A sharp threshold in proof complexity yields lower bounds for satisfiability search

Dimitris Achlioptas
Microsoft Research
One Microsoft Way
Redmond, WA 98052
optas@microsoft.com

Paul Beame*
Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
beame@cs.washington.edu

Michael Molloy†
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
molloy@cs.toronto.edu

September 12, 2003

Abstract

Let $F(\rho n, \Delta n)$ denote a random CNF formula consisting of ρn randomly chosen 2-clauses and Δn randomly chosen 3-clauses, for some arbitrary constants $\rho, \Delta \geq 0$. It is well-known that, with probability $1-o(1)$, if $\rho > 1$ then $F(\rho n, \Delta n)$ has a linear-size resolution refutation. We prove that, with probability $1-o(1)$, if $\rho < 1$ then $F(\rho n, \Delta n)$ has no subexponential-size resolution refutation.

Our result also yields the first proof that random 3-CNF formulas with densities well below the generally accepted range of the satisfiability threshold (and thus believed to be satisfiable) cause natural Davis-Putnam algorithms to take exponential time (to find a satisfying assignment).

1 Introduction

Satisfiability has received a great deal of study as the canonical NP-complete problem. In the last several years the very universality and flexibility that made satisfiability a natural starting point for NP-completeness have also made it the basis for significant progress in the solution of a variety of practical problems including problems in constraint satisfaction [35], planning [28, 27], and symbolic model checking [10]. The basic tools for these advances are some very tight and efficient implementations of satisfiability algorithms using backtracking search based on the Davis-Putnam/DLL (DPLL) procedure [19, 18] and using heuristic search based on hill-climbing and random walks [35, 34]. In a sense, these satisfiability algorithms have become the hammer and there is now a small industry turning computational problems into nails.

In the last twenty years a significant amount of work has been devoted to the study of randomly generated satisfiability instances and the performance of different algorithms on them. Historically, the motivation for studying random instances has been the desire to understand the hardness of "typical" instances. While

* Research supported by NSF grant CCR-9800124.
† Research supported by NSERC and by a Sloan Fellowship

many generative models have been proposed over the years, random k-SAT (described below) is by far the most studied model. One reason for that is that random k-CNF formulas enjoy a number of intriguing mathematical properties, including a form of "expansion" and the existence of 0-1 laws. Another reason is that random k-SAT instances appear hard to deal with computationally for a certain range of the distribution parameters, making them a very popular benchmark for testing and tuning satisfiability algorithms. In fact, some of the better practical ideas in use today come from insights gained by studying the performance of algorithms on random k-SAT instances [25, 24].

Let $C_k(n)$ denote the set of all possible disjunctions of k distinct, non-complementary literals (k-clauses) from some canonical set of n Boolean variables. A random k-CNF formula is formed by selecting uniformly, independently, and with replacement m clauses from $C_k(n)$ and taking their conjunction. We will say that a sequence of random events $\mathcal{E}_n$ occurs with high probability (w.h.p.) if $\lim_{n\to\infty} \Pr[\mathcal{E}_n] = 1$ and with constant probability if $\liminf_{n\to\infty} \Pr[\mathcal{E}_n] > 0$.

One of the most intriguing aspects of random k-CNF formulas is the *Satisfiability Threshold Conjecture* which asserts that for every $k \geq 3$, there exists a constant α_k such that a random k-CNF formula with n variables and $m = \Delta n$ clauses is w.h.p. satisfiable if $\Delta < \alpha_k$ and unsatisfiable if $\Delta > \alpha_k$. Indeed, this is known for $k = 2$ as [14, 20, 23], independently, proved $\alpha_2 = 1$. Moreover, for all $k \geq 2$, it has been proven [21] that there is a sharp transition from the satisfiable regime to the unsatisfiable regime as Δ goes through a critical value $\alpha_k(n)$ (but not that $\alpha_k(n)$ converges with n). Empirical evidence (e.g., [36, 29]) suggests approximate values for α_k, e.g. $\alpha_3 \approx 4.2$. At the same time, for all k it is easy to prove that a random k-CNF formula with Δn clauses is w.h.p. unsatisfiable if $\Delta \geq 2^k \ln 2$ and, recently, it was proved in [6] that such a formula is w.h.p. satisfiable if $\Delta \leq 2^k \ln 2 - O(k)$.

For random k-CNF formulas in the unsatisfiable regime, the behavior of DPLL algorithms, and the more general class of resolution-based algorithms, is well-understood. Specifically, since every unsatisfiable 2-CNF formula has a linear-size resolution refutation, if $\Delta > \alpha_2 = 1$ then even the simplest DPLL algorithms w.h.p. run in polynomial time on a random 2-CNF formula. On the other hand, for $k \geq 3$ a celebrated result of Chvátal and Szemerédi [15] asserts that w.h.p. a random k-CNF formula in the unsatisfiable regime requires an exponentially long resolution proof of unsatisfiability. More precisely, let $res(F)$ and $DPLL(F)$ be the sizes of the minimal resolution and DPLL proofs of the unsatisfiability of a formula F (assume these to be infinite if F is satisfiable). In [15] it was proved that for all $k \geq 3$ and any constant $\Delta > 0$, if F is a random k-CNF formula with n variables and Δn clauses then w.h.p. $res(F) = 2^{\Omega(n)}$ and $DPLL(F) = 2^{\Omega(n)}$. Thus, for $\Delta \geq 2^k \ln 2$ w.h.p. a random k-CNF formula F is unsatisfiable but all its resolution refutations are exponentially long, implying that every DPLL algorithm must take exponential time on F.

Our main result extends the above theorem of [15] by allowing the addition of a random 2-CNF formula on the same n variables. Naturally, since $\alpha_2 = 1$, a formula containing $(1+\epsilon)n$ random 2-clauses w.h.p. will have a polynomial-size refutation, as the 2-clauses alone w.h.p. will have such a refutation. Thus, adding $(1+\epsilon)n$ 2-clauses to a random k-CNF formula with density $2^k \ln 2$, w.h.p. causes the proof complexity to collapse from exponential to linear. Our main result asserts that, in contrast, adding $(1-\epsilon)n$ 2-clauses to a random k-CNF formula w.h.p. has essentially no effect on its proof complexity. More precisely, let $\mathcal{F}^n_{\epsilon,\Delta}$ be the distribution of random CNF formulas with $(1-\epsilon)n$ 2-clauses and Δn 3-clauses, for some arbitrary *constants* $\Delta, \epsilon > 0$. (For simplicity, we focus on $k = 3$; extensions to $k > 3$ are straightforward.)

Theorem 1.1. *For every $\Delta, \epsilon > 0$, if $F \sim \mathcal{F}^n_{\epsilon,\Delta}$, then w.h.p. $res(F) = 2^{\Omega(n)}$ and $DPLL(F) = 2^{\Omega(n)}$.*

Theorem 1.1 represents a sharp threshold in proof complexity, since (combined with the fact $\alpha_2 = 1$) it implies that for every fixed $\Delta > 0$, the proof complexity w.h.p. goes from exponential to linear as the 2-clause density goes through 1. Moreover, for $\Delta > 2.28$ it is known [4] that there exists $\epsilon > 0$ such that formulas from $\mathcal{F}^n_{\epsilon,\Delta}$ are w.h.p. unsatisfiable. Combined with Theorem 1.1, this fact gives a method for proving the first lower bounds on the running times of DPLL algorithms for random *satisfiable* CNF formulas.

More precisely, using standard techniques it is not hard to show that many natural DPLL algorithms when applied to random 3-CNF formulas with Δn clauses, generate at least one unsatisfiable subproblem consisting of a random mixture of 2- and 3-clauses, where the 2-clauses alone are satisfiable. In particular, this is true even for values of Δ for which there is strong empirical evidence of satisfiability, i.e. for Δ significantly below the experimentally observed threshold $\alpha_3 \approx 4.23 \pm 0.05$. By Theorem 1.1, in order to resolve any such subproblem (and backtrack) all DPLL algorithms need exponential time. Thus, we can prove that certain natural DPLL algorithms require *exponential* time significantly below the generally accepted range for the random 3-SAT threshold. As an example, for ORDERED-DLL (which performs unit-clause propagation but, otherwise, sets variables in some a priori fixed random order/sign) we prove

Theorem 1.2. *When* ORDERED-DLL *is applied to a random 3-CNF formula with n variables and $3.81n$ clauses, with constant probability it requires time $2^{\Omega(n)}$.*

Theorem 1.2 sheds light on a widely-cited observation of Selman, Mitchell, and Levesque [36], based on experiments with ORDERED-DLL on small problems, stating that random 3-SAT is easy in the satisfiable region up to the 4.2 threshold, becomes sharply much harder at the threshold and quickly becomes easy again at larger densities in the unsatisfiable region. The upper end of this 'easy-hard-easy' characterization is somewhat misleading since, as we saw, the result of [15] in fact asserts that w.h.p. random 3-CNF formulas only have exponential-size proofs of unsatisfiability above the threshold. By now the rate of decline in proof complexity as the density is increased has been analyzed as well [8]. Our results show that the lower end of this characterization is also somewhat misleading; in fact, Theorem 1.2 shows that the exponentially hard region for ORDERED-DLL begins at least at ratio 3.81, well before ratio 4.2. This concurs with recent experimental evidence that even the best of current DPLL implementations seem to have bad behavior below the threshold [16].

We also note that one highly successful strategy, in practice, for satisfiable formulas is to use a randomized DPLL algorithm and restart it with different random bits if it begins to take too long [25, 24]. While Theorem 1.2 only holds with constant probability, we will see that random restarts are unlikely to reduce the running time of ORDERED-DLL (and similar algorithms) on random k-CNF formulas down to polynomial.

Our proof is similar in general spirit to proofs of other lower bounds for resolution complexity but requires considerably more subtlety. We first prove a number of detailed combinatorial properties of random 2-CNF formulas with $(1-\epsilon)n$ clauses. To do this we consider the standard directed graphs associated with 2-CNF formulas and, for such graphs, we introduce the notion of the *clan* of a vertex. Clans seem to be the appropriate extension of "connected components" in this context, allowing for an amortization of the "boundary" of the 2-CNF formula. By carefully bounding the number of vertices in clans of each size we show that random 2-CNF formulas with $(1-\epsilon)n$ clauses, w.h.p. have properties that guarantee that almost all extensions by linear-sized 3-CNF formulas require exponential size resolution (and DPLL) proofs. This latter argument relies on specialized sharp moment bounds as well as particular properties of clans.

1.1 Background and Related Work

Mixed formulas consisting of 2- and 3-clauses arise for a number of reasons. For example, a frequent observation about converting problems from other domains into satisfiability problems is that they typically become mixed CNF formulas with a substantial number of clauses of length 2 along with clauses of length 3. Another reason is that as DPLL algorithms run, they recursively solve satisfiability on *residual formulas*, restricted versions of their input CNF formula, which are mixtures of clauses of length at least 2. Randomly chosen 3-CNF formulas are an important test case for satisfiability algorithms and when given such formulas as input, many DPLL algorithms produce residual formulas that are mixtures of 2- and 3-clauses that are distributed precisely in the form that we analyze, i.e. are uniformly random. Moreover, as we will see, random mixtures of 2- and 3-clauses, originally introduced as the $(2+p)$-SAT model, are a very convenient means for exploring the interface between computational complexity and phase transitions.

1.1.1 Random $(2+p)$-SAT

As an attempt to "interpolate" between random 2-SAT and random 3-SAT Kirkpatrick et. al. introduced the so-called $(2+p)$-SAT problem in [31]. Here, one considers randomly-generated formulas on n variables where a fraction p of all clauses have length 3 (while the remaining have length 2) and where each clause of length $i = 2, 3$ is chosen uniformly from $C_i(n)$. Using empirical results and non-rigorous techniques of statistical physics, Kirkpatrick et. al. [31, 32, 33] gave evidence that there exists a critical $p_c \approx 0.417$ such that for $p < p_c$ a random $(2+p)$-SAT formula goes from being satisfiable w.h.p. to being unsatisfiable w.h.p. as the 2-clause density goes through $\alpha_2 = 1$. In other words, for $p < p_c$ the 3-clauses seem irrelevant to the formula's satisfiability and random 2-CNF formulas cannot "feel" the presence of up to $p_c/(1-p_c)$ random 3-clauses. They also gave evidence that around p_c the phase transition from satisfiability to unsatisfiability changes character from a so-called second order transition (continuous "order parameter") representative of 2-SAT to a fist-order transition (discontinuous "order parameter") believed to be representative of 3-SAT.

In [4], Achlioptas et. al. proved a number of rigorous results for random $(2+p)$-SAT. In particular, they proved that a formula with $(1-\epsilon)n$ random 2-clauses and Δn random 3-clauses is w.h.p. satisfiable for all $\epsilon > 0$ and $\Delta \leq 2/3$ (and a satisfying assignment can be found by a simple linear-time algorithm), whereas for $\Delta > 2.28$ there exist (sufficiently small) $\epsilon > 0$ such that w.h.p. it is unsatisfiable. These results, respectively, imply $2/5 \leq p_c \leq 0.696$. In [1], it was later conjectured that in fact $p_c = 2/5$, which is equivalent to saying that for every $\delta > 0$ there exists $\epsilon = \epsilon(\delta) > 0$ such that a random formula with $(2/3+\delta)n$ random 3-clauses and $(1-\epsilon)n$ random 2-clauses is w.h.p. unsatisfiable. If true, this statement would have significant implications for the "replica method" of statistical mechanics. Moreover, as we will see in the next section, combined with our Theorem 1.1 it would provide a sharp threshold for the running time of DPLL algorithms on random 3-CNF formulas.

1.1.2 DPLL algorithms below the threshold

By now, there has been a long sequence of results giving improved bounds for the location of the random 3-SAT threshold. The best current bounds assert that a random 3-CNF formula is w.h.p. satisfiable if $\Delta < 3.26$ [5] and w.h.p. unsatisfiable if $\Delta > 4.598$ [26]. Combining this upper bound with the result of [15] we see that every DPLL algorithm w.h.p. takes exponential time on a random 3-CNF formula with $\Delta > 4.598$.

On the other hand, the bound $\Delta < 3.26$ above corresponds to the densities for which a specific DPLL algorithm [5] finds a satisfying truth assignment *without any backtracking* with constant probability. In fact, all lower bounds for the random 3-SAT threshold correspond to values for which this is true for some specific algorithm,[1] with improved bounds resulting from better criteria for branching and value assignment, rather than from "greater search space exploration".

Indeed, almost all algorithms that have been analyzed on random 3-CNF formulas fall in the class of so-called "card-type/myopic algorithms" in the terminology of [3, 5]. Such algorithms seek to create a satisfying truth assignment by setting variables sequentially and by definition: i) they never backtrack, i.e. they stop as soon as a contradiction is generated, ii) they maintain that the residual formula is always a uniformly random mixture of 2- and 3-clauses on the unassigned variables (unit-clauses are satisfied as soon as they occur). In order to maintain the latter property, myopic algorithms use very limited information

[1] Establishing that a random k-CNF formula is satisfiable with constant probability for a given density Δ^* is enough to imply that $\alpha_k \geq \Delta^*$ since by Friedgut's theorem [21] there cannot exist constants $\Delta_1 < \Delta_2$ such that the probability of satisfiability is bounded away from both 0 and 1 for all $\Delta \in [\Delta_1, \Delta_2]$.

to decide which variable(s) to set next and what values to assign (hence their name). Examples of such algorithms are UC (where in the absence of unit clauses a random literal is assigned true) and GUC [13] (where always a random literal in a random shortest clause is assigned true).

It is not hard to prove that the largest density Δ_A for which a myopic algorithm A has constant probability of finding a satisfying assignment is precisely the largest density for which w.h.p. the 2-clause density of the residual formula remains below 1 throughout A's execution (see e.g. [3]). For $\Delta > \Delta_A$ one can endow A with a backtracking scheme (so that the execution of the original myopic algorithm corresponds to the first path explored in the tree of recursive calls) and attempt to analyze its performance. Unfortunately, any non-trivial amount of backtracking makes it hard to have a compact probabilistic model for the residual formula (such as the one corresponding to the original algorithm A). As a result, a probabilistic analysis akin to that possible for $\Delta < \Delta_A$ appears beyond the reach of current mathematical techniques (but see [17] for a non-rigorous analysis based on ideas from statistical physics). This is where our results come in:

If a DPLL algorithm A ever generates a residual formula that is an unsatisfiable random mixture of 2- and 3-clauses with 2-clause density bounded below 1, then w.h.p. A will spend exponential time before backtracking from it.

That is, by Theorem 1.1, once a node in the backtracking search is reached that corresponds to an unsatisfiable random mixture of 2- and 3-clauses (but where the 2-clauses alone are satisfiable), the search cannot leave the sub-tree for an exponentially long time. Standard results (see e.g. [3]) thus imply that with constant probability this is precisely what happens for UC started with $3.81n$ 3-clauses and for GUC started with $3.98n$ 3-clauses. This is because for such initial densities, the corresponding algorithm has constant probability of generating a residual formula $\mathcal{F}^n_{\epsilon,\Delta}$ with Δ and ϵ known to be w.h.p. unsatisfiable by the results of [4].

Theorem 1.3. *Any backtracking extension of* UC *on a random n variable 3-CNF formula with Δn clauses for constant $\Delta \geq 3.81$ requires time $2^{\Omega(n)}$ with constant probability. Also, any backtracking extension of* GUC *on a random n variable 3-CNF formula with Δn clauses for constant $\Delta \geq 3.98$ requires time $2^{\Omega(n)}$ with constant probability.*

We note that the only reason for which Theorem 1.3 is not a high probability result is that with constant probability each algorithm might generate a contradiction and backtrack (thus destroying the uniform randomness of the residual formula) before reaching an unsatisfiable restriction $\mathcal{F}^n_{\epsilon,\Delta}$. Nevertheless, by extending UC and GUC with a natural backtracking heuristic introduced by Frieze and Suen [22], in Section 7 we create natural DPLL algorithms for which the analogue of Theorem 1.3 holds w.h.p.

In fact, we believe that Theorem 1.1 points to a much larger truth than the specific implications for the algorithms and backtracking scheme mentioned above. As will become clear from its proof in the upcoming sections, the conclusion of Theorem 1.1 is quite robust with respect to the probability distribution of the clauses in the mixture. The essential ingredients are as follows. For the 2-CNF subformula, besides satisfiability, the crucial property is that for most literals the associated "tree" of implications is rather small (constant size on average and with a reasonable tail) and has a simple structure. While we only prove this property for random 2-CNF (as generated by backtracking versions of myopic algorithms), it is not hard to imagine that this property would be robust to the branching/value assignments made by any "moderately smart" DPLL algorithm. For the 3-CNF subformula we only need that the variable-clause incidence graph is an expander. Again, while this property is satisfied strongly by arbitrary subformulas of random 3-CNF formulas it suggests that, in fact, random 3-CNF formulas are not the only formulas for which one could hope to prove a result similar to Theorem 1.1. Moreover, it is precisely this richness of expanders that suggests that restarting a DPLL algorithm on a random k-CNF formula is unlikely to yield dramatically different results from run to run (unless, of course, one is willing to restart an exponential number of times).

Finally, as we discuss in section 8, the values 3.81 and 3.98 in Theorem 1.3 will be readily improved with any improvement on the bound for the number Δn of 3-clauses needed to make a formula with $(1-\epsilon)n$ random 2-clauses unsatisfiable. In particular, if it turns out that $\Delta > 2/3$ suffices (as mentioned earlier), then our results would uniformly reduce the onset of exponential behavior to Δ_A for *every* backtracking extension of *every* myopic algorithm A. In other words, we would get that every such DPLL algorithm runs in linear-time for $\Delta < \Delta_A$, but requires exponential time for $\Delta > \Delta_A$.

2 Bounding Resolution Refutation Size

The resolution rule allows one to derive a clause $(A \vee B)$ from two clauses $(A \vee x)$ and $(B \vee \bar{x})$. A resolution derivation of a clause C from a CNF formula F is a sequence of clauses $C_1, \ldots, C_\ell = C$ such that each C_i is either a clause of F or follows from two clauses C_j, C_k for $j, k < i$ using the resolution rule. A resolution refutation of an unsatisfiable formula F is a resolution derivation of the empty clause. The proof inferences define a directed acyclic graph of in-degree 2 whose vertices are the clauses of the proof. The size of a resolution refutation is the number of clauses appearing in the proof. Given F, let $res(F)$ be the length of the shortest resolution refutation of F. The Davis-Putnam/DLL algorithm on a CNF formula F performs a backtracking search for a satisfying assignment of F by extending partial assignments until they either reach a satisfying assignment or violate a clause of F. It is well known that for an unsatisfiable formula F, the tree of nodes explored by any DPLL algorithm can be converted to a resolution refutation of F where the pattern of inferences forms the same tree. Let $DPLL(F)$ be the size of the smallest such refutation, i.e. the size of the smallest DPLL tree associated with F.

For a resolution derivation Π, let $width(\Pi)$ denote the maximum number of literals in any clause of Π. For an unsatisfiable CNF formula let $proofwidth(F)$ be the minimum over all resolution refutations Π of F of $width(\Pi)$. Ben-Sasson and Wigderson [9] showed that to prove lower bounds on resolution proof size it suffices to prove lower bounds on resolution proof width.

Proposition 2.1 ([9]). *There is some constant $c > 0$ such that if all clauses in a formula F have size at most k, $res(F) \geq 2^{c([proofwidth(F)-k]^2/n)}$ and $DPLL(F) \geq 2^{proofwidth(F)-k}$.*

Definition 2.2. *Given a CNF formula F, a literal x is* pure *in F if and only if x appears in F but $\bar{x}$ does not appear in F. We say that $F \Rightarrow_{Res} C$ if and only if there is a resolution derivation of C from F such that in the associated directed acyclic graph there is a path from every clause of F to the clause C.*

The following propositions yield a minor variation of the now standard method for proving lower bounds on the width of resolution proofs [15, 8, 9].

Proposition 2.3. *Let F be a CNF formula and let C be a clause. If $F \Rightarrow_{Res} C$ then every pure literal in F appears in C.*

Proposition 2.4. *Let F be an unsatisfiable CNF formula having clauses of size at most k. If there exist integers $s \geq 2k$ and t such that*

(a) Every subformula of F on at most s variables is satisfiable, and,

(b) every subformula F' of F on v variables, where $\frac{1}{2}s < v \leq s$, contains at least t literals that are pure in F',

then $proofwidth(F) \geq t$.

Proof. Let Π be any resolution refutation of F. To each clause C in Π we associate the subformula F_C of F consisting of those clauses of F that are used in Π to derive C. Observe that $F_C \Rightarrow_{res} C$.

For the empty clause Λ, F_Λ must be unsatisfiable and therefore F_Λ must contain more than s variables. Now, let us follow the graph of the proof backwards starting from Λ, at each step choosing the predecessor whose associated clause has the larger number of variables, provided that number is more than $s/2$. Clearly, this will lead us to a clause C, such that F_C has more than $s/2$ variables and the two predecessors A and B in Π (which must exist since F_C has more than k variables) each contain at most $s/2$ variables.

Since $F_C = F_A \cup F_B$, F_C contains at most $s/2 + s/2 = s$ variables. Therefore by assumption (2.4), F_C contains at least t pure literals. By Proposition 2.3, every pure literal in F_C appears in C, so $proofwidth(F) \geq |C| \geq t$. $\square$

Recall that for each fixed $k \geq 0$, $C_k(n)$ denotes the set of all $2^k\binom{n}{k}$ non-trivial k-clauses on some canonical set of n variables. We will consider a random formula F on n variables formed by selecting uniformly, independently and with replacement $m_2 = m_2(n)$ clauses from $C_2(n)$ and $m_3 = m_3(n)$ clauses from $C_3(n)$. In particular, recall that $\mathcal{F}^n_{\epsilon,\Delta}$ denotes the distribution where $m_2 = (1-\epsilon)n$ and $m_3 = \Delta n$, for some arbitrary *constants* $\Delta, \epsilon > 0$. Our main technical lemma is the following analogue for *mixed* random formulas of similar lemmas from [15, 8, 9] for random k-CNF formulas.

Lemma 2.5. *For every $\Delta, \epsilon > 0$ there exist $\zeta = \zeta(\Delta, \epsilon) > 0$ and $\mu = \mu(\Delta, \epsilon) > 0$ such that for $F \sim \mathcal{F}^n_{\epsilon,\Delta}$*

(a) w.h.p. every subformula of F on $v \leq \zeta n$ variables is satisfiable, and

(b) w.h.p. every subformula of F on v variables with $\frac{1}{2}\zeta n < v \leq \zeta n$ contains at least μn pure literals.

Theorem 1.1 follows immediately from Lemma 2.5, along with Propositions 2.1 and 2.4. However, the presence of 2-clauses in formulas $F \sim \mathcal{F}^n_{\epsilon,\Delta}$ makes the analysis required to prove Lemma 2.5 significantly more involved than the corresponding analysis for random k-CNF formulas, where $k \geq 3$.

Definition 2.6. *Let F be an arbitrary CNF formula. Let*

- *$V(F) = \{x_1, \ldots, x_n\}$ denote the set of variables of F,*
- *$|F|$ denote the number of clauses in F, and*
- *the* degree *of a variable v, $\deg(v)$, be the number of clauses of F containing one of $v, \bar{v}$ (analogously for literals).*
- *We say that a literal ℓ is* near-pure *in F if $\deg(\bar{\ell}) = 1$.*

For F_2, the subformula consisting of the 2-clauses of F, we associate the directed graph $\vec{D}(F_2)$ whose vertex set is $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, and whose edge set is $\{(\bar{x} \to y), (\bar{y} \to x) : (x \vee y)$ is a clause in $F_2\}$.

- *We say that a directed cycle $C = \ell_1 \to \ell_2 \to \cdots \to \ell_q \to \ell_1$ in $\vec{D}(F_2)$ is* pure *if all of $\ell_1, \ldots, \ell_q$ are near-pure in F. (Note that each literal can appear in at most one pure cycle.)*
- *We call a pure literal or pure cycle of a CNF formula F a* pure item *of F.*

We derive Lemma 2.5 from the following lemma.

Lemma 2.7. *For every $\Delta, \epsilon > 0$ there exist $\zeta = \zeta(\Delta, \epsilon) > 0$ and $\mu = \mu(\Delta, \epsilon) > 0$ such that for $F \sim \mathcal{F}^n_{\epsilon,\Delta}$*

(a) w.h.p. every subformula of F on $v \leq \zeta n$ variables contains at least one pure item, and

(b) w.h.p. every subformula of F on v variables with $\frac{1}{2}\zeta n < v \leq \zeta n$ contains at least μn pure literals.

Proof of Lemma 2.5 from Lemma 2.7. Property (b) is identical to that in Lemma 2.5. A formula is *minimally unsatisfiable* if it is unsatisfiable but each of its subformulas is satisfiable. Clearly, every unsatisfiable formula contains a minimally unsatisfiable subformula. Moreover, a minimally unsatisfiable formula F' cannot contain a pure item, since it is easily seen that any satisfying assignment of the subformula F'' of F' obtained by deleting all clauses involving the pure item can be extended to a satisfying assignment of F' by setting the pure or near-pure literals in the pure item to true. Therefore, property (b) implies that F has no minimally unsatisfiable subformula on fewer than ζn variables and hence it has no unsatisfiable subformula on fewer than ζn variables. Thus, properties (a) and (b) imply properties (a) and (b) of Lemma 2.5. □

The proof of Lemma 2.7 occupies Sections 3 to 6. The proof strategy is to first i) establish certain high-probability properties of the 2-clauses of F, then ii) use these properties to prove that all subformulas of F have relatively many pure items, and finally iii) show that the addition of 3-clauses does not significantly reduce these pure items. In fact, we will not only show that large subformulas have many pure items but we will also show that they have relatively few pure cycles, so that they have many pure literals.

The overall argument is subtle because the 2-clauses of F are arbitrarily close to being unsatisfiable themselves and, further, because we need to handle all possible subformulas among the 2-clauses which, unlike the case $k \geq 3$, requires a careful delineation of the different "local neighborhoods" of each variable among the 2-clauses. Indeed, this latter requirement necessitates the introduction of a novel graph-theoretic concept in the digraph associated with the 2-clauses of F that we call "clans".

3 Analyzing Subformulas using Clans

To prove Lemma 2.7 we will in fact prove a stronger lemma. In particular, rather than proving the lemma's assertion for $\mathcal{F}^n_{\epsilon,\Delta}$, we will prove it for an arbitrary formula on n variables formed by starting with a 2-CNF formula satisfying certain properties and adding to it $m_3 = \Delta n$ random 3-clauses. To complete the proof, in Section 5, we prove that F_2 satisfies these properties w.h.p. To describe these properties we need to introduce the following definitions.

Definition 3.1. Let F be an arbitrary 2-CNF formula. For literals x, y appearing in F let us write $x \leadsto_F y$ iff $x = y$ or there exists a directed path in $\vec{D}(F)$ from x to y.

- For each literal x we let $\mathrm{In}_F(x) = \{y : y \leadsto_F x\}$.
- For a set of literals S let $G(S) = G_F(S)$ be the undirected graph formed by considering the subgraph of $\vec{D}(F)$ induced by the vertices corresponding to S and ignoring the direction of arcs.
  - We will say that $\mathrm{In}_F(x)$ is *tree-like* if $G(\mathrm{In}_F(x))$ contains *no* cycle.
  - We will say that $\mathrm{In}_F(x)$ is *simple* if $G(\mathrm{In}_F(x))$ contains *at most one* cycle.
- For each literal x in F, the *clan* of x, $\mathrm{Clan}_F(x) = \mathrm{In}_F(x) \cup \bigcup_{y \in \mathrm{In}_F(x)} \mathrm{In}_F(\bar{y})$.

  Define $\mathrm{Clan}^*_F(x) = \begin{cases} \mathrm{Clan}_F(x) & \text{if } \mathrm{In}_F(y) \text{ is tree-like for all } y \in \mathrm{Clan}_F(x) \\ \{x, \bar{x}\} & \text{otherwise.} \end{cases}$
- For each $i \geq 0$, we let $T_i(F) = |\{x : |\mathrm{Clan}^*_F(x)| \geq i\}|$.

The importance of clans will become apparent in the proof of Lemma 4.2. Roughly, they allow us to identify a relatively small set P^* of pure literals, such that every $x \in F$ belongs to the clan of some member of P^*. Also, in a random formula F w.h.p. a very small number of literals x will have $\mathrm{In}_F(x)$ not tree-like. These literals must be dealt with in a special way; the definition of Clan^* allows us to do so.

Lemma 2.7 will follow readily from the following two lemmas.

Lemma 3.2. *Fix $\rho \in (0,1)$, $C > 0$, and $\Delta > 0$. Let F^* be a formula formed by taking*

- *Any set of clauses from $C_2(n)$ such that the resulting formula F_2^* satisfies:*
  1. *For every literal ℓ, $\mathrm{In}_{F_2^*}(\ell)$ is simple.*
  2. *There are at most $\log n$ literals ℓ such that $\mathrm{Clan}_{F_2^*}(\ell)$ contains a literal ℓ' with $\mathrm{In}_{F_2^*}(\ell')$ not tree-like.*
  3. *For every literal ℓ, $|\mathrm{Clan}_{F_2^*}(\ell)| \leq \log^3 n$.*
  4. *For all $i \geq 3$, $T_i(F_2^*) \leq 2(1-\rho)^i n$.*
- *No more than Δn clauses from $C_3(n)$, chosen uniformly, independently and with replacement.*

There exist $\zeta = \zeta(\Delta, \rho)$ and $\mu = \mu(\Delta, \rho, \zeta) > 0$ such that:

(a) W.h.p. every subformula of F^* on $v \leq \zeta n$ variables has at least one pure item.

(b) W.h.p. every subformula of F^* on v variables with $\frac{1}{2}\zeta n < v \leq \zeta n$ contains at least μn pure literals.

Lemma 3.3. *Fix $\epsilon > 0$ and let F_2 be a random 2-CNF formula formed by selecting uniformly, independently and with replacement $m_2 \leq (1-\epsilon)n$ 2-clauses from $C_2(n)$. There exists $\rho = \rho(\epsilon)$, such that w.h.p. F_2 simultaneously satisfies all four conditions of Lemma 3.2 (where $F_2^* = F_2$).*

If one replaced "Clan" by "In" throughout the statements of the conditions of Lemma 3.2, then Lemma 3.3 would follow quickly from some standard and well-understood properties of random 2-CNF formulae. Intuitively, Lemma 3.3 holds because clans are sufficiently similar in structure to the sets $\mathrm{In}_F(x)$. We defer the lengthy proof to Section 5. We prove Lemma 3.2 in Section 4.

Proof of Lemma 2.7 from Lemma 3.2. Given $\Delta, \epsilon > 0$ we start by applying Lemma 3.3 to get $\rho = \rho(\epsilon)$ and then apply Lemma 3.2 with that ρ. □

4 Proof of Lemma 3.2

Let $\rho, \Delta > 0$, F_2^* be fixed and choose F^* as in the statement of the lemma. Consider any (candidate) subformula H of F^*. Let $v = |V(H)|$ and denote by H_2 the subformula induced by the 2-clauses of H.

The general idea of the argument is as follows. The subformula H_2 has many "loose ends", namely the pure items of H_2 and the literals of $V(H) - V(H_2)$, that must be (mostly) covered by the 3-clauses of H in order for H to have very few pure items. We show that every literal of H is in the clan of one of a small subset of these loose ends. Thus, since the clan sizes are small, the number of loose ends must be large. In order to cover all (or most) of these loose ends, we need a large number of 3-clauses, all of whose variables lie within $V(H)$. However, since the number of variables in H is small, it is highly unlikely that enough of the random 3-clauses will "land" in H. The formal analysis is a bit involved, and require a sharp specialized moment bound to show that the rare large clans do not skew the probabilities too much. We present that moment bound in the next section.

We now work through the details of the argument. Define the set $P = P(H)$ of literals based on H as follows: P consists of the pure literals of H_2, the smallest numbered literal in each pure cycle of H_2, and every literal on the variables of $V(H) - V(H_2)$. Clearly P contains every pure literal of H and also contains one literal from each pure cycle of H (and since pure cycles are disjoint they are represented by distinct literals). So $P(H)$ contains the "loose ends" referred to above.

Lemma 4.1. *For any subformula H of F^*, the number of distinct literals in the 3-clauses of H is at least the number of literals in $P(H)$ that are not contained in pure items in H.*

Proof. We define a one-to-one (but not necessarily onto) mapping from the literals of $P = P(H)$ that are not contained in pure items of H to the literals appearing in the 3-clauses of H. Any literal x in P, that was pure in H_2 or is a literal on $V(H) - V(H_2)$ but is not pure in H, must have $\bar{x}$ in some 3-clause of H and so we map x to $\bar{x}$. The pure cycles of H_2, whose smallest numbered literals form the remainder of P, are disjoint from each other and from the other literals in P. Consider such a cycle C that is pure in H_2 and let $x \in P$ be the smallest numbered literal in C. C will remain pure in H unless there is some y in C such that $\bar{y}$ appears in a 3-clause of H. We map x to $\bar{y}$. The fact that our map is one-to-one follows from the disjointness property of the cycles. $\square$

For convenience throughout the rest of this proof we will write $\mathrm{Clan}(x)$ for $\mathrm{Clan}_{F_2^*}(x)$ and for a set T of literals we will write $\mathrm{Clan}(T) = \bigcup_{x \in T} \mathrm{Clan}(x)$. For any literal x (set of literals T), let $cover(x)$ (resp. $cover(T)$) be the set of literals appearing in $\mathrm{Clan}(x)$ (resp. $\mathrm{Clan}(T)$) together with the complements of those literals.

The next step is to show that there is a small set $P^* \subseteq P$ such that every literal of H lies in $cover(P^*)$. It is easy to see that this is true if we simply take $P^* = P$, and in fact this would be true even if we used a much simpler structure than the clan. However, we need P^* to be smaller than P (roughly half as small will do), and this is the reason that we need to focus on clans.

Lemma 4.2. *For any subformula H of F^* there exists $P^* = P^*(H) \subseteq P = P(H)$ such that*

1. *$cover(P^*)$ contains every literal appearing in H and*
2. *$|P^*| \leq \lfloor \frac{1}{2}(|P| + t_c) \rfloor$ where $t_c = t_c(H)$ is the number of literals $x \in P^*$ such that $\mathrm{In}_{H_2}(x)$ is not tree-like.*

Proof. Let $\hat{P} \subseteq P$ be the set of literals in P on variables in $V(H_2)$. By definition, for every $x \in P - \hat{P}$, $\bar{x} \in P - \hat{P}$. Let $P_{tree} \subseteq \hat{P}$ be the set of all literals $x \in \hat{P}$ with $\mathrm{In}_{F_2^*}(x)$ tree-like. First we prove that for every $x \in P_{tree}$ there is at least one $y \in \hat{P}$, $y \neq x$ such that $\bar{y} \in \mathrm{In}_{F_2^*}(x)$. For $x \in P_{tree}$, $\mathrm{In}_{H_2}(x)$ is tree-like since $\mathrm{In}_{H_2}(x) \subseteq \mathrm{In}_{F_2^*}(x)$. Therefore there is a vertex $z \in \mathrm{In}_{H_2}(x)$ of in-degree 0 in $\vec{D}(H_2)$ such that $z \neq \bar{x}$. Furthermore, since z appears in H_2, $\bar{x} \in \hat{P}$ so we can take $y = \bar{z} \notin \{x, \bar{x}\}$.

Note that $\bar{y} \in \mathrm{In}_{H_2}(x) \subseteq \mathrm{In}_{F_2^*}(x)$ implies $\bar{x} \in \mathrm{In}_{F_2^*}(y)$. Thus we form an undirected graph G with vertex set $\hat{P}$ and an edge $\langle x, y \rangle$ for each pair of literals with $\bar{y} \in \mathrm{In}_{F_2^*}(x)$. Let $P' \supseteq P_{tree}$ be the set of vertices in G of positive degree, consider a spanning forest of the vertices in P', and consider any bipartition of that forest. Let P_1 be the smaller side of that bipartition. Therefore P_1 dominates P', i.e. every vertex in $P' - P_1$ has a neighbor in P_1 and thus $P_1 \cup (\hat{P} - P')$ dominates all of $\hat{P}$. Letting $|\hat{P} - P'| = a$, $|P_1 \cup (\hat{P} - P')| \leq a + \lfloor \frac{1}{2}(|\hat{P}| - a) \rfloor \leq \lfloor \frac{1}{2}(|\hat{P}| + a) \rfloor$. Adding the positive form of each literal in $P - \hat{P}$ to $P_1 \cup (\hat{P} - P')$ we obtain a set P^* of size at most $\lfloor \frac{1}{2}(|P| + a) \rfloor$. Since $\hat{P} - P' \subseteq P^*$ and $P_{tree} \subseteq P'$, $t_c \geq a$ and P^* satisfies the claimed size condition.

By definition of P, $\hat{P}$, and P^*, P^* contains the positive literal corresponding to each variable in $V(H) - V(H_2)$, so $cover(P^*)$ contains all literals on variables in $V(H) - V(H_2)$.

Let x be a literal such that $\bar{x}$ appears in H_2. In the digraph $\vec{D}(H_2)$ walk forward from x until either a sink node is reached or a node on the path is repeated. If we reach a sink of $\vec{D}(H_2)$, the label of that sink is a pure literal y in P which satisfies $x \in \mathrm{In}_{H_2}(y)$. If we reach a repeated node then we have found a cycle in $\vec{D}(H_2)$ and, since all clans contain at most one cycle, this cycle is a pure cycle of H_2. The smallest numbered literal y in this pure cycle is in P and satisfies $x \in \mathrm{In}_{H_2}(y)$. Therefore, in either case there is a literal $y \in P$ such that $x \in \mathrm{In}_{H_2}(y)$.

By definition of P^* either $y \in P^*$ or there is some $z \in P^*$ such that $\bar{y} \in \mathrm{In}_{H_2}(z)$. Therefore $x \in \mathrm{Clan}(z)$ and thus both x and $\bar{x}$ are in $cover(z)$. Thus $cover(P^*)$ contains all literals on $V(H_2)$ as well, so the lemma follows. $\square$

We now show how the property of having few pure items in a subformula H of F^* requires that there are a large number of 3-clauses of F^* whose literals lie entirely in the relatively small set, $cover(P^*(H))$.

Lemma 4.3. *Let T be a set of literals, $t = |T|$. Suppose that H is a subformula of F^* with $P^*(H) = T$, $t_c = t_c(H)$ and at most $t/10$ pure items. Then H, and thus formula F^*, must contain at least $19t/30 - t_c/3$ 3-clauses whose literals are contained in $cover(T)$; further if $t \geq 10t_c$ then there are at least $3t/5$ 3-clauses of F^* whose literals are contained in $cover(T)$.*

Proof. By Lemma 4.2, since $P^*(H) = T$, $|P(H)| \geq 2|T| - t_c = 2t - t_c$. By Lemma 4.1, since H has at most $t/10$ pure items, the 3-clauses of H contain at least $2t - t_c - t/10$ literals and therefore H contains at least $(19t/10 - t_c)/3$ 3-clauses of F^*. By Lemma 4.2, all literals in these clauses are in $cover(P^*) = cover(T)$. In case $t \geq 10t_c$ then this is at least $(19/30 - 1/30)t > 3t/5$. $\square$

We will bound the probability that F^* has a small subformula H with few pure items by bounding the probability for each set of literals T that there is a subformula H of F^* with $P^*(H) = T$ and with at most $|T|/10$ pure items and then summing this bound over all choices of T. This immediately proves part (a) of Lemma 3.2. We will also prove that for any subformula H on a linear number of variables, $|P^*(H)|$ is of linear size but the number of pure cycles is at most polylogarithmic in size and, together with our probability bound, this will prove part (b) of Lemma 3.2.

Lemma 4.4. *Fix $\Delta, \rho > 0$. There is $K = K(\Delta)$ and an $n_0 = n_0(\Delta)$ such that for $n \geq n_0$ and for T a set of literals, $t = |T|$, the probability that F^* has a subformula H with $P^*(H) = T$ and at most $t/10$ pure items is at most*

(a) $R(T) = (K/(tn^2))^{3t/5}|\mathrm{Clan}(T)|^{9t/5}$ if $t \geq \log^4 n$, and at most

(b) $R'(T, t_c) = (K/n^2)^{19t/30 - t_c/3}|\mathrm{Clan}(T)|^{19t/10 - t_c}$ if $t \leq \log^4 n$.

Proof. Since F^* has Δn 3-clauses, for an integer $s \geq 1$, the probability that at least s of them land entirely in $cover(T)$ is at most

$$\binom{\Delta n}{s}\left[\frac{|cover(T)|^3}{8\binom{n}{3}}\right]^s \leq \binom{\Delta n}{s}\left[\frac{|\mathrm{Clan}(T)|^3}{\binom{n}{3}}\right]^s$$
$$\leq [K'/s(n^2)]^s|\mathrm{Clan}(T)|^{3s} \qquad (1)$$

for some constant $K' = K'(\Delta)$. Let $K = 5K'/3$. By assumption about F_2^*, $t_c \leq \log n$ and so if $t > \log^4 n$ then $t > 10t_c$. Thus, by Lemma 4.3, we get the probability upper bound in part (4.4) by setting $s = \lceil 3t/5 \rceil$ and observing that the upper bound in (1) is at most $[K/(tn^2)]^s|\mathrm{Clan}(T)|^{3s}$ and that this is a decreasing function of s (which is therefore at most $R(T)$) for $R(T) \leq 1$. Also, by Lemma 4.3, we get the probability upper bound in part (4.4) by setting $s = \lceil 19t/30 - t_c/3 \rceil$ and observing that (1) is bounded above by $[K'/(n^2)]^s|\mathrm{Clan}(T)|^{3s}$ which is also a decreasing function of s and thus at most $R'(T, t_c)$ for $R'(T, t_c) \leq 1$. $\square$

Lemma 4.5. *Fix $\Delta, \rho > 0$. The probability that there exists some set T of size $t \leq \log^4 n$ and a subformula H of F^* with $P^*(H) = T$ and at most $t/10$ pure items is $o(1)$ in n.*

Proof. Suppose that $t \le \log^4 n$. By assumption about F_2^*, any subformula H with $|P^*(H)| = t$ and with at most $t/10$ pure items would satisfy $t_c(H) \le \log n$ and $|\mathrm{Clan}(P^*(H))| \le t\log^3 n \le \log^7 n$. For each $t \le \log^4 n$ and each t_c, $t_c \le t$, there are at most $\binom{\log n}{t_c}\binom{2n}{t-t_c}$ different sets T with $|T| = t$ containing t_c literals x with non-tree-like $\mathrm{In}_{F_2^*}(x)$. Therefore, by Lemma 4.4(4.4), the probability that there is some subformula H of F^* with $|P^*(H)| = t$, $|t_c(H)| = t_c$ and at most $t/10$ pure items is at most

$$(\log n)^{t_c}(2n)^{t-t_c}(K/n^2)^{19t/30-t_c/3}(\log^7 n)^{19t/10-t_c}$$

which is bounded above by $(K'')^t(\log n)^{14t}n^{-4t/15}$ for some constant $K'' = K''(\Delta,\rho) > 0$. The probability that an H satisfying the conditions of the lemma with $t(H) \le \log^4 n$ exists is then at most

$$\sum_{t_c=1}^{\log n}\sum_{t=t_c}^{\log^4 n}(K''n^{-4/15}\log^{14} n)^t \le K''n^{-4/15}\log^{19} n$$

for n sufficiently large, which is $o(1)$ in n. □

It will be convenient to rewrite the summations over all possible choices of set $T = P^*(H)$ with $|T| = t \ge \log^4 n$ in terms of a probability calculation involving a uniformly chosen random set of literals, T, of size t. Recalling Clan^* from Definition 3.1 , observe that for any such T, $\mathrm{Clan}(T) \le 2\mathrm{Clan}^*(T)$ since there are are most $\log^4 n$ literals in clans of literals x with $\mathrm{In}_{F_2^*}(x)$ not tree-like.

Lemma 4.6. *Fix $\rho > 0$. There is a constant $B = B(\rho) > 0$ such that for any t and for T a set of literals with $|T| = t$ chosen uniformly at random, $\mathbf{E}_T(|\mathrm{Clan}^*(T)|) \le Bt$.*

Proof. Let $B = \sum_{i\ge 1} i(1-\rho)^i$. By assumption, for x chosen uniformly at random from among the $2n$ possible literals, $\mathbf{E}_x(|\mathrm{Clan}^*(x)|) \le 2 + \sum_{i\ge 3} i(1-\rho)^i = B$ and therefore $\mathbf{E}_T(|\mathrm{Clan}^*(T)|) \le Bt$. □

Lemma 4.7. *For every $\rho > 0$ there exists $\alpha = \alpha(\rho) > 0$ such that for all $r \ge 0$ we have for T a set of literals with $|T| = t$ chosen uniformly at random,*

$$\mathbf{Pr}_T(|\mathrm{Clan}^*(T)| > (r+16)\mathbf{E}_T(|\mathrm{Clan}^*(T)|)) < 2\cdot e^{-\alpha\sqrt{r}t}.$$

This lemma is proven in Section 6 using a moment generating function argument.

Lemma 4.8. *Fix $\rho > 0$. There is $K_1 = K_1(\rho)$ such that for any $t > 0$ and for a set of literals T with $|T| = t \ge \log^4 n$ chosen uniformly at random, $E_T(|\mathrm{Clan}(T)|^{9t/5}) \le (K_1 t)^{9t/5}$.*

Proof. Fix an integer $t \ge \log^4 n$ and consider choosing T uniformly at random with $|T| = t$. Since $t \ge \log^4 n$, it suffices to prove the result for $|\mathrm{Clan}^*(T)|$ instead of $|\mathrm{Clan}(T)|$ since the latter is at most twice the former. We divide up the range of possible values of $|\mathrm{Clan}^*(T)|$ into segments of size $\nu(T) = \mathbf{E}_T(|\mathrm{Clan}^*(T)|) \le Bt$ where $B = B(\rho)$ is the constant from Lemma 4.6 and use our tail bounds within each segment. Therefore by Lemma 4.7,

$$\begin{aligned}
\mathbf{E}_T(|\mathrm{Clan}^*(T)|^{9t/5}) &\le (16\cdot\mathbf{E}_T(|\mathrm{Clan}^*(T)|))^{9t/5} \\
&\quad + \sum_{r\ge 0}\mathbf{Pr}_T(|\mathrm{Clan}^*(T)| > (r+16)\nu(T)) \times [(r+17)\nu(T)]^{9t/5} \\
&\le [\nu(T)]^{9t/5} \times (16^{9t/5} + 2\cdot\sum_{r\ge 0} e^{-\alpha\sqrt{r}t}(r+17)^{9t/5}) \\
&\le (Bt)^{9t/5} \times \left(\frac{K_1}{B}\right)^{9t/5} \\
&\le (K_1 t)^{9t/5},
\end{aligned}$$

for some $K_1 = K_1(\alpha, B) = K_1(\rho)$. □

Lemma 4.9. *Fix $\Delta, \rho > 0$. There is $\zeta_0 = \zeta_0(\Delta, \rho) > 0$ such that the probability that F^* has a subformula H with $t = |P^*(H)| \leq \zeta_0 n$ and with at most $t/10$ pure items is $o(1)$ in n.*

Proof. By Lemmas 4.4(4.4) and 4.5, the probability of this event is at most $\sum_{T, \log^4 n \leq |T| \leq \zeta_0 n} R(T)$ plus a term that is $o(1)$ in n. Using Lemma 4.8, we obtain:

$$\begin{aligned}
\sum_{T,|T|=t} R(T) &\leq \binom{2n}{t} (K/(tn^2))^{3t/5} E_T(|\mathrm{Clan}(T)|^{9t/5}) \\
&\leq (2en/t)^t (K/(tn^2))^{3t/5} (K_1 t)^{9t/5} \\
&= ((2e)^5 K^3 K_1^9 t/n)^{t/5} \\
&\leq (K_2 t/n)^{t/5}
\end{aligned}$$

for some constant $K_2 = K_2(\Delta, \rho) > 0$.

Now if we let $\zeta_0 = 1/(32K_2)$ then the probability that such an H exists is at most a term that is $o(1)$ in n plus

$$\sum_{t=\log^4 n}^{n/(32K_2)} (K_2 t/n)^{t/5} \leq \sum_{t \geq \log^4 n} 2^{-t}$$

which is also $o(1)$ in n. $\square$

Lemma 4.9 immediately implies Lemma 3.2(a) since if H has no pure items, then of course it has at most $t/10$ pure items where $t = |P^*(H)|$. It doesn't quite prove Lemma 3.2(b) since we need to rule out H having fewer than μn pure items when $\frac{1}{2}\zeta n < |V(H)| \leq \zeta n$ whereas Lemma 4.9 only rules out subformulas H with up to $\frac{1}{10}|P^*(H)|$ pure items. So Lemma 4.9 falls short on the case where $|P^*(H)| = o(|V(H)|)$. We rectify this problem by showing that if $|V(H)| = \Theta(n)$ then $|P^*(H)| = \Theta(n)$ and using our polylogarithmic upper bound on the number of pure cycles in H.

Lemma 4.10. *Fix Δ, ρ and consider any $\zeta > 0$. Then there exists $\delta = \delta(\zeta, \rho)$ such that if H is a subformula on more than $\frac{1}{2}\zeta n$ variables, then $|P^*(H)| > \delta n$.*

Proof. Choose $I = I(\zeta, \rho) \geq 2$ such that $\sum_{i>I} 2i(1-\rho)^i < \frac{1}{2}\zeta$. Let $P_I^* \subset P^* = P^*(H)$ be the set of items in $P^*(H)$ whose clans have size greater than I. By Lemma 4.2 and the fact that $cover(P^*)$ is closed under complementations of literals, $|cover(P^*)| \geq 2|V(H)|$ which is more than ζn. By condition 4 of Lemma 3.2, $|cover(P_I^*)| \leq \frac{1}{2}\zeta n$, and so we must have

$$|cover(P^* - P_I^*)| \geq |cover(P^*)| - |cover(P_I^*)| > \zeta n - \frac{1}{2}\zeta n > \frac{1}{2}\zeta n.$$

Since each literal x in $P^* - P_I^*$ has $|cover(x)| \leq 2|\mathrm{Clan}(x)| \leq 2I$, this implies that $|P^* - P_I^*| \geq \zeta n/(4I)$ and thus proves the lemma with $\delta = \zeta/(4I)$. $\square$

Proof of Lemma 3.2. Given Δ, ρ we take $\zeta = \zeta_0(\Delta, \rho)$ from Lemma 4.9 and we set $\mu = \delta(\zeta, \rho)/11$ from Lemma 4.10. Part (a) is immediate. If F^* is as in part (b), then it must have at least $\delta(\zeta, \rho)n/10 = \frac{11}{10}\mu n$ pure items. By condition 2 from Lemma 3.2, at most $\log n$ of these items can be pure cycles. Therefore, F^* has at least $\frac{11}{10}\mu n - \log n > \mu n$ pure literals; this proves part (b). $\square$

5 Properties of subcritical random 2-CNF formulae

We will now prove that subcritical random 2-CNF formulas satisfy the properties in Lemma 3.2 w.h.p.

Lemma 5.1. *Let F_2 be random 2-SAT formula formed by picking $m_2 = (1-\epsilon)n$ clauses from $C_2(n)$ uniformly, independently and with replacement. There exists $\rho = \rho(\epsilon) > 0$ such that w.h.p. all of the following hold simultaneously.*

1. *For every literal ℓ, $\mathrm{In}_{F_2}(\ell)$ is simple.*
2. *There are at most $\log n$ literals ℓ, such that $\mathrm{In}_{F_2}(\ell')$ is not tree-like for some $\ell' \in \mathrm{Clan}_{F_2}(\ell)$.*
3. *For every literal ℓ, $|\mathrm{Clan}_{F_2}(\ell)| \le \log^3 n$.*
4. *For all $i \ge 3$, $|\{\ell : |\mathrm{Clan}^*_{F_2}(\ell)| \ge i\}| \le 2(1-\rho)^i n$.*

Proof. To prove this lemma it will be easier to work with random formulas formed by including each of the $4\binom{n}{2}$ possible 2-SAT clauses independently with probability p. In particular, we will prove that each of the four properties holds w.h.p. in such a random formula when $p = (1-\epsilon)/(2n)$, for every $\epsilon > 0$. Given that, using the observations of the paragraph below, it is easy to establish that each of the four properties holds w.h.p. when we pick $m = (1-\epsilon)n$ clauses from $C_2(n)$, for every $\epsilon > 0$. This readily implies the lemma since the intersection of any finite collection of high probability events also holds w.h.p.

First, observe that each of the four properties is monotone decreasing, i.e. adding clauses can only hurt a formula in terms of having each property. Secondly, observe that if a formula contains multiple occurrences of a clause (which could happen when we pick clauses with replacement), we can remove all but one of these occurrences without affecting the property. Further, observe that if $p = (1-\epsilon')/(2n)$, then w.h.p. the resulting formula has at least $(1-\epsilon)n$ 2-clauses for every $\epsilon > \epsilon'$. Moreover, note that the resulting formula is uniformly random conditional on its number of clauses. Finally, note that the same is true for a random formula formed by picking clauses from $C_2(n)$ and removing any duplicates.

To prove that each of the four properties holds w.h.p. when $p = (1-\epsilon)/(2n)$ it will be useful to define for every literal $\ell \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, a set $S = S(\ell)$ of 'related literals', a set $D = D(\ell)$ of 'dangerous questions' and a set $B = B(\ell)$ of 'bad answers'. We first define the sets S and D by applying the following two-part procedure. Initially, we set $S = T = \{\ell\}$ and $D = \emptyset$. (T will be an auxiliary set.)

i) Repeat the following until $T = \emptyset$: choose a literal $x \in T$ and remove it from T; for every literal $y \ne x$, if neither y nor $\bar{y}$ is in S then if $(x \vee \bar{y})$ is in F_2 add y to both S and T; otherwise, if either y or $\bar{y}$ is in S, add $(x \vee \bar{y})$ to D.

When part i) terminates let $S_0 = S$ and $D_0 = D$. If $S_0 = \{\ell_1, \ldots \ell_s\}$ then add $R = \{\bar{\ell}_1, \ldots, \bar{\ell}_s\}$ to S. We will process the elements of R *in sequence*, as described below, with the processing of each element further expanding S. The sets S and D that result after we process all elements of R are the ones we associate with literal ℓ (we will derive B from D). The processing of each $\bar{\ell}_j \in R$ creates a set S_j and amounts to the following procedure, analogous to what we did for ℓ, where here we use an auxiliary set T_j initialized to $T_j = S_j = \{\bar{\ell}_j\}$.

ii) Repeat the following until $T_j = \emptyset$: Choose a literal $x \in T_j$ and remove it from T_j. For every literal $y \ne x$, if neither y nor $\bar{y}$ is in S then if $(x \vee \bar{y})$ is in F_2 add y to S, S_j, and T_j; otherwise, if either y or $\bar{y}$ is in S, add $(x \vee \bar{y})$ to D.
The set B consists of those clauses in D that are also in F_2.

The reason for introducing $S(\ell)$ is that $|S(\ell)|$ can be bounded by considering a "branching process" argument, while at the same time $S(\ell)$ allows us to capture the behavior of $\mathrm{In}_{F_2}(\ell)$ and $\mathrm{Clan}_{F_2}(\ell)$ for a "typical" literal ℓ. Our branching process is analogous to an iterated version of the branching process used to bound the component sizes in sub-critical random graphs [11]. In particular, after running such a process once, we run a new independent process for each vertex in the first process. Naturally, we need some additional twists to handle the fact that we have a directed graph on literals (where each clause creates two directed edges) and the fact that the presence of cycles makes the branching process analogy imperfect.

To prove that each of the first three properties holds w.h.p. we will first show two deterministic relations between S and B (Claim 5.2 below) and then we will give a tail bound for $|S(\ell)|$ for a fixed literal ℓ. To prove the fourth property, we will need to do some additional work in order to show that the number of literals whose clan-star has size at least i is a sharply concentrated random variable.

Claim 5.2.

1. *If $B(\ell) = \emptyset$, then $\mathrm{In}_{F_2}(\ell)$ is tree-like and $\mathrm{Clan}_{F_2}(\ell) = S(\ell)$. Moreover, for all $\ell' \in \mathrm{Clan}_{F_2}(\ell)$, $\mathrm{In}_{F_2}(\ell')$ is tree-like.*
2. *If $|B(\ell)| \leq 1$, then $\mathrm{In}_{F_2}(\ell)$ is simple and $\mathrm{In}_{F_2}(\ell) \subseteq S(\ell)$.*

Proof. Suppose $B(\ell) = \emptyset$, i.e., that F_2 contains no elements of $D(\ell)$. In creating $S_0(\ell)$, we mimicked a search procedure discovering $\mathrm{In}_{F_2}(\ell)$ and did not find any cycles; the only potential locations for edges in $\vec{D}(F_2)$ that lead to $S_0(\ell)$ were those corresponding to clauses in $D_0(\ell)$. Since, by assumption, $F_2 \cap D_0(\ell) = \emptyset$, it follows that the search is complete, $S_0(\ell) = \mathrm{In}_{F_2}(\ell)$ and $\mathrm{In}_{F_2}(\ell)$ is tree-like. Moreover, by the same reasoning, since $F_2 \cap D(\ell) = \emptyset$, each $S_j(\ell)$ is tree-like and

$$\bigcup_{y \in \mathrm{In}_{F_2}(\ell)} \mathrm{In}_{F_2}(\bar{y}) = \bigcup_{j=1}^{|S_0(\ell)|} S_j(\ell)$$

and thus $\mathrm{Clan}_{F_2}(\ell) = S(\ell)$.

Suppose that F_2 contains precisely one element of $D(\ell)$. If that element is not in $D_0(\ell)$ then by the above argument $\mathrm{In}_{F_2}(\ell)$ is tree-like and $\mathrm{In}_{F_2}(\ell) = S_0(\ell) \subset S(\ell)$. therefore we consider the case that the element is in $D_0(\ell)$. There are two kinds of clauses in $B_0(\ell)$: those corresponding to 'internal' edges, those edges (y, x) where $x, y \in S_0(\ell)$, and 'external' edges, those that would create an edge $(\bar{y}, x)$ in $\vec{D}(F_2)$ (and also the edge $(\bar{x}, y)$) where $x, y \in S_0(\ell)$. If the single element of $D_0(\ell) \cap F_2$ corresponds to an internal edge then $S_0(F_2) = \mathrm{In}_{F_2}(\ell)$ and $\mathrm{In}_{F_2}(\ell)$ has precisely one cycle. If that element corresponds to external edges $(\bar{y}, x)$ and $(\bar{x}, y)$ where $x, y \in S_0(\ell)$ then $S_0(\ell)$ is internally tree-like and we find precisely one cycle involving this pair of edges in the subgraph of $G(F_2)$ induced by $S_0(\ell) \cup R(\ell)$. The search for $\mathrm{In}_{F_2}(\ell)$ does not end with $S_0(\ell)$ because we have missed exploring from $\bar{x}$ and from $\bar{y}$. However, in creating $S(\ell)$ we also search from every literal in $R(\ell)$ and, since none of the elements in $D(\ell) - D_0(\ell)$ is in F_2, these searches will fully explore $\mathrm{In}_{F_2}(\bar{x})$ and $\mathrm{In}_{F_2}(\bar{y})$ and thus will finish the exploration of $\mathrm{In}_{F_2}(\ell)$, yielding $\mathrm{In}_{F_2}(\ell) \subseteq S(\ell)$. Furthermore, no additional cycles will be found in these sets and thus $\mathrm{In}_{F_2}(\ell)$ will be simple. $\square$

Claim 5.3. *Fix any literal ℓ and let $S = S(\ell)$. There exists $\sigma = \sigma(\epsilon) > 0$ such that for every $q > 2$, $\Pr[|S| \geq q] < (1 - \sigma)^q$.*

Proof. Let $\mathcal{C}_{p,n}$ be the distribution of the number of vertices in the connected component $C(v)$ of a fixed vertex v in a random graph $G(2n, p = (1 - \epsilon)/(2n))$. Since each clause appears in the formula with probability p it is easy to show (see e.g. [12]) that the size of the tree corresponding to $S_0(\ell)$ and each of the trees corresponding to the different $S_j(\ell)$ is dominated by $\mathcal{C}_{p,n}$.

Let $X \sim \mathcal{C}_{p,n}$ and let

$$W = W_1 + W_2 + \cdots + W_X$$

where the W_i are i.i.d. random variables with $W_i \sim \mathcal{C}_{p,n}$. Observe that $2W$ dominates $|S(\ell)|$ since each $|S_i(\ell)|$ is dominated by W_i for $i \geq 1$. To analyze, $|S(\ell)|$ we will obtain an upper tail bound on W using a standard moment generating function argument. For any $h > 0$ we have

$$\begin{aligned}\Pr[W \geq q] &= \Pr[\exp(hW) \geq \exp(hq)] \\ &\leq \exp(-hq)\mathbf{E}(\exp(hW)) \ ,\end{aligned}$$

using Markov's (or Bernstein's) inequality. Note now that

$$\mathbf{E}(\exp(hW)) = \sum_{k\geq 1} \Pr[X = k]\, \mathbf{E}\left(\prod_{i=1}^{k} \exp(hW_i)\right) \tag{2}$$

$$= \sum_{k\geq 1} \Pr[X = k] \prod_{i=1}^{k} \mathbf{E}\left(\exp(hW_i)\right) \tag{3}$$

$$= \sum_{k\geq 1} \Pr[X = k] \, \left(\mathbf{E}\left(\exp(hW_i)\right)\right)^k \ , \tag{4}$$

where passing from (2) to (3) uses that the random variables W_i are independent while passing from (3) to (4) uses that they are identically distributed.

Let $\Pr[W_i = k] = \Pr[X = k] \equiv p_k$. Let $c = 1 - \epsilon$. To bound (4) we will use the following well-known facts (see, for example, p.156 in [7]). Namely, that asymptotically in n,

$$p_k = (1/c) \left(ce^{-c}\right)^k k^{k-1}/k! \tag{5}$$

and that for all $c < 1$,

$$\sum_{k\geq 1} p_k = 1 \ . \tag{6}$$

Further, let us note that the function $f(x) = xe^{-x}$ is continuous and increasing in $[0, 1)$, going from $f(0) = 0$ to $f(1) = e^{-1}$. In particular for all $0 < \epsilon < 1$, if $c = 1 - \epsilon$, $d = c + \epsilon^2/3$ and $h = \epsilon^3/3$, then $ce^{-c+h} < de^{-d}$ and $d < 1$. Moreover, $de^{-c} < e^{-1}$ which will be useful in bounding $\mathbf{E}(\exp(hW))$. With all this in mind,

$$\begin{aligned}\mathbf{E}(\exp(hW_i)) &= \sum_{k\geq 1} p_k \exp(hk) \\ &= (1/c)\sum_{k\geq 1}(ce^{-c+h})^k k^{k-1}/k! \\ &< (1/c)\sum_{k\geq 1}(de^{-d})^k k^{k-1}/k! \\ &= d/c \ .\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{E}(\exp(hW)) &< \sum_{k\geq 1} p_k (d/c)^k \\ &= (1/c)\sum_{k\geq 1} (de^{-c})^k k^{k-1}/k! \\ &< (1/c)\sum_{k\geq 1} (de^{-c+1})^k \ ,\end{aligned}$$

which is a geometric series with ratio $de^{-c+1} < 1$ depending only on ϵ and therefore converges to some $U(\epsilon) < \infty$. Therefore, for $1 > \epsilon > 0$ there is $\tau = \tau(\epsilon) > 0$ and $q_0 = q_0(\epsilon)$ such that for all $q \geq q_0$,

$$\begin{aligned}\Pr[W \geq q] &< \exp(-hq)U(\epsilon) \\ &= U(\epsilon)\exp\left(-\frac{\epsilon^3}{3}q\right) \\ &\leq (1-\tau)^q \ .\end{aligned}$$

To obtain an upper bound for *all* q as opposed to just those larger than q_0, we observe that for any fixed value of $q > 1$, $\Pr[W \geq q] < 1$, and so there exists $\sigma' = \sigma'(\tau, q_0)$ such that for all $q > 1$,

$$\Pr[W \geq q] < (1-\sigma')^q \ . \tag{7}$$

Now, letting $\sigma = \sigma'/2$, we see that

$$\Pr[|S(\ell)| \geq q] \leq \Pr[W \geq q/2] < (1-\sigma')^{q/2} < (1-\sigma)^q.$$

□

Claim 5.4. *W.h.p. there are at most* $\log n$ *literals for which* $|B(\ell)| > 0$ *and no literals for which* $|B(\ell)| > 1$.

Proof. Observe that $|D(\ell)| \leq 2|S(\ell)|^2$ and that each element of $D(\ell)$ appears in the formula with probability $p < 1/n$. Therefore, for any ℓ,

$$\Pr[|B(\ell)| > 0] \leq p\mathbf{E}(2|S(\ell)|^2) < 10p + \sum_{k>2}(1-\sigma)^k 2k^2 p = C/n \ ,$$

for some constant $C = C(\epsilon)$. Therefore, the expected number of literals ℓ with $|B(\ell)| > 0$ is bounded by $2C$ and by Markov's inequality, w.h.p. there are at most $\log n$ such literals. Similarly, observe that

$$\Pr[|B(\ell)| > 1] \leq p^2\mathbf{E}\left(\binom{2|S(\ell)|^2}{2}\right) = o(1/n)$$

and therefore w.h.p. no such literals exist. □

Proof of 1: By Claim 5.4 w.h.p. for every ℓ we are in one of the two cases of Claim 5.2. In both cases $\mathrm{In}_{F_2}(\ell)$ is simple.
Proof of 2: By Claim 5.4, w.h.p. there are at most $\log n$ literals not falling in the first case of Lemma 5.2. It is only for such literals that $\mathrm{Clan}_{F_2}(\ell)$ could contain some ℓ' such that $\mathrm{In}_{F_2}(\ell')$ is not tree-like.
Proof of 3: First observe that by (7) there exists a constant C such that w.h.p. for every ℓ, $S(\ell)$ has size at most $C\log n$. Since w.h.p. every $|B(\ell)| \leq 1$, every $\mathrm{In}_{F_2}(\ell)$ is contained in its associated $S(\ell)$ and therefore has size at most $C\log n$. Therefore for any ℓ,

$$\begin{aligned}|\mathrm{Clan}_{F_2}(\ell)| &\leq |\mathrm{In}_{F_2}(\ell)| + \sum_{y\in \mathrm{In}_{F_2}(\ell)} |\mathrm{In}_{F_2}(\bar{y})| \\ &\leq C\log n + (C\log n)^2 \\ &\leq \log^3 n \ .\end{aligned}$$

Proof of 4: We will choose ρ later as a function of ϵ. Observe that we only need to consider ℓ such that $B(\ell) = \emptyset$ since $B(\ell) \neq \emptyset$ precisely when $\mathrm{Clan}_{F_2}(\ell)$ has a literal ℓ' with $\mathrm{In}_{F_2}(\ell')$ not tree-like.

For these literals, by Claim 5.2, $\mathrm{Clan}_{F_2}(\ell) = S(\ell)$. Therefore for $i \geq 3$, by linearity of expectation and Claim 5.3,

$$\mathbf{E}(T_i(F_2)) < 2n \times (1-\sigma)^i. \tag{8}$$

Our next step is to show that for each i there exists $Q_i \leq \mathbf{E}\left(T_i(F_2)\right)$ such that w.h.p. for all $i \geq 3$,

$$|T_i(F_2) - Q_i| < n^{3/4} \quad . \tag{9}$$

To prove (9) we will need to do some work before appealing to a concentration inequality. The reason for this is that, a priori, replacing a single clause in F_2 could change $T_i(F_2)$ dramatically, for some i; luckily, this is an unlikely event. To capture this last fact we will introduce a family of random variables U_i with the following properties: i) w.h.p. $U_i(F_2) = T_i(F_2)$ for all i, and ii) by definition (of the U_i), replacing a clause in F_2 can affect each U_i by at most $\mathrm{polylog}(n)$. Thus, appealing to a large deviation inequality for the U_i will yield the desired result.

The random variables U_i are motivated by the following observation.

Observation 5.5. *If $x \in \mathrm{Clan}_{F_2}(y)$ then $\bar{y} \in \mathrm{Clan}_{F_2}(\bar{x})$. Thus, if $B = \max_x |\mathrm{Clan}_{F_2}(x)|$, then no literal appears in more than B clans.*

Recall now that for every literal ℓ, $\mathrm{Clan}^*(\ell) \subseteq \mathrm{Clan}(\ell)$ and w.h.p. $|\mathrm{Clan}(\ell)| \leq \log^3 n$ for all ℓ. Therefore, the above observation suggests that when adding/removing a single arc $\vec{e}$ in $\vec{D}(F_2)$ there are at most $\log^3 n$ literals for which $\mathrm{Clan}^*_{F_2}(\ell)$ changes. This is because $\mathrm{Clan}_{F_2}(\ell)$ can change only if it contains one of the two endpoints of $\vec{e}$ and, by our observation, each endpoint of $\vec{e}$ appears in at most $\log^3 n$ clans.

This leads us to introduce the notion of the *domponent*, $\mathrm{Domp}(\ell)$, of a literal ℓ. The domponents of all literals in a 2-SAT formula F are determined as follows. We first associate with each arc $\vec{e}$ in $\vec{D}(F)$ a $count(\vec{e})$ equal to the number of clan-stars in which $\vec{e}$ is present ($\vec{e}$ is present in a clan-star if it was followed at least once in determining that clan-star). We then create a subgraph $\vec{D'}(F)$ of $\vec{D}(F)$ by removing all arcs $\vec{e}$ such that $count(\vec{e}) \geq \log^3 n$. The domponent of each literal ℓ is then its clan-star in $\vec{D'}(F)$. If for a literal ℓ, $\mathrm{Domp}(\ell) = \mathrm{Clan}^*_{F_2}(\ell)$ then we will say that $\mathrm{Domp}(\ell)$ is *good*. Analogously to $T_i(F_2)$ we let

$$U_i(F_2) = |\{x : |\mathrm{Domp}(x)| = i \text{ and } \mathrm{Domp}(x) \text{ is good}\}| \quad .$$

Note that for all i, by definition, $U_i(F_2) \leq T_i(F_2)$ and therefore $\mathbf{E}(U_i(F_2)) \leq \mathbf{E}(T_i(F_2))$. Further, note that by part 3 of the lemma w.h.p. $\vec{D}(F) = \vec{D'}(F)$. Therefore, to prove (9) it suffices to take $Q_i = \mathbf{E}(U_i(F_2))$ and prove that w.h.p. for all i, $|U_i(F_2) - \mathbf{E}(U_i(F_2))| < n^{3/4}$.

To prove that the random variables U_i are concentrated around their expectation we consider the probability space corresponding to the m_2 independent choices of clauses from $C_2(n)$ that determine F_2. We claim that for any possible set of values for these choices (i.e. for any set of clauses), changing the value of any single random variable (i.e. replacing a clause with some other clause) can only affect $\mathrm{Domp}(\ell)$ for at most $4\log^3 n$ literals. To prove this claim we break-down the replacement of a clause to four steps corresponding to the four arcs that are removed/added in $\vec{D}(F_2)$. The claim then follows by the fact that the removal/addition of each such arc can affect $\mathrm{Domp}(\ell)$ for at most $\log^3 n$ literals. This last assertion follows trivially from the fact that, by the definition of domponents, the arc $\vec{e}$ being added (removed), cannot be traversed (have been traversed) during the determination of the domponents more than $\log^3 n$ times.

Given the above claim, we can apply the following inequality of McDiarmid [30] to get that the probability of each D_i deviating by $n^{3/4}$ is bounded by $\exp(-n^{1/5})$. The union bound then implies that w.h.p. no D_i deviates by that much.

Theorem 5.6 ([30]). *Let $\mathbf{X} = (X_1, X_2, \dots, X_n)$ be a family of independent random variables with each X_k taking values in a set A_k. Suppose that the real-valued function f defined on $\prod A_k$ satisfies*

$$|f(\mathbf{x}) - f(\mathbf{x}')| \leq c_k$$

whenever the vectors $\mathbf{x}$ and $\mathbf{x}'$ differ only in the k-th coordinate. Let μ be the expected value of the random variable $f(\mathbf{X})$. Then for any $t \geq 0$,

$$\begin{aligned} \Pr[f(\mathbf{X}) \geq \mu + t] &\leq \exp\left(-2t^2/\sum c_k^2\right) \qquad \textit{and} \\ \Pr[f(\mathbf{X}) \leq \mu - t] &\leq \exp\left(-2t^2/\sum c_k^2\right) \ . \end{aligned}$$

Combining (8) and (9) we get that there exists $\sigma = \sigma(\epsilon) > 0$ such that w.h.p. for $i \geq 3$,

$$T_i(F_2) \leq 2n \times (1-\sigma)^i + n^{3/4} \ . \tag{10}$$

Further, recall that by (7)

$$\text{w.h.p.} \quad T_i(F_2) = 0 \text{ for all } i \geq C \log n \ . \tag{11}$$

Let us now choose $\phi < \sigma$ such that $(1-\phi)^{C \log n} \geq n^{-1/4}$. Thus, for all $i < C \log n$

$$2n \times (1-\phi)^i \geq 2n^{3/4} \ . \tag{12}$$

We claim that w.h.p. for all $i \geq 3$,

$$T_i(F_2) \leq (4n) \times (1-\phi)^i \ . \tag{13}$$

If $i \geq C \log n$ then (13) holds by (11). If $i < C \log n$ then by (10), (13) and $\phi < \sigma$, respectively,

$$\begin{aligned} T_i(F_2) &\leq 2n \times (1-\sigma)^i + n^{3/4} \\ &\leq 2n \times (1-\sigma)^i + 2n \times (1-\phi)^i \\ &\leq 4n \times (1-\phi)^i \ . \end{aligned}$$

By (10) and (13) it follows that there is $\rho < \phi$ such that w.h.p. for all $i \geq 3$, $T_i(F_2) \leq 2n \times (1-\rho)^i$. $\square$

6 Proof of Lemma 4.7

We will prove a somewhat more general concentration statement, cast in terms of picking weighted balls without replacement.

Lemma 6.1. *Let $\mathcal{B}$ be a set of m weighted balls, each ball x having integer $\mathrm{weight}(x) \geq 0$. Let B_i denote the number of balls with weight i and suppose that there is a $\rho > 0$ such that*

$$B_i \leq (1-\rho)^i m, \quad \textit{for all } i \geq 0 \ . \tag{14}$$

Then there is an $\alpha' > 0$ such that for every $\xi \geq 1$ and $1 \leq t \leq m/2$, if we choose a random subset $R \subseteq \mathcal{B}$ of t balls, and let $W = \sum_{x \in R} \mathrm{weight}(x)$ then

$$\Pr[W > 4(1+\xi)^2 \mathbf{E}(W)] < 2\exp(-\alpha' \xi t) \ .$$

Lemma 4.7 follows from Lemma 6.1 by setting $m = 2n$, $B_i = |\{x : \mathrm{Clan}^*(x) = i+2\}|$, $\alpha = \alpha'/3$, and $4(1+\xi)^2 = r + 16$ and observing that $\xi = \sqrt{4 + r/4} - 1 \geq \max\{1, \sqrt{r}/3\}$.

Proof of Lemma 6.1. We start by considering W to be defined in the following, equivalent, manner. Let S be an infinite sequence of balls formed by choosing balls uniformly, independently and *with replacement* from $\mathcal{B}$. Let W be the sum of the weights of the first t distinct elements of S.

Let us consider the prefix $P = p_1, p_2, \ldots, p_d$ of S where $d = 2(1+\xi) \times t$. In particular, let us form a random set $R' \subseteq \mathcal{B}$, by scanning P linearly and adding to R' every ball not seen before, until either $|R'| = t$ or we exhaust P. Let

$$W' = \sum_{x \in R'} \mathrm{weight}(x) \qquad \text{and} \qquad Q = \sum_{i=1}^{d} \mathrm{weight}(p_i) \ .$$

Then, by (the miracle of) linearity of expectation, we see that $\mathbf{E}(Q) = 2(1+\xi)\mathbf{E}(W)$ and, thus, for any $\xi > 0$

$$\begin{aligned} \Pr[W > 4(1+\xi)^2\mathbf{E}(W)] &\leq \Pr[W' > 4(1+\xi)^2\mathbf{E}(W)] + \Pr[W' \neq W] \\ &\leq \Pr[Q > 4(1+\xi)^2\mathbf{E}(W)] + \Pr[W' \neq W] \\ &\leq \Pr[Q > (2+\xi)\mathbf{E}(Q)] + \Pr[W' \neq W] \ . \end{aligned}$$

For $W' \neq W$ to occur it must be that we picked $2(1+\xi)t$ balls out of m balls with replacement and got fewer than t distinct balls. We start by proving that for all $\xi \geq 1$, the probability of this event is bounded above by $\exp(-\xi t/2)$. For this, we first observe that the expected number of balls drawn after drawing the i-th distinct ball and until drawing the $(i+1)$-st distinct ball is $m/(m-i)$. Therefore, since $t \leq m/2$, it follows that after drawing $2(1+\xi)t$ balls we expect at least $(1+\xi)t$ distinct balls. To prove the probability bound we will use Theorem 5.6. In particular, we let X_i be the label of the i-th ball drawn and we let f be the number of distinct balls. Clearly, the random variables $\{X_i\}$ are independent and we can take $c_k = 1$ for all k. Therefore we get that the probability we draw fewer than t distinct balls, for all $\xi \geq 1$, is bounded above by

$$\Pr[f(\mathbf{X}) < t] \leq \Pr[f(\mathbf{X}) < \mu - \xi t] \leq \exp\left(-\frac{2(\xi t)^2}{2(1+\xi)t}\right) \leq \exp(-\xi t/2) \ .$$

We will prove below that $\Pr[Q > (2+\xi)\mathbf{E}(Q)] < \exp(-\theta\xi t)$ for some $\theta = \theta(\rho) > 0$. Combining this with the estimate for $W' \neq W$ we get that for $\xi \geq 1$ the probability of having $W > 4(1+\xi)^2\mathbf{E}(W)$ is at most $\exp(-\theta\xi t) + \exp(-\xi t/2) \leq 2\exp(-\alpha'\xi t)$ for $\alpha = \min\{\phi, 1/2\}$ as required.

To prove our tail bound on Q we first note that for any $h > 0$,

$$\begin{aligned} \Pr[Q > (2+\xi)\mathbf{E}(Q)] &= \Pr[\exp(hQ) > \exp((2+\xi)h\mathbf{E}(Q))] \\ &\leq \mathbf{E}(\exp(hQ)) \times \exp(-(2+\xi)h\mathbf{E}(Q)) \ . \end{aligned} \tag{15}$$

Now let $\{Q_i\}_{i=1}^d$ be i.i.d.r.v. defined by $Q_i = \mathrm{weight}(p_i)$. Thus, $Q = \sum_{i=1}^d Q_i$ and as a result

$$\begin{aligned} \mathbf{E}(\exp(hQ)) &= \mathbf{E}\left(\prod_{i=1}^{d} \exp(hQ_i)\right) \\ &= \prod_{i=1}^{d} \mathbf{E}\left(\exp\left(hQ_i\right)\right) \\ &= \left(\mathbf{E}\left(\exp\left(hQ_i\right)\right)\right)^d \ . \end{aligned} \tag{16}$$

To simplify notation let us replace Q_i with T in the rest of the proof and let $\mu = \mathbf{E}(T)$.

To go from (17) to (18) we use (14). To go from (18) to (19) we require $h < \rho$, which suffices to guarantee the sum's convergence. Finally, to go from (19) to (20) we use that for $h > 0$, $e^{-h} > 1 - h$.

$$\begin{aligned}
\mathbf{E}(\exp(hT)) &= \sum_{i=0}^{\infty} \Pr[T = i] \exp(hi) \\
&= \sum_{i=0}^{\infty} \Pr[T = i] \left(1 + hi + (\exp(hi) - hi - 1)\right) && (17) \\
&\leq 1 + h\mu + \sum_{i=1}^{\infty} (1-\rho)^i (\exp(hi) - hi - 1) && (18) \\
&= 1 + h\mu + (1-\rho)\left(\frac{1}{\rho - 1 + \exp(-h)} - \frac{h+\rho}{\rho^2}\right) && (19) \\
&< 1 + h\mu + (1-\rho)\left(\frac{1}{\rho - h} - \frac{h+\rho}{\rho^2}\right) && (20) \\
&= 1 + h\mu + \frac{h^2(1-\rho)}{\rho^2(\rho - h)} \ . && (21)
\end{aligned}$$

Now, substituting $h = \rho^3$ in (21) we get (22), while (23) follows from $\mu \geq 1 > (\rho+1)^{-1}$.

$$\begin{aligned}
\mathbf{E}(\exp(\rho^3 T)) &< 1 + \rho^3\mu + \frac{\rho^3}{\rho+1} && (22) \\
&< 1 + 2\rho^3\mu \ . && (23)
\end{aligned}$$

Note now that, by (15) and (16), for all $h > 0$,

$$\begin{aligned}
\Pr[Q > (2+\xi)\mathbf{E}(Q)] &\leq \left(\frac{\mathbf{E}(\exp(hT))}{\exp((2+\xi)h\mathbf{E}(T))}\right)^{2(1+\xi)t} \\
&\leq \left(\frac{\mathbf{E}(\exp(hT))}{\exp(2h\mathbf{E}(T))}\right)^{2(1+\xi)t} \times \exp(-2h\xi\mu t) \ . && (24)
\end{aligned}$$

Taking $h = \rho^3$, (23) implies that the ratio in (24) is bounded by 1. Thus, since $\mu \geq 1$, if $\theta = 2\rho^3$, then

$$\Pr[Q > (2+\xi)\mathbf{E}(Q)] \leq \exp(-\theta\xi t) \ .$$

□

7 Implications for Satisfiability Algorithms

A number of algorithms for finding satisfying assignments for CNF formulas operate by building a partial assignment step by step. These algorithms commit to the assignments made at each step and operate on a *residual formula*, in which clauses already satisfied have been removed, while the remaining clauses have been shortened by the removal of their falsified literals. We call such algorithms *forward search* algorithms and they include the myopic algorithms UC and GUC mentioned in the introduction, as well as several more sophisticated variants [13, 14, 2, 5]. During the execution of any such algorithm a partial assignment may produce clauses of size 1 (unit clauses) in the residual formula which in turn create additional *forced* choices in the partial assignment, since the variables appearing in unit clauses have only one possible assignment if the formula is to be satisfied. The choices made by a forward search algorithm when no unit clauses are present are called *free*. As we saw, in UC a free choice amounts to assigning a random value to a random unassigned variable; in GUC a random literal in a random clause of smallest size in the residual formula is satisfied; the branching rule of ORDERED-DLL amounts to assigning 0 to the smallest-numbered unassigned variable (which makes a simple forward search version of ORDERED-DLL probabilistically equivalent to UC for random k-CNF).

We are interested in extensions of forward search algorithms to complete algorithms via backtracking. In any such extension, if a path in the search tree leads to a contradiction, the algorithm must begin backtracking by undoing all the (forced) choices up to the last free choice and flipping the assignment to that variable. From there, perhaps the simplest option would be for the algorithm to act as if it had reached this point without backtracking and apply the original heuristic to decide which variable(s) to set next. An alternative heuristic which we call FS-backtracking (inspired by [22]) is the following: When a contradiction is reached, record the portion of the assignment between the last free choice and the contradiction; these literals become *hot*. After flipping the value of the last free choice, instead of making the choice that the original heuristic would suggest, give priority to the complements of the hot literals in the order that they appeared; once the hot literals are exhausted continue as with the original heuristic. FS-backtracking is quite natural in that this last part of the partial assignment got us into trouble in the first place.

A key property of FS-backtracking that is useful in our analysis, as in that of [22], is that as long as the value of each variable in a partial assignment has been flipped at most once, the residual formula is uniformly random conditional on the number of clauses of each size. This property will be very useful for us as it would allow us to apply Theorem 1.1 to residual formulas generated after some backtracking has already occurred. We emphasize that while the original motivation for introducing FS-backtracking is technical convenience, FS-backtracking is also in fact a genuinely good algorithmic idea. Specifically, given a forward search algorithm A, let us write A-SIMPLE to denote its extension using simple backtracking and A-FS for its extension using FS-backtracking. Initial experiments comparing ORDERED-DLL-FS to ORDERED-DLL (which uses simple backtracking) on random formulas at ratios between 3.8 and 4.0 show that the histogram of run-times of FS-backtracking is *significantly better* than that of simple backtracking throughout the range.

Any DPLL algorithm A has the property that for any residual subformula F' created by A, either A satisfies F' or A produces a resolution refutation of F'. Thus, to prove lower bounds for DPLL algorithms, our plan is to prove that each such algorithm is likely to arrive at a point during its execution in which the residual formula F' is unsatisfiable but any resolution refutation of F' must have exponential size, implying that A must run for exponential time beyond that point.

Let us say that a DPLL algorithm is at a t-*stage* if precisely t variables have been set.

Definition 7.1. *Let $\epsilon = 10^{-4}$. A t-stage of a DPLL algorithm is* bad *if the residual formula at that stage is the union of a random 3-CNF formula with $(2.281 \pm \epsilon)(n-t)$ clauses and a random 2-CNF formula with $(0.999 \pm \epsilon)(n-t)$ 2-clauses, where $t \leq n/2$.*

Recall from our discussion in the introduction that formulas as in Definition 7.1 above are w.h.p. unsatisfiable while, by our Theorem 1.1, w.h.p. all their resolution refutations have exponential size.

Lemma 7.2. *Let $\Delta_{\text{UC}} = \Delta_{\text{ORDERED-DLL}} = 3.81$ and let $\Delta_{\text{GUC}} = 3.98$.*

1. *For each $A \in \{$UC,ORDERED-DLL,GUC$\}$, an execution of* any *backtracking extension of A on a random 3-CNF formula with $\Delta_A n$ clauses reaches a* bad *t-stage with constant probability.*

2. *For each $A \in \{$UC,ORDERED-DLL,GUC$\}$, an execution of algorithm A-*FS *on a random 3-CNF formula with $\Delta_A n$ clauses reaches a* bad *t-stage w.h.p.*

Corollary 7.3. *Let* $\Delta_{\text{UC}} = \Delta_{\text{ORDERED-DLL}} = 3.81$ *and let* $\Delta_{\text{GUC}} = 3.98$.

1. *For each* $A \in \{\text{UC},\text{ORDERED-DLL},\text{GUC}\}$*, an execution of* any *backtracking extension of* A *on a random 3-CNF formula with* $\Delta_A n$ *clauses takes time* $2^{\Omega(n)}$ *with constant probability.*
2. *For each* $A \in \{\text{UC},\text{ORDERED-DLL},\text{GUC}\}$*, an execution of algorithm* A-FS *on a random 3-CNF formula with* $\Delta_A n$ *clauses takes time* $2^{\Omega(n)}$ *w.h.p.*

Note that in Lemma 7.2 and Corollary 7.3 when we refer to ORDERED-DLL we consider *any* algorithm that extends the first branch of the standard version of ORDERED-DLL that does simple backtracking.

Proof of Lemma 7.2. The lemma follows from results in [13, 3, 22]. Below we outline these results and show how they can be combined. The original analyses in these papers were largely geared towards understanding the ratios between clauses and variables at which random k-CNF formulas remain satisfiable almost surely, particularly in the case that $k = 3$. In fact, virtually the only method known for determining lower bounds on the satisfiability threshold for 3-CNF formulas is based on analyzing such algorithms. These analyses apply primarily to forward search algorithms, such as UC and GUC.

A forward search algorithm is a prefix of any of its backtracking extensions — it corresponds to the first path explored in the backtracking search tree: We will show that our full DPLL algorithms reach bad t-stages by proving that the corresponding prefixes of those executions reach such bad t-stages.

We restate the previous analyses of some of the forward search algorithms on random 3-CNF formulas. The key property shown in all of these analyses is that when they are run on uniformly random formulas, the residual formula at each stage in these prefixes remains uniformly random conditional only on the number of clauses of each length. To state this more precisely, let $\mathcal{V}(t)$ denote the set of variables not assigned a value after t steps and let $C_j(t)$ denote the number of clauses in the residual formula with length j after t steps. Then, for each t, the set of j-clauses in the residual formula is distributed as a set of $C_j(t)$ clauses drawn uniformly, with replacement among all $2^j\binom{|\mathcal{V}(t)|}{j}$ j-clauses on the variables in $\mathcal{V}(t)$.

Given the above claim, to prove the lemma it suffices to prove that starting with a random 3-SAT formula with Δn clauses, with suitable probability there exists t such that the residual formula after t steps has the appropriate number of j-clauses for each $0 \le j \le 3$; i.e., it remains now to analyze the values of $C_j(t)$ as a function of t for the various algorithms. As is usual in such analyses, although the forward search algorithm would stop precisely when a 0-clause in the residual formula is created, we first do the analysis of the evolution of the residual formula without taking into account this stopping condition and then prove that with appropriate probability no 0-clause is created.

For $j = 2, 3$, it can be shown that the number of j-clauses at time t can be approximated by the scaled solution to a pair of differential equations. In particular, the following claims were proved in [13, 3] for UC,ORDERED-DLL and in [22] for GUC.

UC,ORDERED-DLL: For any $\delta > 0$, w.h.p. for all $0 \le t \le (1-\delta)n$,

$$\begin{aligned} C_3(t) &= \Delta(1-t/n)^3 \cdot n + o(n) \ , &(25)\\ C_2(t) &= \frac{3\Delta}{2}(t/n)(1-t/n)^2 \cdot n + o(n) \ . &(26)\end{aligned}$$

For any $\Delta > 2/3$ let α be the unique solution to $6\Delta x - 3\Delta x^2 + 4\ln(1-x) = 0$.
GUC: For any $\delta > 0$, w.h.p. for all $0 \le t \le (\alpha-\delta)n$,

$$\begin{aligned} C_3(t) &= \Delta(1-t/n)^3 \cdot n + o(n) \ , &(27)\\ C_2(t) &= \left(\frac{3\Delta}{2}t/n - \frac{3\Delta}{4}(t/n)^2 + \ln(1-t/n)\right)(1-t/n) \cdot n + o(n) \ . &(28)\end{aligned}$$

For the number of 1- and 0-clauses we will use another key claim which, intuitively, amounts to saying that if the density of the residual 2-CNF subformula remains bounded away from 1, then 1-clauses do not accumulate and with positive probability no 0-clauses are ever generated. More precisely, if for a given t_0 there exist $\delta, \epsilon > 0$ such that $t_0 \leq (1-\delta)n$ and w.h.p. for all $0 \leq t \leq t_0$, $C_2(t) \leq (1-\epsilon)(n-t)$, then with probability $\rho = \rho(\delta, \epsilon) > 0$, $C_1(t_0) + C_0(t_0) = 0$. (Note that since 0-clauses are never removed from the residual formula, having $C_0(t_0) = 0$ means that no 0-clauses were generated during the first t_0 steps.)

To gain some intuition for the last claim we observe that for all of the algorithms A we consider and all $t = 0, \ldots, n-1$, the expected number of unit clauses generated in step t is $C_2(t)/(n-t) + o(1)$. Since each algorithm can satisfy (and thus remove) one unit clause in each step, unit clauses will not accumulate as long as this rate is bounded above by $1 - \epsilon$ for some $\epsilon > 0$. In fact, under this condition, $C_1(t)$ behaves very much like the queue size in a stable server system. In particular, there exist constants $M = M(\delta)$ and $k > 0$ such that w.h.p. $C_1(t) < \log^k n$ for all t, and w.h.p. $\sum_t C_1(t) < Mn$. This implies, that the number of 0-clauses generated is dominated by a Poisson random variable with constant mean (the constant depending on M). Moreover, there is an $\ell > 0$ such that w.h.p. there is no period of $\log^\ell n$ consecutive steps in which C_1 is strictly positive.

Now, by substituting the given values for Δ, ϵ in equations (25)–(28) we see that indeed there exists $t \leq n/2$ such that at time t w.h.p. we have the right number of 2- and 3-clauses for a bad configuration. Moreover, up to that t, w.h.p. the density of 2-clauses stays uniformly below 1 and, therefore, with positive probability we indeed get a bad configuration. In particular, for UC,ORDERED-DLL, if $\Delta = \Delta_{\text{UC}} = 3.81$, this occurs when $t \approx .22625n$. For GUC, if $\Delta = \Delta_{\text{GUC}} = 3.98$, this occurs when $t \approx .243n$. This yields our positive probability results for arbitrary backtracking versions of UC, ORDERED-DLL, and GUC.

FS-Backtracking. As we saw above, as long as the density of the residual 2-CNF subformula is bounded below 1, the number of 1-clauses in one of the forward search algorithms behaves like a random walk with negative drift and a barrier at 0. As a result, it is natural to divide an algorithm's execution into *epochs*, where a step t ends an epoch if $C_1(t) = 0$. From our discussion above, each epoch has constant expected length and w.h.p. no epoch lasts more than a polylogarithmic number of steps.

Frieze and Suen [22] developed a method for improving the success probability of the above forward search algorithms with a small amount of backtracking using the notion of epoch. This limited backtracking allows one to backtrack to the beginning of the current epoch (but not further into the past). This epoch begins with a free choice followed by a sequence of forced choices. As in the usual backtracking algorithms, in Frieze and Suen's method one flips the value of the assignment made by the last free choice but, unlike usual backtracking algorithms, in their method one also flips the value of the assignment to *all* variables set so far during the current epoch. After all these values are flipped, if there are any unit clauses remaining then these propagations are done to finish the epoch. If a 0-clause is generated during this epoch after the flip then the algorithms fails. After the epoch is complete then all assignments are fixed and the algorithm continues as before.

Frieze and Suen's method does not do full backtracking and therefore, like the forward search algorithms, is an incomplete search procedure. It is easy to check that our modification, FS-backtracking, extends it to a complete backtracking search in such a way that the residual formulas that occur in their limited backtracking algorithms at the end of each epoch also appear as residual formulas using FS-backtracking. Although Frieze and Suen applied their method only to GUC, creating a procedure they called GUCB, it is clear that it can be used and analyzed in exactly the same manner for any algorithm having the property that the residual formula is uniformly random conditioned on the number of clauses of each length.

The first observation of Frieze and Suen's analysis is that the residual formula resulting after the flip is uniformly random conditional on the number of clauses of each length. (This was the motivation for the particular form of backtracking and would not be true if we did not flip all variables set so far during the epoch.) To see this, we separate the clauses of the residual formula at the beginning of the last epoch into *volatile clauses*, those containing a variable whose value is tentative in that may be flipped, and the remaining non-volatile clauses. Clearly, and in every step during the epoch, the set of non-volatile clauses remains uniformly random conditional on their size. Each volatile clause may contain literals that agree or disagree with the tentative value assignment. If a volatile clause contains any variable that disagrees with the tentative assignment then, when the assignment is flipped, the clause will be satisfied and therefore will disappear from the residual formula. It remains to consider the volatile clauses that only contain literals that agree with the tentative assignment. After the flip, these clauses will be shortened by the removal of the literals that agree with the tentative assignment. Before the tentative assignment was flipped, the only thing "exposed" about such clauses is that they contained one of these literals (since they were immediately satsified by it) and therefore the remaining literals in these clauses are uniformly random. Thus, the formula as a whole is uniformly random conditional on the number of clauses of each length.

The other key observation is that the number of volatile clauses that re-enter the residual formula as the result of a flip is at most polylogarithmic. This is because there are only a polylogarithmic number of variables flipped (by the epoch-length argument) and no variable appears in more than, say, $\log^2 n$ clauses, since we are dealing with sparse random formulas. As shown in [22], this implies that once the assignment has been flipped the probability of a second 0-clause being generated by that flipped assignment (together with its resulting unit propagation) is very small. In particular, this probability is so small that combined with the fact that each epoch's probability of requiring a flip is $O(1/n)$, it implies that w.h.p. no 0-clause is ever generated.

As a result, by considering epochs instead of individual steps, we get that w.h.p. at the end of each epoch there are no 1- or 0-clauses. Furthermore, the number of j-clauses, $j = 2, 3$ after t steps is still given by equations (25)–(28) (the $o(n)$ term absorbing the effect of any flips). Thus, after t variables have been set, where $t \approx .22625n$ for ORDERED-DLL-FS and UC-FS, and $t \approx .243n$ for GUC-FS, we see that each algorithm w.h.p. will be in a bad t-stage. □

8 Further Research

Our upper bounds on the number of 3-clauses needed to cause exponential behavior in satisfiability algorithms will be readily improved with any improvement on the $2.28n$ upper bound for unsatisfiability in random $(2+p)$-SAT. That is, if it is shown that for some $\epsilon > 0$ and $2/3 \leq \Delta < 2.28$, random formulas with $(1-\epsilon)n$ 2-clauses and Δn 3-clauses are unsatisfiable w.h.p. then the bounds of 3.81 and 3.98 will be immediately reduced. In fact, if Δ is reduced to $2/3$, to match the lower bound, then our results immediately imply the following remarkably sharp behavior: every card-type algorithm A is such that it operates in linear time with constant probability up to some threshold β_A but any backtracking extension of A requires exponential time with constant probability for all ratios larger than β_A. In fact, if A uses FS-backtracking then it would work in linear time almost surely at ratios below β_A and require exponential time almost surely above β_A.

It seems quite likely that one can extend our w.h.p. analysis to the simple backtracking versions of UC, GUC, ORDERED-DLL, and other card-type algorithms.

References

[1] D. Achlioptas. *Threshold Phenomena in random graph coloring and satisfiability*. PhD thesis, Department of Computer Science, University of Toronto, 1999.

[2] D. Achlioptas. Setting 2 variables at a time yields a new lower bound for random 3-SAT. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 28–37, Portland,OR, May 2000.

[3] D. Achlioptas. A survey of lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 265(1–2):159–185, 2001.

[4] D. Achlioptas, L. M. Kirousis, E. Kranakis, and D. Krizanc. Rigorous results for random $(2+p)$-SAT. *Theoretical Computer Science*, 265(1–2):109–129, 2001.

[5] D. Achlioptas and G. B. Sorkin. Optimal myopic algorithms for random 3-SAT. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 590–600. IEEE, 2000.

[6] Dimitris Achlioptas and Yuval Peres. The random k-SAT threshold is $2^k \ln 2 - O(k)$. In *35th Annual ACM Symposium on Theory of Computing*, pages 223–231, San Diego, CA, 2003.

[7] N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method.* John Wiley & Sons, 1992.

[8] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability of random k-CNF formulas. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 561–571, Dallas, TX, May 1998.

[9] E. Ben-Sasson and A. Wigderson. Short proofs are narrow – resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.

[10] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Procceedings, 5th International Conference, TACAS'99*, pages 193–207, Berlin, Germany, 1999. Springer-Verlag.

[11] B. Bollobás. *Random graphs*. Academic Press, London-New York, 1985.

[12] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson. The scaling window of the 2-SAT transition. *Random Structures and Algorithms*, 18(3):201–256, 2001.

[13] M.T. Chao and J. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics. *Information Science*, 51:289–314, 1990.

[14] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627, Pittsburgh, PA, October 1992. IEEE.

[15] V. Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[16] C. Coarfa, D. D. Demopoulos, A. San Miguel Aguirre, D. Subramanian, and M. Y. Vardi. Random 3-SAT: The plot thickens. In *Proceedings 6th International Conference on Principles and Practice of Constraint Programming*, Singapore, September 2000.

[17] S. Cocco and R. Monasson. Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-Satisfiability problem. *Phys. Rev. Lett.*, 86(8):1654–1657, 2001.

[18] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[19] M. Davis and H. Putnam. A computing procedure for quantification theory. *Communications of the ACM*, 7:201–215, 1960.

[20] W. Fernandez de la Vega. On random 2-SAT. Manuscript, 1992.

[21] E. Friedgut. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, 12:1017–1054, 1999.

[22] A. Frieze and S. Suen. Analysis of two simple heuristics on a random instance of k- SAT. *Journal of Algorithms*, 20(2):312–355, 1996.

[23] A. Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences*, 53:469–486, 1996.

[24] C. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proceedings Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 431–437, Madison, WI, 1998.

[25] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. Automat. Reason.*, 24(1-2):67–100, 2000.

[26] S. Janson, Y. C. Stamatiou, and M. Vamvakari. Bounding the unsatisfiability threshold of random 3-SAT. *Random Structures Algorithms*, 17(2):103–116, 2000.

[27] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Principles of Knowledge Representation and Reasoning: Proceedings of the address=Cambridge, MA, Fifth International Conference (KR'96)*, pages 374–384, 1996.

[28] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th AAAI*, pages 1194–2001, Portland, OR, 1996.

[29] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random formulas. *Science*, 264:1297–1301, May 1994.

[30] C. J.H. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics, Proceedings of the 12th British Combinatorial Conference*, pages 148–188. Cambridge University Press, 1989.

[31] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Phase transition and search cost in the $(2+p)$-SAT problem. In *4th Workshop on Physics and Computation*, Boston, MA, 1996.

[32] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. $2+p$-SAT: relation of typical-case complexity to the nature of the phase transition. *Random Structures Algorithms*, 15(3-4):414–435, 1999.

[33] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic phase transitions. *Nature*, 400:133–137, 1999.

[34] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the 13th IJCAI*, pages 290–295, Chambéry, France, 1993.

[35] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, San Jose, CA, 1992.

[36] B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.