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

December 6, 2001

Abstract

We give the first example of a sharp threshold in proof complexity. More precisely, we show that for any sufficiently small $\epsilon > 0$ and $\Delta > 2.28$, random formulas consisting of $(1-\epsilon)n$ 2-clauses and Δn 3-clauses, which are known to be unsatisfiable almost certainly, almost certainly require resolution and Davis-Putnam proofs of unsatisfiability of exponential size, whereas it is easily seen that random formulas with $(1+\epsilon)n$ 2-clauses (and Δn 3-clauses) have linear size proofs of unsatisfiability almost certainly.

A consequence of our result also yields the first proof that typical random 3-CNF formulas at ratios below the generally accepted range of the satisfiability threshold (and thus expected to be satisfiable almost certainly) cause natural Davis-Putnam algorithms to take exponential time to find satisfying assignments.

1 Introduction

The satisfiability problem has received a great deal of study as the canonical NP-complete problem. In the last several years the very universality and flexibility of the satisfiability problem that made it a natural starting point for NP-completeness has also made it the basis for significant progress in the solution of a variety of practical problems including problems in constraint satisfaction [34], planning [27, 26], and symbolic model checking [11]. 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 [34, 33]. In a sense, these satisfiability algorithms have become the hammer and there is now a small industry turning computational problems into nails.

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

A folklore property of these satisfiability algorithms is that they work extremely well at handling typical satisfiable formulas, although outlying satisfiable formulas can cause very bad behavior for any specific algorithm. (One highly successful strategy is to use randomized DPLL algorithms and re-start them with different random bits if they begin to take too long [24, 23].)

It has previously been shown that DPLL algorithms are exponentially inefficient for proving unsatisfiability for typical unsatisfiable formulas [16, 10, 8, 9]. We give a method to prove the first lower bounds on the running times of DPLL algorithms for typical *satisfiable* CNF formulas. We show that certain natural DPLL algorithms when applied to random 3-CNF formulas for which there is strong empirical evidence of satisfiability (density below 3.98) almost certainly generate subproblems consisting of unsatisfiable mixed formulas of 2- and 3-clauses for which we can prove that exponential-size proofs of unsatisfiability are required. The key to this result is the proof complexity lower bound for these mixed formulas.

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* $\Delta, \epsilon > 0$. Let $res(F)$ and $DLL(F)$ be the sizes of the minimal resolution and Davis-Putnam/DLL proofs of the unsatisfiability of F. 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 $\liminf_{n\to\infty} \Pr[\mathcal{E}_n] > 0$.

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 $DLL(F) = 2^{\Omega(n)}$.*

This bound is interesting in its own right since it yields the first example of a sharp threshold for the proof complexity of unsatisfiable formulas. More precisely, for $\Delta > 2.28$ formulas from $\mathcal{F}^n_{\epsilon,\Delta}$ are unsatisfiable almost certainly [4] and require exponential-size resolution and DPLL proofs of this fact, but if we slightly increase the number of 2-clauses from $(1-\epsilon)n$ to $(1+\epsilon)n$, the 2-CNF subformula alone becomes unsatisfiable almost certainly and this unsatisfiability problem becomes trivial for resolution or DPLL proofs.

Mixed formulas consisting of 2- and 3-clauses arise for two natural reasons. 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. Also, as DPLL algorithms run, they recursively solve satisfiability on restricted versions of their input CNF formula which are mixtures of clauses of length at least 2. Randomly-chosen k-CNF formulas are an important test case for satisfiability algorithms. When given randomly-chosen 3-CNF formulas as input, many DPLL algorithms produce restricted mixed formulas that are distributed precisely in the form that we analyze.

A fundamental conjecture about random k-CNF formulas says that for each $k \geq 2$, there is a constant α_k, the *satisfiability threshold*, such that a random k-CNF formula of clause-variable ratio Δ is almost certainly satisfiable for $\Delta < \alpha_k$ and almost certainly unsatisfiable if $\Delta > \alpha_k$. It is known that 1 is the threshold for random 2-SAT [15, 22] and empirically, it appears that there is a threshold for random 3-SAT around 4.2 [35, 28]. The best proven bounds show that random 3-SAT formulas are almost certainly satisfiable for $\Delta < 3.26$ [5] and almost certainly unsatisfiable for $\Delta > 4.598$ [25] and it is known that the 'phase' transition from satisfiable to unsatisfiable formulas is asymptotically sharp [20]. The proven lower bounds on the satisfiability thresholds have all been derived by analyzing specific DPLL-like algorithms without backtracking ('card-type/myopic algorithms' in the terminology of [1, 5]). The formulas we consider are at densities bounded well below the empirical 4.2 threshold and thus should be almost certainly satisfiable.

An extension of random 3-SAT problems to mixed formulas has led to the study of the so-called $(2+p)$-SAT problem. Here one looks at randomly-generated formulas on n variables with a mix of clauses of length 2 and 3 where the fraction of clauses of length 3 is p. Using a mixture of empirical results and heuristic, non-rigorous techniques of statistical physics, Kirkpatrick et. al. [30, 31, 32] gave evidence that one could add up to roughly $2n/3$ random 3-clauses (corresponding to p around 0.4) and have no impact on typical satisfiability problems in that the point at which the threshold is reached is dependent solely on the number of 2-clauses. Furthermore they suggested that the phase transition for the $(2+p)$-SAT problem itself changed character from a so-called first-order transition (discontinuous "order parameter") representative of 2-SAT to a second-order transition (continuous "order parameter") believed to be representative of 3-SAT at this point and suggested that this second-order phase transition somehow characterizes NP-hard problems. (This latter conjecture has been put in further perspective by the results in [3].)

In [4], Achlioptas et. al. proved a number of rigorous results about $(2+p)$-SAT. They showed that adding $2n/3$ randomly chosen 3-clauses to a random formula with $(1-\epsilon)n$ 2-clauses yields a formula that is almost surely satisfiable (and this assignment can be found by a simple card-type algorithm), while adding $2.28n$ random 3-clauses yields an almost certainly unsatisfiable formula. Moreover, they proved that the transition from satisfiability to unsatisfiability is sharp and empirical evidence suggests that the $2n/3$ bound is much closer to the truth [32]. Furthermore $2n/3$ is the provable limit for showing satisfiability by card-type algorithms [1] and, in [2], it was conjectured that it is tight for all algorithms.

The card-type algorithms analyzed for 3-SAT have the property that they make irrevocable choices to the partial assignment and at each point in their execution the restricted formula that remains is an unbiased random formula characterized by a pair of integers (c_2n, c_3n) describing the number of 2- and 3-clauses respectively that remain. The algorithms will succeed almost certainly if they exhaust the 3-clauses without c_2 ever reaching 1. In fact, if they reach $((1-\epsilon)n, 2n/3)$ without c_2 ever reaching 1 they will succeed with constant probability [4].

One can extend the card-type algorithms, such as UC (unit clauses first) and GUC [14], into full backtracking DPLL algorithms in a variety of different ways so that the execution of the original algorithm is the first path explored in the tree of recursive calls. If the original card-type algorithm reaches $((1-\epsilon)n, c_3n)$ where c_3 is at least 2.28, the resulting formula is almost certainly unsatisfiable, and by our argument it almost certainly requires an exponential-size DPLL proof. Thus, once this node has been explored in the backtracking search, the search cannot leave the sub-tree for an exponentially long time.

Our results show that this happens with constant probability for UC started with $3.81n$ 3-clauses and for GUC started with $3.98n$ 3-clauses (it would happen almost certainly except that the card-type algorithm might reach a dead end before then) and thus with constant probability *any* backtracking extension of UC and GUC applied to random 3-CNF formulas above these ratios will run for exponential time. Furthermore, by generalizing a limited backtracking heuristic used by Frieze and Suen [21] into a full backtracking heuristic we create natural DPLL algorithms, UC-FS and GUC-FS, extending UC and GUC respectively, for which we can show that above exponential lower bounds hold almost certainly rather than just with constant probability.

These results shed light on a widely-cited and repeated observation of Selman, Mitchell, and Levesque [35], based on experiments with ORDERED-DLL, a backtracking version of an algorithm probabilistically equivalent to UC, on small problems, 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 it is known that at any constant ratio above the threshold any DPLL or resolution algorithm almost certainly requires exponential size proofs of unsatisfiability [16]. By now the rate of decline in proof complexity as the ratio is increased has been analyzed as well [8].

Our new results show that the lower end of this characterization is also somewhat misleading; in fact, our results show 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 [17].) More precisely we show:

Theorem 1.2. *Any backtracking extension of* UC *or* ORDERED-DLL *on a random n variable 3-CNF formula with rn clauses for constant $r \geq 3.81$ requires time $2^{\Omega(n)}$ with constant probability and any backtracking extension of* GUC *on a random n variable 3-CNF formula with rn clauses for constant $r \geq 3.98$ requires time $2^{\Omega(n)}$ with constant probability.*

We also show that for some natural and effective backtracking extensions of these algorithms the bounds hold w.h.p.

As we discuss in section 8, 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 upper bound for unsatisfiability in random $(2+p)$-SAT. In fact if it turns out that, as was conjectured in [2], for every $\delta > 0$ there exists $\epsilon > 0$ such that a random formula with $(2/3+\delta)n$ 3-clauses and $(1-\epsilon)n$ 2-clause is unsatisfiable w.h.p. then our results would imply a perfectly sharp characterization for these DPLL algorithms.

Our lower bound on the proof complexity of mixed formulas is similar in general spirit to other lower bounds for resolution algorithms 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 with $(1-\epsilon)n$ clauses, w.h.p. have properties that guarantee that almost all extensions by linear-sized 3-CNF formulas yield exponential size resolution (and DPLL) proofs. The latter argument relies on specialized sharp moment bounds as well particular properties of clans.

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 refutation of an unsatisfiable CNF formula F begins with the clauses of F and by a sequence of inferences using the resolution rule derives the empty clause. 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 a tree. Let $DLL(F)$ be the size of the smallest such refutation, i.e. the size of the smallest DPLL tree associated with F.

Definition 2.1. *Let F be an an arbitrary CNF formula.*

- *Given a set F of clauses, a literal x is* pure *in F if and only if x appears in F but $\neg x$ does not appear in F.*
- *$V(F)$ denotes the set of variables of F.*
- *$|F|$ denotes the number of clauses in F.*
- *The* degree *of a variable v, $\deg(v)$, is the number of clauses of F containing one of $v, \bar{v}$ (analogously for literals). The* average degree *of F is $\left(\sum_v \deg(v)\right)/|V(F)|$.*
- *With any 2-CNF formula F on variables $\{x_1, \ldots, x_n\}$ we associate the following directed graph $\vec{D}(F)$:*
  - *The vertex set is $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$.*

– *The edge set is*

$$\{(\bar{x} \to y), (\bar{y} \to x) : (x \vee y) \text{ is a clause in } F\} \ .$$

- *We say that a literal* ℓ *is* near-pure *in* F *if* $\deg(\bar{\ell}) = 1$.
- *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.)*
- F *is* μ-pure *if it has at most* $\mu|V(F)|$ *pure items, i.e. pure literals and cycles.*

The *boundary* of F, $b(F)$, is the set of pure items in F. Note that this generalizes the definition of boundary from [10] where the boundary was defined to be those literals whose variables appeared in precisely one clause of F. Let ν be any sub-additive measure on sets of clauses, i.e. satisfying $\nu(A \cup B) \leq \nu(A) + \nu(B)$. Define the *satisfiability* ν*-threshold for* F, $s_\nu(F)$, to be $\min_{G \subseteq F;\, G \text{ unsatisfiable}} \nu(G)$.

Given a set of clauses F we say that $F \Rightarrow_{Res} C$ if and only if there is a resolution derivation of C from F that uses all clauses in F. Define the ν*-sub-critical expansion of* F, $e_\nu(F) = \max_{s \leq s_\nu(F)} \min\{|b(G)| : s/2 \leq \nu(G) < s\}$. The following are extensions of [10].

Lemma 2.2. *If* $F \Rightarrow_{Res} C$ *then* C *contains at least* $|b(F)|$ *literals.*

Proof. This is trivial for pure literals since, once they appear in the derivation, they cannot be removed. For pure cycles, observe that once a literal from the pure cycle appears in a clause in the derivation, all clauses derived from it will also contain a literal from that cycle. The argument is by induction on the proof in topological order: Suppose that a literal x from the cycle appears in clause A and does not appear in clause D where D is derived from A and some clause B by the resolution rule. Therefore x must appear negatively in B and the resolution rule must use x. The only way that $\bar{x}$ can be in B is if B is derived using the one negative clause $\bar{x} \vee y$ that appears in the pure cycle containing x. Therefore by the induction hypothesis B contains some literal in this pure cycle and this literal will appear in D. Since each literal is in at most one pure cycle, the lemma follows. □

Using Lemma 2.2 the following is immediate from [10].

Proposition 2.3. *Let* ν *be any sub-additive measure on sets of clauses. If all clauses in a formula* F *have size at most* k, *then* $res(F) = 2^{\Omega([e_\nu(F)-k]^2/n)}$ *and* $DLL(F) \geq 2^{e_\nu(F)-k}$.

For each n, let us fix a canonical set of n variables and for each fixed $k \geq 0$, let $C_k(n)$ denote the set containing all $2^k\binom{n}{k}$ non-trivial k-clauses on our canonical set of 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 $C_3(n)$.

Let $\mathcal{F}^n_{\epsilon,\Delta}$ be the distribution where $m_2 = (1-\epsilon)n$ and $m_3 = \Delta n$, for some arbitrary *constants* $\Delta, \epsilon > 0$.

Lemma 2.4. *For every* $\Delta, \epsilon > 0$ *there exist* $\zeta = \zeta(\Delta, \epsilon) > 0$ *and* $\mu = \mu(\Delta, \epsilon) > 0$ *such that*

1. *W.h.p.* F *contains no unsatisfiable subformula on* $v \leq \zeta n$ *variables.*
2. *W.h.p.* F *contains no* μ*-pure subformula on* $\frac{1}{2}\zeta n \leq v \leq \zeta n$ *variables.*

Corollary 2.5. *For* $F \sim \mathcal{F}^n_{\epsilon,\Delta}$, *w.h.p.,* $res(F) = 2^{\Omega(n)}$ *and* $DLL(F) = 2^{\Omega(n)}$.

The overall proof strategy will be to analyze almost certain properties of the 2-clauses of F and show that such subformulas must have large boundaries and then to show that the addition of 3-clauses does not significantly reduce the size of the boundary. (Since the 2-clause subformula of F alone is almost certainly satisfiable, the fact that the two clauses alone have a large boundary does not contradict the fact that 2-SAT is easy for resolution.) This argument is subtle because the 2-clauses of F are so close to being unsatisfiable themselves and because we need to handle all possible subformulas among the 2-clauses. The latter requirement necessitates the introduction of a somewhat unusual graph-theoretic concept in the digraph associated with the 2-clauses of F.

3 Bounding Purity

By Lemma 2.2, any minimal unsatisfiable formula must have no pure items. Therefore, it will be convenient to restate Lemma 2.4 in terms of the following definition.

Definition 3.1. A subformula H of F on v variables is (μ, ζ)-*pure* if: i) $v < \frac{1}{2}\zeta n$ and H is 0-pure, or, ii) $\frac{1}{2}\zeta n \leq v \leq \zeta n$ and H is μ-pure.

Hence, Lemma 2.4 can be restated as

Lemma 3.2. *For every $\Delta, \epsilon > 0$, there exists $\zeta = \zeta(\Delta, \epsilon) > 0$, $\mu = \mu(\Delta, \epsilon)$ such that w.h.p. F contains no (μ, ζ)-pure subformula.*

To prove Lemma 2.4 we will need need to consider two possibilities for the fraction of 3-clauses appearing in a potential (μ, ζ)-pure subformula. In particular, it will be relatively easy to prove that w.h.p. every subformula rich in 3-clauses is not (μ, ζ)-pure (Lemma 3.5). Proving the same assertion for subformulas where the fraction of 3-clauses is arbitrarily small (Lemma 3.4) will be harder, comprising Sections 4 and 5. More precisely, let us say that

Definition 3.3. *A formula F is δ-*rich *if it has at least $\delta|V(F)|$ 3-clauses, and δ-*poor *otherwise.*

Lemma 3.2 will follow readily from the following two lemmas.

Lemma 3.4. *For every $\Delta, \epsilon > 0$ and all $\zeta \leq \zeta_1(\Delta, \epsilon)$, there exist $\delta_1 = \delta_1(\Delta, \epsilon, \zeta) > 0$ and $\mu_1 = \mu_1(\Delta, \epsilon, \zeta)$ such that w.h.p. $\mathcal{F}^n_{\epsilon,\Delta}$ contains no δ_1-poor, (μ_1, ζ)-pure subformula.*

Lemma 3.5. *For every $\Delta, \epsilon, \delta > 0$, and all $\zeta \leq \zeta_2(\Delta, \epsilon, \delta)$, there exists $\mu_2 = \mu_2(\Delta, \epsilon, \zeta)$ such that w.h.p. $\mathcal{F}^n_{\epsilon,\Delta}$ contains no δ-rich, (μ_2, ζ)-pure subformula.*

Proof of Lemma 3.2. Given $\Delta, \epsilon > 0$ we start by applying Lemma 3.4 with $\zeta = \zeta_1(\Delta, \epsilon)$. This determines a value $\delta^* = \delta_1(\Delta, \epsilon, \zeta_1) > 0$. We then apply Lemma 3.5 with $\delta = \delta^*$ and $\zeta = \zeta_2(\Delta, \epsilon, \delta^*) > 0$.

As a result, for $\zeta = \min\{\zeta_1(\Delta, \epsilon), \zeta_2(\Delta, \epsilon, \delta^*)\} > 0$ and $\mu = \max\{\mu_1(\Delta, \epsilon, \zeta), \mu_2(\Delta, \epsilon, \zeta)\} > 0$, we see that w.h.p. there are no δ^*-poor, (μ, ζ)-pure subformulas and no δ^*-rich, (μ, ζ)-pure subformulas. □

To prove Lemma 3.5 we will need the following

Lemma 3.6. *For every $\Delta, \epsilon, \beta > 0$ there is $\zeta = \zeta(\Delta, \epsilon, \beta) > 0$ such that w.h.p. no subformula of $F \sim \mathcal{F}^n_{\epsilon,\Delta}$ on $v \leq \zeta n$ variables has average degree at least $2 - \beta/2$ and at least βv 3-clauses.*

Proof. For a given $\beta > 0$ let α be such that $2\alpha + 3\beta = 2 - \beta/2$, implying $\alpha + 2\beta - 1 = \beta/4$. We will bound the expected number of subformulas on v variables having αv 2-clauses and βv 3-clauses. Using this bound we will show that there exists $\zeta = \zeta(\Delta, \epsilon, \beta) > 0$ such that the total expected number of such subformulas for $1 \leq v \leq \zeta n$ variables is $o(1)$.

For a fixed v, we can bound the expected number, $Q_\beta(v)$, of such subformulas by

$$\begin{aligned}
& \binom{n}{v}\binom{(1-\epsilon)n}{\alpha v}\binom{\Delta n}{\beta v}\left(\frac{v}{n}\right)^{2\alpha v+3\beta v} && (1)\\
\leq\; & \left(\frac{en}{v}\right)^{v}\left(\frac{e(1-\epsilon)n}{\alpha v}\right)^{\alpha v}\left(\frac{e\Delta n}{\beta v}\right)^{\beta v}\left(\frac{v}{n}\right)^{2\alpha v+3\beta v} \\
=\; & \left(\frac{(1-\epsilon)^{\alpha}e^{1+\alpha+\beta}\Delta^{\beta}}{\alpha^{\alpha}\beta^{\beta}}\times\left(\frac{v}{n}\right)^{\alpha+2\beta-1}\right)^{v} \\
=\; & \left(K\times\left(\frac{v}{n}\right)^{\beta/4}\right)^{v}\quad, && (2)
\end{aligned}$$

for some (constant) $K = K(\beta,\Delta,\epsilon)$. It is easy to see that for $\zeta = \zeta(\beta, K)$ sufficiently small and all n sufficiently large, if $v \leq \zeta n$ the right hand side of (2) is decreasing with v. Therefore,

$$\sum_{1\leq v\leq \zeta n} Q_\beta(v) \leq \sum_{v=1}^{8/\beta} K n^{-\beta/4} + \sum_{v>8/\beta} K \times \frac{(8/\beta)^{\beta/4}}{n^2} = o(1).$$

□

Proof of Lemma 3.5. We first observe that any μ-pure subformula H on v variables must contain at most μv pure literals and therefore have average degree at least $2(1-\mu)$. Moreover, if H is δ-rich, it contains at least δv 3-clauses. We will take $\mu_2 = \delta/4$. Thus, any δ-rich, μ_2-pure subformula must have average degree at least $2-\delta/2$ and contain at least δv 3-clauses. But from Lemma 3.6, there exists $\zeta_2 = \zeta(\Delta,\epsilon,\delta) > 0$ such that w.h.p. F contains no such subformula on $v \leq \zeta_2 n$ variables. □

To prove Lemma 3.4 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 6, we prove that F_2 satisfies these properties w.h.p.

4 The δ-poor case

As mentioned earlier, we will prove the assertion of Lemma 3.4 for formulas formed by starting with a 2-CNF formula satisfying certain properties and adding m_3 random 3-clauses. To describe these properties we need to introduce the following definitions.

Definition 4.1. Let F be an arbitrary 2-CNF formula. For literals x, y appearing in F let us write $x \rightsquigarrow_F y$ iff $x = y$ or there exists a directed path in $\vec{D}(F)$ from x to y.

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

- $T_i(F) = |\{x : |\mathrm{Clan}^*_F(x)| \geq i\}|$.

Lemma 3.4 will follow readily from the following two lemmas.

Lemma 4.2. *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.*

For all $\zeta \leq \zeta_0(\Delta, \rho)$, there exist $\delta = \delta_0(\Delta, \rho, \zeta) > 0$, and $\mu = \mu_0(\Delta, \rho, \zeta) > 0$ such that, w.h.p. F^* contains no δ-poor, (μ, ζ)-pure subformula.

Lemma 4.3. *Fix $\epsilon > 0$ and let F_2 be a random 2-SAT 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 4.2 (where $F^*_2 = F_2$).*

Proof of Lemma 3.4. Given $\Delta, \epsilon > 0$ we start by applying Lemma 4.3 to get $\rho = \rho(\epsilon)$ and then apply lemma 4.2 with that ρ. Thus, we get that for all $\zeta \leq \zeta_0(\Delta, \rho)$, there exist $\delta = \delta_0(\Delta, \rho, \zeta) > 0$, and $\mu = \mu_0(\Delta, \rho, \zeta) > 0$ such that, w.h.p. $\mathcal{F}^n_{\epsilon,\Delta}$ contains no δ-poor, (μ, ζ)-pure subformula. To conclude the proof we set $\zeta_1(\Delta, \epsilon) = \zeta_0(\Delta, \rho(\epsilon))$ and similarly for δ_1, μ_1. ☐

5 Proof of Lemma 4.2

Let $\rho, \Delta > 0$, F^*_2 be fixed and choose F^* as in the statement of the lemma. For any $\delta, \zeta, \mu > 0$, consider any (candidate) δ-poor, (μ, ζ)-pure 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, the pure items of H_2, that must be (mostly) covered by the 3-clauses of H in order to for H to have very few pure items. More generally, H may itself create new loose ends but then must cover most of them up again. We show we can cover the literals appearing in H by the clans of the loose ends. Further since the clan sizes are typically small we get a large number of loose ends, the set $P(H)$ below, each of which must be covered by a different 3-clause literal.

In the case that the set of variables in H is large, we will set parameters so that the contribution of all large clans is so small that the δ-poverty of H simply doesn't allow enough 3-clause literals to cover enough loose ends. In the case that the set of variables in H is not so large, we get to use the fact that *every* loose end must be covered but we need a probability argument. Intuitively, it seems unlikely that the random 3-clauses of the formula will exactly cover all loose ends, both those from H_2 and the ones that the 3-clauses themselves generate. One subtlety of the probability analysis is that the new loose ends generated depend on the 3-clauses themselves. In order to make this analysis work, we need stronger bounds on the set of variables in H; we show that, except possibly for the case of the very small number of clans that are not tree-like, we get two loose ends per clan instead of just one and this is enough to make the probability calculation work. Finally, we use a sharp specialized moment bound to show that the rare large clans do not skew the probabilities too much and derive the claimed result.

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).

Lemma 5.1. *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. □

Lemma 5.2. *For any subformula H of F^*, for each literal x on the variables in $V(H)$, there is some literal $y \in P(H)$ such that $x \in \mathrm{Clan}_{F_2^*}(y)$.*

Proof. If the literal x is on $V(H) - V(H_2)$ then $x \in P$ so we can take $y = x$. If x is a literal on $V(H_2)$ and $\bar{x}$ appears in H_2, then in the digraph $\vec{D}(H_2)$ walk forward from x as far as possible, since all clans contain at most one cycle, either we reach a sink of $\vec{D}(H_2)$, in which case the label of that sink is a pure literal y in P which satisfies $x \in \mathrm{In}_{H_2}(y) \subseteq \mathrm{In}_{F_2}(y)$, or we reach a pure cycle of H_2 in which case the smallest numbered literal y in this cycle satisfies $x \in \mathrm{In}_{H_2}(y) \subseteq \mathrm{In}_{F_2^*}(y)$. Clearly in either case, both x and $\bar{x}$ appear in $\mathrm{Clan}_{F_2^*}(y)$. □

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 $\mathrm{Clan}(T) = \bigcup_{x \in T} \mathrm{Clan}(x)$.

Lemma 5.3. *For any $\zeta \leq 1$, there are $\mu_0 = \mu_0(\zeta, \rho)$, $n_0 = n_0(\zeta)$ and $\delta_0 = \delta_0(\zeta, \rho) > 0$ such that if $n \geq n_0$, $\mu \leq \mu_0$, $\delta \leq \delta_0$, and $\zeta n/2 \leq v = |V(H)|$ then H is not a μ-pure subformula of F^*.*

Proof. Suppose that H is a μ-pure subformula of F^* and $\zeta n/2 \leq V(H)$. Since H has at most δv 3-clauses and at most μv items remain pure in H, by Lemma 5.1, $|P| \leq (3\delta + \mu)v$. Choose $I = I(\zeta, \rho) \geq 2$ such that $\sum_{i>I} 2i(1-\rho)^i < \zeta/3$, let $n_0 = n_0(\zeta)$ be such that $\log^4 n \leq \zeta n/6$ for all $n \geq n_0$, and let $\mu_0 = \mu_0(\zeta, \rho) = 1/(4I)$ and $\delta_0 = \delta_0(\zeta, \rho) = 1/(4I)$. Therefore $|P| \leq v/I$. By Lemma 5.2, $|\mathrm{Clan}(P)| \geq 2v$. Let $L_I = \{x : |\mathrm{Clan}(x)| > I\}$. Using the bound on the sizes of $T_i(F_2^*)$ and the fact that there are at most $\log^4 n$ elements in clans of literals ℓ with non-tree-like $\mathrm{In}_{F_2^*}(\ell)$, $|\mathrm{Clan}(P \cap L_I)| \leq |\mathrm{Clan}(L_I)| \leq \sum_{i>I} 2i(1-\rho)^i n + \log^4 n < \zeta n/3 + \zeta n/6 = \zeta n/2 \leq v$. Therefore $|\mathrm{Clan}(P - L_I)| \geq |\mathrm{Clan}(P) - \mathrm{Clan}(P \cap L_I)| > v$. However, $|\mathrm{Clan}(P - L_I)| \leq |P| \cdot I \leq v$ which is a contradiction. □

It remains to show that the probability there is a 0-pure subformula H of F^* with $|V(H)| < \zeta n/2$ is $o(1)$ in n. Suppose that H is 0-pure and $v = |V(H)| < \zeta n/2$. 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 their complements.

Lemma 5.4. *For any subformula H of F^* that is 0-pure 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 $V(H) - V(H_2)$. Further for every literal x such that $\bar{x}$ appears in H_2, as shown in the proof of Lemma 5.2, $x \in \mathrm{In}_{H_2}(y)$ for some literal y in P. 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. □

We will bound the probability that F^* has a 0-pure subformula H by bounding $\Pr[P^*(H) = T]$ for each set of literals T and summing over all choices of T.

Lemma 5.5. *Let T be a set of literals, $t = |T|$, and suppose that H is a 0-pure subformula of F^* with $P^*(H) = T$ and $t_c = t_c(H)$. Then H and thus formula F^* must contain at least $2t/3 - t_c/3$ 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 5.4, since $P^*(H) = T$, $|P(H)| \geq 2|T| - t_c = 2t - t_c$. By Lemma 5.1, since H is 0-pure the 3-clauses of H contain at least $|P(H)| = 2t - t_c$ literals and therefore H contains at least $(2t - t_c)/3$ 3-clauses of F^*. By Lemma 5.4, all literals in these clauses are in $cover(P^*) = cover(T)$. In case $t \geq 10t_c$ then this is at least $(2/3 - 1/30)t > 3t/5$. □

Lemma 5.6. *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 a random F^* has a 0-pure subformula H with $P^*(H) = T$ is at most*

1. *$R(T) = (K/(tn^2))^{3t/5} |\mathrm{Clan}(T)|^{9t/5}$ if $t_c(H) \leq t/10$, and at most*
2. *$R'(T, t_c) = (K/n^2)^{2t/3 - t_c/3} |\mathrm{Clan}(T)|^{2t - t_c}$ 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

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

for some constant $K' = K'(\Delta)$. Let $K = 5K'/3$. By Lemma 5.5, if $t_c(H) \leq t/10$ then we get the probability upper bound in part 1 by setting $s = \lceil 3t/5 \rceil$ and observing that the upper bound in (3) is at most $[K/(tn^2)]^s|\mathrm{Clan}(T)|^{3s}$ and that this is a decreasing function of s (and thus at most $R(T)$) for $R(T) \leq 1$. Also, by Lemma 5.5, we get the probability upper bound in part 2 by setting $s = \lceil 2t/3 - t_c/3 \rceil$ and observing that (3) 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 $R'(T, t_c) \leq 1$. □

Lemma 5.7. *Fix $\Delta, \rho > 0$. The probability that there exists some set T of literals and a 0-pure subformula H of F^* with $P^*(H) = T$, and $|T| \leq \log^4 n$ is $o(1)$ in n.*

Proof. Suppose that $t \leq \log^4 n$ and suppose that H is a 0-pure subformula of F^* with $t_c(H) = t_c$ and $t = |P^*(H)|$. By assumption about F_2^*, $t_c \leq \log n$ and $|\mathrm{Clan}(P^*(H))| \leq t\log^3 n \leq \log^7 n$. For each $t \leq \log^4 n$ and each t_c, $t_c \leq t$, 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 5.6.2, the probability that there is some 0-pure subformula H of F^* with $|P^*(H)| = t$ and $|t_c(H)| = t_c$ is at most

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

which is bounded above by $(K'')^t(\log n)^{14t}n^{-t/3}$ for some constant $K'' = K''(\Delta, \rho) > 0$. The probability that an H satisfying the conditions of the lemma with $t(H) \leq \log^4 n$ exists is then at most

$$\sum_{t_c=1}^{\log n}\sum_{t=t_c}^{\log^4 n}(K''n^{-1/3}\log^{14} n)^t \leq K''n^{-1/3}\log^{18} n$$

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 $|T| = t \geq \log^4 n$ in terms of a probability calculation involving a uniformly chosen random set of literals, T, of size t. Observe that for any such T, $\mathrm{Clan}(T) \leq 2\mathrm{Clan}^*(T)$ since there are are most $\log^4 n$ literals in clans of literals x with $\mathrm{In}_{F_2^*}(x)$ not tree-like.

Lemma 5.8. *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, $\mathbf{E}_T(|\mathrm{Clan}^*(T)|) \leq Bt$.*

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

Lemma 5.9. *For every $\rho > 0$ there exists $\alpha = \alpha(\rho) > 0$ such that for all $r \geq 0$ we have for T a set of literals with $|T| = t$ chosen uniformly at random,*

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

Proof. See appendix. □

Lemma 5.10. *Fix $\rho > 0$. There is $K_1 = K_1(\rho, C)$ such that for any $t > 0$ and for a set of literals T with $|T| = t \geq \log^4 n$ chosen uniformly at random, $E_T(|\mathrm{Clan}(T)|^{9t/5}) \leq (K_1 t)^{9t/5}$.*

Proof. Fix an integer $t \geq \log^4 n$ and consider choosing T uniformly at random with $|T| = t$. Since $t \geq \log^4 n$, 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 $\nu(T) = \mathbf{E}_T(|\mathrm{Clan}^*(T)|) \leq Bt$ where $B = B(\rho)$ is the constant from Lemma 5.8 and use our tail bounds within each segment. Therefore by Lemma 5.9,

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

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

Lemma 5.11. *Fix $\Delta, \rho > 0$. There is $\zeta_0 = \zeta_0(\Delta, \rho) > 0$ such that the probability that a random F^* has a 0-pure subformula H with $v = |V(H)| \leq \zeta_0 n/2$ is $o(1)$ in n.*

Proof. By Lemmas 5.6.1 and 5.7, the probability of this event is at most $\sum_{T, |T| \geq \log^4 n} R(T)$ plus a term that is $o(1)$ in n. By Lemma 5.10,

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

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

Now if we let $\zeta_0 = 1/(32K_2)$ then for $v \leq \zeta_0 n/2$, $t \leq v \leq n/(32K_2)$. Therefore by Lemma 5.6, 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. □

Lemma 4.2 follows immediately from Lemmas 5.3 and 5.11.

6 Properties of subcritical random 2-CNF formulae

We will now prove that subcritical random 2-CNF formulas satisfy the properties in Lemma 4.2 w.h.p.

Lemma 6.1. *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, $|\mathrm{Clan}_{F_2}(\ell)| \leq \log^3 n$.*
4. *For all $i \geq 3$, $|\{\ell : |\mathrm{Clan}^*_{F_2}(\ell)| \geq i\}| \leq 2(1-\rho)^i\, n$.*

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 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 $y \neq x$, 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 $y \neq x$, 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 [12]. 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 6.2 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 6.2.

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. ☐

Claim 6.3. *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. Since each clause appears in the formula with probability p it is easy to show (see e.g. [13]) that the tree corresponding to $S_0(\ell)$ and each of the trees corresponding to the different $S_j(\ell)$ is dominated by the connected component of a fixed vertex in a random graph $G(2n, p = (1-\epsilon)/(2n))$.

Let $X \stackrel{D}{=} |C(v)|$, where v is a fixed vertex in $G(2n, p = (1-\epsilon)/(2n))$ and $|C(v)|$ denotes the number of vertices in its connected component $C(v)$. Now, let

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

where the W_i are i.i.d. random variables with $W_i \stackrel{D}{=} |C(v)|$. 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

$$\begin{aligned} \mathbf{E}(\exp(hW)) &= \sum_{k\geq 1} \Pr[X=k]\, \mathbf{E}\left(\prod_{i=1}^{k} \exp(hW_i)\right) && (4) \\ &= \sum_{k\geq 1} \Pr[X=k] \prod_{i=1}^{k} \mathbf{E}\left(\exp(hW_i)\right) && (5) \\ &= \sum_{k\geq 1} \Pr[X=k]\, \left(\mathbf{E}\left(\exp(hW_i)\right)\right)^k \ , && (6) \end{aligned}$$

where passing from (4) to (5) uses that the random variables W_i are independent while passing from (5) to (6) uses that they are identically distributed.

Let $\Pr[W_i = k] = \Pr[X = k] \equiv p_k$. Let $c = 1 - \epsilon$. To bound (6) we will use the following well-known facts (see, for example, p.156 in [6]). Namely, that asymptotically in n,

$$p_k = (1/c)\left(ce^{-c}\right)^k k^{k-1}/k! \tag{7}$$

and that for all $c < 1$,

$$\sum_{k\geq 1} p_k = 1 \ . \tag{8}$$

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 $q > 1$,

$$\Pr[W \geq q] < (1-\sigma')^q \ . \tag{9}$$

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 6.4. *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 6.4 w.h.p. for every ℓ we are in one of the two cases of Claim 6.2. In both cases $\mathrm{In}_{F_2}(\ell)$ is simple.
Proof of 2: By Claim 6.4, w.h.p. there are at most $\log n$ literals not falling in the first case of Lemma 6.2. 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 (9) 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 6.2, $\mathrm{Clan}_{F_2}(\ell) = S(\ell)$. Therefore for $i \geq 3$, by linearity of expectation and Claim 6.3,

$$\mathbf{E}(T_i(F_2)) < 2n \times (1-\sigma)^i. \tag{10}$$

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

$$|T_i(F_2) - Q_i| < n^{3/4} \quad . \tag{11}$$

To prove (11) 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 6.5. *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 (11) 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 [29] 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 6.6 ([29]). *Let* $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ *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 (10) and (11) we get that there exists $\sigma = \sigma(\epsilon) > 0$ such that w.h.p. for $i \geq 3$,

$$T_i(F_2) \leq 2n \times (1-\sigma)^i + n^{3/4} \ . \tag{12}$$

Further, recall that by (9)

$$\text{w.h.p. } T_i(F_2) = 0 \text{ for all } i \geq C \log n \ . \tag{13}$$

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

$$2n \times (1-\phi)^i \geq 2n^{3/4} \ . \tag{14}$$

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

$$T_i(F_2) \leq (4n) \times (1-\phi)^i \ . \tag{15}$$

If $i \geq C \log n$ then (15) holds by (13). If $i < C \log n$ then by (12), (15) 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 (12) and (15) it follows that there is $\rho < \phi$ such that w.h.p. for all $i \geq 3$, $T_i(F_2) \leq 2n \times (1-\rho)^i$. □

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 values of additional variables at each step and simplify the resulting formula based on these values to obtain a *residual formula*, in which satisfied clauses have been removed and the remaining clauses have been shortened by the removal of their falsified literals. We call such algorithms *forward search* algorithms. These algorithms include UC, ORDERED-DLL, GUC mentioned in the introduction as well as several more sophisticated variants []. 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 by the algorithm made when there are no unit clauses are *free* choices. In UC this free choice is a random assignment to a random unassigned variable; in ORDERED-DLL this is an assignment of 0 to the smallest-numbered unassigned variable and in GUC this is the assignment that satisfies a random literal in a random clause of smallest size in the residual formula.

We now analyze natural DPLL algorithms that are backtracking extensions of these forward search algorithms to a full tree search. In these extensions, if the first path in the tree search fails and finds a contradiction, a DPLL algorithm backtracks, undoing the forced choices up to the last free choice, flips the assignment to that variable, calls it a forced choice rather than a free choice, and then continues. At this point there are many options for how to continue, each of which yields a different DPLL algorithm. Probably the simplest option would be to act as if the algorithm had reached this point without backtracking and apply the original heuristic. An alternative heuristic we call FS-backtracking (inspired by [21]) 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.

Given a forward search algorithm A, we write A-SIMPLE to denote its extension using simple backtracking and A-FS for the DPLL algorithm extending A using FS-backtracking. Initial experiments comparing ORDERED-DLL-FS to ORDERED-DLL-SIMPLE 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. A key property of FS-backtracking that is useful in our analysis, as in that of [21], is that at any time when the value of any variable in the partial assignment has flipped at most once during the algorithm's execution, the reduced formula is uniformly random conditional on the number of clauses of each size.

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', The main idea for our DPLL lower bounds is to show that each of the algorithms we consider is likely to arrive at a point during its execution in which the residual formula is unsatisfiable but any resolution proof of this fact requires exponential size; thus the algorithm must run for exponential time beyond this point.

Define a *stage* during the execution of a DPLL algorithm to be a sequence of steps during which the partial assignment of values to variables is constant. A t-*stage* is a stage in which the value of precisely t variables has 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 3-CNF formula with $(2.281 \pm \epsilon)(n-t)$ clauses and a 2-CNF formula with $(0.999 \pm \epsilon)(n-t)$ 2-clauses.*

Recall from our discussion in the introduction that if the residual formula at a bad t-stage is uniformly distributed with these properties then it is unsatisfiable almost certainly [4] and by Theorem 1.1 any such resolution proof almost certainly requires exponential size.

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

1. *For each $A \in \{$UC,ORDERED-DLL,GUC$\}$, there exists $p_A > 0$ such that the execution of* any *algorithm that is a backtracking extension of $A \in \{$UC,ORDERED-DLL,GUC$\}$ on a random 3-CNF formula with $r_A n$ clauses reaches a* bad *t-stage with $t \leq n/2$ such that the residual formula is uniformly distributed, with probability at least p_A.*

2. *For each $A \in \{$UC,ORDERED-DLL,GUC$\}$, an execution of algorithm A-*FS *on a random 3-CNF formula with $r_A n$ clauses reaches a* bad *t-stage with $t \leq n/2$ such that the residual formula is uniformly distributed w.h.p.*

Proof. The lemma follows readily from results in [14, 1, 21]. 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 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 rn 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 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 [14, 1] for UC,ORDERED-DLL and in [21] for GUC.

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

$$C_3(t) = r(1-t/n)^3 \cdot n + o(n) \ , \tag{16}$$

$$C_2(t) = \frac{3r}{2}(t/n)(1-t/n)^2 \cdot n + o(n) \ . \tag{17}$$

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

$$C_3(t) = r(1-t/n)^3 \cdot n + o(n) \ , \tag{18}$$

$$C_2(t) = (\frac{3}{2}rt/n - \frac{3}{4}r(t/n)^2 + \ln(1-t/n))(1-t/n) \cdot n + o(n) \ . \tag{19}$$

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 $t_0 \le (1-\delta)n$ and w.h.p. for all $0 \le t \le t_0$, $C_2(t) \le (1-\epsilon)(n-t)$, 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 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. $\sum_{t=0}^{n-1} C_1(t) < Mn$. 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 r, ϵ in equations (16)–(19) we see that indeed w.h.p. we have the right number of 2- and 3-clauses for a bad configuration. Moreover, w.h.p. the condition on the density of 2-clauses also holds and, therefore, with positive probability we indeed get a bad configuration. In particular, for UC,ORDERED-DLL, if $r = r_{\mathrm{UC}} = 3.81$, this occurs when $t \approx .22625n$ with positive probability. For GUC, if $r = r_{\mathrm{GUC}} = 3.98$, this occurs when $t \approx .243n$. This yields our positive probability results for arbitrary backtracking versions of UC, ORDER-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 [21] 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 the algorithm learned 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 [21], 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 (16)–(19) (the $o(n)$ term absorbing the effect of any flips). Thus the existence of a bad t-stage for $t \leq n/2$ holds w.h.p. $\square$

Corollary 7.3. *Let* $r_{\mathrm{UC}} = r_{\mathrm{ORDERED\text{-}DLL}} = 3.81$ *and let* $r_{\mathrm{GUC}} = 3.98$. *For* $A \in \{\mathrm{UC},\mathrm{ORDERED\text{-}DLL},\mathrm{GUC}\}$*, there exists* $p_A > 0$ *such that an execution of* any *backtracking extension of algorithm* $A \in \{\mathrm{UC},\mathrm{ORDERED\text{-}DLL},\mathrm{GUC}\}$ *on a random 3-CNF formula with* $r_A n$ *clauses takes time* $2^{\Omega(n)}$ *with probability at least* p_A*. For* $A \in \{\mathrm{UC},\mathrm{ORDERED\text{-}DLL},\mathrm{GUC}\}$*, an execution of algorithm* A-FS *on a random 3-CNF formula with* $r_A n$ *clauses takes time* $2^{\Omega(n)}$ *w.h.p.*

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 $2/3 \leq r < 2.28$, random formulas with $(1 - \epsilon)n$ 2-clauses and rn 3-clauses are unsatisfiable w.h.p. then the bounds of 3.81 and 3.98 will be immediately reduced. In fact, if r 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. A survey of lower bounds for random 3-SAT via differential equations. *Theoret. Comput. Sci., to appear*. Available as http://www.research.microsoft.com/~optas/lbsurvey.ps.

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

[3] D. Achlioptas, A. Chtcherba, I. Istrate, and C. Moore. The phase transition in 1-in-k SAT and NAE 3-SAT. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms (Washington, DC, 2001)*, pages 721–722. ACM, New York, 2001.

[4] D. Achlioptas, L. M. Kirousis, E. Kranakis, and D. Krizanc. Rigorous results for random $(2+p)$-SAT. *Theoret. Comput. Sci., to appear.*

[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] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 1992.

[7] D. Angluin and L. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.

[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 R. Impagliazzo. Random CNF's are hard for the polynomial calculus. In *Proceedings 40th Annual Symposium on Foundations of Computer Science*, pages 415–421, New York,NY, October 1999. IEEE.

[10] E. Ben-Sasson and A. Wigderson. Short proofs are narrow – resolution made simple. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 517–526, Atlanta, GA, May 1999.

[11] 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.

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

[13] 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*.

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

[15] 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.

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

[17] 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.

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

[21] 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.

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

[23] 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, 1998.

[24] 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.

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

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

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

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

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

[30] 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).*

[31] 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.

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

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

[34] 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, 1992.

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

A Proof of Lemma 5.9

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

Lemma A.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*

$$B_i \leq (1-\rho)^i m, \quad \textit{for all } i \geq 0 \ . \tag{20}$$

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 5.9 follows from Lemma A.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 A.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 $\xi > 0$

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

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 6.6. 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 $h > 0$,

$$\begin{aligned}
&\Pr[Q > (2+\xi)\mathbf{E}(Q)] \\
&= \Pr[\exp(hQ) > \exp((2+\xi)h\mathbf{E}(Q))] \\
&\leq \mathbf{E}(\exp(hQ)) \times \exp(-(2+\xi)h\mathbf{E}(Q)) \ . \qquad (21)
\end{aligned}$$

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

$$\begin{aligned}
\mathbf{E}(\exp(hQ)) &= \mathbf{E}\left(\prod_{i=1}^d \exp(hQ_i)\right) \\
&= \prod_{i=1}^d \mathbf{E}\left(\exp\left(hQ_i\right)\right) \\
&= \left(\mathbf{E}\left(\exp\left(hQ_i\right)\right)\right)^d \ . \qquad (22)
\end{aligned}$$

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 (23) to (24) we use (20). To go from (24) to (25) we require $h < \rho$, which suffices to guarantee the sum's convergence. Finally, to go from (25) to (26) we use that for $h > 0$, $e^{-h} > 1 - h$.

$$\begin{aligned}
&\mathbf{E}(\exp(hT)) \\
&= \sum_{i=0}^{\infty} \Pr[T = i] \exp(hi) \\
&= \sum_{i=0}^{\infty} \Pr[T = i] \left(1 + hi + (\exp(hi) - hi - 1)\right) \qquad (23) \\
&\leq 1 + h\mu + \sum_{i=1}^{\infty} (1-\rho)^i (\exp(hi) - hi - 1) \qquad (24) \\
&= 1 + h\mu + (1-\rho)\left(\frac{1}{\rho - 1 + \exp(-h)} - \frac{h+\rho}{\rho^2}\right) \qquad (25) \\
&< 1 + h\mu + (1-\rho)\left(\frac{1}{\rho - h} - \frac{h+\rho}{\rho^2}\right) \qquad (26) \\
&= 1 + h\mu + \frac{h^2(1-\rho)}{\rho^2(\rho - h)} \ . \qquad (27)
\end{aligned}$$

Now, substituting $h = \rho^3$ in (27) we get (28), while (29) follows from $\mu \geq 1 > (\rho+1)^{-1}$.

$$\begin{aligned}
\mathbf{E}(\exp(\rho^3 T)) &< 1 + \rho^3\mu + \frac{\rho^3}{\rho+1} \qquad (28) \\
&< 1 + 2\rho^3\mu \ . \qquad (29)
\end{aligned}$$

Note now that, by (21) and (22), for all $h > 0$,

$$\begin{aligned}
&\Pr[Q > (2+\xi)\mathbf{E}(Q)] \\
&\leq \left(\frac{\mathbf{E}(\exp(hT))}{\exp((2+\xi)h\mathbf{E}(T))}\right)^{2(1+\xi)t} \\
&\leq \left(\frac{\mathbf{E}(\exp(hT))}{\exp(2h\mathbf{E}(T))}\right)^{2(1+\xi)t} \times \exp(-2h\xi\mu t) \ . \qquad (30)
\end{aligned}$$

Taking $h = \rho^3$, (29) implies that the ratio in (30) is bounded by 1. Thus, since $\mu \geq 1$, if $\theta = 2\rho^3$, then

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

$\square$