On Markov chains for #Clustered-Monotone-SAT and other hard counting problems with easy decision version

Eleni Bakali

Abstract—TotP is the class of all self-reducible counting problems in #P having decision version in P. Two TotP-complete problems under parsimonious reductions are Size-of-Subtree, and #CM-SAT. It is an important theorem that #SAT reduces to #CM-SAT under approximation preserving reductions. Sampling and approximate counting are often performed with the use of Markov chains. However for #SAT it is known that we cannot design irreducible Markov chains whose state space is the set of satisfying assignments of an input formula, due to a scattering phenomenon of the set of solutions. On the contrary, the set of solutions of #CM-SAT, as well as of Size-of-Subtree, is connected in a particular way that permits the design of irreducible Markov chains. We design and study some Markov chains whose state space is the set of solutions of the above problems. We analyse their mixing time, their stationary distributions, and the complexity of computing the normalizing factors and the size of the support of the stationary distributions. Finally, we discuss the results in relation to the hardness of counting in TotP, and to other open problems in complexity theory.

Index Terms—backtracking tree, computational complexity, counting complexity, Markov chains, randomized algorithms, sampling, #CM-SAT, easy decision version, TotP

1 Introduction

THE complexity class #P [28] is the class of functions that count the number of accepting paths of an NPTM (Non-deterministic Polynomial Time Turing Machine). Equivalently, #P is the class of functions that count the number of solutions of a problem in NP. We often use the term 'counting problem' for such functions.

The set of all self-reducible counting problems in #P having decision version in P is contained in a complexity class called TotP [1]. By saying 'decision version' of a counting problem, we mean the problem of deciding whether the value of the corresponding function on an input is non-zero. Equivalently, the decision version examines if there are any solutions to the corresponding NP problem, or any accepting computational paths of the corresponding NPTM. TotP is a proper subclass of #P unless P=NP [1].

There is a great number of problems, from many different scientific areas, that belong to TotP. For example such problems are: counting matchings, computing the determinant of a matrix, computing the partition function of several models from statistical physics, like the Ising and the hard-core model, counting colourings of a graph with a number of colors greater than the maximum degree, counting bases of a matroid, computing the volume of a convex body, counting independent sets, and many more.

In [4] Bakali et.al. showed the first completeness results for TotP, under parsimonious reductions, i.e. reductions that preserve the number of solutions. The parsimoniously complete problems are representative of a class, in the sense that all the problems of the class are reduced to the complete ones, so each algorithmic result for one of the complete problems is automatically transferred to the whole class.

Two such problems are Size-of-Subtree and Clustered-Monotone-SAT (abr. #CM-SAT). The Size-of-Subtree is the problem of computing the size of a backtracking tree, without traversing it exhaustively, a problem firststudied by Knuth [5]. #CM-SAT [4] is the problem of computing the number of solutions of a logical formula with the following properties: There is a tree of polynomial height whose nodes correspond to the satisfying assignments (or solutions) of this formula, such that for every given solution it is easy to get any adjacent to this solution, by making some appropriate transformations in polynomial time. In addition, it is easy to find the satisfying assignment corresponding to root of this tree in polynomial time.

The #CM-SAT is in particular interesting because it turns out that the general #SAT reduces to #CM-SAT under approximation preserving reductions [4]. That means that every logical formula can be reduced to another formula with the above mentioned properties so that we can approximate the number of solutions to the first one if and only if we can approximate the number of solutions to the second one.

This fact is important because of the following phenomenon that has arised from the study of random SAT. For formulas considered hard, the satisfying assignments are widely scattered in the space of all assignments [7], [8], [9]. The solutions form clusters where is hard to find even one solution, and moreover even if you are given one solution in some cluster, it is hard to find another one in a different cluster. The solutions do not seem to relate to each other in any algorithmically tractable way. This scattering explains the reason of failure of many algorithmic approaches for #SAT, and particularly of methods that rely on Markov chains; In order to have a unique stationary distribution, it is essential for a Markov chain to be irreducible, i.e. it must be able to move from any state to any other at some time. But as we mentioned before, it

• *Eleni Bakali is currently pursuing PhD degree program in theoretical computer science in the National Technical University of Athens, Greece. E-mail: mpakali@corelab.ntua.gr*

is not known how to construct a Markov chain with a state space that consists of the set of solutions of a formula with a scattered set of solutions.

On the contrary, for the TotP-complete version of #SAT, namely #CMSAT, the situation is completely different. The solutions of an input formula build an easy neighbourhood structure, in particular a tree of polynomial height, such that from any solution it is easy to move to its neighbouring solutions.

This suggests that the study of Markov chains for #CM-SAT as well as for other TotP-complete problems, which present the same connectivity property of solutions, is a natural research direction. This is the direction that we follow in this paper.

1.1 Some first approaches and arising questions

A simple random walk on a perfect (i.e. full and complete) binary tree mixes in polynomial time with respect to the number of the nodes of the tree. However the tree under consideration is the one we mentioned earlier, whose nodes are the satisfying assignments of some input formula. As we said, for any input to #CM-SAT, the associated tree has polynomial height with respect to the size of the input, so we consider all running times with respect to the height of the tree. So the size of the perfect binary tree is exponential with respect to its height, and thus the running time of the simple random walk is exponential, as well.

An intuitive reason for the slow mixing of the simple random walk on the perfect binary tree is that from any internal node the probability of going a level down is twice the probability of going a level up, (since every internal node has two children, but one parent).

So we thought of designing a Markov chain where the probability of going a level up equals the probability of going a level down. If n is the height of the tree, we can prove that the mixing time of this Markov chain equals the mixing time of a simple random walk on a path of length n, which is well known to be $O(n^2)$.

We then generalize this Markov chain to every binary tree, not necessarily full or complete, and a number of questions arise:

- Does this Markov chain on an arbitrary binary tree converge quickly?
- Which is its stationary distribution?
- Can we compute the normalizing factor of the stationary distribution?
- Can we compute the size of the support of the stationary distribution? Note that the size of the support equals the size of the tree, and thus corresponds exactly to the counting problem.
- Can we reduce counting to computing the normalizing factor of the stationary distribution? Note that for the uniform distribution the normalizing factor is just a multiple of the size of the support, but as we will see, this is not the case for the stationary distributions of our Markov chains.
- Can we have an FPRAS for the counting problem? (An FPRAS is a polynomial time approximation algorithm with an arbitrarily small multiplicative error). If not, what is the reason for this difficulty?

1.2 Main results and discussion

We first analyse two Markov chains on the perfect binary tree: (a) the simple random walk and (b) an Markov chain that from any internal node the probability of going to the parent is double the probability of going to each child. We prove that the second Markov chain is rapidly mixing.

We then generalize the second Markov chain to an arbitrary binary tree. We define a family of Markov chains $(P_S)_S$, each having as states the nodes of a binary tree S, as follows.

Let S be a subtree of the perfect binary tree T of height n, containing the root of T. We define the Markov chain P_S over the nodes of S, with the following transition probabilities.

$p_S(i,j) = 1/2$ if j is the parent of i,
$p_S(i,j) = 1/4$ if j is a child of i,
$p_S(i,j) = 0$ for every other $j \neq i$, and
$p_S(i,i) = 1 - \sum_{j \neq i} p_S(i,j)$.

We prove the following.

- For every S the corresponding Markov chain P_S converges in polynomial time with respect to the height of the tree n. Note that this means that the running time is logarithmic in the size of the tree, when this size is exponentially big.
- The stationary distribution of P_S is $\pi_S(u) = a \cdot 2^{n-i}$, where i is the depth of node u and a is the normalizing factor of π_S, i.e. a constant such that $\sum_{u \in V(S)} \pi_S(u) = 1$.
- The normalizing factor can be approximated in polynomial time. The exact computation of the normalizing factor is hard.
- The computation of the size of the support can be reduced to the computation of the normalizing factor. However the exact as well as the approximate computation of the normalizing factor is hard.

The last two results seem contradictory. We can approximate the normalizing factor, and we can reduce the computation of the size of the support to the computation of the normalizing factor, but we cannot approximate the size of the support. The reason why this happens is that the reduction is subtractive, and subtractive reductions do not preserve multiplicative errors. So if we use the approximation of the normalizing factor to approximate the size of the support, the error of the latter could be exponentially big.

A remark arising from our results is the following. According to [3] it is proved that for the uniform distribution over the set of solutions to a problem in #P, the tasks of sampling, computing the size of the support (aka counting), and computing the normalizing factor are computationally equivalent. As it turns out this is not the case for our distributions. In fact we show that an FPRAS for the normalizing factor implies an FPRAS for counting if and only if NP=RP.

A final corollary deriving from our results is a new additive error approximation algorithm for any problem in TotP. This

algorithm exploits the special characteristics of TotP, and thus opens new research directions regarding derandomization. Note that an additive error approximation algorithm for problems in #P is already known ([12], chapter 6.2.2), but its derandomization is a long standing open question, proved to be as hard as proving circuit lower bounds [13]. See section 6 for more details on this matter.

2 Preliminaries

2.1 #P and TotP

The model of computation is the non-deterministic polynomial time Turing machine (NPTM). For each NPTM M there is a polynomial p_M bounding its running time. When an NPTM M returns *yes* or *no*, we denote $acc_M(x)$ the number of accepting computational paths of M on input $x \in \Sigma^*$, where Σ is some alphabet.

Definition 1. *[28] #P is the class of functions f for which there exists a polynomial-time decidable binary relation R and a polynomial p such that* $\forall x \in \Sigma^*$, $f(x) = \{y \in \{0,1\}^* \mid |y| = p(|x|) \wedge R(x,y)\}$. *Equivalently,* $\#P = \{acc_M : \Sigma^* \to \mathbb{N} \mid M \text{ is a NPTM}\}$.

Definition 2. *FP is the class of functions in #P computable in polynomial time.*

Definition 3. *The **decision version** of a function in #P is the language* $L_f = \{x \in \Sigma^* \mid f(x) > 0\}$.

Definition 4. *[10]* $\#PE = \{f : \Sigma^* \to \mathbb{N} \mid f \in \#P \text{ and } L_f \in P\}$.

Definition 5. *[19]* $TotP = \{tot_M : \Sigma^* \to \mathbb{N} \mid M \text{ is a NPTM}\}$, *where* $tot_M(x) = \#(\text{all computation paths of } M \text{ on input } x) - 1$.

The most important theorem about TotP is that TotP contains all self-reducible problems in #P, with decision version in P.

Intuitively a function is self-reducible if we can reduce the computation of its value on an instance to the computations of its values on a polynomial number of other instances, and the depth of this recursion is polynomial. For example #SAT is self-reducible, since the number of satisfying assignments of a formula ϕ equals the sum of satisfying assignments of two formulas ϕ_0 and ϕ_1, where ϕ_0, (respectively ϕ_1) is ϕ with its first variable fixed to 0 (respectively 1).

Definition 6. *A function* $f : \Sigma^* \to \mathbb{N}$ *is called poly-time self-reducible if there exist polynomials r and q, and polynomial time computable functions* $h : \Sigma^* \times \mathbb{N} \to \Sigma^*$, $g : \Sigma^* \times \mathbb{N} \to \mathbb{N}$, *and* $t : \Sigma^* \to \mathbb{N}$ *such that for all* $x \in \Sigma^*$:

1. *f can be processed recursively by reducing x to a polynomial number of instances* $h(x,i)$ $(0 \le i \le r(|x|))$, *i.e. formally*
   $\forall x\ f(x) = t(x) + \sum_{i=0}^{r(|x|)} g(x,i) f(h(x,i))$
2. *the recursion terminates after at most polynomial depth, i.e. formally the depth of the recursion is* $q(|x|)$, *and* $\forall x \forall j_1, j_2, \ldots, j_{q(|x|)} \in \{0,\ldots,r(|x|)\}$
   $f(h(\ldots h(h(x,j_1),j_2)\ldots,j_{q(|x|)}))$ *can be computed in polynomial time.*
3. *every instance invoked in the recursion is of* $poly(|x|)$ *size, i.e. formally*
   $\forall x\ \forall k \le q(|x|)\ \forall j_1,j_2,\ldots,j_k \in \{0,\ldots,r(|x|)\}$
   $|h(\ldots h(h(x,j_1),j_2)\ldots,j_k| \in poly(|x|)$.

Theorem 1. *[1] (a)* $FP \subseteq TotP \subseteq \#PE \subseteq \#P$. *The inclusions are proper unless P=NP.*
(b) TotP is the Karp closure of self-reducible #PE functions.

2.2 Size-of-Subtree and #CM-SAT

We focus on two counting problems, Size-of-Subtree and #CM-SAT, that are both TotP-complete under parsimonious reductions [4].

In this paper we use the convention that the **height** of a tree is the maximum length of a path from the root to a leaf, and that the **depth** of a node is its distance from the root. A tree of height n, has $n + 1$ **levels**, corresponding to the depths of the nodes, and the root is in depth or level 0.

Definition 7. *Let T be the perfect binary tree of height n. Let S be a subtree of height n containing the root of T, given in succinct representation, e.g. by a polynomial computable predicate* $R_S : V(T) \to \{0,1\}$ *such that* $R_S(u) = 1$ *iff* $u \in V(S)$, *where* $V(\cdot)$ *denotes the set of vertices. The counting problem Size-of-Subtree is to compute the size of* $V(S)$ *for an input tree S.*

We will not give the precise definition of #CM-SAT, see [4]. We are just going to describe it in words. #CM-SAT is #SAT restricted to formulas with the following special properties.

Consider a clustering of the space of solutions $\{0,1\}^n$, where each cluster contains all assignments that have their first k variables fixed to some values. A formula of #CM-SAT has got at most one satisfying assignment in each cluster, and it is easy to decide whether such an assignment exists and, if so, easy to find it.

Moreover, if we label each cluster according to their fixed values, then we have the following certain kind of monotonicity among the clusters. Consider a rooted complete binary tree with 2^k nodes. Suppose you traverse the tree in BFS order putting to each node of the tree a label of a cluster in lexicographic order (see Fig. 1). Then a formula of #CM-SAT has the property that if a cluster has no solutions, then all its descendant clusters (with respect to the tree structure) have no solutions.

An input to #CM-SAT consists of a number k and a formula with the above properties.

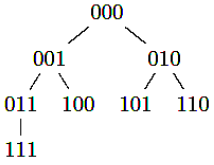

Fig. 1: The complete binary tree T_3.

The important theorem about #CM-SAT is that #SAT is reduced to it under approximation preserving reductions [4].

This means that every formula of SAT can be transformed in polynomial time to another formula with the above mentioned properties, such that if we can approximate the number of satisfying assignments of the second formula, we can also approximate the number of satisfying assignments of the first one. This implies that if can approximate the number of solutions to the second formula, we can actually decide whether the first formula is satisfiable.

Theorem 2. *[4]* $\#SAT \leq_{AP} \#CM\text{-}SAT$.

Since Size-of-Subtree is TotP-complete under parsimonious reductions [4], the results extend to any problem in TotP. For an arbitrary problem #A in TotP, there is a tree S such that its nodes are in one-to-one correspondence with the solutions of A. We will not get into that. For more details see [4].

In fact in the presentation of our results we focus on Size-of-Subtree, but it is easy to see from the problems definitions that the results extend to #CM-SAT in a straight forward way: The solutions of an input formula of #CM-SAT form a subtree of the complete binary tree of height k. Thus such a formula can be considered as a succinct representation of an input for Size-of-Subtree.

2.3 Kinds of approximation

We are concerned with two kinds of approximation, additive and multiplicative. Additive refers to the computation of some probability, and multiplicative refers to any real quantity we want to estimate. We call FPRAS an algorithm with an arbitrarily small multiplicative error.

Definition 8. *We call* ***FPRAS*** *for a function* $f: \Sigma^* \to \mathbb{R}$ *a probabilistic algorithm A such that for all* $x \in \Sigma^*$, *and for all* $\epsilon > 0$ *computes a value A(x) in time poly*$(|x|, \epsilon^{-1})$ *such that*

$$Pr[(1-\epsilon) f(x) \leq A(x) \leq (1+\epsilon) f(x)] \geq \frac{3}{4}.$$

Definition 9. *We call* ***additive*** *approximation to a probability p, a number* $\hat{p} = p \pm \xi$ *for some* $\xi \in (0,1)$. *In the case of Size-of-Subtree the quantity under consideration is* $p \equiv |S| / 2^n$. *In the case of the Circuit Acceptance Probability Estimation problem (CAPE) [13] the quantity under consideration is* $p \equiv$ *#sat. assignments*$/2^n$, *where n is the number of input gates of the given circuit.*

2.4 Basics of Markov chains

Let $\{X_t\}_{t\geq 0}$ be a Markov chain over a finite state space $\mathcal{X}$ with transition probabilities p_{ij}.

Let $p_x^{(t)}$ be the distribution of X_t when starting from state x.

Let π be the stationary distribution, and let $\tau_x(\epsilon) = \min\{t : \|p_x^{(t)} - \pi\| \leq \epsilon\}$ be the *mixing time* when starting from state x.

An ergodic Markov chain is called *time reversible* if $\forall i,j \in \mathcal{X}$, $p_{ij}\pi_i = p_{ji}\pi_j$.

Let H be the *underlying graph* of the chain, for which we have an edge with weight $w_{ij} = p_{ij}\pi_i = p_{ji}\pi_j$ for each $i,j \in \mathcal{X}$.

A Markov chain is called *lazy* if $\forall i \in, p_{ii} \geq 1/2$.

In [3] the conductance of a time reversible Markov chain is defined, as follows.

Definition 10. *(Conductance) Let* $Y \subseteq \mathcal{X}$. *We define*

$$\Phi(Y) = \frac{\sum_{i\in Y,\, j\notin Y} w_{ij}}{\sum_{i\in Y} \pi_i}$$

The conductance of the Markov chain is

$$\Phi(H) = \min \Phi(Y)$$

where the minimum is taken over all $Y \subseteq \mathcal{X}$ *such that* $0 < \sum_{i\in Y} \pi_i \leq 1/2$.

Lemma 1. *[3] For any lazy, time reversible Markov chain*

$$\tau_x(\varepsilon) \leq const \times \left[\frac{1}{\Phi(H)^2} (\log \pi_x^{-1} + \log \varepsilon^{-1}) \right].$$

Lemma 2. *For any Markov chain, and any* $Y \subseteq \mathcal{X}$ *with* $\pi(Y) \leq 1/2$

$$\tau_{mix} \geq \frac{1}{4\Phi(Y)}$$

3 The simple random walk on the perfect binary tree

Consider the simple random walk R on a full complete binary tree T of height n. I.e. the transition probabilities are

r(i,j) = 1⁄3 for all i≠j,
r(i,i) = 1⁄3 for i=root,
r(i,i) = 2⁄3 for i=leaf, and
r(i,i) = 0 in any other case.

Theorem 3. *The simple random walk W converges to the uniform distribution* $U(i) = (2^n - 1)^{-1}$ *for every node i.*

Proof. It is easy to check that for all nodes i,j

$$\sum_i U(i) r(i,j) = U(j).$$

□

Theorem 4. *The mixing time of W is exponential in n.*

Proof. Let Y be the set of nodes of the left subtree of T. $U(Y) = \frac{1}{2} - \frac{1}{2^n - 1}$. There is only one edge e = (i,j) leaving from Y ; i is the left child of the root, and j is the root. The edge e has weight $w_e = r_{ij}Ui = \frac{1}{3}\frac{1}{2^n - 1}$. Thus $\Phi(Y) \leq 1/2$, and so from lemma 2 we have $\tau_{mix} \geq \frac{3(2^n - 3)}{8}$.

□

3 A rapidly mixing Markov chain on the perfect binary tree

Consider a Markov chain Q on the perfect binary tree T of height n, with transition probabilities

q(i,j) = 1⁄2 if j is the parent of i,
q(i,j) = 1⁄4 if j is a child of i,
q(i,j) = 0 for every other j≠i, and
$q(i,i) = 1 - \sum_{j\neq i} p_S(i,j)$.

Theorem 5. *The stationary distribution of Q is* $\pi_Q(v) = 2^{-depth(v)}(n+1)^{-1}$ *for all v.*

Proof. It is easy to check that for all nodes i,j

$$\sum_i \pi_Q(i)q(i,j) = \pi_Q(j) .$$

□

Theorem 6. *The mixing time of Q when starting form the root, is polynomial in n, thus logarithmic in the size of T.*

Proof. Let $\{L_t\}_{t\geq 0}$ be a Markov chain on a path with n + 1 nodes, defined as L_t = depth(Q_t), where depth(v) denotes the depth of node v in T. The transition probabilities of L are

$$l(i,j) = \Pr[L_t = j \mid L_{t-1} = i] = \Pr[depth(Q_t) = j \mid depth(Q_{t-1}) = i]$$

thus

l(i,j) = 1⁄2 for all i≠j,

l(0,0) = l(n,n) = 1⁄2.

So L is a simple random walk on a path of length n + 1. Its stationary distribution is the uniform over the nodes of the path and its mixing time is polynomial in n. The latter is well known, but for the sake of completeness we present briefly a proof:

First we consider the lazy version of the chain, i.e. with probability 1⁄2 we do nothing, and with probability 1⁄2 we follow the rules of L. Conductance Φ(Y) is minimized for Y_1 = {0,1,...,⌊(n+1)/2 ⌋}, $w_{(n+1)/2,(n+1)/2+1}$ =1/2(n+1) , so Φ(Y_1) ≥ 1/(n+1). Thus from lemma 1 for ϵ = 1/4 we get that the mixing time, starting form any initial node x, is O(n^2 log n).

We return to the proof of the theorem. When L has mixed to its stationary distribution, Q has mixed to a distribution that is uniform over the levels of T. To complete the proof observe that the symmetries of the perfect binary tree imply that when starting from the root, for every step t, the distribution of Q_t depends only on the depth of the nodes, i.e. the probability Pr[Q_t = v] is the same for all v in the same depth. The latter can also be proved by induction in t.

So at the time that L has mixed to the uniform distribution, Q has mixed to π_Q.

□

5 A GENERALIZATION FOR AN ARBITRARY BINARY TREE

We generalize the Markov chain of the previous section to an arbitrary binary tree, not necessarily full or complete. I.e. the state space consists of the nodes of a tree of height n, and the transition probabilities are as in the previous section.

Definition 11. *Let S be a subtree of the perfect binary tree T of height n, containing the root of T. We define the Markov chain P_S over the nodes of S, with the following transition probabilities.*
$p_S(i,j)$ = 1⁄2 if j is the parent of i,
$p_S(i,j)$ = 1⁄4 if j is a child of i,
$p_S(i,j)$ = 0 for every other j≠i, and
$p_S(i,i) = 1 - \sum_{j\neq i} p_S(i,j)$.

5.1 Stationary distribution and mixing time.

Consider the following family of distributions $(\pi_S)_S$.

Definition 12. *Let S be a binary tree of height n and let V (S) be the set of nodes of S. For all u ∈ V (S) $\pi_S(u) = a_S \cdot 2^{n-i}$, where i is the depth of node u and a_S is the normalizing factor of π_S, i.e. a constant such that $\sum_{u\in V(S)} \pi_S(u) = 1$.*

Theorem 7. *The stationary distribution of P_S is π_S as defined in 12.*

Proof. It is easy to check that

$$\sum_i \pi_S(i)p_S(i,j) = \pi_S(j) .$$

□

Note For simplicity of notations from now on we will assume that S is fixed and omit it from the subscripts in P_S,p_S,π_S,α_S, unless we refer to more than one tree in the same proof.

Now we will prove that P is rapidly mixing, i.e. mixes in time polynomial in the height of the tree S.

The following lemma proves two properties of the Markov chains we defined, needed for the proofs that will follow.

Lemma 3. *Let R be a binary tree of height n, and let a_R be the normalizing factor of the stationary distribution π_R of the above Markov chain (def. 11, 12). It holds $a_R^{-1} \leq (n+1)2^n$, and $\pi_R(root) \geq 1/(n+1)$.*

Proof. Let r_i be the number of nodes in depth i.

$$1 = \sum_{u\in S} \pi_R(u) = \sum_{i=0}^{n} \sum_{level(u)=i} \pi_R(u) = \sum_{i=0}^{n} r_i a_R 2^{n-i}$$

$$\Rightarrow \frac{1}{a_R} = \sum_{i=0}^{n} r_i 2^{n-i}$$

which is maximized when the r_i's are maximized, i.e. when the tree is perfect binary (lets say T), in which case $r_i = 2^i$ and $\alpha_T^{-1} = (n+1)2^n$. This also implies that for the root of R it holds $\pi_R(root) = \alpha_R \cdot 2^n \geq 1/(n+1)$.

□

Proposition 1. *The mixing time of P, when starting from the root, is polynomial in the height of the tree n.*

Proof. First of all, we will consider the lazy version of the Markov chain, i.e. in every step, with probability 1⁄2 we do nothing, and with probability 1⁄2 we follow the rules as in definition 11. The mixing time of P is bounded by the mixing time of its lazy version. The stationary distribution is the same. The Markov chain is time reversible, and the underlying graph is a tree with edge weights $w_{uv} = \pi_u p_{uv} = 2^i\alpha \times 1/8 = 2^{i-3}\alpha$, if we suppose that u is the father of v and $2^i\alpha$ is the probability π_u.

The quantity π_{root}^{-1} is O(n), as we showed in lemma 3.

Now it suffices to show that 1⁄Φ(H) is polynomial in n.

Let $\mathcal{X}$ be the set of the nodes of S, i.e. the state space of the Markov chain P. We will consider all possible Y ⊆ $\mathcal{X}$ with 0 ≤ π(Y) ≤ 1⁄2. We will bound the quantity

$$\Phi(Y) = \frac{\sum_{i\in Y, j\notin Y} w_{ij}}{\sum_{i\in Y} \pi_i} .$$

If Y is connected and does not contain the root of S, then it is a subtree of S, with root let say u, and $\pi_u = \alpha 2^k$ for some k∈ℕ . We have

$$\sum_{i\in Y, j\notin Y} w_{ij} \geq w_{u,\, father(u)} = 2^{k-2}a .$$

Now let Y ′ be the perfect binary tree with root u and height the same as Y , i.e. k. We have

$$\sum_{i\in Y} \pi_i \leq \sum_{i\in Y'} \pi_i = \sum_{j=0}^{k} 2^{k-j} a \times 2^j =$$

$$2^k(k+1)a \le 2^k(n+1)a$$

where this comes if we sum over the levels of the tree Y '. So it holds

$$\frac{\sum w_{ij}}{\sum \pi_i} \ge \frac{2^{k-2}a}{2^k(n+1)a} = \frac{1}{4(n+1)}.$$

If Y is the union of two subtrees of S, not containing the root of S, and the root of the first one is an ancestor of the root of the second one, then the same arguments hold, where now we take as u the root of the first subtree.

If Y is the union of λ subtrees not containing the root of S, for which it holds that no one's root is an ancestor of any other's root, then we can prove a same bound as follows. Let $Y_1,...Y_\lambda$ be the subtrees, and let $k_1,k_2,...,k_\lambda$, be the respective exponents in the probabilities of the roots of them, in the stationary distribution. Then as before

$$\sum w_{ij} \ge 2^{k_1-2}a + 2^{k_2-2}a + ... + 2^{k_\lambda-2}a$$

and

$$\sum_{i\in Y}\pi_i = \sum_{j=1...\lambda}\sum_{i\in Y_j}\pi_i \le 2^{k_j}(n+1)a$$

thus

$$\frac{\sum w_{ij}}{\sum \pi_i} \ge \frac{a\sum_{j=1...\lambda}2^{k_j-2}}{(n+1)a\sum_{j=1...\lambda}2^{k_j}} = \frac{1}{4(n+1)}.$$

If Y is a subtree of S containing the root of S, then the complement of Y , i.e. S \ Y is the union of λ subtrees of the previous form. So if we let Y_i,k_i be as before, then

$$\sum w_{ij} = \sum_{j=1...\lambda} 2^{k_j-2}$$

and since from hypothesis π(Y) ≤ 1⁄2, we have

$$\sum_{i\in Y}\pi_i \le \sum_{i\in S\setminus Y}\pi_i \le (n+1)a\sum_{j=1...\lambda}2^{k_j}$$

thus the same bound holds again.

Finally, similar arguments imply the same bound when Y is an arbitrary subset of S i.e. an arbitrary union of subtrees of S.

In total we have 1⁄Φ(H) ≤ 4(n + 1).

□

Note that this result implies mixing time $O(n^2 \log n)$. This agrees with the theorem in the previous section, stating that on the full binary tree the mixing time should be as much as the mixing time of a simple random walk over the levels of the tree, i.e. over a chain of length n.

5.2 The complexity of computing the normalizing factor

We analyse the complexity of exactly and approximately computing the normalizing factors of the family of distributions $(\pi_S)_S$ (def 12).

Theorem 8. *Computing the normalizing factor a_s of any distribution π_s in the family of def. 12*

1) *is TotP-hard under Turing reductions,*
2) *FPRAS is possible, i.e. approximation with arbitrarily small multiplicative error,*
3) *exact computation is impossible deterministically (or respectively probabilistically) if NP ≠ P (or* respectively NP ≠ RP).

Proof.

1) We will reduce the computation of the size of S to the computation of the normalizing factors of the above probability distributions $(\pi_S)_S$. The reduction is Turing (i.e. the most general kind of reduction, as opposed to other kinds of reductions with extra properties, like the parsimonious ones). Since Size-of-Subtree is TotP-complete under parsimonious reductions, we conclude that computing the normalizing factor is TotP-complete under Turing reductions. It remains to show the following proposition.

Proposition 2. *Let S be a binary tree of height n, and ∀i = 0...n, let S_i be the subtree of S that contains all nodes up to depth i, and let a_{Si} be the corresponding normalizing factors defined as above. Then*

$$|S| = \frac{1}{a_{S_n}} - \sum_{k=0}^{n-1}\frac{1}{a_{S_k}}.$$

Proof of Proposition 2. For i = 1,...,n let r_i be the number of nodes in depth i. So $|S| = r_0 + ... + r_n$.

Obviously if S is not empty,

$$r_0 = 1 = \frac{1}{a_{S_0}} \tag{1.1}$$

We will prove that ∀k = 1...n

$$r_k = \frac{1}{a_{S_k}} - 2\frac{1}{a_{S_{k-1}}} \tag{1.2}$$

so then

$$|S| = \frac{1}{a_{S_0}} + \sum_{k=1}^{n}\left(\frac{1}{a_{S_k}} - 2\frac{1}{a_{S_{k-1}}}\right) = \frac{1}{a_{S_n}} - \sum_{k=1}^{n}\frac{1}{a_{k-1}}.$$

We will prove claim (1.2) by induction.

For k = 1 we have

$$\sum_{u\in S_1}\pi_{S_1}(u) = 1 \Rightarrow a_{S_1}\cdot r_1 + 2a_{S_1}\cdot r_0 = 1 \Rightarrow$$

$$r_1 = \frac{1}{a_{S_1}} - 2r_0 = \frac{1}{a_{S_1}} - 2\frac{1}{a_{S_0}}.$$

Suppose claim (1.2) holds for k < i ≤ n. We will prove it holds for k = i.

$$\sum_{u\in S_i}\pi_{S_i}(u) = 1 \Rightarrow \sum_{k=0}^{i}2^{i-k}a_{S_i}\cdot r_k = 1 \Rightarrow$$

$$r_i = \frac{1}{a_{S_i}} - \sum_{k=0}^{i-1}2^{i-k}r_k$$

and substituting r_k for k = 0,...,i − 1 by (1.1) and (1.2), we get

$$r_i = \frac{1}{a_{S_i}} - 2\frac{1}{a_{S_{i-1}}}.$$

(end proof Prop. 2)□

2) Now we give an FPRAS for the computation of the normalizing factor α_S, using the previously defined Markov chain P_S .

Proposition 3. *For any binary tree R of height n we can estimate a_R, within (1 ± ζ) for any ζ > 0, with probability 1 − δ for any δ > 0, in time poly(n,ζ^{-1},log δ^{-1}).*

Proof of Proposition 3. Let R be a binary tree of height n.

We can estimate α_R as follows.

As we saw, $\pi_R(root) = 2^n\alpha_R$, and we observe that this is always at least $1/(n+1)$ (which is the case when R is full binary). So we can estimate $\pi_R(root)$ within $(1 \pm \zeta)$ for any $\zeta > 0$, by sampling m nodes of R according to π_R and taking, as estimate, the fraction $= \sum_{i=1}^{m} X_i$, where $X_i = 1$ if the i-th sample node was the root, else $X_i = 0$.

It is known by standard variance analysis arguments (see e.g. the unbiased estimator theorem in [11]) that we need $m = O(\pi_R(root) \cdot \zeta^{-2}) = poly(n,\zeta^{-1})$ to get

$$\Pr\left[(1-\zeta)\pi_R(root) \le \hat{p} \le (1+\zeta)\pi_R(root)\right] \ge \frac{1}{3}.$$

We can boost up this probability to $1 - \delta$ for any $\delta > 0$, by repeating the above sampling procedure $t = O(\log \delta^{-1})$ times, and taking as final estimate the median of the t estimates computed each time, (see the median trick in [11]).

The random sampling according to π_R can be performed by running the Markov chain defined earlier, on the nodes of R. Observe that the deviation ϵ from the stationary distribution can be negligible and be absorbed into ζ, with only a polynomial increase in the running time of the Markov chain.

Finally, the estimate for α_R is $\hat{a}_R = 2^{-n}\hat{p}$, and it holds

$$\Pr\left[(1-\zeta)a_R \le \hat{a}_R \le (1+\zeta)a_R\right] \ge (1-\delta).$$

(end proof Prop. 3)□

3) The third part of theorem 8 is proved as follows. #IS is the problem of counting the number of all independent sets (of all sizes) of an input graph. #IS ∈ TotP. If NP≠P (respectively NP≠RP) then #IS does not admit FPTAS (respectively FPRAS) [14], [2].

Since the computation of the normalizing factors of the family of distributions $(\pi_S)_S$ is TotP-hard, #IS is reduced to it, so if we could exactly compute the normalizing factors of $(\pi_S)_S$, we could also compute exactly #IS, which would imply NP=P (respectively NP=RP if the computation is randomised).

□

5.3 The complexity of computing the size of the support

We analyse the complexity of computing the size of the support of the family of distributions $(\pi_S)_S$ (def. 12). We then discuss two remarks that result from this analysis.

Theorem 9. *Computing the size of the support of any distribution in family $(\pi_S)_S$ (def. 12)*

1) *is TotP-complete under parsimonious reductions,*
2) *reduces to exactly computing the normalizing factors,*
3) *additive error approximation (see def. 9) is possible in randomized polynomial time, (note that the size of the support is in general exponential in n),*
4) *exact computation is impossible deterministically (or respectively probabilistically) if NP ≠ P (or respectively NP ≠ RP),*
5) *multiplicative polynomial factor approximation is impossible deterministically (or respectively probabilistically) iff NP ≠ P (or respectively NP ≠ RP).*

Proof.

1) From definition 12, a positive probability is given to every node of the corresponding input tree S, so the size of the support equals exactly the size of the tree. The statement of the theorem follows from the TotP-completeness of Size-of-Subtree under parsimonious reductions [4].

2) The statement follows from proposition 2.

3) We first prove the following proposition.

Proposition 4. *For all $\xi > 0, \delta > 0$ we can get an estimate $|\hat{S}|$ of $|S|$ in time poly$(n,\xi^{-1},\log \delta^{-1})$ such that*

$$\Pr\left[|S| - \xi 2^n \le |\hat{S}| \le |S| + \xi 2^n\right] \ge 1-\delta.$$

Proof of Proposition 4. Let $\zeta = \xi/(2(n+1))$ and $C = \zeta/(1+\zeta)$, thus poly(ε^{-1}) = poly(ζ^{-1}) = poly(n,ξ^{-1}).

So according to proposition 3 we have in time poly$(n,\xi^{-1},\log \delta^{-1})$ estimations $\forall i = 1,\ldots,n$

$$(1-\varepsilon)a_{S_i} \le \hat{a}_{S_i} \le (1+\varepsilon)a_{S_i} \quad (1.3)$$

We will use proposition 2. Let $A = 1/a_{S_n}$ and $B = \sum_{k=0}^{n-1} 1/a_{S_k}$, so $|S| = A - B$, and clearly $B \le A$.

From (1.3) we have $(1/1+\varepsilon)A \le \hat{A} \le 1/(1-\varepsilon)A \Leftrightarrow$

$(1-\zeta)A \le \hat{A} \le (1+\zeta)A$ and similarly $(1-\zeta)B \le \hat{B} \le (1+\zeta)B$.

Thus $(1-\zeta)A - (1+\zeta)B \le \hat{A} - \hat{B} \le (1+\zeta)A - (1-\zeta)B \Leftrightarrow$

$A - B - \zeta(A+B) \le \hat{A} - \hat{B} \le A - B + \zeta(A+B)$, and since $A \ge B$, we have

$|S| - 2\zeta A \le |\hat{S}| \le |S| + 2\zeta A$. And since from lemma 3 the maximum A is $2^n(n+1)$, we have

$|S| - 2\zeta(n+1)2^n \le |\hat{S}| \le |S| + 2\zeta(n+1)2^n \Leftrightarrow$

$|S| - \xi \cdot 2^n \le |\hat{S}| \le |S| + \xi \cdot 2^n$.

(end proof Prop. 4)□

To conclude the proof, let $p = |S| / 2^n$. For all $\xi > 0$, $\delta > 0$ we can get an estimation in time poly$(n,\xi^{-1},\log \delta^{-1})$ such that

$$\Pr[p - \xi \le \hat{p} \le p + \xi] \ge 1 - \delta$$

which satisfies the definition of additive approximation for $|S|$, i.e. the size of the support of π_S.

4) The arguments are simiral to those for the third part of theorem 8. #IS ∈ TotP and does not admit FPTAS (respectively FPRAS) if NP≠P (respectively NP≠RP) [14], [2].

From the hardness of computing the size of the support of $(\pi_S)_S$, it turns out that we cannot compute it exactly, else we could also compute exactly #IS, which would imply NP=P (respectively NP=RP if the computation is randomised).

5) **Fact 1**: *If NP=RP then all problems in TotP admit an FPRAS.*

Proof of fact 1: In [6] Stockmeyer has proven that an FPRAS, with access to a Σ_2^p oracle, exists for any problem in #P. If NP=RP then $\Sigma_2^p = RP^{RP} \subseteq BPP$. Finally it is easy to see that an FPRAS with access to a BPP oracle, can be replaced by another FPRAS, that simulates the oracle calls itself.

Fact 2: *If NP≠RP then it is impossible to have an FPRAS for every problem TotP.*

Proof of fact 2: The problem #IS is inapproximable if NP≠RP, and it belongs to TotP.

Finally, it is proven that for any self reducible problem in #P, an algorithm with a polynomial multiplicative error can be transformed to FPRAS [3].

Since the computation of the size of the support of π_S is TotP-complete under parsimonious reductions, we conclude that it admits a polynomial multiplicative approximation algorithm if and only if NP=P (respectively NP=RP if the algorithm is randomised).

□

6 Remarks and further research

Remark 1. We have considered three computational tasks related to any probability distribution: sampling, computing the normalizing factor, and computing the size of the support. For the uniform distribution the normalizing factor is a multiple of the size of the support, and sampling is equivalent to existence of FPRAS for the size of the support [3]. However for a general distribution not only their complexity is unknown; it is even unclear whether these three tasks are equivalent or not.

For the family of distributions we studied here (definition 12) it turns out that the three tasks are not all equivalent unless NP=RP.

We also showed that exact computation of the size of the support reduces to exactly computing the normalizing factor (fact 2), but not under approximation preserving reductions. The previous arguments imply that such an approximation preserving reduction, between the two tasks, exists if and only if NP=RP.

Open problem 1. Thus it is an open problem equivalent to NP vs RP the existence of an approximation preserving reduction of the computation of the size of the support to the computation of the normalizing factors of distributions $(\pi_S)_S$ (def. 12).

Remark 2. We showed how to get a probabilistic additive approximation to the problem Size-of-Subtree. We should mention that an additive approximation could be also obtained by a simple random sampling process that chooses m = poly(n) nodes of the full binary tree T of height n uniformly at random, and takes as estimate of the size of S, the proportion of those m samples that belong to S. This is an application of the general method of [12] chapter 6.2.2.

Our alternative method exploits some special characteristics of TotP, and thus cannot be generalized for every problem in #P.

It is known that derandomizing the general simple method is as difficult as proving circuit lower bounds [13]. However we don't know a similar relationship between circuit lower bounds and deterministic additive approximation, restricted to problems in TotP, thus it might not be equally difficult to derandomize our method.

Open problem 2. It is an open problem how to derandomize the additive error approximation algorithms for the size of the support, in subexponential time. This would yield a deterministic solution of the CAPE problem (see def. 9), within additive approximation, for families of circuits, for which counting the number of accepting inputs is in TotP, (we will call it TotP-CAPE.)

Note that the best (exact, and additive error) deterministic algorithm known for CAPE, on an arbitrary circuit, is by exhaustive search. Derandomizing it faster than the exhaustive search algorithm, i.e. even in time $2^{\gamma n}\text{poly}(n)$ for some $\gamma < 1$, would yield NEXP $\not\subseteq$ P/poly [13]. The latter is a long standing conjecture.

Open problem 3. A final open problem, is whether we can achieve derandomization of the same task in polynomial time. Such a result would also imply a solution to the CAPE problem in deterministic polynomial time for depth-two AC^0 circuits (i.e. DNF's and CNF's). The best deterministic algorithm known until now is of time $n^{O(\log \log n)}$ [16]. (For more on AC^0-CAPE, see the survey [17] p.13, and [18] for an older result.)

7 Related work

TotP is defined in [19], some of its properties and relationships to other classes are studied in [1], [10], [20], and completeness is studied in [4].

Regarding the Size-of-Subtree: In [5] Knuth provides a probabilistic algorithm practically usefull, but with an exponential worst case error. Modifications and extensions of Knuth's algorithm have been presented and experimentaly tested in [21], [22], [23], without signifcant improvements for worst case instances. There are also many heuristics and experimental results on the problem ristricted to special backtracking algorithms, or special instances, see e.g. [24] for more references. Surprisingly there exist FRAS's for random models of the problem [25], [26]. In [27] quantum algorithms for the problem are studied. In [6] Stockmeyer provided unconditional lower bounds for the problem under a model of computation which is different from the Turing machine, namely a variant of the (non-uniform) decision tree model.

The relationship between approximate counting and uniform sampling has been studied in [3].

There are numerous papers regarding algorithmic and hardness results for individual problems in #P and TotP, e.g. [28], [29], [30], [2], [31]. However, apart from the backtracking-tree problem, other TotP-complete problems have not been studied algorithmically yet.

There is also a huge literature on Markov chains and computational questions related to them, like those considered in this paper, i.e. determining the mixing time, the stationary distribution, and the complexity of computing the partition function (i.e. the normalizing factor) of the stationary distribution. Such algorithms are known e.g. as Glauber dynamics, Gibbs sampling, Metropolis - Hastings algorithm, etc [32], [33], [34]. These Markov chains are designed for individual problems that concern the computation of a weighted sum over the set of solutions to a combinatorial problem. Such problems are for example the hard-core model and the Potts model from statistical physics. The literature on this matter is enormus, we mention only indicatevely [35], [36], [37], [38], [39], [40].

ACKNOWLEDGMENT

I want to thank Stathis Zachos, Dimitris Fotakis and Aris Pagourtzis for their useful comments. I also want to thank Eleni Chamou, Petros Pantavos and Gerasimos Palaiopanos for writing assistance.

REFERENCES

[1] Aris Pagourtzis, Stathis Zachos: "The Complexity of Counting Functions with Easy Decision Version." *MFCS 2006*: 741-752
[2] Martin E. Dyer, Alan M. Frieze, Mark Jerrum: "On Counting Independent Sets in Sparse Graphs." *SIAM J. Comput.* 31(5): 1527-1541 (2002)
[3] Alistair Sinclair, Mark Jerrum: "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains." *Inf. Comput.* 82(1): 93-133 (1989)
[4] Eleni Bakali, Aggeliki Chalki, Aris Pagourtzis, Petros Pantavos, Stathis Zachos: "Completeness Results for Counting Problems with Easy Decision." *CIAC 2017*: 55-66
[5] Donald E. Knuth. 1974. "Estimating the Efficiency of Backtrack Programs." *Technical Report. Stanford University*, Stanford, CA, USA.
[6] Larry J. Stockmeyer: "On Approximation Algorithms for #P." *SIAM J. Comput.* 14(4): 849-861 (1985)
[7] Dimitris Achlioptas: "Random Satisfiability." *Handbook of Satisfiability* 2009: 245-270
[8] Dimitris Achlioptas, Federico Ricci-Tersenghi: "Random Formulas Have Frozen Variables." *SIAM J. Comput.* 39(1): 260-280 (2009)
[9] Dimitris Achlioptas, Amin Coja-Oghlan, Federico Ricci-Tersenghi: "On the solution-space geometry of random constraint satisfaction problems." *Random Struct. Algorithms* 38(3): 251-268 (2011)
[10] A. Pagourtzis. "On the complexity of hard counting problems with easy decision version." *3rd Panhellenic Logic Symposium*, Anogia, Crete, 2001.
[11] Alistair Sinclair, CS271 Randomness and Computation, lecture notes, Fall 2011, https://people.eecs.berkeley.edu/ sinclair/cs271/n10.pdf
[12] Oded Goldreich: "Computational complexity - a conceptual perspective." Cambridge University Press 2008, ISBN 978-0-521-88473-0, pp. I-XXIV, 1-606
[13] Ryan Williams: "Improving Exhaustive Search Implies Superpolynomial Lower Bounds." *SIAM J. Comput.* 42(3): 1218-1244 (2013)
[14] Alistair Sinclair, CS294 Markov Chain Monte Carlo: Foundations and Applications, lecture notes, Fall 2009 https://people.eecs.berkeley.edu/ sinclair/cs294/f09.html
[15] Sanjeev Arora, Boaz Barak: "Computational Complexity - A Modern Approach." Cambridge University Press 2009, ISBN 978-0-521-42426-4, pp. I-XXIV, 1-579
[16] Parikshit Gopalan, Raghu Meka, Omer Reingold: "DNF sparsification and a faster deterministic counting algorithm." *Computational Complexity* 22(2): 275-310 (2013)
[17] Ryan Williams: "Algorithms for Circuits and Circuits for Algorithms." *IEEE Conference on Computational Complexity* 2014: 248-261
[18] Michael Luby and Boban Velickovic. "On deterministic approximation of DNF." *Algorithmica*, 16(4/5):415–433 (1996)
[19] Aggelos Kiayias, Aris Pagourtzis, Kiron Sharma, Stathis Zachos: "Acceptor-Definable Counting Classes." *Panhellenic Conference on Informatics* 2001: 453-463.
[20] Evangelos Bampas, Andreas-Nikolas Göbel, Aris Pagourtzis, Aris Tentes: "On the connection between interval size functions and path counting." *Computational Complexity* 26(2): 421-467 (2017)
[21] Paul Walton Purdom Jr.: "Tree Size by Partial Backtracking." *SIAM J. Comput.* 7(4): 481-491 (1978)
[22] Pang C. Chen: "Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs." *SIAM J. Comput.* 21(2): 295-315 (1992)
[23] Philip Kilby, John K. Slaney, Sylvie Thiébaux, Toby Walsh: "Estimating Search Tree Size." *AAAI* 2006: 1014-1019
[24] Gleb Belov, Samuel Esler, Dylan Fernando, Pierre Le Bodic, George L. Nemhauser: "Estimating the size of search trees by sampling with domain knowledge. " *IJCAI* 2017: 473-479
[25] Martin Fürer, Shiva Prasad Kasiviswanathan: "An Almost Linear Time Approximation Algorithm for the Permanen of a Random (0-1) Matrix." *FSTTCS* 2004: 263-274
[26] Vaisman, R., Kroese, D.P.: "Stochastic Enumeration Method for Counting Trees." *Methodol. Comput. Appl. Probab.* 19: 31-73 (2017)
[27] Andris Ambainis, Martins Kokainis: "Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games." *STOC* 2017: 989-1002
[28] Leslie G. Valiant: "The Complexity of Computing the Permanent." *Theor. Comput. Sci.* 8: 189-201 (1979)
[29] M. Jerrum and A. Sinclair: "The Markov chain Monte-Carlo method: an approach to approximate counting and integration." *Approximation Algorithms for NP-hard Problems* (Dorit Hochbaum, ed.), PWS, pp. 482– 520 (1996)
[30] Richard M. Karp, Michael Luby, Neal Madras: "Monte-Carlo Approximation Algorithms for Enumeration Problems." *J. Algorithms* 10(3): 429-448 (1989)
[31] Dror Weitz: "Counting independent sets up to the tree threshold." *STOC* 2006: 140-149
[32] Stuart Geman, Donald Geman: "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Trans. Pattern Anal. Mach. Intell.* 6(6): 721-741 (1984)
[33] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E.: "Equations of state calculations by fast computing machines." *J. Chem. Phys.*, 21(6): 1087–1092 (1953)
[34] Hastings, W.: "Monte Carlo sampling methods using Markov chains and their application." *Biometrika*, 57: 97–109 (1970)
[35] Nayantara Bhatnagar, Allan Sly, Prasad Tetali: "Reconstruction Threshold for the Hardcore Model." *APPROX-RANDOM* 2010: 434-447
[36] Jean Barbier, Florent Krzakala, Lenka Zdeborova and Pan Zhang : "The hard-core model on random graphs revisited." *J. Phys.:* Conf. Ser. 473 : 012021 (2013)
[37] G. Brightwell and P. Winkler: "Hard constraints and the Bethe lattice: adventures at the interface of combinatorics and statistical physics." *Proc. Int'l. Congress of Mathematicians*, volume III : 605– 624 (2002)
[38] A. D. Sokal. "Chromatic polynomials, Potts models and all that." *Physica A: Statistical Mechanics and its Applications*, 279(1):324–332, (2000)
[39] D. J. Welsh and C. Merino: "The Potts model and the Tutte polynomial." *Journal of Mathematical Physics*, 41(3):1127–1152 (2000)
[40] F.-Y. Wu: "The Potts model." *Reviews of modern physics*, 54(1):235 (1982)