition on intermediate nodes, the correlation decays (see Section 2.1 in the full version). SSM thus guards against the worst-case boundary conditions that might be created by the recursive procedure.

Let the $(\Delta - 1)$-ary k-uniform hypertree $\mathbb{T}_{k,\Delta}$ be the recursively-defined hypergraph in which each vertex has $\Delta - 1$ "descending" hyperedges, each containing $k - 1$ new vertices.

▶ **Observation 1.** Let $k \geq 2$. For $\Delta \geq 6$, strong spatial mixing *does not hold* on $\mathbb{T}_{k,\Delta}$.

Observation 1 follows from the fact that the infinite $(\Delta - 1)$-ary tree $\mathbb{T}_{2,\Delta}$ can be embedded in the hypertree $\mathbb{T}_{k,\Delta}$, and from well-known facts about the phase transition on $\mathbb{T}_{2,\Delta}$.

Observation 1 prevents the generalisation of Liu and Lu's algorithm [9] so that it applies for $\Delta \geq 6$, even with an edge-size lower bound k. The problem is that the construction of the computation tree involves constructing intermediate instances in which the arity of a hyperedge can be as small as 2. So, even if we start with a k-uniform hypergraph, the computation tree will contain instances with small hyperedges. Without strong spatial mixing, these small hyperedges cause problems in the analysis. Lu, Yang and Zhang [10] discuss this problem and say "How to avoid this effect is a major open question whose solution may have applications in many other problems." This question motivates our work.

To overcome this difficulty, we introduce a new amortisation technique in the analysis. Since lack of correlation decay is caused primarily by the presence of small-arity hyperedges within the intermediate instances, we keep track of such hyperedges. Thus, we track not only the correlation, but also combinatorial properties of the intermediate instances in the computation tree. Using this idea, we obtain the following result.

▶ **Theorem 2.** *There is an FPTAS for* $\#\mathsf{HyperIndSet}(3, 6)$.

Note that $\#\mathsf{HyperIndSet}(2, 6)$ is NP-hard to approximate due to [15], so our result is tight for $\Delta = 6$. This also shows that $\Delta = 6$ is the first case where the complexity of approximately counting independent sets differs on hypergraphs and graphs, as for $\Delta \leq 5$ both admit an

FPTAS [9]. Moreover, Theorem 2 is stronger than the best MCMC algorithm [2] when $\Delta = 6$ as [2] only works for $k \geq 8$.

We also apply our technique to large k.

▶ **Theorem 3.** *There exists a constant k_0 such that for all positive integers $k \geq k_0$ and Δ satisfying $k \geq 1.66\Delta$ there is an FPTAS for the problem* $\#\mathsf{HyperIndSet}(k, \Delta)$.

In the large k case, our result is worse than that obtained by analysis of the MCMC algorithm [2] ($k \geq 1.66\Delta$ rather than $k \geq \Delta + 2$) but it is incomparable since our algorithm is deterministic rather than randomised. The reason that our bound is worse is mainly due to technical difficulty in the analysis that we will explain shortly. In fact, we believe that, in the long run, analysis of the correlation-decay based algorithm is more likely to reveal the exact critical threshold than analysis of MCMC simulation, although other new ideas will probably be required in order to achieve sufficiently precise analysis.

The main technical difficulty in correlation-decay analysis is bounding a function that we call the "decay rate". This boils down to solving an optimization problem with $(k-1)(\Delta-1)$ variables. In previous work (e.g. [13]), this optimization has been solved using a so-called "symmetrization" argument, which reduces the problem to a univariate optimization via convexity. However, the many variables represent different branches in the computation tree. Since our analysis takes the shape of intermediate instances in the tree into consideration, the symmetrization argument does not work for us, and different branches take different values at the maximum. This problem is compounded by the fact that the shape of the sub-tree consisting of "bad" intermediate instances is heavily lopsided, and the assignment of variables achieving the maximum is far from uniform. Given these problems, there does not seem to be a clean solution to the optimization in our analysis. Instead of optimizing, we give an upper bound on the maximum decay rate. In Theorem 2, as k and Δ are small, the number of variables is manageable, and our bounds are much sharper than those in Theorem 3. On the other hand, because of this, the proof of Theorem 3 is much more accessible, and we will use Theorem 3 as a running example to demonstrate our technique.

We also provide some insight on the hardness side. Recall that for graphs it is NP-hard to approximate $\#\mathsf{IS}$ beyond the uniqueness threshold ($\Delta = 6$) [15]. We prove that it is NP-hard to approximate $\#\mathsf{HyperIndSet}(6, 22)$ (Corollary 29). In contrast, we show in the full version that uniqueness holds on the 6-uniform Δ-regular hypertree iff $\Delta \leq 28$ (Corollary 36). Thus, efficient approximation schemes cease to exist well below the uniqueness threshold on the hypertree. In fact, we show that this discrepancy grows exponentially in k: for large k, it is NP-hard to approximate $\#\mathsf{HyperIndSet}(k, \Delta)$ when $\Delta \geq 5 \cdot 2^{k/2}$ (Theorem 28 and Corollary 30), despite the fact that uniqueness holds on the hypertree for all $\Delta \leq 2^k/(2k)$ (Lemma 37 in the full version). Theorem 28 follows from a rather standard reduction to the hard-core model on graphs. Nevertheless, it demonstrates that the computational-threshold phenomena in the hypergraph case ($k > 2$) are different from those in the graph case ($k = 2$).

As mentioned earlier, there are models where efficient (randomised) approximation schemes exist (based on MCMC simulation) even though SSM does not hold. In fact, this can happen even when uniqueness does not hold. A striking example is the ferromagnetic Ising model (with external field). As [14] shows, there are parameter regimes where uniqueness holds but strong spatial mixing fails. It is easy to modify the parameters so that even uniqueness fails. Nevertheless, Jerrum and Sinclair [6] gave an MCMC-based FPRAS that applies for all parameters and for general graphs (with no degree bounds). It is still an open question to give a correlation decay based FPTAS for the ferromagnetic Ising model.

We conclude this section by giving an outline of the rest of the paper. In Section 2, we give some preliminaries. We reformulate $\#\mathsf{HyperIndSet}(k, \Delta)$ as the problem of counting

satisfying assignments in monotone CNF formulas, which will allow us to use the computation tree of Liu and Lu [9]. In Section 3, we give an overview of our proof approach, describing the main idea behind our new amortisation technique. In Section 4, we give the main ingredients which we use to prove Theorem 3 (large k). Section 5 describes the main lemma which yields Theorem 2 ($k = 3$). Section 6 gives the formal statements and proofs of the hardness results.

2 Preliminaries: monotone CNF formulas & the computation tree

The problem of counting the independent sets of a hypergraph has an equivalent formulation in terms of monotone CNF formulas. Given a hypergraph $H = (V, \mathcal{F})$, let C be a Boolean formula with variable set V. For each hyperedge, construct a clause which is the disjunction of all variables corresponding to vertices in the hyperedge. Let C be the conjunction of all such clauses. Note that C is a monotone formula – no variable is negated. Also, independent sets of H are in one-to-one correspondence with satisfying assignments of C – a variable is assigned value "true" in an assignment if and only if it is out of the corresponding independent set. Going the other direction, any monotone CNF formula can be viewed as a hypergraph. In the technical sections of this paper, we use the monotone CNF formulation and, in particular, the computation tree of Liu and Lu [9]. Below we give the relevant definitions and notation; our notation aligns as much as possible with that of [9].

Let C be an instance of a monotone CNF formula. We will denote the set of variables in C by V and set $n := |V|$. Variables in V will be denoted by $x_1, x_2, \ldots$ and clauses in C by $c_1, c_2, \ldots$. The arity of a clause c will be denoted by $|c|$, i.e., $|c|$ is the number of variables appearing in the clause c. We assume throughout that no variable appears twice in the same clause. For a variable $x \in V$, we denote by $d_x(C)$ the number of clauses where x appears. When x and C are clear from context, we will simply use d to denote $d_x(C)$. When C is clear from context, we will use Δ to denote $\max_{x \in V} d_x(C)$ and we will say that C is a formula with max degree Δ. Let $\mathcal{C}_{k,\Delta}$ be the set of all monotone CNF formulas which have max degree Δ and whose clauses have arity at least k. Note that some formulas in $\mathcal{C}_{k,\Delta}$ may have some clauses with arbitrarily large arities.

Our goal is to approximately count the number of satisfying assignments of a formula $C \in \mathcal{C}_{k,\Delta}$, which we denote by $Z(C)$. Since C is monotone, an assignment $\sigma : V \to \{0, 1\}$ is satisfying if, for every clause in C, there is at least one variable $x \in c$ with $\sigma(x) = 1$. Note that $Z(C) > 0$ since the all-1 assignment satisfies every monotone CNF formula. For convenience, we will use the simplified notation "$x = 1$" to denote (the set of) satisfying assignments of C in which x is set to 1, and we similarly use "$x = 0$". We associate the formula C with a probability distribution in which each satisfying assignment has probability mass $1/Z(C)$. We will denote probabilities with respect to this distribution by $\Pr_C(\cdot)$.

Let x be a variable in V. Define $R(C, x) := \frac{\Pr_C(x=0)}{\Pr_C(x=1)}$, this is well-defined since $\Pr_C(x = 1) > 0$ by the monotonicity of C. In fact, the monotonicity of C also implies that $0 \leq R(C, x) \leq 1$, where the upper bound follows from the fact that, for every satisfying assignment with $x = 0$, flipping the assignment of x to 1 does not affect satisfiability. Our interest in the quantity $R(C, x)$ stems from the following simple lemma (the proof follows the argument in [9, Appendix A] and is given for completeness in the full version).

▶ **Lemma 5.** *Let k and Δ be positive integers. Suppose that there is a polynomial-time algorithm (in n and $1/\varepsilon$) that takes an n-variable formula $C \in \mathcal{C}_{k,\Delta}$, a variable x of C, and an $\varepsilon > 0$ and computes a quantity $\widehat{R}(C, x)$ satisfying $|\widehat{R}(C, x) - R(C, x)| \leq \varepsilon$. Then, there exists an FPTAS which approximates $Z(C)$ for every $C \in \mathcal{C}_{k,\Delta}$.*

Liu and Lu [9] established that a computation tree approach gives a recursive procedure for *exactly* calculating $R(C, x)$ for any monotone CNF formula C and any variable $x \in C$. We next give the details of this recursive procedure (see [9, Lemma 5]). First, we introduce the following definitions (see Definitions 6 and 7 in the full version).

We call the variable x *forced* in C if x appears in a clause of arity 1 in C. We call the variable x *free* if x does not appear in any clause of C. We call the clause c *redundant* in C if there is a clause c' in C such that c is a (strict) superset of c' (note that removing c from C does not affect the set of satisfying assignments of C). We next give the details of the computation tree. The nodes in the computation tree will be pairs (C, x) such that

$$C \text{ is a monotone CNF formula and } x \text{ is a variable which is not forced in } C. \tag{1}$$

Let C, x satisfy (1). We first perform a pre-processing step on C which involves (i) initially removing all of the redundant clauses, (ii) then, removing all clauses of arity 1. Note that part (ii) of the preprocessing step removes all forced variables that were present in C; at the time of the removal, forced variables appear only in clauses of arity 1 since part (i) of the preprocessing step has already removed all redundant clauses in C (and hence all clauses of arity greater than 1 that contain forced variables). Denote the formula after the completion of the preprocessing step by $\widetilde{C}$. Note that every clause in $\widetilde{C}$ is also a clause in the initial formula C, so x is not forced in $\widetilde{C}$. Further, since removing redundant clauses does not change the set of satisfying assignments of C, and x is not forced in C, we have that $R(\widetilde{C}, x) = R(C, x)$.

If x is free in $\widetilde{C}$ (the formula after the pre-processing step), then the start node (C, x) is (declared) a leaf of the computation tree (note that in this case $R(C, x) = 1$). In the sequel, we assume that x is not free in $\widetilde{C}$. Denote by $\{c_i\}_{i\in[d]}$ the clauses where x occurs in $\widetilde{C}$ and let $w_i = |c_i| - 1$ (note that $d \geq 1$). We will use $\mathbf{w}$ to denote the vector $(w_1, \ldots, w_d)$. The variables in clause c_i other than x will be denoted by $x_{i,1}, \ldots, x_{i,w_i}$. For the pair (C, x), we next construct pairs $(C_{i,j}, x_{i,j})$ for $i \in [d]$ and $j \in [w_i]$, where $C_{i,j}$ is an appropriate subformula obtained from $\widetilde{C}$, roughly, by hard-coding (some of) the occurrences of the variables in $\widetilde{C}$ to either 1 or 0 (this will be explained below and will henceforth be referred to as pinning)[1].

Precisely, for $i \in [d]$, let C_i be the formula obtained from $\widetilde{C}$ by removing clauses $c_1, \ldots, c_{i-1}$ (note that this has the same effect as pinning the occurrences of x in these clauses to 1) and pinning the occurrences of x in $c_{i+1}, \ldots, c_d$ to 0. For $j \in [w_i]$, the formula $C_{i,j}$ is obtained from C_i by further removing clause c_i and pinning all the occurrences of $x_{i,1}, \ldots, x_{i,j-1}$ to 0.

In the full version, we prove that the pairs $(C_{i,j}, x_{i,j})$ satisfy (1) for all $i \in [d]$, $j \in [w_i]$. We next state the relation between $R(C, x)$ and the $R(C_{i,j}, x_{i,j})$'s. It is proved in [9, Lemma 5], using a Weitz-type telescopic expansion (see also the full version for the argument), that

$$R(C, x) = \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{R(C_{i,j}, x_{i,j})}{1 + R(C_{i,j}, x_{i,j})}\right). \tag{2}$$

By applying (2) recursively, it is not hard to see that one can compute the quantity $R(C, x)$ *exactly*. Of course, exact computation using this scheme will typically require exponential time, so as in [9] we will stop the recursion at some (small) depth L to keep the computations feasible within polynomial time. This will yield a quantity $R(C, x, L)$ and the hope is that, by choosing L appropriately, the error $|R(C, x, L) - R(C, x)|$ will be sufficiently small.

[1] Note that our notation for i, j is different from the one in [9]; there, the roles of i, j are interchanged.

In light of (2), and analogously to [9], we define $R(C, x, L)$ for integer $L \geq 0$ as follows. Namely, for integer L we set

$$R(C,x,L) = \begin{cases} 1, & \text{if } x \text{ is free in } \widetilde{C} \text{ or } L \leq 0, \\ \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{R(C_{i,j}, x_{i,j}, L-l_{w_i})}{1+R(C_{i,j}, x_{i,j}, L-l_{w_i})}\right), & \text{otherwise,} \end{cases} \tag{4}$$

where $l_{w_i} := \lceil \log_6(w_i + 1) \rceil$ (note that $l_1 = \ldots = l_5 = 1$). The choice of l_{w_i} is such that the size of the computation tree is polynomial for $L = O(\log n)$; the particular choice of the logarithm base in the definition of l_{w_i} is not important as long as it is a big enough constant.

Using correlation decay techniques together with a new method to account for the shape of the computation tree, we will show the following key lemma (proved in Section 5).

▶ **Lemma 10.** *Let $\Delta = 6$. There exist constants α, τ with $0 < \alpha < 1$ and $\tau > 0$ such that the following holds for all integers L. Let C be a monotone CNF formula whose clauses all have arity greater than or equal to 3 and, further, each variable occurs in at most Δ clauses. Then, for the quantity $R(C, x, L)$ defined recursively from (4), it holds that*

$$|R(C,x,L) - R(C,x,\infty)| \leq \tau\alpha^L.$$

In the following section, we give an overview of our approach to proving Lemma 10. We also prove an analogous lemma in the case where k, Δ are large, see Lemma 11 in Section 4.

Proof of Theorems 2 and 3 assuming Lemmas 10 and 11. Invoke Lemmas 5, 10 and 11 and the reformulation of $\#\mathsf{HyperIndSet}(k, \Delta)$ in terms of the monotone CNF problem. ◀

3 Proof Approach

To prove Lemma 10, the standard approach so far in the literature has been to show that, for a node (C, x) in the computation tree, the quantity $|R(C, x, L) - R(C, x, \infty)|$ is bounded by $\alpha \max_{i,j} |R(C_{i,j}, x_{i,j}, L-1) - R(C_{i,j}, x_{i,j}, \infty)|$ for some constant $0 < \alpha < 1$ and then, inductively, to deduce that $|R(C, x, L) - R(C, x, \infty)|$ decays exponentially in L. This approach has been extremely successful when strong spatial mixing holds [14, 8, 9, 13, 18, 10].

In our setting, this inductive approach is problematic since, inside the computation tree, we are faced with the possibility that the formula at the root of a subtree has many arity-2 clauses. For $\Delta \geq 6$, these subtrees prohibit the application of the above proof scheme since they are in non-uniqueness and hence the desired step-by-step decay is no longer present.

While arity-2 clauses are problematic, clauses with larger arity do at least lead to good decay of correlation in a single step. Thus, our approach is to do an amortised analysis. In a single step, we track both the one-step decay of correlation and the immediate creation of arity-2 clauses, which will later lead to worse decay. More formally, instead of tracking the quantity $R(C, x, L)$, we track the quantity $m(C, x, L) = \delta^{b(C)} R(C, x, L)$ where $b(C)$ denotes the number of arity-2 clauses in the formula C (the "bad" clauses) and $\delta \in (0, 1)$ is an appropriate constant. (In fact, in Section 5, we define $m(C, x, L)$ as $\delta^{b(C)}\Phi(R(C, x, L))$ for an appropriate function Φ, see (28).)

Crucially, note that the root formula C satisfies $|m(C, x, L) - m(C, x, \infty)| = |R(C, x, L) - R(C, x, \infty)|$, since by the assumption in Lemma 10 we have that $b(C) = 0$. Thus, the key step in the proof of Lemma 10 is to show that the quantity $|m(C, x, L) - m(C, x, \infty)|$ decays exponentially with L; we will show that, for some constant $\alpha \in (0, 1)$, for an arbitrary node (C, x) in the computation tree, it holds that

$$|m(C,x,L) - m(C,x,\infty)| \leq \alpha \max_{i,j} |m(C_{i,j}, x_{i,j}, L-1) - m(C_{i,j}, x_{i,j}, \infty)|. \tag{5}$$

Since arity-2 clauses are the source of problems it is important not to create too many of them. Therefore, we also have to be careful in the construction of the computational tree. We achieve this by carefully ordering the clauses that we process at each step to avoid creating arity-2 clauses as much as possible.

Unfortunately, the quantity $m(C, x, L)$ is more complicated than the plain message $R(C, x, L)$ which has been studied before since it incorporates combinatorial information about the formula C and thus it does not satisfy a simple recursion (unlike $R(C, x, L)$). Nevertheless, we are able to define a multi-variable quantity κ (see (32)) and to show that when $\kappa \leq 1$, inequality (5) holds.

The technical details of applying the approach are quite intricate in the context of Lemma 10. In Section 4, we first apply the approach to the case of large k, where the proof is (significantly) shorter. There, instead of tracking the number of clauses of arity 2, we track (roughly) the aggregate arities of the clauses in C. Other than that, the high-level proof approach is similar to what is described above. In Section 5, we give an outline of the more difficult proof of Lemma 10.

4 The case of large k

In this section, we show the following lemma. Let $c := 0.565$ and $\beta \sim 1.65115$ be the solution to $2^{0.0001}c^\beta = 2 \times 0.9997(1 - c^\beta)c^2$.

▶ **Lemma 11.** *Let $\beta \sim 1.65$ be defined as above. Let k be a sufficiently large positive integer and let Δ be a positive integer satisfying $k \geq \beta\Delta + 3$. There are real numbers α and τ satisfying $0 < \alpha < 1$ and $\tau > 0$ such that for every $C \in \mathcal{C}_{k,\Delta}$ and every integer L,*

$$|R(C, x, L) - R(C, x, \infty)| \leq \tau\alpha^L, \tag{6}$$

where the quantity $R(C, x, L)$ is defined recursively from (4)*.*

To estimate the error $|R(C, x, L) - R(C, x, \infty)|$, we will track a specific quantity $m(C, x, L)$ which is assigned to each node in the computation tree. Let C be the original monotone CNF formula and let (N, x) be a node in the computation tree. As explained earlier, each clause c' of N is obtained from a clause c of C by pinning a certain number of variables to 0 (possibly none), which effectively is the same as removing those variables. We call these 0-pinnings *deficits* and let $\max\{0, k - |c'|\}$ be number of deficits of c'. Note that a clause of arity larger than k is considered to have no deficit, although some variables of it may have been pinned to 0. Let $D(N) = \sum_{c' \in N} \max\{0, k - |c'|\}$ denote the total deficits of N. Also observe that if a clause c of C does not show up in N, it does not contribute any deficits. For any node (N, x) in the computation tree, let $m(N, x, L) := \delta^{D(N)} R(N, x, L)$ where $\delta \in (0, 1)$ is a constant that we will choose later.

Now let us calculate how the number of deficits changes in one step of the recursion. Let (N, x) be a node in the computation tree. Note that pinning any variable to 1 will remove all clauses containing it, and will therefore eliminate all deficits of these clauses. Moreover, for a clause of arity 2, pinning any of its variables to either 0 or 1 will eliminate the whole clause due to the preprocessing step. Let $b_2(N, x)$ (or simply b_2 when N and x are clear from the context) denote the number of arity-2 clauses containing x in N. In the recursion, we will always order the clauses so that arity-2 clauses come last. Thus, clauses $c_1, \ldots, c_{d-b_2}$ each have arity at least 3 and clauses $c_{d-b_2+1}, \ldots, c_d$ each have arity 2. Due to these arity-2 clauses, the deficits in every branch below (N, x) will decrease by at least $b_2(k - 2)$.

Now consider an integer i in the range $1 \leq i \leq d - b_2$. Recall that in N_i, we pin appearances of x prior to i to 1, thus eliminating deficits in clauses $c_1, \ldots, c_{i-1}$. Together with the removal

of clause c_i, these removals decrease the total deficits by $\sum_{t=1}^{i} \max\{0, k-1-w_t\}$, where $w_t = |c_t| - 1$. Let $s_i = \sum_{t=1}^{i} \max\{0, k-1-w_t\}$. We also pin all appearances of x after i to 0, increasing the total number of deficits by at most $d - i - b_2$. Here we have a "$-b_2$" because pinning a variable to 0 in a arity-2 clause does not increase the deficits. Moreover, in $N_{i,j}$, we pin all appearances of $j-1$ other variables to 0. Each of these pinnings may contribute at most $\Delta - 1$ deficits, giving a total increase of at most $(\Delta-1)(j-1)$. Next consider $i \geq d - b_2 + 1$. For such an i, clause c_i has arity 2, and it is easy to see that deficits do not increase in the branch corresponding to $N_{i,j}$. Also note that the preprocessing steps do not increase deficits. Hence we have the following upper bounds on $D(N_{i,j})$ for $i \in [d]$:

$$D(N_{i,j}) \leq \begin{cases} D(N) - b_2(k-2) - s_i + d - i - b_2 + (\Delta-1)(j-1) & \text{if } 1 \leq i \leq d - b_2, \\ D(N) - b_2(k-2) - s_{d-b_2} & \text{if } d - b_2 + 1 \leq i. \end{cases} \tag{7}$$

The key to our analysis is to bound the correlation decay of $m(C, x, L)$. We will analyse the recursion for $m(C, x, L)$ based on that of $R(C, x, L)$. Recall that the recursion for $R(C, x, L)$ depends on the function $F^{d,\mathbf{w}}(\mathbf{r})$ implicitly defined by (4), i.e.,

$$F^{d,\mathbf{w}}(\mathbf{r}) := \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{r_{i,j}}{1 + r_{i,j}}\right), \tag{8}$$

where $r_{i,j} \in [0,1]$ for all $i \in [d]$, $j \in [w_i]$ (so $R(C, x, L) = F^{d,\mathbf{w}}(\{R(C_{i,j}, x_{i,j}, L - l_{w_i})\})$).

For $i \in [d]$, the following quantity $\rho_{\delta,\alpha}^{\mathbf{w},i}$ will roughly upper bound the sensitivity of $|m(C, x, L) - m(C, x, \infty)|$ to the i-th clause in which x appears in C, or more precisely, to the quantities $|m(C_{i,j}, x_{i,j}, L - l_{w_i}) - m(C_{i,j}, x_{i,j}, \infty)|$ for $j \in [w_i]$:

$$\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r}) := \begin{cases} \alpha^{-l_{w_i}} \delta^{s_i + i - d + b_2} \sum_{j=1}^{w_i} \delta^{-(j-1)(\Delta-1)} \left| \frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,j}} \right| & \text{if } 1 \leq i \leq d - b_2, \\ \alpha^{-l_{w_i}} \delta^{s_{d-b_2}} \left| \frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,1}} \right| & \text{if } d - b_2 + 1 \leq i \leq d. \end{cases} \tag{10}$$

Note that $\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$ depends also on $r_{i',j}$ with $i' \neq i$. The decay rate of $|m(C, x, L) - m(C, x, \infty)|$ in terms of L, for all formulas C with max degree Δ, will be captured by the aggregation of $\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$'s, namely, $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r}) := \delta^{b_2(k-2)} \sum_{i=1}^{d} \rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$. The main technical lemma of the section is the following.

▶ **Lemma 12.** *For all sufficiently large Δ, for any $k \geq \beta\Delta + 3$ where $\beta \sim 1.65115$ is defined earlier, there exists constants $0 < \delta < 1$, $0 < \alpha < 1$, and $U > 0$ such that, for all $\mathbf{0} \leq \mathbf{r} \leq \mathbf{1}$, $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ is at most 1 when $d \leq \Delta - 1$ and at most U when $d = \Delta$.*

Before sketching the proof of Lemma 12, let's show that it is sufficient to imply Lemma 11.

Proof of Lemma 11 (Sketch). Let δ, α be as in Lemma 12. For $C \in \mathcal{C}_{k,\Delta}$, we have $D(C) = 0$, so $|m(C, x, L) - m(C, x, \infty)| = |R(C, x, L) - R(C, x, \infty)|$.

The lemma will thus follow by showing that for every node (N, x) of the computation tree the quantity $|m(N, x, L) - m(N, x, \infty)|$ decays roughly as α^L. The proof is by induction on L; the base cases $L \leq 0$ are easy to show since $\delta \in (0, 1)$. For the induction step, one needs the following key inequality (obtained by considering the gradient of $F^{d,\mathbf{w}}$):

$$|m(N, x, L) - m(N, x, \infty)| \leq \max_{\mathbf{r}} \left\{ \left(\sum_{i=1}^{d} \sum_{j=1}^{w_i} \delta^{D(N) - D(N_{i,j})} \alpha^{-l_{w_i}} \left| \frac{\partial F^{d,\mathbf{w}}(\mathbf{r})}{\partial r_{i,j}} \right| \right) \right\} \times \max_{i,j} \left\{ \alpha^{l_{w_i}} \left| m(N_{i,j}, x_{i,j}, L - l_{w_i}) - m(N_{i,j}, x_{i,j}, \infty) \right| \right\}. \tag{16}$$

Our bounds (7) on $D(N)-D(N_{i,j})$ for $i\in[d]$, $j\in[w_i]$ imply that the $\max_{\mathbf{r}}\{\cdot\}$ expression in (16) is upper bounded by $\max_{\mathbf{r}}\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ (since $\delta\in(0,1)$). Thus, by Lemma 12, one obtains step-wise decay of $m(N,x,L)$. (The case $d=\Delta$ arises only at the root formula.) ◀

Proof of Lemma 12 (Sketch). Let $\alpha=0.9999$ and δ be such that $\delta^{\Delta}=c$ (recall, $c=0.565$).

The proof has two main steps. In the first step, one obtains a bound on $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ that does not depend on $\mathbf{r}$ (i.e., we bound the terms which depend on the "messages" $R(C,x,L)$). We do this for each clause c_i; the bound we obtain depends only on the arity of the clause, or, for the purposes of this proof, on w_i. The second step is then to maximize over the possible values of the w_i's.

For $w_i\geq 2$, let $\mu(w_i):=\max\{2,w_i\left(2-2^{1-w_i}\right)^{-1}\}$. The result of the first step is the following bound on $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ (note that $\mathbf{r}$ is completely eliminated).

$$\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{b_2(k-2)}\left(\sum_{i=1}^{d-b_2}\alpha^{-l_{w_i}}\delta^{s_i+i-d+b_2-(w_i-1)(\Delta-1)}2^{-w_i}\mu(w_i)+\alpha^{-1}b_2\right),\tag{21}$$

Using the fact that $\alpha^{-l_{w_i}}\leq\alpha^{-1}w_i^{0.0001}$ for any $w_i\geq 2$, we obtain that, if $w_i>k-1$, then the right hand side of (21) increases if we replace w_i with $k-1$. Henceforth we may assume that $w_i\leq k-1$, which allows us to rewrite $s_i=s_{i-1}+k-1-w_i$. Using the fact that $\delta^{\Delta}=c$, we then obtain

$$\alpha\,\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{k+b_2(k-1)}\eta_d(\mathbf{w})+b_2\delta^{b_2(k-2)},\quad\eta_d(\mathbf{w}):=\sum_{i=1}^{d-b_2}\delta^{s_{i-1}}w_i^{0.0001}(2c)^{-w_i}\mu(w_i).\tag{22}$$

Convexity arguments show that the maximum of $\eta_d(\mathbf{w})$ is achieved at $\mathbf{w}=\mathbf{w}'$, where $w_i'=k-1$ for $1\leq i\leq t$ and $w_i'=2$ for $t+1\leq i\leq d-b_2$ for some $0\leq t\leq d-b_2-1$. We thus obtain

$$\eta_d(\mathbf{w})\leq\eta_d(\mathbf{w}')\leq\Delta(k-1)^{1.0001}(2c)^{-(k-1)}\left(2-2^{2-k}\right)^{-1}+\frac{2^{0.0001}}{2c^2(1-\delta^{k-3})}\leq 0.9998c^{-\beta},\tag{26}$$

where the last inequality follows from $\frac{2^{0.0001}}{2c^2(1-\delta^{k-3})}\leq\frac{2^{0.0001}}{2c^2}\cdot\frac{1}{1-c^{\beta}}=0.9997c^{-\beta}$ (using the definition of c,δ) and that for large Δ and k, we have $\Delta(k-1)^{1.0001}(2c)^{-(k-1)}\left(2-2^{2-k}\right)^{-1}<0.0001$ (using that $2c>1$ and $k\geq\beta\Delta+3$). Now, plug (26) into (22):

$$\alpha\,\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{b_2(k-2)}\left(0.9998\,\delta^k c^{-\beta}+b_2\right)\leq c^{\beta b_2}\left(0.9998+b_2\right)\leq\alpha,\tag{27}$$

where the last inequality follows from $c^{\beta}<1/2$. (27) yields $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq 1$, as claimed. ◀

5 A finer analysis to treat $k\geq 3$, $\Delta=6$

In this section, we describe the main technical ingredient to prove Lemma 10 (which in turn was critical in deducing Theorem 2 in Section 2).

Let (C,x) be a non-leaf node in the computation tree, where the root node is a monotone CNF formula with max degree $\Delta=6$ all of whose clauses have arity at least $k=3$. Denote by $b(C)$ the total number of clauses of arity 2 in C. Recall that $\widetilde{C}$ is the formula obtained from C after the preprocessing step. Also, $c_1,\ldots,c_d$ are the clauses in which x appears in $\widetilde{C}$ and $w_i=|c_i|-1$. We have $d\geq 1$ since (C,x) is a non-leaf node and $w_i\geq 1$ for all i because of property (1). Let b_3 denote the number of clauses such that $|c_i|=3$ and let b_2 denote the number of clauses such that $|c_i|=2$.

When processing the node (C, x), we will order the clauses in which x appears in $\widetilde{C}$ so that clauses with $|c_i| = 3$ are processed first. We thus have that $d, b_2, b_3 \in \mathbb{Z}$ and $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{Z}^d$ satisfy

$$\begin{aligned} &1 \le d, \quad 0 \le b_2, b_3 \le d, \quad b_2 + b_3 \le d \\ &w_i = 2 \text{ for } i = 1, \ldots, b_3, \quad w_i \ge 3 \text{ or } w_i = 1 \text{ for } i = b_3 + 1, \ldots, d. \end{aligned} \tag{30}$$

As already described in Section 3, to prove Lemma 10, we will track a quantity $m(C, x, L)$ which is assigned to each node in the computation tree. Let $\eta := \eta(\Delta) = (1/2)^\Delta$. Define $m(C, x, L)$ by

$$m(C, x, L) := \delta^{b(C)}\, \Phi(R(C, x, L)) \tag{28}$$

where $\delta \in (0, 1]$, and $\Phi : [\eta, 1] \to \mathbb{R}$ satisfies:

$$\Phi \text{ is continuously differentiable on } [\eta, 1], \text{ and } \varphi := \Phi' \text{ satisfies } \varphi(z) > 0 \text{ for } z \in [\eta, 1]. \tag{29}$$

For $\Delta = 6$, the value of δ will be later chosen to be $\delta = 9789/10000$ and Φ will be specified in the upcoming equation (46). Also, note that for all nodes (C, x) in the computation tree the quantity $m(C, x, L)$ is well-defined; this is because of the property (1) and the lower bound $R(C, x, L) \ge \eta$ (follows from (4), see Remark 8 in the full version).

We will show Lemma 10 with $\alpha := 1 - 10^{-4}$. In particular, we aim to show that α upper bounds the decay rate of $|m(C, x, L) - m(C, x, \infty)|$, i.e., $|m(C, x, L) - m(C, x, \infty)|$ decays roughly as α^L (modulo a multiplicative constant). As in the proof of Lemma 11, this yields Lemma 10 when applied to the root formula of the tree (since it has no arity-2 clauses).

Ler $\mathbf{r}$ be a real vector with components $r_{i,j}$ for $i \in [d]$ and $j \in [w_i]$, $\mathbf{0} \le \mathbf{r} \le \mathbf{1}$. For $i \in [d]$, we will use the following analogue of the quantity $\rho^{\mathbf{w},i}_{\delta,\alpha}(\mathbf{r})$ defined in Section 4 (see (10)):

$$\rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) := \alpha^{-l_{w_i}} \sum_{j=1}^{w_i} \left(\frac{1}{\delta}\right)^{(j-1)(\Delta-1)} \frac{1}{\varphi(r_{i,j})} \left|\frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,j}}\right|, \tag{31}$$

where recall that the function $F^{d,\mathbf{w}}(\mathbf{r})$ gives the recursion (4) that $R(\cdot,\cdot,\cdot)$ satisfies (cf. (8)). Once again, the quantity $\rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r})$ will roughly upper bound the sensitivity of $|m(C, x, L) - m(C, x, \infty)|$ to the quantities $|m(C_{i,j}, x_{i,j}, L) - m(C_{i,j}, x_{i,j}, \infty)|$ for $j \in [w_i]$. The decay rate of $|m(C, x, L) - m(C, x, \infty)|$ will be captured by the following quantity:

$$\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r}) := \varphi(F^{d,\mathbf{w}}(\mathbf{r}))\, \delta^{b_2} \left(\sum_{i=1}^{b_3} \left(\frac{1}{\delta}\right)^{b_3 - i} \rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) + \sum_{i=b_3+1}^{d} \rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) \right). \tag{32}$$

Finally, we specify the function Φ. For $z \in (0, 1]$, let

$$\Phi(z) := \frac{1}{\chi\psi} \log\left(\frac{z^\chi}{\psi - z^\chi}\right) \text{ with } \chi = 1/2,\ \psi = 13/10, \text{ so } \varphi(z) = \Phi'(z) = \frac{1}{z(\psi - z^\chi)}. \tag{46}$$

Our main technical lemma is the following (proved in Section 6 of the full version).

▶ **Lemma 14.** *Let $\Delta = 6$, $\chi = 1/2$, $\psi = 13/10$, $\delta = 9789/10000$, $\alpha = 1 - 10^{-4}$ and Φ be defined from (46). There exists a constant $U > 0$ such that, for all $d, b, \mathbf{w}$ satisfying (30), for all $(1/2)^{\Delta-1}\mathbf{1} \le \mathbf{r} \le \mathbf{1}$, it holds that $\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r})$ is at most 1 when $d \le \Delta - 1$ and at most U when $d = \Delta$.*

Lemma 10 can be derived from Lemma 14 using an argument analogous to the one used to derive Lemma 11 from Lemma 12, see Section 5 in the full version for the details.

The proof of Lemma 14 is significantly harder however than the proof of Lemma 12. For one thing, the analysis has to account for the (irrational) function φ (without which the step-wise contraction is simply not true) present in the expression for $\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r})$. Further, the two-step procedure that we followed in the proof of Lemma 12, that is, maximising first over $\mathbf{r}$ and then maximising over $\mathbf{w}$, is far too crude. Instead, we need to slowly reduce the number of the variables akin to the approach in [9]. In our case, however, to bound tightly the contribution of δ, we need to prove asymmetric inequalities in several variables with irrational expressions (see the key Lemmas 15, 19 and 20 in the full version). To keep the length of the paper reasonable and easier to read through, we rigorously verify certain two-variable inequalities using a computer; as the reader will notice, in some cases, even reducing to such two-variable inequalities requires a significant analytical effort (see the proofs of the aforementioned lemmas, for example).

6 Hardness Results

We prove the hardness results stated in the Introduction. We will work with the problem $\#\mathsf{HyperIndSet}(k,\Delta)$ (instead of the monotone CNF formulation). The proof is via a reduction to the independent set model on graphs which was used by Bordewich *et al.* [2]. The precise inapproximability results for the hard-core model had not yet been proved at the time [2] was written, so we carry out the details explicitly to obtain the bound that their reduction gives.

We use the inapproximability result of Sly and Sun [16] for the hard-core model. Recall, for a graph $G=(V,E)$, the partition function of G in the hard-core model with parameter $\lambda>0$ is given by $Z_G(\lambda):=\sum_I \lambda^{|I|}$ where the sum ranges over all independent sets I of G.

▶ **Theorem 27** ([16]). *For $\Delta\geq 3$, let $\lambda_c(\Delta):=(\Delta-1)^{\Delta-1}/(\Delta-2)^{\Delta}$. For all $\lambda>\lambda_c(\Delta)$, it is* NP*-hard to approximate $Z_G(\lambda)$ on Δ-regular graphs G, even within an exponential factor.*

▶ **Theorem 28.** *Let $k\geq 2$, $\Delta\geq 3$ be integers. Suppose that $2^{\lceil k/2\rceil}-1<\frac{(\Delta-2)^{\Delta}}{(\Delta-1)^{\Delta-1}}$. Then, it is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$, even within an exponential factor.*

Proof (Sketch). Let $\lambda:=1/(2^{\lceil k/2\rceil}-1)$. Let G be a Δ-regular n-vertex graph. Construct a hypergraph H by replacing every vertex of G by a (distinct) set of $\lceil k/2\rceil$ vertices and place a hyperedge among the corresponding sets of vertices for each edge of G. Then $Z_H=(2^{\lceil k/2\rceil}-1)^n\,Z_G(\lambda)$, from which the theorem easily follows (see full version for details). ◀

▶ **Corollary 29.** *Let $k=6$, $\Delta=22$. It is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$.*

▶ **Corollary 30.** *Let $k\geq 2$. For $\Delta\geq 5\cdot 2^{k/2}$, it is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$.*

References

1 M. Bordewich, M. E. Dyer, and M. Karpinski. Path coupling using stopping times and counting independent sets and colorings in hypergraphs. *Random Struct. Algorithms*, 32(3):375-399, 2008.

2 M. Bordewich, M. E. Dyer, and M. Karpinski. Stopping Times, Metrics and Approximate Counting. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, LNCS 4051:108-119, 2006.

3 M. Dyer and C. Greenhill. On Markov Chains for Independent Sets. *J. Algorithms*, 35(1):17-49, 2000.

4 A. Galanis, Q. Ge, D. Štefankovič, E. Vigoda, and L. Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Struct. Algorithms*, 45(1):78-110, 2014.

5 A. Galanis, D. Štefankovič, and E. Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *CoRR*, abs/1203.2226, 2012. Preprint is available from the arXiv at: http://arxiv.org/abs/1203.2226.

6 M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993.

7 L. Li, P. Lu, and Y. Yin. Approximate counting via correlation decay in spin systems. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 922–940, 2012.

8 L. Li, P. Lu, and Y. Yin. Correlation Decay up to Uniqueness in Spin Systems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67-84, 2013.

9 J. Liu and P. Lu, FPTAS for Counting Monotone CNF. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1531–1548, 2015. Full version available from the arXiv at http://arxiv.org/abs/1311.3728.

10 P. Lu, K. Yang, and C. Zhang. FPTAS for Hardcore and Ising Models on Hypergraphs. Available from the arXiv at http://arxiv.org/abs/1509.05494

11 M. Luby and E. Vigoda. Fast convergence of the Glauber dynamics for sampling independent sets. *Random Struct. Algorithms*, 15(3-4):229-241, 1999.

12 E. Mossel, D. Weitz, and N. Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probab. Theory Related Fields*, 143(3-4):401–439, 2009.

13 A. Sinclair, P. Srivastava, D. Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 1549–1563, 2015.

14 A. Sinclair, P. Srivastava, and M. Thurley. Approximation algorithms for two-state antiferromagnetic spin systems on bounded degree graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 941–953, 2012.

15 A. Sly. Computational Transition at the Uniqueness Threshold. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 287–296, 2010.

16 A. Sly and N. Sun. The computational hardness of counting in two-spin models on d-regular graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 361–369, 2012.

17 D. Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing* (STOC), 140–149, 2006.

18 Y. Yin and C. Zhang. Spatial mixing and approximate counting for Potts model on graphs with bounded average degree. Available from the arXiv at http://arxiv.org/abs/1507.07225