Analysis of Pure Literal Elimination Rule for Non-uniform Random (MAX) k-SAT Problem with an Arbitrary Degree Distribution

Oleksii Omelchenko, Andrei A. Bulatov

School of Computing Science, Simon Fraser University, Canada
{oomelche, abulatov}@cs.sfu.ca

Abstract

MAX k-SAT is one of the archetypal NP-hard problems. Its variation called random MAX k-SAT problem was introduced in order to understand how hard it is to solve instances of the problem on average. The most common model to sample random instances is the uniform model, which has received a large amount of attention. However, the uniform model often fails to capture important structural properties we observe in the real-world instances.

To address these limitations, a more general (in a certain sense) model has been proposed, the configuration model, which is able to produce instances with an arbitrary distribution of variables' degrees, and so can simulate biases in instances appearing in various applications.

Our overall goal is to expand the theory built around the uniform model to the more general configuration model for a wide range of degree distributions. This includes locating satisfiability thresholds and analysing the performance of the standard heuristics applied to instances sampled from the configuration model.

In this paper we analyse the performance of the pure literal elimination rule. We provide an equation that given an underlying degree distribution gives the number of clauses the pure literal elimination rule satisfies w.h.p. We also show how the distribution of variable degrees changes over time as the algorithm is being executed.

Introduction

MAX SAT and its variant MAX k-SAT are known NP-hard problems even for $k = 2$. Hence, it is unlikely there exists an efficient algorithm, unless P=NP, and so we must rely on heuristics and approximation algorithms. Therefore, it is natural to ask what heuristics are good at solving *typical* SAT instances.

There are many ways to define which instances are typical. One of the approaches is to construct a random distribution of SAT formulas, and call formulas sampled from the distribution as representative or typical. By constructing appropriate distributions we may imitate formulas coming from different domains.

The most well-studied random model of k-CNF formulas is the *uniform* model $F_k(n, rn)$, which samples equiprobably formulas having n variables and rn k-clauses, where quantity $r > 0$ is called *density* (Achlioptas 2009). It turns out that many key properties of the model depend on the density. One such property is the *satisfiability threshold*, a critical density $\rho_k = \rho_k(n)$, which depends on k (and maybe on n), such that with high probability (w.h.p.) ϕ is satisfiable whenever $r \leq (1 - \epsilon)\rho_k(n)$, and unsatisfiable if $r \geq (1 + \epsilon)\rho_k(n)$ for $\epsilon > 0$. Finding the satisfiability threshold for different values of k has been a very active and fruitful research direction, see, e.g., (Achlioptas 2009; Ding, Sly, and Sun 2015; Mitchell, Selman, and Levesque 1992; Larrabee and Tsuji 1992; Chvatal and Reed 1992; Goerdt 1996; Achlioptas 2001; Achlioptas and Sorkin 2000; Achlioptas and Peres 2004).

A similar quantity related to MAX k-SAT, the optimization version of k-SAT, is the (expected) number of clauses in a formula from $F_k(n, rn)$ that can be simultaneously satisfied. Although this value is not known, a straightforward argument gives a lower bound of $(1 - 2^{-k})rn$ and in the case of MAX 2-SAT (Coppersmith et al. 2003) places this value between $(3/4r + 0.34\sqrt{r})n$ and $(3/4r + 0.5\sqrt{r})n$.

Lower bounds on the satisfiability thresholds and on the number of satisfiable clauses of MAX k-SAT instances are often proved by analyzing relatively simple heuristics. For example, the $(1 - 2^{-k})rn$ bound is obtained as the expected number of satisfiable clauses, and this simple technique can be derandomized using the *method of conditional expectations* (Erdös and Selfridge 1973). A number of more sophisticated heuristics of different types have been analyzed as well, see, e.g., (Broder, Frieze, and Upfal 1993; Luby, Mitzenmacher, and Shokrollahi 1998; Molloy 2005; Achlioptas and Sorkin 2000; Achlioptas 2001; Kaporis, Kirousis, and Lalas 2002, 2006; Hajiaghayi and Sorkin 2003; Frieze and Suen 1996; Alekhnovich and Ben-Sasson 2007), for a survey see (Achlioptas 2009). In this paper we focus on the pure literal elimination heuristic. Given a CNF ϕ this algorithm simply selects a variable that appears in ϕ in only one polarity, that is, either only negated or only unnegated, assigns it the value that satisfies all its occurrences, and removes the clauses that get satisfied. (Broder, Frieze, and Upfal 1993) (see also (Kim 2008)) analysed the pure literal elimination rule for solving uniform random 3-SAT and MAX 3-SAT. They obtained an expression, which shows how many clauses are satisfied by each iteration of the algorithm, and proved that the maximal number of clauses

which can be satisfied by an assignment of pure literals is concentrated around the function $T_n(r, k)$. This function is however hard to calculate when r is constant. A somewhat easier to use expressions for estimating the efficacy of the pure literal elimination rule for any $k \geq 3$ were given in (Mitzenmacher 1997; Molloy 2005).

Most of the results we have mentioned so far concern the uniform random instances. However, there is a whole line of research suggesting that in many aspects random uniform formulas do not resemble natural instances coming from the real-world problems. For example, many industrial CNF instances exhibit *scale-free* property (Ansótegui, Bonet, and Levy 2009), they seem to consist of *clusters* of tightly connected communities (Ansótegui et al. 2019), have a rather smaller *fractal dimension* (Ansótegui et al. 2014), and other structural features not present in random instances (Ansótegui et al. 2008; Beyersdorff and Kullmann 2014). Moreover, it is unlikely that there exists a universal algorithm performing well on all instances coming from different domains.

One way to tune the Random k-SAT model so that random instances resemble instances coming from specific domains is to use more general random model. Several such models have been suggested, see, (Ansótegui, Bonet, and Levy 2009; Ansótegui, Bonet, and Levy 2019; Ansótegui et al. 2019; Ansótegui et al. 2014; Friedrich et al. 2017). In this paper, we use a random model called *configuration model*, parametrized by a sequence of random variables. In order to generate a k-CNF with n variables we fix a random variable ξ_i with values in $\mathbb{N}$ distributed accordingly to a distribution $\mathbb{D}_i$ for each variable v_i. The random variable ξ_i represents the degree or the number of occurrences of v_i in the formula. Then the degree ξ_i of v_i is sampled from $\mathbb{D}_i$ and we create ξ_i *clones* of v_i; each clone is negated with probability 1/2. Finally, the set of clones is randomly partitioned into k-sets to form k-clauses.

Depending on the distributions of ξ_i the configuration model allows one to generate a wide variety of random k-CNFs. For example, if every ξ_i is a constant we are dealing with CNFs with a fixed degree sequence. This case has been studied in (Cooper, Frieze, and Sorkin 2007) for 2-SAT. If every $\mathbb{D}_i$ is the same Poisson distribution, the configuration model is also known as the Poisson Cloning model. (Kim 2004) proves that in many aspects the Poisson Cloning model is equivalent to the uniform Random (MAX) k-SAT. (Omelchenko and Bulatov 2021b) and (Omelchenko and Bulatov 2021a) study several aspects of the configuration model when the distributions are heavy-tailed.

The overall research goal is therefore to expand the results known for the uniform model to the configuration model. The main difficulty in this enterprise is that it is not always possible to use the same well established and efficient tools in both theoretical and practical aspects of the problem. On the theoretical side, when the distributions $\mathbb{D}_i$ are not as well behaved as Poisson, especially if they are heavy-tailed or do not have higher moments, the standard probability theory tools such as concentration inequalities, do not apply, and have to be replaced with more complicated and less efficient ones. For example, (Borovkov and Borovkov 2008) and (Omelchenko and Bulatov 2019) obtained some (relatively weak) concentration bounds for heavy-tailed distributions. On the practical side, weak concentration properties mean that in order to achieve a visible trend in experimental results one needs to handle formulas with billions of variables.

In this work we study how the pure literal elimination rule performs in the configuration model, and how the distribution of degrees of variables affects the number of clauses which the rule can satisfy. Our main contribution is Theorem 1. The theorem gives an equation that depends on the "averaged" distribution of degrees and only assumes that the random variables ξ_i have finite first moments. Solving the equation one can find how many clauses the pure elimination rule satisfies. Note that a similar result for the Poisson Cloning model was obtained by (Kim 2008). Our secondary result is Lemma 6, which shows how the degree distribution evolve as the pure literal heuristics transforms the instance. This information is useful for further analysis of k-CNF formulas obtained *after* they have been processed with the pure literal elimination heuristic.

Our analysis uses two important tools: the Weak Law of Large Numbers and the Wormald's differential equations method. The former one gives us some concentration property, while the latter is a standard tool in analysis of SAT algorithms, see (Achlioptas 2001; Achlioptas and Sorkin 2000).

The paper is organized as follows. After reminding the basic definitions and notation, we introduce the configuration model. The main part of the paper is the "Analysis of Pure Literal Elimination Algorithm" section, where we give a number of lemmas needed to obtain the main result of this work, Theorem 1. We conclude the paper with a number of experimental results which show how the pure literal elimination rule performs on different random instances obtained from the configuration model in reality. Due to space restrictions many proofs will be given in Supplemental materials.

Notations & Definitions

MAX SAT, MAX k-SAT, and Random MAX k-SAT. Let $x_1, \ldots, x_n$ be boolean variables, and n will always denote their number. A *literal* is either a variable or its negation. A literal is *negative*, when it is a negated variable, and *positive* otherwise. A *clause* is a disjunction of literals, and a k-clause is a clause with exactly k not necessarily distinct literals. Then a *CNF formula* is a set of clauses, while a *k-CNF formula* is a set of k-clauses. So, we treat every CNF formula as a set of clauses, and each clause is a set of k literals.

The *MAX SAT* problem is an optimization problem where given a CNF formula ϕ we need to determine an assignment that satisfies the maximal number of clauses in ϕ. The *MAX k-SAT* is a special case of MAX SAT, where the given formula ϕ is k-CNF.

Random MAX k-SAT problem is a variant of MAX k-SAT problem, where instead of an arbitrary k-CNF formula we are given a random k-CNF formula, sampled from some probabilistic distribution over k-CNF formulas with n vari-

ables. The distribution we use in this paper is configuration model (see the "Configuration Model" section for details).

Probability Reminder. We say that a sequence of events A_n happens *with high probability* (w.h.p.), when $\Pr[A_n] \to 1$ with $n \to \infty$. We use acronyms *r.v.* for "random variable", *r.vs.* for "random variables", and *u.a.r.* for "uniformly at random". We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

The next lemma shows that the sum of n independent r.vs. $\xi_1, \xi_2, \ldots, \xi_n$ with finite expectations $\mathbb{E}|\xi_i| < \infty$ is concentrated around its mean, i.e. $\sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n} \mathbb{E}\xi_i + o(n)$ w.h.p. This result is known as the *Weak Law of Large Numbers* (WLLN), however, its classical versions require the r.vs. ξ_i's to be either identically distributed and/or to have moments beyond expectation. Variation of the WLLN we need and state below does not ask for such premises, while its proof is only a minor modification of the now classical proof of Khintchin's Law of Large Numbers (Borovkov 2013).

Lemma 1 (WLLN)**.** *Let $\xi_1, \xi_2, \ldots, \xi_n$ be a collection of n independent r.vs. with finite expectation $\mathbb{E}|\xi_i| < \infty$. Then it holds w.h.p.*

$$\sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n} \mathbb{E}\xi_i + o(n).$$

To analyse the evolution of algorithms, we rely on Wormald's *differential equations method* theorem (Wormald 1995, 1999). Some improvements to the method and its applications can be found in (Bohman 2009; Bohman and Keevash 2010; Warnke 2014, 2019). The main idea of the method is simple. Suppose you have a number of co-evolving in time scalar random processes $X_i(t)$, where $t \in \mathbb{N}$ and $i \in [\ell]$. Our goal is to describe how those processes evolve, and approximate their trajectory in time. The processes may affect each other in some intricate way during their evolution, which complicates their description. The theorem states that if the processes are "good", namely: (a) given state of all processes at any time t, the expected one time change of each process can be well-approximated (up to $o(1)$ factor) by some function $f_i\left(\frac{t}{n}, \frac{X_1(t)}{n}, \frac{X_2(t)}{n}, \ldots, \frac{X_\ell(t)}{n}\right)$; (b) it is highly unlikely that any process during its evolution will jump over $n^{1/2}$; and finally (c) functions f_i's satisfy Lipschitz condition, then we can be confident that the processes are concentrated around the deterministic trajectory, which can be obtained from a solution to the system of differential equations, for as long as all the above three conditions hold (see Supplemental Materials for a precise statement of the theorem).

Finally, we denote by $\xi \sim \mathbb{D}$ the fact that a r.v. ξ is distributed according to a probability distribution $\mathbb{D}$.

Configuration Model. *Configuration model* $\mathbb{C}_n^k((\xi_i)_{i=1}^n)$ is a probability distribution of k-CNF formulas with n variables. The model is parametrized by an ordered sequence of n independent random variables ξ_i with support on $\mathbb{N}^+$. We write $\phi \sim \mathbb{C}_n^k((\xi_i)_{i=1}^n)$, when formula ϕ is sampled from the configuration model. We do not require ξ_i's to be identically distributed as each $\xi_i \sim \mathbb{D}_i$ may come from its own distribution $\mathbb{D}_i$. But we do require them to be independent and with finite expectation $\mathbb{E}\xi_i < \infty$ for all $i \in [n]$.

Sampling formulas from $\mathbb{C}_n^k((\xi_i)_{i=1}^n)$ is usually done by constructing such formulas. By $\mathcal{V}(\phi)$ we denote the set of all variables in a sampled formula ϕ. Then $|\mathcal{V}(\phi)| = n$ always. The *degree* of a variable $v \in \mathcal{V}(\phi)$ is the number of times it appears in ϕ as an atom; each occurrence of v in ϕ we call a *clone* of v. Let $\mathcal{S}(\phi)$ be the set of all clones of all variables in ϕ. Each clone in $\mathcal{S}(\phi)$ can be positive or negative thus corresponding to a positive or negative literal. A literal in ϕ is said to be *pure literals*, if ϕ does not contain its negation.

A formula in the configuration model is generated in two steps. The first step, called CREATECLONES$((\xi_i)_{i=1}^n, k)$, starts with a sequence $(\xi_i)_{i=1}^n$ of random variables and creates clones of variables and assigns a sign to each of them. In order to do that for each $i \in [n]$ we sample degree d_i of each variable $v_i \in \mathcal{V}(\phi)$ from ξ_i and create d_i clones of v_i. If the total number of clones is not a multiple of k we discard these clones and start all over. Otherwise we assign a sign to each clone equiprobably. Let $\mathcal{S}$ denote the resulting set of (signed) clones. During the process we say that a clone $c \in \mathcal{S}(\phi)$ is *paired*, when there is a clause in ϕ, which contains this clone c; otherwise, the clone is said to be *unpaired*.

In the second step, called CONFIGURATIONMODEL-CNF$(\mathcal{S}, k)$, we randomly partition the set $\mathcal{S}$ into k-element subset that form clauses of the formula.

Note that as long as the second step results in a random partition of $\mathcal{S}$ into k-element subsets, it makes no difference how exactly partitioning is done. The most basic way is to pick k-element subsets from $\mathcal{S}$ until all the clones are put into some clause. However, often it is convenient to choose a more elaborate way, which may ease the analysis of the process. For example, we can use any rule (deterministic or probabilistic) to pick the very first clone for every clause, since in the end all clones must end up in some clauses. This observation was one of the key features used in (Cooper, Frieze, and Sorkin 2007; Omelchenko and Bulatov 2021b) to produce an alternative algorithm called TSPAN for constructing 2-CNF formulas. On the other hand, this freedom of picking the very first clone is coupled with the restriction that the other $k-1$ clones of every clause must be picked u.a.r. without replacement from the set of unpaired clones. When these two conditions are satisfied, we can be sure that the formulas produced by the alternative algorithms are equivalent to the formulas from $\mathbb{C}_n^k((\xi_i)_{i=1}^n)$.

To end this section, we introduce several quantities, which we frequently use in the subsequent analysis. First, let $S_n = |\mathcal{S}(\phi)|$ be the total number of clones in ϕ. Then, clearly, the number of clauses in ϕ is $|\phi| = \frac{S_n}{k}$. Let $\gamma := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\xi_i$ denote the average degree of variables, and since all $\mathbb{E}\xi_i < \infty$, it follows that $\gamma < \infty$. Moreover, as the expectations of ξ_i's are finite, we have the following simple result:

Lemma 2. *It holds that $S_n = (1 + o(1))\,\gamma n$ w.h.p.*

Let ξ be a r.v. over $\mathbb{N}$ with probability distribution function

$$\Pr[\xi = d] = \frac{1}{n}\sum_{i=1}^{n} \Pr[\xi_i = d] \quad \text{for all } d \in \mathbb{N}. \quad (1)$$

It turns out that many quantities in the configuration model can be expressed via this "averaged" r.v. ξ. For example, $\gamma = \mathbb{E}\xi$, and so $S_n = (1 + o(1))\, n\mathbb{E}\xi$ w.h.p. Next, we will abuse notation by using the same letter p with different subscripts to denote two different probabilities distinguishing them by the number of indices. Let $p_d := \Pr[\xi = d]$, while we use $p_{i,j}$ to denote the probability that a u.a.r. chosen variable produces i positive clones and j negative clones. The next lemma expresses $p_{i,j}$ in terms of p_{i+j}.

Lemma 3. *The probability a randomly chosen variable produces exactly i positive and j negative clones is*

$$p_{i,j} = 2^{-(i+j)} \binom{i+j}{i} \Pr[\xi = i+j] = 2^{-(i+j)} \binom{i+j}{i} p_{i+j}.$$

Finally, let $N_{i,j}$ be the number of *variables*, which produced exactly i positive and j negative clones. Then by Lemma 1 and Lemma 3, it readily follows that w.h.p.

$$N_{i,j} = (1 + o(1))\, np_{i,j}. \tag{2}$$

Analysis of Pure Literal Elimination

As was mentioned in the Introduction, our goal here is to analyse the efficiency of the pure literal elimination heuristic on random k-CNF instances sampled from the configuration model.

The main entities of interest in pure literal elimination rule are pure literals. As was mentioned in previous sections, we call a literal ℓ pure in a $k-$CNF formula ϕ if its complement literal $\bar{\ell}$ does not appear in ϕ. Hence, we can safely satisfy ℓ, and eliminate all clauses in which ℓ is present, thus, simplifying ϕ. We continue this elimination process as long as we have pure literals to satisfy. Observe, that during clause elimination it may happen that a literal, which was not initially pure, becomes pure if all occurrences of its complement literal happen to be in the eliminated clauses, which fuels the algorithm with new pure literals.

For the sake of analysis we slightly modify the algorithm (keeping the same outcome) by employing deferred decision. That is, instead of feeding the algorithm a complete k-CNF formula, we instead "reveal" only those clauses that contain pure literals, satisfy them, update the set of pure literals, and repeat this process as long as we have unpaired clones of pure literals. This way the unrevealed part of the formula remains random, and so probabilistic analysis can be applied.

The deferred decision version of the pure literal elimination algorithm is given in Algorithm 1. We start with creation of clones for all n variables by calling the CREATECLONES$((\xi_i)_{i=1}^n, k)$ procedure with $(\xi_i)_{i=1}^n$ being a sequence of r.vs. from which degrees are sampled. It may happen that some variables possess only clones of one sign (it happens w.h.p. when we have $p_d > 0$ for some constant d). Such variables form an initial pool of pure literals.

We generate formula ϕ iteratively, one clause at a time. We start with an empty formula. Next, as long as we have pure literals, we pick one of them u.a.r. without replacement. Observe that if we satisfy the chosen literal l, then every clause in which a clone of l is present is satisfied as well.

Algorithm 1: Pure literal elimination algorithm with deferred decision.

```
1:  function PURELITERAL((ξ_i)_{i=1}^n, k)
2:      φ_1 ← {};
3:      S ← CREATECLONES((ξ_i)_{i=1}^n, k);
4:      while there are clones in S with no complementary
        counterparts do
5:          c ← u.a.r. picked clone from the set of clones
            with no complementary counterparts in S;
6:          Satisfy the literal ℓ associated with c;
7:          C ← the set of all unpaired clones of ℓ;
8:          while C ≠ ∅ do
9:              c ← pick an arbitrary clone from C;
10:             cl ← sample k − 1 clones u.a.r. w/o replace-
                ment from S − {c};
11:             cl ← {c} ∪ cl;
12:             Mark cl as satisfied;
13:             φ_1 ← φ_1 ∪ {cl};
14:             C ← C − cl;
15:             S ← S − cl;
16:         end while
17:     end while
18:     return CONFIGURATIONMODELCNF(S, k);
19: end function
```

Hence, to know how many clauses we satisfy by satisfying l, we calculate how many clauses contain clones of l. To do so take all clones of the chosen pure literal l and distribute them in the following manner. Pick an arbitrary unpaired clone of l and form a new 1-clause out of it. Next, to finish the formation of the new clause, sample other $k-1$ unpaired clones from $\mathcal{S}$ (line 1.10), and append them to the newly created 1-clause (line 1.11). We obtain a complete k-clause, which we add to the formula ϕ (line 1.13). Since we have paired clones in the new clause, they cannot appear in any other clauses, and so we remove them from the sets $\mathcal{C}$ and $\mathcal{S}$ (lines 1.14 and 1.15 respectively). We continue formation of new clauses containing clones of l as long as there remain unpaired clones of it. This generation process of clauses containing clones of l continues until we deplete all its unpaired clones. After that we pick next pure literal, if there exists any, and continue producing new clauses. Observe that the algorithm not just creates new clauses but actually satisfies them as well.

Just like in the original algorithm, it may happen that during elimination process some literals, which were not pure initially, become pure, when all clones of their complements get paired. Or, in other words, all their complement counterparts appear only in clauses, which we have already satisfied. In that case we add the newly-formed pure literals to the pool of all not-yet-paired pure literals, and continue the pairing process until we exhaust all the unpaired clones of pure literals. When all pure literals are used up we generate a k-CNF from the remaining clones in the usual way.

To analyse Algorithm 1, we need to introduce a few quantities that play major role in the analysis. The first one is the number of *variables* with i unpaired positive clones and j unpaired negative clones at time $t \in \mathbb{N}$, which we denote

by $N_{i,j}(t)$. Time t here measures how many clauses Algorithm 1 has formed, or, equivalently, the total number of times the inner loop (lines 8-16) has been executed. Hence, initially we have $t = 0$. As we form new clauses and t increases, $N_{i,j}(t)$ changes as well. If it happens that Algorithm 1 runs long enough for t to reach $|\phi| = \frac{S_n}{k}$, we conclude that the pure literal elimination rule was able to find a satisfying assignment. Also note that due to the random nature of the pairing process, $N_{i,j}(t)$ for all $i, j \geq 0$ are in fact random processes.

Note that $\sum_{i\geq 1}(N_{i,0}(t) + N_{0,i}(t))$ measures the number of pure literals at time t, since $N_{i,0}(t)$ and $N_{0,i}(t)$ count the number of variables with i unpaired clones all of which belong to a single literal. Hence, the algorithm runs as long as $\sum_{i\geq 1}(N_{i,0}(t) + N_{0,i}(t)) > 0$. Let t_0 denote the moment of time, when the algorithm stops, i.e. when the quantity $\sum_{i\geq 1}(N_{i,0}(t_0) + N_{0,i}(t_0))$ hits zero. Note that t_0 is a random variable, due to the random nature of dynamics of the processes $N_{i,j}(t)$. Clearly, to study how well the pure literal elimination heuristic performs on random instances from $\mathbb{C}_n^k((\xi_i)_{i=1}^n)$, we need to learn the distribution of t_0.

In order to apply the differential equations method to approximate dynamics of the processes $N_{i,j}(t)$, we must verify that the processes are "good". The next lemma proves that the processes $N_{i,j}(t)$ are indeed good candidates for approximation using the differential equations method.

Lemma 4. *Let $H(t) := \bigcup_{0\leq t'\leq t}\bigcup_{i,j\geq 1}\{N_{i,j}(t')\}$ be the complete history of the evolution of the processes $N_{i,j}(t')$ up to and including time t. Then for all $i, j \geq 1$ and all $0 < t < t_0$, it holds:*

1. $\mathbb{E}\left[N_{i,j}(t+1) - N_{i,j}(t) \mid H(t)\right]$
$$= f_{i,j}\left(\frac{t}{n}, \frac{N_{i,j}(t)}{n}, \frac{N_{i+1,j}(t)}{n}, \frac{N_{i,j+1}(t)}{n}\right) + o(1),$$
where
$$f_{i,j}\left(\tau, n_{i,j}(\tau), n_{i+1,j}(\tau), n_{i,j+1}(\tau)\right) = -\frac{k-1}{\gamma - k\tau}(i+j)n_{i,j}(\tau) + \frac{k-1}{\gamma - k\tau}\Big[(i+1)n_{i+1,j}(\tau) + (j+1)n_{i,j+1}(\tau)\Big];$$
2. $\Pr\left[\left|N_{i,j}(t+1) - N_{i,j}(t)\right| > k \mid H(t)\right] = 0$;
3. $f_{i,j}\left(\tau, n_{i,j}(\tau), n_{i+1,j}(\tau), n_{i,j+1}(\tau)\right)$ *is continuous and satisfies Lipschitz condition for* $0 \leq \tau \leq \frac{t_0}{n} < \frac{\gamma}{k}$.

Remark. *As it follows from the lemma, we use $n_{i,j}(\tau) = \frac{N_{i,j}(\tau n)}{n}$ with $\tau = \frac{t}{n}$, to denote the scaled number of variables with i positive and j negative unpaired clones at the scaled time τ.*

Hence, now we construct the system of differential equations describing the dynamics of the scaled number of variables $n_{i,j}(\tau)$ for all $i, j \geq 1$

$$\begin{aligned}\frac{d\, n_{i,j}}{d\tau} &= f_{i,j}\left(\tau, n_{i,j}(\tau), n_{i+1,j}(\tau), n_{i,j+1}(\tau)\right)\\ &= -\frac{k-1}{\gamma - k\tau}(i+j)n_{i,j}(\tau)\\ &+ \frac{k-1}{\gamma - k\tau}\Big[(i+1)n_{i+1,j}(\tau) + (j+1)n_{i,j+1}(\tau)\Big], \qquad (3)\end{aligned}$$

with initial values $n_{i,j}(0) = \frac{N_{i,j}(0)}{n} = 2^{-(i+j)}\binom{i+j}{i}p_{i+j}$. which follows from (2) after plugging in the initial number of variables $N_{i,j}$. The system has a single solution.

Lemma 5. *The system* (3) *defined for all $i, j \geq 1$ together with initial values $n_{i,j}(0) = 2^{-(i+j)}\binom{i+j}{i}p_{i+j}$ has a unique solution*

$$n_{i,j}(\tau) = 2^{-(i+j)}\binom{i+j}{i} \times \sum_{\ell\geq 1}\binom{\ell}{i+j} z(\tau)^{i+j}\left(1 - z(\tau)\right)^{\ell-(i+j)},$$

where

$$z(\tau) = \left(1 - \frac{k\tau}{\gamma}\right)^{1-\frac{1}{k}}.$$

Now, we apply differential equations method to approximate dynamics of the processes $N_{i,j}(t)$.

Lemma 6. *Let $N_{i,j}(t)$ be the number of variables with i positive and j negative unpaired clones at time t, where $i, j \geq 1$. Then it holds w.h.p. that $N_{i,j}(t) = n \cdot n_{i,j}\left(\frac{t}{n}\right) + o(n)$, where function $n_{i,j}\left(\frac{t}{n}\right)$ is the solution from Lemma 5 for all $i, j \geq 1$.*

Now after learning how $N_{i,j}(t)$'s evolve over time, we determine when the pure literal elimination algorithm stops and how many clauses it satisfies. First let $C(t)$ denote the number of unpaired *clones* of pure literals at time t, and let $c(\tau) := \frac{C(\tau n)}{n}$ be its scaled version at scaled time τ. Then we have the following result:

Lemma 7. *The number of unpaired pure clones at time $0 \leq t \leq t_0$ is*

$$C(t) = n \cdot c\left(\frac{t}{n}\right) + o(n),$$

where

$$\begin{aligned}c(\tau) &= \gamma - k\tau - z(\tau)\gamma + z(\tau)\sum_{\ell\geq 1}\ell p_\ell\left(1 - \frac{z(\tau)}{2}\right)^{\ell-1}\\ &= \gamma - k\tau - z(\tau)\gamma - \frac{z(\tau)}{2}\frac{d}{dx}\left[G\left(1 - \frac{x}{2}\right)\right]\Big|_{x=z(\tau)}.\end{aligned}$$

Here $G(x) = \mathbb{E}\left[x^\xi\right]$ is the probability-generating function of the r.v. ξ given by (1)*.*

Now, given Lemma 7, the stopping time of Algorithm 1 becomes almost self-evident.

Theorem 1. *Let $\phi \sim \mathbb{C}_n^k(\{\xi_i\}_{i=1}^n)$, where $\mathbb{E}\xi_i < \infty$. Then the pure literal elimination heuristic satisfies w.h.p. $(1 + o(1))\,\tau_0 n$ clauses with $\tau = \tau_0 > 0$ being the smallest solution of the equation*

$$\gamma - k\tau - z(\tau)\gamma - \frac{z(\tau)}{2}\frac{d}{dx}\left[G\left(1 - \frac{x}{2}\right)\right]\Big|_{x=z(\tau)} = 0, \quad (4)$$

where $z(\tau)$ is the function defined in Lemma 5.

Proof. Recall that the Algorithm 1 stops as soon as all pure clones get paired. The number of unpaired pure clones at time t is $C(t) = n \cdot c\left(\frac{t}{n}\right) + f(n)$ w.h.p., where $f(n) = o(n)$,

as it follows from Lemma 7. Introduce an increasing function $\lambda(n)$, such that $f(n) \ll \lambda(n) \ll n$. For the sake of analysis instead of stopping the algorithm when all pure clones get exhausted, we stop its execution when $C(t)$ becomes less than $\lambda(n)$. Although, the algorithm could still continue working and satisfy more clauses, but as it will become apparent from the proof, difference in the number of clauses that Algorithm 1 satisfies, which are not satisfied by stopping the algorithm at the $\lambda(n)$ mark is of order at most $\lambda(n) = o(n)$, which is negligible comparing to the number of clauses we do satisfy.

Hence, it follows that the algorithm's stopping time t_0 is the time, when $C(t_0) \leq \lambda(n)$ for the first time. When we consider the same condition in terms of the scaled version of the number of unpaired pure clones, we have that $\tau_0 = \frac{t_0}{n}$ is the time, when $c(\tau_0) = \frac{C(t_0)}{n} \leq \frac{\lambda(n)}{n} = o(1)$. In other words, $\tau = \tau_0$ is the first time $c(\tau)$ crosses 0 (up to $o(1)$ additive term, since $c(\tau)$ is a smooth continuous function).

Therefore, by finding the closest to zero $\tau_0 > 0$, which satisfies equation (4) gives us the stopping time of Algorithm 1, from which we obtain the number of satisfied clauses $t_0 = (1 + o(1))\,\tau_0 n$ (the $(1 + o(1))$ multiplicative factor here is caused by us stopping the algorithm at $\lambda(n) = o(n)$ level instead of when $C(t)$ turns to 0). □

Note that if $\xi \sim D(\theta)$ for some parametric probability distribution over $\mathbb{N}^+$, then the evolution of the scaled number of pure clones can be itself viewed as not only a function of scaled time τ, but also of parameter θ, i.e. $c(\tau) = c_\theta(\tau)$. In that case the most interesting values of θ are the ones, when $c_\theta(\tau)$ only touches zero at $\tau = \tau_0$ and bounces back as τ increases. In that case we should see sharp and sudden changes in the efficacy of the pure literal algorithm in the neighbourhood of such critical θ's. We demonstrate examples of this phenomenon in the next section, where we discuss our experiment.

Experiments

To verify our results experimentally, we picked 4 probability distributions over $\mathbb{N}^+$, coming from different classes:

1. *Subgaussian distribution* $S(\mu, \sigma)$ with probability distribution function $\Pr[S(\mu,\sigma) = x] = C_{\mu,\sigma} e^{-\left(\frac{x-\mu}{\sigma}\right)^2}$, where $C_{\mu,\sigma}$ is its normalizing constant. $S(\mu, \sigma)$ resembles normal distribution but with support for whole numbers only. As the name suggests, subgaussian distribution comes from the class of subgaussian distributions.
2. *Poisson distribution* $P(\lambda)$ with distribution $\Pr[P(\lambda) = x] = \frac{\lambda^{x-1} e^{-\lambda}}{(x-1)!}$. The distribution is identical to classical Poisson distribution but with support defined only for positive natural numbers. The Poisson distribution is an example of exponentially decaying distributions.
3. *Log-normal distribution* $L(\eta, \sigma)$ distributed as $\Pr[L(\eta,\sigma) = x] = \frac{M_{\eta,\sigma}}{x} e^{-\left(\frac{\log x - \mu}{\sigma}\right)^2}$, where $M_{\eta,\sigma}$ is the normalizing constant. $L(\eta, \sigma)$ was inspired by the classical continuous log-normal distribution, and it belongs to the class of distributions with tail's decay rate being in-between exponentially decaying distributions and the ones with polynomially heavy tails.
4. *Zeta distribution* $Z(\alpha)$ with distribution $\Pr[Z(\alpha) = x] = \frac{x^{-\alpha}}{\zeta(\alpha)}$, where $\zeta(\alpha) = \sum_{x \geq 1} x^{-\alpha}$ is the Riemann zeta function. $Z(\alpha)$ is a canonical example of a discrete heavy-tailed distribution.

Remark. *For $S(\mu, \sigma)$ and $L(\eta, \sigma)$ we fix their second parameter $\sigma = 1$, and now all 4 distributions parametrized by a single parameter.*

The goal of the experiment is to measure performance of the pure literal elimination rule on random k-CNFs formulas having different degree distributions. The distributions we picked cover a wide spectrum of tail dynamics at infinity, hence, this should help us to develop an intuition on how distributions of degrees affect performance of the pure literal elimination.

Let us denote by $D(\theta)$ an arbitrary one-parameter probability distribution with θ being the distribution's parameter. Let $\mathcal{D} = \{S(\mu, 1), P(\lambda), L(\eta, 1), Z(\alpha)\}$ be the set of the 4 chosen distributions. For every fixed $D(\theta) \in \mathcal{D}$ let θ_γ be the value of its parameter, so that the expected value $\mathbb{E}[D(\theta_\gamma)] = \gamma$.

Next we describe setup of the experiment. First, we fix a list of 13 average degrees $\mathcal{E} = \{3, 4, 5, \ldots, 14, 15\}$. Then for every $D(\theta) \in \mathcal{D}$ and every $\gamma \in \mathcal{E}$, we estimate θ_γ. Next, we produce random[1] 3-CNF formulas $\phi \sim \mathbb{C}^3_n((\xi_i)_{i=1}^n)$, where $\xi_i \sim D(\theta_\gamma)$ are i.i.d. r.vs. with $\mathbb{E}\xi_i = \gamma$, and n, the number of variables of sampled formulas, we fix to be 100,000. We solve the generated formulas with pure literal elimination algorithm, and record how many clauses the algorithm satisfies. Let the number of satisfied clauses be T, and its scaled version $\tau_0 = \frac{T}{n} = 10^{-5} T$. Then with every sampled formula ϕ we associate a tuple $(\phi, D(\theta_\gamma), \gamma, \tau_0)$, where ϕ is the formula itself, $D(\theta_\gamma)$ is the distribution, which was used for degree sequence generation, γ is the average degree, and finally τ_0 is the scaled number of clauses of ϕ that the pure literal algorithm was able to satisfy. By sampling many formulas for each $D(\theta) \in \mathcal{D}$ and $\gamma \in \mathcal{E}$, we should be able to get a good estimate of the "true" τ_0 value. In our experiment we produced 7,777 instances for each distribution and each average degree γ.

After collecting enough data points, we averaged τ_0 for each γ and every $D(\theta)$, and plotted this estimate of τ_0 on Figure 1. We also calculated the 0.99 confidence interval of the obtained estimates, which we also added to the figure (surprisingly, whiskers, marking the bounds of the confidence intervals, are almost indiscernible, which means that the estimates seem to converge rapidly to their true values. We say it is rather surprising, since, based on our experience, quantities derived from heavy-tailed distributions quite often do not exhibit nice convergence rates. One case of this phenomenon can be seen, for example, in (Omelchenko and Bulatov 2021b)).

Also for each $D(\theta) \in \mathcal{D}$ we calculate numerically $\tau_0 = \tau_0(\theta)$ by solving the equation (4). Note that now $\tau(\theta)$ can

[1] We use std::random_device() from the standard library of C++ to generate seeds for PRG.

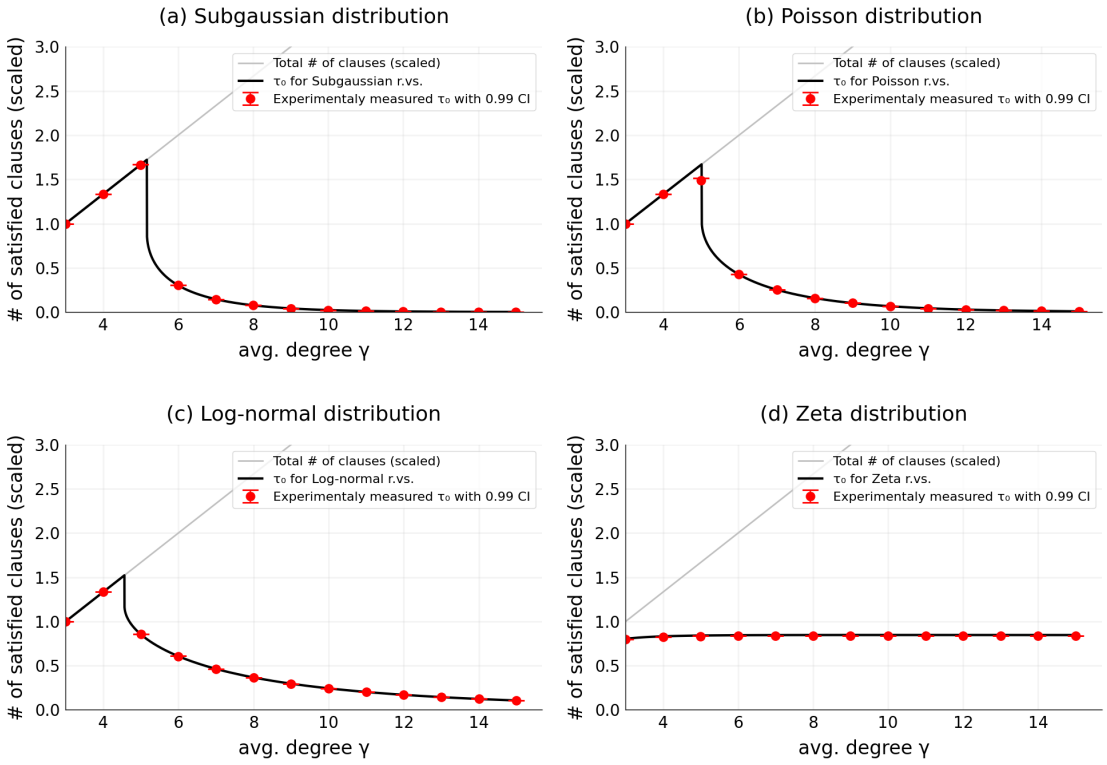

Figure 1: Experimentally obtained efficacy of the pure literal heuristic on formulas from $\mathbb{C}_n^3((\xi_i)_{i=1}^n)$ vs. theoretically predicted efficacy, where r.vs. ξ_i's follow (a) subgaussian, (b) Poisson, (c) log-normal, and (d) zeta distribution. Black curves represent the scaled number of satisfied clauses $\tau_0(\gamma)$ as a function of γ. Grey slanted lines show the typical number of total clauses, and round markers are experimentally obtained estimates of $\tau_0(\gamma)$ with whiskers showing bounds of the 0.99 confidence intervals.

be viewed as a function of the distribution's parameter, which, in turn, controls the expected value $\gamma(\theta)$. Hence, we can plot $(\gamma(\theta), \tau_0(\theta))$ by varying θ. We pick θ's so that $\gamma(\theta) \in [3, 15]$, and calculate corresponding $\tau_0(\theta)$. Obtained functions of the "true" values of τ_0 are represented as black curves on Figure 1. Additionally, we plot as grey slanted lines the scaled average number of clauses that formulas with average degree γ have, i.e. for each γ this quantity is equal to $\frac{|\phi|}{n} = \frac{\gamma}{k} = \frac{\gamma}{3}$.

As it follows from the experiment, we have obtained good evidence supporting Theorem 1. The "true" values of the scaled number of satisfied clauses τ_0, given by equation 4, predict well how the pure literal elimination performs in reality. There are some more observations we can make from the experimental data. It seems the heuristic performs really well, when the distribution of degrees has a rapidly decaying tail (like Poisson and subgaussian), and γ is at most around 5. However, for even slightly larger γ's performance of the pure literal heuristic drops significantly and it becomes almost useless. Hence, this average degree $\gamma = 5$ and the corresponding values of θ_γ for Poisson and subgaussian distributions seem to be critical.

However, the more heavy-tailed distributions exhibit a somewhat reversed dynamic. The pure literal algorithm does not seem to perform well on them for lower values of γ. For example, the algorithm was able to satisfy only around 1/2 of all clauses of formulas with log-normally distributed degrees when $\gamma = 5$, while on subgaussian r.vs. it was able to satisfy all or almost all clauses at the same average degree. But as we increase γ, the rule starts performing better on the more heavy-tailed distributions comparing to their light-tailed counterparts. Consider, for example, Poisson and zeta distributions. When $\gamma > 10$, the algorithm is unable to satisfy any reasonable number of clauses in case of Poisson distributed degrees, but for zeta distribution efficacy of the pure literal elimination rule drops much and much slower as we increase γ.

References

Achlioptas, D. 2001. Lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 265.

Achlioptas, D. 2009. Random Satisfiability. *Frontiers in Artificial Intelligence and Applications*, 185.

Achlioptas, D.; and Peres, Y. 2004. The Threshold for Random k-SAT Is 2^k Log 2-O(k). *Journal of the American Mathematical Society*, 17(4): 947–973.

Achlioptas, D.; and Sorkin, G. 2000. Optimal myopic algo-

rithms for random 3-SAT. *Annual Symposium on Foundations of Computer Science - Proceedings*.

Alekhnovich, M.; and Ben-Sasson, E. 2007. Linear Upper Bounds for Random Walk on Small Density Random 3-CNFs. *SIAM J. Comput.*, 36(5): 1248–1263.

Ansótegui, C.; Bonet, M. L.; Giráldez-Cru, J.; and Levy, J. 2014. The Fractal Dimension of SAT Formulas. In Demri, S.; Kapur, D.; and Weidenbach, C., eds., *Automated Reasoning*, 107–121. Cham: Springer International Publishing. ISBN 978-3-319-08587-6.

Ansótegui, C.; Bonet, M. L.; Giráldez-Cru, J.; Levy, J.; and Simon, L. 2019. Community Structure in Industrial SAT Instances. *J. Artif. Intell. Res.*, 66: 443–472.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. On the Structure of Industrial SAT Instances. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, 127–141. Berlin, Heidelberg: Springer-Verlag. ISBN 3642042430.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2019. Phase Transition in Realistic Random SAT Models. In Sabater-Mir, J.; Torra, V.; Aguiló, I.; and Hidalgo, M. G., eds., *Artificial Intelligence Research and Development - Proceedings of the 22nd International Conference of the Catalan Association for Artificial Intelligence, CCIA 2019, Mallorca, Spain, 23-25 October 2019*, volume 319 of *Frontiers in Artificial Intelligence and Applications*, 213–222. IOS Press.

Ansótegui, C.; Bonet, M. L.; Levy, J.; and Manyà, F. 2008. Measuring the Hardness of SAT Instances. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, 222–228. AAAI Press. ISBN 9781577353683.

Beyersdorff, O.; and Kullmann, O. 2014. Unified Characterisations of Resolution Hardness Measures. In *SAT*.

Bohman, T. 2009. The triangle-free process. *Advances in mathematics (New York. 1965)*, 221(5): 1653–1677.

Bohman, T.; and Keevash, P. 2010. The early evolution of the H-free process. *Inventiones mathematicae*, 181(2): 291–336.

Borovkov, A.; and Borovkov, K., eds. 2008. *Asymptotic analysis of random walks: heavy-tailed distributions*, volume 118 of *Encyclopedia of mathematics and its applications*. Cambridge University Press. ISBN 978-0-51172-139-7.

Borovkov, A. A. 2013. *Probability Theory by Alexandr A. Borovkov.* Universitext. Springer London : Imprint: Springer, 1st ed. 2013. edition. ISBN 1-4471-5200-X.

Broder, A. Z.; Frieze, A. M.; and Upfal, E. 1993. On the Satisfiability and Maximum Satisfiability of Random 3-CNF Formulas. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, 322–330. USA: Society for Industrial and Applied Mathematics. ISBN 0898713137.

Chvatal, V.; and Reed, B. 1992. Mick gets some (the odds are on his side) (satisfiability). In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, 620–627.

Cooper, C.; Frieze, A.; and Sorkin, G. B. 2007. Random 2-SAT with Prescribed Literal Degrees. *Algorithmica*, 48(3): 249–265.

Coppersmith, D.; Gamarnik, D.; Hajiaghayi, M. T.; and Sorkin, G. B. 2003. Random MAX SAT, random MAX CUT, and their phase transitions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, 364–373. ACM/SIAM.

Ding, J.; Sly, A.; and Sun, N. 2015. Proof of the Satisfiability Conjecture for Large k. STOC '15, 59–68. New York, NY, USA: Association for Computing Machinery. ISBN 9781450335362.

Erdös, P.; and Selfridge, J. 1973. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3): 298–301.

Friedrich, T.; Krohmer, A.; Rothenberger, R.; Sauerwald, T.; and Sutton, A. M. 2017. Bounds on the Satisfiability Threshold for Power Law Distributed Random SAT. In Pruhs, K.; and Sohler, C., eds., *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, 37:1–37:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Frieze, A. M.; and Suen, S. 1996. Analysis of Two Simple Heuristics on a Random Instance of k-SAT. *J. Algorithms*, 20(2): 312–355.

Goerdt, A. 1996. A Threshold for Unsatisfiability. *J. Comput. Syst. Sci.*, 53(3): 469–486.

Hajiaghayi, M. T.; and Sorkin, G. B. 2003. The satisfiability threshold of random 3-SAT is at least 3.52. Technical report, IBM.

Kaporis, A. C.; Kirousis, L. M.; and Lalas, E. G. 2002. The Probabilistic Analysis of a Greedy Satisfiability Algorithm. In Möhring, R. H.; and Raman, R., eds., *Algorithms - ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17-21, 2002, Proceedings*, volume 2461 of *Lecture Notes in Computer Science*, 574–585. Springer.

Kaporis, A. C.; Kirousis, L. M.; and Lalas, E. G. 2006. The probabilistic analysis of a greedy satisfiability algorithm. *Random Struct. Algorithms*, 28(4): 444–480.

Kim, J. H. 2004. The Poisson Cloning Model for Random Graphs, Random Directed Graphs and Random k-SAT Problems. In Chwa, K.; and Munro, J. I., eds., *Computing and Combinatorics, 10th Annual International Conference, COCOON 2004, Jeju Island, Korea, August 17-20, 2004, Proceedings*, volume 3106 of *Lecture Notes in Computer Science*, 2. Springer.

Kim, J. H. 2008. Finding cores of random 2-SAT formulae via Poisson cloning. *CoRR*, abs/0808.1599.

Larrabee, T.; and Tsuji, Y. 1992. Evidence for a Satisfiability Threshold for Random 3CNF Formulas. Technical report, USA.

Luby, M.; Mitzenmacher, M.; and Shokrollahi, M. A. 1998. Analysis of Random Processes via And-Or Tree Evaluation. In Karloff, H. J., ed., *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, 364–373. ACM/SIAM.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and Easy Distributions of SAT Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, 459–465. AAAI Press. ISBN 0262510634.

Mitzenmacher, M. 1997. Tight thresholds for the pure literal rule. Technical report.

Molloy, M. 2005. Cores in random hypergraphs and Boolean formulas. *Random Struct. Algorithms*, 27(1): 124–135.

Omelchenko, O.; and Bulatov, A. 2021a. Satisfiability and Algorithms for Non-uniform Random k-SAT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 3886–3894.

Omelchenko, O.; and Bulatov, A. A. 2019. Concentration inequalities for sums of random variables, each having power bounded tails. https://arxiv.org/abs/1903.02529. Online; accessed 6 March 2019.

Omelchenko, O.; and Bulatov, A. A. 2021b. Satisfiability Threshold for Power Law Random 2-SAT in Configuration Model. *Theoretical Computer Science*.

Warnke, L. 2014. When does the K4-free process stop? *Random structures & algorithms*, 44(3): 355–397.

Warnke, L. 2019. On Wormald's differential equation method. *ArXiv*, abs/1905.08928.

Wormald, N. 1999. *The differential equation method for random graph processes and greedy algorithms*, 73–155. Wydawnictwo Naukowe Pwn.

Wormald, N. C. 1995. Differential Equations for Random Processes and Random Graphs. *The Annals of applied probability*, 5(4): 1217–1235.