The efficiency of resolution and Davis-Putnam procedures *

Paul Beame†
Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195
beame@cs.washington.edu

Richard Karp
Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195
karp@cs.washington.edu

Toniann Pitassi‡
Computer Science Department
University of Toronto
Toronto, ON Canada
toni@cs.toronto.edu

Michael Saks§
Department of Mathematics
Rutgers University
New Brunswick, NJ
saks@math.rutgers.edu

October 9, 2001

Abstract

We consider several problems related to the use of resolution-based methods for determining whether a given boolean formula in conjunctive normal form is satisfiable. First, building on work of Clegg, Edmonds and Impagliazzo, we give an algorithm for unsatisfiability that when given an unsatisfiable formula of F finds a resolution proof of F. The runtime of our algorithm is subexponential in the size of the shortest resolution proof of F. Next we investigate a class of backtrack search algorithms for producing resolution refutations of unsatisfiability, commonly known as Davis-Putnam procedures and provide the first asymptotically tight average-case complexity analysis for their behavior on random formulas. In particular, for a simple algorithm in this class, called *ordered DLL* we prove that the running time of the algorithm on a randomly generated k-CNF formula with n variables and m clauses is $2^{\Theta(n(n/m)^{1/(k-2)})}$ with probability $1-o(1)$. Finally, we give new lower bounds on $\mathrm{res}(F)$, the size of the smallest resolution refutation of F, for a class of formulas representing the pigeonhole principle, and for randomly generated formulas. For random formulas, Chvátal and Szemerédi had shown that random 3-CNF formulas with a linear number of clauses require exponential size resolution proofs and Fu extended their results to k-CNF formulas. These proofs apply only when the number of clauses is $\Omega(n \log n)$. We show that a lower bound of the form 2^{n^γ} holds with high probability even when the number of clauses is $n^{(k+2)/4-\epsilon}$.

*Preliminary versions of these results appeared in the 37th IEEE Symposium on Foundations of Computer Science [BP96] and the 30th ACM Symposium of Theory of Computing [BKPS98].

†Research supported by NSF grant CCR-9303017.

‡ Research supported by NSF grant CCR-9457782 and US-Israel BSF Grant 95-00238. This work was done while the author was at the University of Arizona.

§ Research supported by NSF grant CCR-9700239. This work was done while the author was on sabbatical at University of Washington.

1 Introduction

The satisfiability problem for boolean formulas in conjunctive normal form (*CNF formulas*) plays a central role in computer science. Historically, it was the "first" NP-complete problem. It is the natural setting in which to formulate a wide variety of constraint satisfaction problems. Its companion problem, finding a proof of unsatisfiability of a given unsatisfiable formula, plays an important role in artificial intelligence, where it is referred to as *propositional theorem proving*, and also in circuit testing.

In the last three decades, a tremendous amount of research has been directed towards understanding the mathematical structure of the satisfiability problem, and to developing algorithms for satisfiability testing and propositional theorem proving. Much of this research has centered around the method of *resolution*. The *resolution principle* says that if C and D are clauses and x is a variable, then any assignment that satisfies both of the clauses $C \vee x$ and $D \vee \neg x$ also satisfies $C \vee D$. The clause $C \vee D$ is said to be a *resolvent* of the clauses $C \vee x$ and $D \vee \neg x$ *on the variable* x. A *resolution refutation* for a CNF formula F consists of a sequence of clauses $C_1, C_2, \ldots, C_s$ where (i) each clause C_i is either a clause of F, or is a resolvent of two previous clauses and (ii) C_s is the empty clause, denoted by Λ. We can represent the proof as an acyclic directed graph on vertices $C_1, \ldots, C_s$ where each clause of F has in-degree 0, and any other clause has in-degree 2 with its two in-arcs from the two clauses that produced it. It is well known that resolution is a *sound* and *complete* propositional proof system, i.e., a formula F is unsatisfiable if and only if there is a resolution refutation for F. Resolution is the most widely studied approach to propositional theorem proving, and there is a large body of research exploring *resolution algorithms*, i.e., algorithms that on input an unsatisfiable formula F, output a resolution refutation of F.

Any resolution algorithm can be used trivially to test satisfiability of an arbitrary (satisfiable or unsatisfiable) formula F, since F is satisfiable if and only if the algorithm finds no refutation. Nearly all satisfiability testers that have been studied in the literature can be derived in this way from resolution algorithms, and we say that such satisfiability testers are *resolution-based*.

One fundamental approach to satisfiability testing is to use backtrack search to look for a satisfying assignment. Algorithms that use this approach are commonly called Davis-Putnam procedures, but we will refer to them as *DLL algorithms*, after Davis, Logemann and Loveland who first considered them ([DLL62]). A DLL algorithm can be described recursively as follows. First check whether F is *trivially satisfiable* (has no clauses) or is *trivially unsatisfiable* (contains an empty clause) and if so stop. Otherwise, select a literal l_i (a variable or the complement of a variable) and apply the search algorithm recursively to search for a satisfying assignment for the formula $F\lceil_{l_i=0}$ obtained by setting $l_i = 0$ in F. If the search succeeds, then we have an assignment for F. Otherwise, repeat the search with the formula $F\lceil_{l_i=1}$. If neither of these searches finds a satisfying assignment then F is not satisfiable.

A particular DLL algorithm is specified by a *splitting rule*, which is a subroutine that for each recursively constructed formula, determines the next splitter (literal to recurse on) and the assignment to try first. In general, the splitting rule may depend on the details of the structure of the original formula, and on the results of the computation in other recursive calls. For a particular formula F, different splitting rules may result in vastly different running times.

If the splitter for some given formula F is a literal l such that l is contained in a *unit clause* (a clause C of size one) then the $l = 0$ branch falsifies C and thus terminates immediately. Effectively, the algorithm fixes $l = 1$. A splitting rule is said to use *unit propagation* if, for any formula F that has a *unit clause* (clause of size one) the splitter is chosen to be a literal in such a clause. Virtually all splitting rules considered in the literature use unit propagation, and it can be shown that any splitting rule can be modified so that it uses unit propagation at the cost of a factor of at most $O(n)$ in the running time of the algorithm, where n is the number of underlying variables. We will only consider algorithms that use unit propagation.

The simplest such splitting rule is: fix an ordering of the variables $x_1, \ldots, x_n$. For a sub-formula F' obtained by fixing some variables, if there is a unit clause, the splitter is the first literal belonging to such a clause. Otherwise select the first unfixed variable. The algorithm obtained from this splitting rule is called *ordered DLL*.

The execution of a DLL algorithm A on formula F can be represented by a labeled rooted binary tree, denoted $T_A(F)$, in the usual way. Each node corresponds to a recursive call. Each internal node is labeled by its splitter and the two out-edges correspond to the possible assignments. For any node, the path from the root to that node defines a partial assignment (*restriction*) of the variables, and the recursive call at that node is applied to the sub-formula obtained by applying the restriction to the original formula. Each leaf is either a *success leaf*, i.e., all of the original clauses are satisfied by the associated restriction, or a *failure leaf*, i.e., at least one original clause is falsified by the restriction. Each failure leaf is labeled by one of the original clauses that if falsifies. F is unsatisfiable if and only if all leaves are failure leaves, in which case the tree as labeled above (with internal nodes labeled by splitters and leaves labeled by falsified clauses) is called a *DLL-refutation* of F. The size of the refutation is defined to be the number of nodes of the tree. It is easy to see that a formula F has a DLL refutation if and only if it is unsatisfiable. Thus DLL-refutations form a complete and sound proof system.

Given a DLL refutation, it is not hard to show by induction that if we start from the clauses labeling the leaves, and work towards the root, we can label each internal node by a clause which is a resolvent of the clauses labeling its two children, and the root will be labeled by the empty clause. This tree is now the directed graph representation of a resolution refutation, and thus the DLL proof system can be viewed naturally as a restricted version of resolution.

This paper focuses on some problems concerning resolution refutations, DLL refutations, and algorithms for satisfiability. Of central importance are two parameters defined for any unsatisfiable formula F:

- $\text{res}(F)$, the size of the smallest resolution refutation of F,
- $\text{DLL}(F)$, the size of the smallest DLL refutation of F.

We define $\text{res}(F) = \text{DLL}(F) = \infty$ for satisfiable formulas. It follows from the above discussion that $\text{DLL}(F) \geq \text{res}(F)$ for all formulas F. Furthermore, for any DLL-procedure for satisfiability, $\text{DLL}(F)$ is a lower bound for its running time on F.

Our results fall into three groups. The first group of results shows that the resolution and DLL proof systems are, to some extent, *automatizable* in the sense that for each of these systems, there is an algorithm that on input an unsatisfiable formula F, finds a refutation of F within the proof system in time that can be upper bounded non-trivially in terms of the size of the optimal refutation within that system. In particular, we show that for any formula F in n variables and m clauses that has a DLL refutation of size at most S, there is an algorithm running in $n^{O(\log S)}m$ time that *finds* a DLL refutation of F. The analogous algorithm for resolution uses $2^{O(\sqrt{n \log S} \log n)}m$ time to find a resolution refutation for a formula F possessing one of size S. Clegg, Edmonds, and Impagliazzo [CEI96] have given algorithms to determine unsatisfiability within these time bounds but they do not produce resolution or DLL refutations.

The second group of results concerns lower bounds for general resolution proofs. We develop simpler methods for obtaining resolution lower bounds than previously used. We illustrate this by first showing a very simple lower bound on the resolution proof complexity of the pigeonhole principle which also significantly improves on the best previous lower bounds for this problem. Our main object of study for resolution however is that of randomly chosen k-CNF formulas and we use our technique to obtain much stronger lower bounds on the resolution proof complexity of such formulas.

Our third group of results analyzes the complexity of proving unsatisfiability for random formulas (above the threshold) using DLL algorithms, the algorithms that are used most commonly in practice for satisfiability-testing. We obtain the first non-trivial upper bound for resolution proofs of unsatisfiability of random formulas by showing that one of the simplest of all DLL algorithms, ordered DLL, has a running time that is qualitatively similar to the size of the best possible resolution proofs. There is still a gap between our upper bounds for DLL and the lower bounds for resolution, but we show that our analysis for ordered DLL is tight. We also make progress towards showing that our upper bound is tight for all DLL algorithms by extending our lower bound for ordered DLL to a broader class of DLL algorithms.

Our techniques result in significant simplifications and improvements of previous algorithms and lower bounds. A preliminary version of this work [BP96] pointed out that further simplification could be obtained by finding a direct relationship between res(F) and the minimum b for which F has a proof with all clauses at most b. Recently, Ben-Sasson and Wigderson [BSW99] have developed such a characterization of res(F) and DLL(F). Using this characterization, one can derive some of our general resolution bounds more simply. We conclude our paper with a discussion of this improvement and other directions for further research.

Because our bounds for random formulas for both DLL and general resolution are our most significant results but are necessarily spread over several sections of the paper, we discuss them now in more detail.

Random k-CNF formulas

We consider the usual random k-CNF model, which is defined in terms of three integer parameters n, k and m. A formula is generated by selecting m clauses of size k independently and uniformly from the set of all clauses of size k on n variables. We denote this distribution by $\mathcal{F}_m^{k,n}$, and write $F \sim \mathcal{F}_m^{k,n}$ to mean that F is selected from this distribution. The ratio $\Delta = m/n$ is referred to as the *clause density*.

The random k-CNF model has been widely studied for several good reasons. First, it is an intrinsically natural model, analogous to the random graph model, that sheds light on fundamental structural properties of the satisfiability problem. Second, for appropriate choice of parameters, randomly chosen formulas are empirically difficult for satisfiability, and are a commonly used benchmark for testing satisfiability algorithms. (See, for example, the encyclopedic survey of the SAT problem in [GPFW97].) Lastly, it is a useful model for evaluating the effectiveness of a particular propositional proof system: strong lower bounds on proof size for random k-CNF formulas attest to the fact that the proof system in question is ineffective on average.

A fundamental conjecture about the random k-CNF formula model, (see [CS88, BFU93, CF90, CR92, FS96, KKKS98]) says that there is a constant θ_k, the *satisfiability threshold*, such that a random k-CNF formula of clause density Δ is almost certainly satisfiable for Δ bounded below θ_k (as n gets large), and almost certainly unsatisfiable if Δ is bounded above θ_k. There is considerable empirical and analytic evidence for this. Recently Friedgut [Fri99] showed that for each n and k there is a threshold $\theta_k(n)$ such that for any $\epsilon > 0$, random k-CNF formulas with clause density $\Delta \leq \theta_k(n) - \epsilon$ are almost certainly satisfiable and those with clause density $\Delta \geq \theta_k(n) + \epsilon$ are almost certainly unsatisfiable. However, he does not rule out the possibility that $\theta_k(n)$ fails to converge to a constant. It is known that $\theta_2 = 1$ is independent of n [CR92, Goe96] and that for each k, $\theta_k(n)$ is bounded between two constants b_k and d_k that are independent of n, e.g., $3.26 \leq \theta_3(n) \leq 4.596$ [AS00, JSV00].

The threshold indicates three distinct ranges of clause density for investigating complexity. For Δ at the threshold, an effective algorithm must be able to distinguish between unsatisfiable and satisfiable instances. Below the threshold, a random formula is almost certainly satisfiable and the problem of interest is to find a satisfying assignment quickly.

Above the threshold, the formula is almost certainly unsatisfiable and we have the two closely related questions (i) what is the typical size of the smallest unsatisfiability proof? and (ii) how quickly can an algorithm find a proof?

Several empirical studies of DLL procedures on random k-CNF formulas have been done, e.g., by Selman, Mitchell, and Levesque [SML96] and Crawford and Auton [CA96]. The former applies ordered DLL (defined earlier) to random k-CNF formulas for various values of Δ. The curves in [SML96, CA96] show very low complexity for Δ below the threshold, a precipitous increase in complexity at the threshold, and a speedy decline to low complexity above the threshold.

Much has been made of the analogy with statistical physics [KS94], and there has been a suggestion that the computational complexity at the threshold is evidence of a critical phenomenon in complex systems and based on underlying edge-of-chaos behavior present only near the threshold. The empirical observation that satisfiability is easy below the threshold is supported by analytical work. The proofs of the aforementioned lower bounds on θ_k were obtained by analyzing some DLL algorithm and showing that it almost certainly finds a satisfying assignment in linear time, provided that Δ is below some specified constant.

In their seminal paper, Chvátal and Szemerédi [CS88] showed that for any fixed Δ above the threshold there is a constant $\kappa_\Delta > 0$ such that $\text{res}(F) \geq 2^{\kappa_\Delta n}$ almost certainly if F is a random k-CNF formula of clause density Δ. (This result substantially improved previous work of Franco and Paull [FP83] with showed sub-exponential time lower bounds for refuting the same class of formulas using particular DLL algorithms.) On the other hand, Fu [Fu95] showed that $\text{res}(F)$ is almost certainly polynomial in n for $m = \Omega(n^{k-1})$. These results together with the empirical work motivate the problem of determining the best constant κ_Δ for which $\text{res}(F) \geq 2^{\kappa_\Delta n}$ with prob $1 - o(1)$ for random k-CNF formulas F of density Δ. The lower bound in [CS88] as presented does not give bounds on the dependence of κ_Δ on Δ, but rough estimates show that for 3-CNF formulas the bound decreases as $1/\Delta^{\Omega(\Delta^4)}$. This implies that the lower bound declines extremely quickly and becomes trivial when the number of clauses grows above $n \log^{1/4} n$. For larger clause size k Fu [Fu95] obtained better bounds, but with a similar exponential drop-off. Is there really such a sharp decline in complexity for random formulas above the threshold?

Our new lower bounds show that the drop-off in κ_Δ is at most polynomial in Δ. We show that for any constant $\epsilon > 0$ there is a constant $a_\epsilon > 0$ such that random 3-CNF formulas the complexity of resolution proofs is almost certainly at least $2^{a_\epsilon(n/\Delta^{4+\epsilon})}$ and obtain similar results for random k-CNF formulas larger values of k. In particular, our results imply that even random formulas with moderately large clause densities require proofs of weakly exponential size. More precisely, for any $k \geq 3$ and $\epsilon > 0$, we show that there is a $\gamma > 0$ such that almost all k-CNF formulas in n variables with at most $n^{(k+2)/4-\epsilon}$ clauses require resolution refutations of size at least 2^{n^γ}. For example, a random 3-CNF formula with $n^{5/4-\epsilon}$ clauses requires exponentially large resolution proofs.

Although these resolution bounds show that the decline in κ_Δ is at most inverse polynomial in Δ, it is not immediately clear that even a polynomial decline with Δ is achievable. There seem to be no previously known non-trivial upper bounds on the running time of algorithms on random instances above the threshold. We prove the first such non-trivial upper bound by showing that for $F \sim \mathcal{F}_m^{k,n}$, the size of the DLL-refutation of F produced by ordered DLL is $2^{O(n/\Delta^{1/(k-2)})} n^{O(1)}$. Thus κ_Δ does indeed decline as a fixed power of Δ.

At the upper end, our result shows that when $m = \Omega(n^{k-1}/\log^{k-2} n)$, the algorithm runs in polynomial time, improving on Fu's $\Omega(n^{k-1})$ bound on the number of clauses needed for polynomial-size resolution proofs but also giving an algorithm to find such proofs.

There is a gap between the exponents on Δ given by our lower bounds for general resolution and our upper bound for ordered DLL. What is the optimal exponent for general resolution or for DLL, which is more interesting since such bounds would have implications for practical satisfiability testing? We show that our upper bound for ordered DLL is indeed tight in that ordered DLL requires proofs of $2^{\Omega(n/\Delta^{1/(k-2)})}$ size.

Can a different splitting rule achieve a better exponent than ordered DLL on random formulas? We expand our understanding of DLL algorithms by showing that in the case $k = 3$, for $m = \Omega(n^{3/2} \log^{\epsilon} n)$, the lower bound for ordered DLL extends to a larger class of algorithms, whose splitting rule (aside from unit propagation) is independent of the formula. The key step in proving this lower bound is Lemma 30 which applies to *any* DLL splitting rule and therefore may be of independent interest. It shows that with high probability, along every path of a DLL tree that is "not too long", the number of unit clauses generated cannot be very large. While Lemma 30 is general, the remainder of the proof of the lower bound unfortunately depends heavily on the independence of the splitting rule from the formula.

Overall, our results show that for $F \sim \mathcal{F}^{k,n}_{\Delta n}$, $\log_2 \mathrm{res}(F)$ decays as a fixed power of Δ, suggesting that there is not an isolated point of complexity at the threshold, but rather a slow and gradual decline in complexity as Δ increases.

2 Preliminary definitions

Let $X = \{x_1, \ldots, x_n\}$ be a set of boolean variables. Following usual parlance, an assignment ρ of 0-1 values to some subset of the variables is called a *restriction*. We will abuse notation and identify ρ with the set of literals set to 1 by ρ. We write $v(\rho)$ for the set of variables that are assigned values by ρ.

Similarly, a clause C over the variables X can be viewed as a set of literals, and we write $v(C)$ for the underlying set of variables. If $\mathcal{C}$ is a set of clauses, or F is a CNF formula, we write $v(\mathcal{C})$ or $v(F)$ for the underlying sets of variables.

If C is a clause and ρ is a restriction then ρ satisfies C if it sets some literal of C to 1. If ρ does not satisfy C, we define $C\lceil_\rho$, the *restriction of C by ρ* to be the clause obtained from C by deleting all literals set to 0 by ρ. For a formula F the *restriction of F by ρ*, $F\lceil_\rho$, is the formula obtained by removing all clauses satisfied by ρ, and replacing any other clause C of F by $C\lceil_\rho$.

If $\mathcal{P} = (C_1, \ldots, C_r)$ is a resolution refutation of a formula F and ρ is a restriction we can construct a refutation of $F\lceil_\rho$, denoted $\mathcal{P}\lceil_\rho= (D_1, \ldots, D_r)$, as follows. For ease of description, we allow the proof to contain clauses that are identically 1. For $i \in [r]$ having defined $D_1, \ldots, D_{i-1}$, if ρ satisfies C_i then let D_i be the identically 1 clause. Otherwise, (i) if C_i is a clause of F, let $D_i = C_i\lceil_\rho$. and (ii) if C_i is the resolvent of C_j and C_k on variable x, then if either D_j or D_k is 1, then let D_i equal the other one, and otherwise let D_i be the resolvent of D_j and D_k on x. It is not hard to show that after deleting the 1-clauses, the result is a resolution proof of $F\lceil_\rho$.

A resolution refutation $\mathcal{P}$ is said to be *b-bounded* if all of the clauses appearing in it have size at most b.

For the purpose of generating test formulas, the most natural model of a random k-CNF formula on n variables with clause density Δ is to choose $m = \Delta n$ clauses independently with replacement. This distribution, which we denote $\mathcal{F}^{k,n}_m$, is the one analyzed in [CS88]. Another model, which is used in [Fri99], is to choose each of the possible clauses independently with probability $p = m/\binom{n}{k}2^k$; call this $\mathcal{F}^{k,n}(p)$. An easy argument shows that when considering properties of formulas that are monotone (or anti-monotone) with respect to sets of clauses, the almost certain properties under both distributions are the same up to a change from m to $m \pm o(m)$. This is just a natural extension of the similar (and more precise) equivalences for the random graph model as shown for example in [AV79]. We generally assume the distribution $\mathcal{F}^{k,n}_m$. We write

$F \sim \mathcal{F}$ to mean F is a random formula selected according to distribution $\mathcal{F}$.

We make frequent use of two well-known tail bounds for the binomial distribution (see [ASE92], Appendix A).

Proposition 1. *If Y is a random variable distributed according to the binomial distribution $B(n,p)$ then*

1. *Pr*$[Y < np/4] \leq 2^{-(np)/2}$
2. *Pr*$[Y > Cnp] \leq (\frac{e}{C})^{-Cnp}$

3 Automatizability of DLL and resolution

The quantity $\mathrm{res}(F)$ (resp. $\mathrm{DLL}(F)$) tells us the size of the smallest resolution refutation (resp. DLL refutation) of F. A fundamental problem is to find effective algorithms for constructing resolution refutations and DLL refutations whose size is "close" to optimal. This is the *automatizibility problem* for proof systems, which was formalized in [BPR97].

DEFINITION 2. *Let $\mathcal{S}$ be an arbitrary propositional proof system.* [1] *For the unsatisfiable formula F, let $s(F)$ denote the size of the smallest refutation of F in $\mathcal{S}$. Then $\mathcal{S}$ is said to be automatizable if there exists a deterministic algorithm that takes as input an unsatisfiable formula F on n variables and m clauses, and outputs an $\mathcal{S}$-refutation of f in time polynomial in $s(F)$ and n and m. More generally $\mathcal{S}$ is $q(s,n,m)$-automatizable if there exists a deterministic algorithm that runs in time $q(s(F),n,m)$ and outputs an $\mathcal{S}$-refutation of F (whose size is necessarily also bounded by $q(s(F),n,m)$).*

Theorem 3.

1. *The DLL proof system is $q(s,n,m)$-automatizable for $q(s,n,m) = n^{O(\log s)}m$.*
2. *The resolution proof system is $q(s,n,m)$-automatizable for $q(s,n,m) = 2^{O(\sqrt{n \log s}\log n)}m$.*

These results, especially the second, fall short of the desired polynomial automatizability. Nevertheless, even the second is strong enough that if $\mathrm{res}(F)$ is sub-exponential, $2^{o(n)}$, then our algorithm finds a sub-exponential size resolution refutation in sub-exponential time. The results are closely related to, and motivated by, previous results of Clegg, Edmonds and Impagliazzo. Their results concern the polynomial calculus proof system (called Groebner in [CEI96]), which is more general than the resolution system. In their paper, they proposed and analyzed satisfiability testing algorithms based on the Groebner basis algorithm from commutative algebra. When run on an unsatisfiable formula F, their algorithm produces a refutation in the polynomial calculus proof system (but not necessarily a resolution refutation). They give two algorithms for this, the first of which finds a refutation in time bounded above by $O((\mathrm{DLL}(F))^{\log n})$ and the second finds a refutation in time bounded above by $O(2^{\sqrt{n \mathrm{res}(F)}\log n})$. In other words, provided that F has a short DLL refutation (resp., resolution refutation), their first (resp. second) algorithm finds a refutation that is "not too big", but in the stronger polynomial calculus proof system. Our two algorithms, which closely parallel theirs, achieve comparable running times, but produce, respectively, a DLL refutation and a resolution refutation.

Proof of Theorem 3. The theorem asserts the existence of two algorithms, which on input an unsatisfiable formula F, find, respectively, a DLL-refutation and a resolution refutation, within a specified time bound. The two algorithms are most easily described together.

[1] We will not provide a general definition of propositional proof system, since we are focusing exclusively on the two concrete systems, resolution and DLL that we have defined above. The interested reader can readily formulate a general definition or consult [CR77].

First, we need a subroutine, called **Bounded-search**, which takes as input F and an integer parameter b and finds a b-bounded resolution refutation of F if one exists. It is not hard to implement this subroutine in time $T_0(n, m, b) = n^{O(b)+O(m)poly(n)}$, e.g., by listing the b-bounded clauses of F and, for each clause on the list, resolve it with each clause preceding it (if possible) and add the resolvent to the end of the list, if it is of size at most b and doesn't duplicate anything on the list. If the algorithm constructs the empty clause then we have the desired refutation, otherwise, there is no such refutation.

The main algorithm called **Resolution-search** also takes as input F and an auxiliary parameter b. First we use **Bounded-search**(F, b) to find a b-bounded resolution refutation for F if it exists. If not, then for each of the $2v(F)$ literals l, apply **Resolution-search** to the formula $F\lceil_{l=1}$, in order to identify the literal l for which **Resolution-search**$(F\lceil_{l=1})$ terminates fastest. These $2n$ calls to **Resolution-search** are executed in a sequence of parallel rounds; in round i the ith step of each of the $2n$ calls is performed. As soon as the first of the calls terminates, say for literal l^*, all of the other calls are aborted, except the call corresponding to $\neg\, l^*$, which is run to completion. The output of **Resolution-search**(F, b) consists of the derivation of the singleton clause l^*, followed by the derivation of the singleton clause $\neg\, l^*$ followed by $\emptyset$. (Note that the parallel search for the literal l^* described above can be replaced by a more space efficient "doubling search" which in iteration i runs each of the recursive calls one by one from the beginning for 2^i steps, stopping the first time that one of the calls terminates. The time analysis below can be modified to apply to this variant.)

The analysis of the algorithm will rely on the following technical fact, which is easily proved by induction:

Proposition 4. *Suppose that $T(n, s)$ is a function defined for nonnegative integers n and $s > 0$ that satisfies, for some positive increasing function $h(n)$, positive constant C and $\lambda > 1$:*

$$\begin{array}{rcll} T(0, s) & \leq & h(0) & \\ T(n, s) & \leq & h(n) & \textit{if } s \leq 1 \\ T(n, s) & \leq & h(n) + CnT(n-1, \frac{s}{\lambda}) + T(n-1, s) & \textit{if } n \geq 1 \textit{ and } s > 1 \end{array}$$

Then $T(n, s) \leq h(n)(1 + C^{\log_\lambda s} n^{2\log_\lambda s})$.

We now prove the first part of the automatization theorem. Here we use the above algorithm with $b = 0$. It is not hard to see that in this case, the output by **Resolution-search** can be viewed as a DLL-refutation, since the DAG associated to the proof is a tree. We upper bound the running time of the algorithm in terms of DLL(F). Let $T_1(n, s; m)$ denote the maximum running time of **Resolution-search**$(F, 0)$ over all formulas F with at most n variables and m clauses and for which DLL$(F) \leq s$. Consider a DLL-refutation of size at most s and let x_i be the splitting variable at the root. The left and right branches of the tree give refutations for $F\lceil_{x_i}$ and $F\lceil_{\neg x_i}$, and the smaller of these is of size at most $s/2$. Hence at least one of the recursive calls terminates after at most $T_1(n-1, s/2; m)$ steps, and so the literal l^* is found after at most that number of rounds. The time for each round can be bounded above by Cn for some constant C. Once l^* is found, it takes at most $T_1(n-1, s; m)$ steps to complete the call to **Resolution-search**$(F\lceil_{\neg\, l^*})$. Thus, we conclude that for fixed m, $T_1(n, s; m)$ satisfies the recurrence of the above Proposition with $h(n) = T_0(n, m, b)$ and $\lambda = 2$. We conclude that $T_1(n, s; m) = n^{O(\log s)}O(m)$ as required to prove the first automatization result.

For the second result, first define, for a set $\mathcal{P}$ of clauses, $\mathcal{P}[b]$ to be the subset of clauses of size greater than b. For a formula F, let res(F, b) denote the minimum of $|\mathcal{P}[b]|$ over all resolution refutations of F (so that for $b < 0$, res(F) = res(F, b)). Let $T_2(n, s; m, b)$ denote the maximum time needed by **Resolution-search**(F, b) on (n, m)-formulas F satisfying res$(F, b) \leq s$. Note that $T_2(n, s; m, b) \leq T_0(n, b)$ if $s < 1$ and $T_2(0, s; m, b) = O(1)$. Suppose n and s are both at least 1. Let F be an (n, m)-formula and let $\mathcal{P}$ be a resolution refutation of F with $|\mathcal{P}[b]| \leq s$. For a literal l, let $c(b, l)$ be the number of clauses of $\mathcal{P}[b]$

containing l. The average of $c(b, l)$ over literals is greater than $|\mathcal{P}[b]|b/2n$ and hence there exists a literal l with $c(b, l) > b|\mathcal{P}[b]|/2n$. Note that the refutation $\mathcal{P}\lceil_{l=1}$ of $F\lceil_{l=1}$ has at most $|\mathcal{P}[b]|(1 - \frac{b}{2n})$ clauses, and hence $T_2(n, s; m, b)$ satisfies the recurrence for T in the proposition with $\lambda = \frac{2n}{2n-b}$ and $h(n) = T_0(n, b)$. Applying the proposition, we conclude that $T_2(n, b) \leq T_0(n, b)n^{O(\frac{n}{b}\log s)}$. Choosing $b = \sqrt{n \log s}$, yields an upper bound of $2^{O(\sqrt{n \log s}\log n)}O(m)$, to complete the proof of the theorem. $\square$

Remark. The result of Ben Sasson and Wigderson mentioned in the introduction implies that the **Bounded-search** routine is sufficient to automatize resolution. More specifically, they show that for any formula F, if DLL$(F) \leq s$ then **Bounded-search** with $b = O(\log s)$ finds a resolution refutation of F, and if res$(F) \leq s$ then **Bounded-search** with $b = O(\sqrt{n \log s})$ finds a resolution refutation of F.

4 Lower bounding resolution proof complexity

For any unsatisfiable formula on n variables, res$(F) \leq$ DLL$(F) \leq 2^n + 1$, since a DLL proof of an n-variable formula is a binary tree of maximum depth n. Unless $coNP = NP$ one would expect that there are formulas where res(F) is super-polynomial in $|F|$, but it is not obvious how to prove such lower bounds. In a breakthrough paper, Haken [Hak85] proved the first exponential lower bounds in general resolution for a class of formulas related to the pigeon-hole principle. Haken obtained his bounds using an elegant new technique called "bottleneck counting". The technique was developed further in [Urq87] to give more general bounds on resolution refutations. Building on Haken's and Urquhart's arguments, Chvátal and Szemerédi [CS88] used the bottleneck counting method to show that for a sufficiently large constant c, almost certainly a random 3-CNF formula with cn clauses requires an exponential length resolution refutation. Fu [Fu95] recently extended this bound to apply when the number of clauses is larger, but for 3-CNF formulas it gives no improvement on [CS88].

All of these proofs have the same general structure. To prove a lower bound on res(F) for all F belonging to some specified class of formulas $\mathcal{F}$, the first step is to prove a result of the following type: for some larger class $\mathcal{F}'$ of formulas, any resolution refutation of $G \in \mathcal{F}'$ must have a "large" clause.

In the second step, which is typically the more involved part, an arbitrary resolution refutation of F is considered. Each clause in the proof is viewed as allowing certain truth assignments to flow through it, namely those that it falsifies. Using the result of the first step, one shows that every truth assignment must flow through some "complex" or "large" clause that only permits a small number of truth assignments to pass (thus the term "bottleneck counting"). Therefore the number of clauses in the refutation must be large. The complications in the argument come in making the association between complex clauses and truth assignments.

Our method uses something very much like the first step, but replaces the second step by an argument that leads to stronger results with simpler arguments. We assume for contradiction that $F \in \mathcal{F}$ has a small proof. We use this assumption to show that F can be modified to a formula $F' \in \mathcal{F}'$ that has a proof with no large clauses, contradicting the first step.

We consider two methods for modifying the formula F to get F'. The first is to apply a restriction, i.e., fix a small set of variables. The second is to *augment* F, i.e., add some additional clauses to F. The restriction method is equivalent to the special case of the augmentation method where the clauses that are added are all unit clauses.

4.1 Lower bounds for the pigeonhole principle

We illustrate our approach to lower bounds for general resolution proofs by giving a very simple proof of the exponential lower bounds for the class $\{\neg\ PHP^m_n : m > n\}$ of pigeonhole principle formulas considered by Haken. The variables of the formula $\neg\ PHP^m_n$ correspond to the entries $P_{i,j}$ of an $m \times n$ boolean matrix (we think of the rows as corresponding to"pigeons" and the columns as corresponding to "holes"). Its clauses are: (1) $P_{i,1} \vee P_{i,2} \vee ... \vee P_{i,n}$, for each $i \leq m$; (each row has at least one 1, or every pigeon goes into a hole) and (2) $\neg P_{i,k} \vee \neg P_{j,k}$, for each $i, j \leq m$, $k \leq n$, $i \neq j$, (each column has at most one 1, or every hole gets at most one pigeon). Since $m > n$, this is trivially unsatisfiable. Note that the number of clauses in $\neg\ PHP^m_n$ is $m + \binom{m}{2}n \leq m^3$. We prove:

Theorem 5. *For $n \geq 2$, any resolution refutation of $\neg PHP^n_{n-1}$ has size at least $2^{n/20}$.*

Proof. As in the lower bound proof of Haken [Hak85], a truth assignment to the underlying variables $P_{i,j}$ is *critical* if it defines a one-to-one, onto map from $n-1$ rows (pigeons) to $n-1$ columns (pigeon-holes), with the remaining pigeon not mapped to any hole. A critical assignment where i is the pigeon left out is called i*-critical*. In what follows we will only be interested in critical truth assignments.

Let C be a clause. The monotone clause $M(C)$ associated to C is obtained by replacing each occurrence of a negative literal $\neg\ P_{i,k}$ by the set of literals $\{P_{l,k} \mid l \neq i\}$. It is easy to check that C and $M(C)$ are satisfied by precisely the same set of critical assignments.

We will be interested in restrictions corresponding to partial matchings that one obtains by repeated choosing an i, j, and setting $P_{i,j} = 1$, $P_{i,j'} = 0$ for $j' \neq j$, and $P_{i',j} = 0$ for $i' \neq i$. Observe that if one begins with a resolution refutation of $\neg PHP^n_{n-1}$ and one chooses such a partial matching restriction that sets t variables to 1 then the result is resolution refutation of $\neg PHP^{n-t}_{n-t-1}$. Furthermore, the restriction applied to the monotone conversion of each clause results in the same clause as doing the monotone conversion of the clause first and then applying the restriction. (The transformation to monotone clauses, due to Sam Buss, is not essential, but will make our argument slightly cleaner.)

So let $C_1, ..., C_S$ be a resolution refutation of $\neg PHP^n_{n-1}$ and $M_1 = M(C_1), \ldots, M_S = M(C_S)$ be its monotone conversion. Say that a clause M_t is *large* if it has at least $n^2/10$ (positive) literals, i.e. at least 1/10-th of all the variables. To show that $S \geq 2^{n/20}$, we will show that the number L of large clauses is at least $2^{n/20}$. Assume for contradiction that $L < 2^{n/20}$. Let $d_{i,j}$ denote the number of large clauses containing $P_{i,j}$. By averaging, there is an i, j with $d_{i,j} \geq L/10$. Choose such an i, j, and apply the restriction $P_{i,j} = 1$, and $P_{i,j'} = 0$ for $j' \neq j$ and $P_{i',j} = 0$ for $i' \neq i$. Applying this restriction we obtain a monotone conversion of a refutation of $\neg PHP^{n-1}_{n-2}$ with at most $9L/10$ large clauses. Applying this argument iteratively $\log_{10/9} L$ many times, we are guaranteed to have knocked out all large clauses. Thus, we are left with a refutation of $\neg PHP^{n'}_{n'-1}$, where

$$n' \geq n - \log_{10/9} L = (1 - (\log_{10/9} 2)/20)n > 0.671n,$$

and where no clause in the refutation is large. But this contradicts the following lemma (originally due to Haken [Hak85]) which states that such a refutation must have a clause whose monotone conversion has size at least $2(n')^2/9 > n^2/10$.

Lemma 6. *Any resolution refutation of $\neg PHP^n_{n-1}$ must contain a clause C such that $M(C)$ has at least $2n^2/9$ literals.*

Proof. Given a clause C, let

$$badpigeons(C) = \{i \mid \text{there is some } i\text{-critical assignment } \alpha \text{ falsifying } C\}$$

Define the complexity $\text{comp}(C) = |badpigeons(C)|$.

Let $\mathcal{P}$ be a resolution refutation of $\neg PHP^n_{n-1}$, and consider the complexity of the clauses that appear in $\mathcal{P}$. The complexity of each initial clause is at most 1 and the complexity of the final false clause is n.

Note that if we use the resolution rule to derive a clause C from two previous clauses C' and C'', we have that $\text{comp}(C) \leq \text{comp}(C') + \text{comp}(C'')$, since any assignment falsifying C must also falsify at least one of C' or C''. If C is the first clause in the proof with $\text{comp}(C) > n/3$, we must have $n/3 < \text{comp}(C) \leq 2n/3$. We will show that $M(C)$ contains a large number of variables.

For $comp(C) = t$ will now show that $M(C)$ has at least $(n-t)t \geq 2n^2/9$ distinct literals mentioned. Fix some $i \in badpigeons(C)$, and let α be an i-critical truth assignment falsifying C. For each $j \notin badpigeons(C)$, consider the j-critical assignment, α', obtained from α by replacing i by j, that is by mapping i to the place that j was mapped to in α. This assignment satisfies C, and differs from α only in one place: if α mapped j to l, then α' maps i to l. Since C and $M(C)$ agree on all critical assignments and $M(C)$ is monotone, it must contain the variable $P_{i,l}$.

Running over all $n-t$ j's not in $badpigeons(C)$ (using the same α), it follows that $M(C)$ must contain at least $n-t$ distinct variables $P_{i,l}$, $l \leq n$. Repeating the argument for all $i \in badpigeons(C)$ shows that C contains at least $(n-t)t$ positive literals. □

□

We note that Theorem 5 improves somewhat upon Haken's bound of $2^{n/577}$ although our major interest is in its simpler proof rather than in the better size bound. Buss and Turán [BT88] extend Haken's argument to show that $\neg PHP^m_n$ requires super-polynomial size resolution lower bounds as long as $m < n^2/\log n$. Our argument can be extended to re-derive their result.

5 Resolution lower bounds for random formulas

The previous section gave a simple proof that $\text{res}(F)$ is large for a specific class of formulas. Abstractly, we can summarize the approach as follows. We assume for contradiction that F has a small proof. In particular F has a proof with a small number of large clauses. We then modify F (in the above case, restrict some variables) to obtain another formula F' having a proof with no large clauses. We then obtain a contradiction by showing that any proof of F' contains a large clause.

In this section we show how the same idea can be used to obtain simple and improved lower bounds on $\text{res}(F)$ that hold with high probability when F is a randomly chosen formula of a given clause density.

5.1 Resolution refutations usually require big clauses

The first main ingredient is a result (essentially from [CS88]) that provides a set of parameterized conditions on a formula F that imply that any resolution refutation of F has at least one large clause. We then show that when F is a random formula of clause density Δ, these conditions hold for certain values of the parameters. We need some definitions.

DEFINITION 7. *For a real number σ, a set of clauses $\mathcal{C}$ is σ-*sparse *if $|\mathcal{C}| \leq \sigma|v(\mathcal{C})|$ where $v(\mathcal{C})$ is the set of variables appearing in $\mathcal{C}$.*

DEFINITION 8. *If $\mathcal{C}$ is a set of clauses and l is a literal, we say that l is* pure *in $\mathcal{C}$ if some clause of $\mathcal{C}$ contains l and no clause of $\mathcal{C}$ contains $\neg\, l$.*

DEFINITION 9. *For $s \geq 1$ and $\epsilon \in (0,1)$, the following properties are defined for formulas F:*

> *Property $A(s)$: Every set of $r \leq s$ clauses of F is 1-sparse.*
>
> *Property $B_\epsilon(s)$: For r satisfying $s/2 < r \leq s$, every subset of r clauses of F has at least ϵr pure literals.*

The following result is essentially due to Chvátal and Szemerédi (and is closely related to Haken's argument in Lemma 6).

Proposition 10. *Let $s > 0$ be an integer and F be a CNF formula. If properties $A(s)$ and $B_\epsilon(s)$ both hold for F, then F has no $\epsilon s/2$-bounded proof.*

Proof. The result holds trivially if F is satisfiable, so assume that F is unsatisfiable. We say that a set S of clauses implies a clause C if every assignment that satisfies all of the clauses in S satisfies C. Since F is unsatisfiable, F implies any clause C. The complexity of a clause C with respect to F, comp(C) is the minimum size of a set of clauses that implies C.

Let $\mathcal{P}$ be a resolution refutation of F.

Claim. If F satisfies $A(s)$ then there is a clause $C \in \mathcal{P}$ for which $s/2 < \text{comp}(C) \leq s$.

It is easy to see that if S is a minimal set of clauses that implies C and the literal l is pure in S then l is in C. Hence for the C given by the above claim, property $B_\epsilon(F)$ implies that $|C| \geq \epsilon\text{comp}(C) \geq \epsilon s/2$.

So it suffices to prove the claim. We first note that if $\mathcal{C}$ is a set of clauses such that any subset is 1-sparse, then it is satisfiable. Indeed, the sparsity condition is equivalent to the hypothesis of the Hall theorem on systems of distinct representatives and the conclusion of the theorem is that there is a one-to-one mapping sending each clause $C \in \mathcal{C}$ to a variable $v_C \in C$. We can thus satisfy each clause C by appropriately fixing v_C.

Now if S implies Λ then S is unsatisfiable, so $A(s)$ implies comp$(\Lambda) > s$. Choose C to be the first clause in $\mathcal{P}$ with comp$(C) \geq s/2$. Since C is the resolvent of two previous clauses C_h, C_j and comp$(C) \leq$ comp(C_h) + comp(C_j) we conclude that comp$(C) < s$.

□

Lemma 11. *For each integer $k \geq 3$ and $\epsilon > 0$, there are constants $C(k), c_\epsilon(k) > 0$ such that the following holds. Let m, n be integers with $m = \Delta n$ for some $\Delta \geq 1$. Let $F \sim \mathcal{F}_m^{k,n}$.*

1. *If $s \leq C(k)n/\Delta^{1/(k-2)}$, then F satisfies $A(s)$ probability $1 - o(1)$ in s.*
2. *If $s \leq c_\epsilon(k)n/\Delta^{2/(k-2-\epsilon)}$ then F satisfies $B_\epsilon(s)$ with probability $1 - o(1)$ in s.*

These lemmas are proved by elementary combinatorial probability. We defer the proof until section 5.3 where we state and prove a generalization (Lemma 17).

5.2 The formula augmentation method

Armed with these results we give a very simple proof that a random k-CNF F of density Δ satisfies $res(F) \geq 2^{n/\Delta^{O(1)}}$ with probability $1 - o(1)$. An *augmentation* of a formula F is a formula obtained by adding additional clauses to F. As we now describe, augmentations can simplify proofs.

We say that a clause C *subsumes* a clause D if $C \subset D$. Suppose that $\mathcal{P}$ is a proof of F and let G be a CNF formula. We can obtain a proof of the augmented formula $F \wedge G$, denoted $\mathcal{P}\lceil_G$, as follows: For each clause $D \in \mathcal{P}$, if there is a clause C in G such that C subsumes D replace D by C and propagate this simplification forward through the rest of the proof by (possibly) shortening clauses that were produced using D.

Observe that in the case that G consists of clauses of size 1, G corresponds naturally to a restriction ρ, and there is a close correspondence between the proofs $\mathcal{P}\lceil_G$ and $\mathcal{P}\lceil_\rho$.

Following our general approach, suppose we want to prove that $res(F)$ is big. Assuming for contradiction that F has a small proof $\mathcal{P}$, we show that for some integer s and $\epsilon > 0$, there is a G such that (i) G subsumes all clauses of size $\epsilon s/2$ of $\mathcal{P}$ and such that (ii) $F \wedge G$ satisfies $A(s)$ and $B_\epsilon(s)$. This is a contradiction since (i) implies that $\mathcal{P}\lceil_G$ is an $\epsilon s/2$ bounded refutation of $F \wedge G$, while (ii) and Proposition 10 imply that no such refutation is possible.

We will realize this approach by considering G chosen at random from some distribution $\mathcal{G}$. We say that the distribution $\mathcal{G}$ satisfies property $g(b, M)$, for $b, M > 0$ if for any clause C of size at least b, if $G \sim \mathcal{G}$ then Pr[G does not subsume C] $\leq 1/M$.

Proposition 12. *Let F be a formula. Let $s, M \geq 1$ and $\epsilon > 0$, and let $\mathcal{G}$ be a distribution over formulas that satisfies $g(\epsilon s/2, M)$. Then*

$$res(F) \geq M \times Pr_{G \sim \mathcal{G}}[F \wedge G \text{ satisfies both } A(s) \text{ and } B_\epsilon(s)].$$

Proof. The conclusion follows immediately from the chain of inequalities:

$$\begin{aligned} \mathrm{res}(F)/M &\geq \Pr_{G \sim \mathcal{G}}[\mathcal{P}\lceil_G \text{ is not } \epsilon s/2 \text{ bounded}] \\ &\geq \Pr_{G \sim \mathcal{G}}[F \wedge G \text{ satisfies both } A(s) \text{ and } B_\epsilon(s)]. \end{aligned}$$

The second inequality is immediate from Proposition 10. For the first inequality, if $\mathcal{P}$ is a proof of F of size res(F), it has at most res(F) clauses of size at least $\epsilon s/2$, and by property $g(\epsilon s/2, M)$ the probability that there is a clause not subsumed by G is at most res$(F)/M$. $\square$

The above is stated for a fixed formula F. For distributions over formulae we have:

Theorem 13. *Let $\mathcal{F}$ be a distribution over formulas. Let $s, M \geq 1$ and $\epsilon > 0$ and suppose that $\mathcal{G}$ is a distribution over formulas that satisfies $g(\epsilon s/2, M)$. Then*

$$\begin{aligned} Pr_{F \sim \mathcal{F}}[res(F) < M/2] \leq\ & 2(Pr_{F \sim \mathcal{F}, G \sim \mathcal{G}}[F \wedge G \text{ does not satisfy } A(s)\ ] \\ &+ Pr_{F \sim \mathcal{F}, G \sim \mathcal{G}}[F \wedge G \text{ does not satisfy } B_\epsilon(s)]). \end{aligned}$$

Proof. For a formula F, let

$$p(F) = Pr_{G \sim \mathcal{G}}[F \wedge G \text{ does not satisfy } A(s)] + Pr_\rho[F\lceil_\rho \text{ does not satisfy } B_\epsilon(s)].$$

By Proposition 12, res$(F) > (1 - p(F))M$. Thus $\Pr_{F \sim \mathcal{F}}[\mathrm{res}(F) < M/2] \leq \Pr_F[p(F) > 1/2] < 2E_F[p(F)]$, where $E[\cdot]$ denotes expectation. This last quantity is equal to the right hand side of the claimed inequality. $\square$

We now use a form of self-reduction to obtain:

Theorem 14. *For each $\mu > 0$, there exists a constant $a_\mu > 0$ such that if $F \sim \mathcal{F}_m^{k,n}$ for $m = \Delta n$ with $\Delta > 1$, then res$(F) \geq 2^{a_\mu n/\Delta^{1+4/(k-2)+\mu}}$ with probability $1 - o(1)$ in n. In particular, when $k = 3$ this reduces to res$(F) \geq 2^{a_\mu n/\Delta^{5+\mu}}$.*

Proof. We apply Theorem 13 to the case $\mathcal{F} = \mathcal{F}_m^{k,n}$ by choosing $\mathcal{G} = \mathcal{F}$. Note that $F \wedge G$ has the distribution $\mathcal{F}_{2m}^{k,n}$, so Lemma 11 implies that for any $\epsilon > 0$ and for some constant $c_\epsilon(k) > 0$, if $s \leq c_\epsilon(k)n/(2\Delta)^{2/(k-2-\epsilon)}$ then $F \wedge G$ satisfies $A(s)$ and $B_\epsilon(s)$ with probability $1 - o(1)$. (Note that the requirement on s for $B_\epsilon(s)$ is more stringent than that for $A(s)$). Next, we choose M as large as possible so that $\mathcal{G}$ satisfies $g(\epsilon s/2, M)$. For a clause C of size at least $\epsilon s/2$, the probability that a single randomly chosen clause subsumes C is $\binom{\epsilon s/2}{k}/(2^k\binom{n}{k})$. If $G \sim \mathcal{G}$, then the probability that none of its m clauses subsume C is at most $(1-\binom{\epsilon s/2}{k}/(2^k\binom{n}{k}))^m \leq 2^{-d_\epsilon m s^k \Delta/n^{k-1}}$ for some constant d_ϵ. Substituting $s = c_\epsilon n/\Delta^{2/(k-2-\epsilon)}$, we have that, for sufficiently small ϵ, $\mathcal{G}$ satisfies $g(\epsilon s/2, 2^{e_\epsilon n \Delta^{1+4/(k-2)+O(\epsilon)}})$ for some constant e_ϵ. Hence with probability $1 - o(1)$, res$(F) \geq 2^{\Omega(n/\Delta^{1+4/(k-2)+\alpha})}$ for any $\alpha > 0$. $\square$

Corollary 15. *For any $k \geq 3$ and $\epsilon > 0$, there is a constant γ such that almost all k-CNF formulas in n variables with at most $n^{2k/(k+2)-\epsilon}$ clauses require resolution proofs of size at least 2^{n^γ}.*

These results provide strong lower bounds on res(F) for random formulas. However, note that as k gets large, the exponent in the lower bound of res(F) tends to n/Δ, while in the upper bound obtained by using ordered DLL, the exponent is $n/\Delta^{1/(k-2)}$. We'd like to close this gap.

Observe that for property $A(s)$, Lemma 11 requires $s = O(n/\Delta^{1/(k-2)})$ while for property $B_\epsilon(s)$, it requires $s \leq O(n/\Delta^{2/(k-2-\epsilon)})$. It turns out that one way to significantly close the above gap would be to show that the second part of Lemma 11 holds if we weaken the bound on s:

Problem. Is it true that if $F \sim \mathcal{F}_m^{k,n}$ then F satisfies $B_\epsilon(s)$ with probability near 1, for $s \leq n\Delta^{1/(k-2-\epsilon)+o(1)}$?

If this were true, then the argument used in the above theorem would be improved substantially to: for $F \sim \mathcal{F}_m^{k,n}$, with probability near 1, res$(F) \geq 2^{n/\Delta^{2/(k-2-\epsilon)}}$, which is very comparable to the upper bound. The corollary is improved so that m can be as large as $n^{(k-\epsilon)/2}$. We discuss this problem further in the last section of the paper.

Lacking an affirmative answer to the above problem, we look for other ways to improve our result. In the next section, we'll see that we can narrow the gap substantially using random restrictions instead of augmentations.

5.3 The random restriction method

We now apply an approach analogous to the above, using restrictions instead of augmentations. As mentioned above, applying a restriction can be viewed as applying an augmentation consisting of clauses of size one, but we use the language of restrictions because it is more familiar and natural.

Specializing Theorem 13 to the case of restrictions yields the following. If $\mathcal{R}$ is a probability distribution over the set of restrictions, we say that $\mathcal{R}$ has *property* $R(b, M)$ if for $b, M > 0$ for any clause C on X of size at least b, Pr[ρ does not satisfy C] $\leq 1/M$. Then

Theorem 16. *Let $\mathcal{F}$ be a distribution over k-bounded formulae. Let s, $M \geq 1$ and $\epsilon > 0$ and suppose that $\mathcal{R}$ is a distribution over restrictions that satisfies $R(\epsilon s/2, M)$. Then*

$$\begin{aligned} Pr_{F\sim\mathcal{F}}[res(F) < M/2] &\leq 2(Pr_{F\sim\mathcal{F},\rho\sim\mathcal{R}}[F\lceil_\rho \text{ does not satisfy } A(s)\,] \\ &\quad + Pr_{F\sim\mathcal{F},\rho\sim\mathcal{R}}[F\lceil_\rho \text{ does not satisfy } B_\epsilon(s)\,]). \end{aligned}$$

We will use this result to get a lower bound on $\mathrm{res}(F)$ for random k-CNF formulas, using the distribution $\mathcal{R}_t$ over restrictions where we first choose $v(\rho) \subseteq X$ by selecting each variable independently with probability t/n and then set the selected variables uniformly at random.

Lemma 11 needs to be generalized to formulas obtained from a random k-CNF by applying a random restriction.

Lemma 17. *For each integer $k \geq 3$ and $\epsilon > 0$ there are constants $C(k), c_\epsilon(k) > 0$ such that the following holds. Let m, n, s, t be integers with $m = \Delta n$ for some $\Delta \geq 1$. Let $F \sim \mathcal{F}_m^{k,n}$ and $\rho \sim \mathcal{R}_t$.*

1. *If $t \leq C(k)n/m^{1/k}$ and $s \leq C(k)n/\Delta^{1/(k-2)}$, then $F\lceil_\rho$ satisfies $A(s)$ probability $1 - o(1)$ in s.*
2. *If $s, t \leq c_\epsilon(k)n/\Delta^{2/(k-2-\epsilon)}$ then $F\lceil_\rho$ satisfies $B_\epsilon(s)$ with probability $1 - o(1)$ in s.*

The proofs of these lemmas require a preliminary result. Let F and ρ be as in the statements of the lemmas. Let M denote the event that $F\lceil_\rho$ contains an empty clause. For $r, q > 0$, let $Q(r, q)$ denote the event that there exists a set R of at most r variables such that $v(C) \subseteq R$ for at least q non-empty clauses C of $F\lceil_\rho$.

Proposition 18. *Let $m \geq n \geq t \geq k \geq 3$ be positive integers.*

1. *If $t \leq n/m^{1/(k-1)}$ then $Pr[M] \leq m^{-1/(k-1)}$.*
2. *Assume $n \geq 2k^2$. For $r, q \geq 1$,*

$$Pr[Q(r,q)] \leq \left(\frac{ne}{r}\right)^r \left(\frac{2ekmr(t+r)^{k-1}}{qn^k}\right)^q,$$

Proof. For the first part, if C is a fixed k-clause then $C\lceil_\rho$ is empty iff ρ sets all literals in C to 0. Thus $\Pr_C[C\lceil_\rho \text{ is empty}] = (t/2n)^k$. Therefore $\Pr[M]$ is upper bounded by the expected number of empty clauses of $\mathcal{F}\lceil_\rho$ which is $m(\frac{t}{2n})^k \leq m^{-1/(k-1)}$.

For the second part, let $R \subseteq X$ with $|R| = r \geq 1$. Let C denote a randomly chosen clause and $I = v(C) \cap R$. Let $W(C)$ denote the event that $C\lceil_\rho$ is nonempty and its variables are contained in R. Then $\Pr[W(C)] = \sum_{i=1}^k \Pr[|I| = i]\Pr[v(C) - I \subseteq v(\rho) : |I| = i]$. Now $\Pr[v(C) - I \subseteq v(\rho) : |I| = i] = (t/n)^{k-i}$, while $\Pr[|I| = i] = \binom{r}{i}\binom{n-r}{k-i}/\binom{n}{k} \leq \binom{r}{i}\frac{k!}{(k-i)!}(n-k)^i \leq \binom{k}{i}(\frac{r}{n-k})^i \leq 2\binom{k}{i}(\frac{r}{n})^i$, where the last inequality holds since we assume $n \geq 2k^2$. Thus:

$$\begin{aligned} \Pr[W(C)] &\leq 2\sum_{i=1}^{k}\binom{k}{i}(\frac{r}{n})^i(\frac{t}{n})^{k-i} \\ &\leq \frac{2kr}{n}\sum_{j=0}^{k-1}\binom{k-1}{j}(\frac{r}{n})^j(\frac{t}{n})^{k-1-j} \\ &= \frac{2kr}{n}(\frac{r+t}{n})^{k-1}. \end{aligned}$$

Calling this latter probability p, if F is a random formula, the number of clauses of F for which $W(C)$ holds has the binomial distribution, $B(m,p)$. The probability that at least q clauses of F are contained in S after ρ is applied is bounded above by:

$$\text{Pr}[B(m,p) \geq q] \leq \binom{m}{q} p^q \leq \left(\frac{2ekmr(r+t)^{k-1}}{qn^k}\right)^q, \tag{1}$$

Summing this over the $\binom{n}{r} \leq (en/r)^r$ subsets of X of size r we obtain the desired upper bound on Pr$[Q(r,q)]$. $\square$

Proof. **of Lemma 17**.

We begin with the first part. The probability that $A(s)$ fails is at most $\sum_{r=1}^{s} \text{Pr}[Q(r,r+1)] + \text{Pr}[M]$. By the first part of Proposition 18, Pr$[m] = o(1)$ in m and hence also in s. For $r \leq s$ we have:

$$Q(r,r+1) \leq \left(\frac{ne}{r}\right)^r \left(\frac{2ekmr(r+t)^{k-1}}{(r+1)n^k}\right)^{r+1} \tag{2}$$

$$\leq \frac{r}{en}\left(\frac{2ekm(r+t)^{k-1}}{(r+1)n^{k-1}}\right)^{r+1} \tag{3}$$

For $t \leq r \leq s$, and for some constant $C_1(k) > 0$, if $s \leq C_1(k)n/\Delta^{1/(k-2)}$, the quantity (3) is at most $\frac{r}{2^r n}$. Similarly, for $1 \leq r < t$, the quantity (3) is at most $\frac{r}{n}(C_2(k)m(\frac{t}{n})^{k-1})^{r+1}$ which is at most $\frac{r}{2^r n}$ for $t \leq C_3(k)n/m^{1/(k-1)}$ for some positive constants $C_2(k), C_3(k)$. Thus the $\sum_{r=1}^{s} Q(r,r+1)$ is at most $\frac{1}{n}\sum_{r=1}^{s}\frac{r}{2^r} < 2/n$ which is $o(1)$ in n and hence in s.

Finally, in the hypothesis of the lemma, we take $C(k) = \min(C_1(k), C_2(k))$.

Now for the second part of the lemma. We first observe:

$$\text{Pr}[\neg B_\epsilon(s)] \leq \sum_{\lfloor s/2\sigma \lceil \leq r \leq s/\sigma} \text{Pr}[Q(r,\sigma r)] + \text{Pr}[M]$$

where $\sigma = 2/(k+\epsilon)$. To see this, suppose that $B_\epsilon(s)$ fails. We want to show that either M holds or $Q(r,\sigma r)$ holds for some r in the given range. Assume M does not hold. Since $B_\epsilon(s)$ fails, there is a collection $\mathcal{C}$ of w clauses that has at most ϵw pure literals with $s/2 \leq w \leq s$. We upper bound $|v(\mathcal{C})|$. The sum of the clause sizes is wk and if u is the number of pure literals, the impure variables contribute at most $wk-u$ to this sum. Each impure variable appears in at least two clauses so the number of impure variables is at most $\frac{wk-u}{2}$ and thus $|v(\mathcal{C})| \leq \frac{wk-u}{2} + u = \frac{wk+u}{2} \leq \frac{w(k+\epsilon)}{2}$. Extend the set $v(\mathcal{C})$ to a set R of size $r = \lfloor\frac{w(k+\epsilon)}{2}\rfloor$, so that $\lfloor\frac{s}{2\sigma}\rfloor \leq r \leq \frac{s}{\sigma}$ and R contains at least σr clauses. Thus $Q(r,\sigma r)$ holds.

So it suffices to upper bound the sum of $Q(r,\sigma r)$ for $\lfloor\frac{s}{2\sigma}\rfloor \leq r \leq \frac{s}{\sigma}$. We have:

$$\begin{aligned} Q(r,\sigma r) &\leq \left(\frac{ne}{r}\right)^r \left(\frac{2ekm(r+t)^{k-1}}{\sigma n^k}\right)^{\sigma r} \\ &= \left(\frac{(ne)^{1/\sigma}C_1 m(r+t)^{k-1}}{\sigma r^{1/\sigma} n^k}\right)^{\sigma r} \\ &\leq \left(\frac{C_4 m r^{-1/\sigma}(r+t)^{k-1}}{n^{k-1/\sigma}}\right)^{\sigma r} < 2^{-\sigma r} \end{aligned}$$

for appropriate constant $C_4(k)$, provided that

$$\frac{C_4 m r^{-1/\sigma}(r+t)^{k-1}}{n^{k-1/\sigma}} \leq \frac{1}{2}.$$

For $\sigma = 2/(k+\epsilon)$, this last condition is satisfied if both s and t are at most $c_\epsilon(k)n\cdot(\frac{n}{m})^{2/(k-2-\epsilon)}$, for some constant $c_\epsilon(k) \geq 0$. Now the total failure probability for property $B_\epsilon(s)$ is at most $\sum_{r=\lfloor\frac{s}{2\sigma}\rfloor}^{s} 2^{-\sigma r} + \Pr[M]$ which is clearly $o(1)$ in s and Part 2 is proved. □

Next, in order to apply Theorem 16, we determine as large an M as possible such that the distribution $\mathcal{R}_t$ satisfies $R(\epsilon s/2, M)$, where s and t are the largest numbers satisfying the hypotheses of both parts of the previous lemma.

Lemma 19. *$\mathcal{R}_t$ satisfies $R(\epsilon s/2, e^{\epsilon st/4n})$.*

Proof. For a fixed clause C and for $\rho \sim \mathcal{R}_t$, a variable x that is in C is fixed by ρ to satisfy C with probability $t/2n$ so the probability that ρ doesn't satisfy a given clause C is at most $(1-t/2n)^{|C|} \leq e^{-|C|t/2n}$. □

Applying Theorem 16 using Lemmas 17 and 19 yields

Theorem 20. *For any integer $k \geq 3$ and $\epsilon > 0$ there is a constant $c'_\epsilon > 0$ such that the following holds. Let $F \sim \mathcal{F}_m^{k,n}$ where $m = \Delta n$ with Δ at least the satisfiability threshold θ_k.*

- *If $m \leq c'_\epsilon n^{2(k-1)/(k+\epsilon)}$ then res$(F) \geq 2^{\Omega(n^{1-1/(k-1)}/\Delta^{1/(k-1)+2/(k-2-\epsilon)})}$ with probability $1-o(1)$ in n.*
- *If $m \geq c'_\epsilon n^{2(k-1)/(k+\epsilon)}$ then res$(F) \geq 2^{\Omega(n/\Delta^{4/(k-2-\epsilon)})}$ with probability $1-o(1)$ in n.*

Proof. Let $C(k)$ and $c_\epsilon(k)$ be the constants from Lemma 17. Let $s = c_\epsilon(k)n/\Delta^{2/(k-2-\epsilon)}$. The two cases of the theorem correspond to the two constraints on t in Lemma 17. The constraint $t \leq C(k)n/m^{1/(k-1)}$ is the more stringent constraint if and only if $m^{1/(k-1)} \geq (c_\epsilon(k)/C(k))(m/n)^{2/(k-2-\epsilon)}$. This holds if and only if $n \geq c''(k)m^{1-(k-2-\epsilon)/(2k-2)} = c''(k)m^{(k+\epsilon)/(2k-2)}$ for some constant $c''(k) > 0$, i.e. if $m \leq c'_\epsilon n^{(2k-2)/(k+\epsilon)}$ for some constant $c'_\epsilon > 0$. In this case, applying Lemma 19 yields the first bound. In the case that $m > c'_\epsilon n^{(2k-2)/(k+\epsilon)}$, the constraint $t \leq c_\epsilon n(n/m)^{2/(k-2-\epsilon)}$ is the more stringent and Lemmas 19 and 17 yield the second bound. □

When $k = 3$, the largest m can be in the above and still get a lower bound of the form 2^{n^γ}, is $n^{6/5-\epsilon}$, which matches the result obtained for $k = 3$ in Corollary 15. For $k \geq 4$ we obtain the following corollary.

Corollary 21. *For $k \geq 4$, and $\epsilon > 0$, there is a constant $\gamma > 0$ such that almost all k-CNF formulas with at most $n^{(k+2)/4-\epsilon}$ clauses require resolution proofs of size at least 2^{n^γ}.*

Proof. To see this, assume that $m = n^{(k+2)/4-\epsilon}$. Since $k \geq 4$ if we take $\epsilon' = 4\epsilon$ then

$$\begin{aligned}
(k+2)/4-\epsilon &= 1+(k-2-\epsilon')/4 \\
&> 1+(k-2-\epsilon')/(k+\epsilon') \\
&= (2k-2)/(k+\epsilon')
\end{aligned}$$

Thus we can apply the second bound of Theorem 20 to derive that k-CNF formulas with at most m clauses almost certainly require resolution proofs of size $2^{\Omega(n/\Delta^{4/(k-2+\epsilon')})}$ which is $2^{\Omega(n^{2\epsilon'/(k-2+\epsilon')})}$. □

5.4 The Deletion Argument

In this subsection, we sketch a variant of the restriction approach, which, in a preliminary version of this work [BKPS98], was shown to yield the following theorem.

Theorem 22. *For each $\gamma > 0$, there exists a constant a_γ such that for all $m \geq n$, if F is a 3-CNF formula chosen according to $\mathcal{F}_m^{3,n}$ then with probability $1 - o(1)$, res$(F) \geq 2^{a_\gamma (n/\Delta^{4+\gamma})}$.*

Observing that this bound is non-trivial for $m = o(n^{5/(4+\gamma)})$ and combining with Corollary 21 we obtain:

Corollary 23. *For $k \geq 3$, and $\epsilon > 0$, there is a constant $\gamma > 0$ such that almost all k-CNF formulas with at most $n^{(k+2)/4-\epsilon}$ clauses require resolution proofs of size at least 2^{n^γ}.*

The proof of Theorem 22 in [BKPS98] is rather technical. The recent work of Ben-Sasson and Wigderson [BSW99] referred to earlier, shows how one can derive the same bound in a substantially simpler way given Lemmas 10 and 11. Therefore, we do not include our entire proof, but instead give a short sketch.

The major bottleneck in the argument of section 5.1 is the upper bound on t needed for Lemma 17. Indeed, for t much larger than $n/\sqrt{m}$, there is a substantial probability that $A(s)$ does not hold for $F\lceil_\rho$. In particular, the bound computed in the proof of Lemma 17 on the probability that a clause of a random F becomes empty is nearly tight; an easy computation shows that the probability that no clause of a random F becomes empty when ρ is applied is $e^{-\Theta(mt^3/n^3)}$ and so, since the presence of an empty clause in $F\lceil_\rho$ violates $A(s)$ we have that $t = o(n/m^{1/3})$. (The creation of unit clauses under the restriction placed an even stronger limitation on t).

To overcome this limitation, we want to avoid the creation of clauses of size 0 or 1 in $F\lceil_\rho$. To do this we modify the distribution on restrictions so that ρ may depend on F. The general idea is to first choose a random restriction and then delete any assignment that sets more than one variable in any clause of F. For technical reasons one must also delete assignments to variables that share some clause with too many other variables. By careful arguments one can show that appropriate analogues of Theorem 16 and Lemma 17, allowing the condition $t \leq cn/\sqrt{m}$ on the size of the restriction to be eliminated.

6 The behavior of DLL on random formulas

We now analyze particular DLL algorithms when applied to random formulas and show that we can obtain quite good upper bounds using a very simple splitting rule and that even certain generalizations of this splitting rule require much larger proofs than our current lower bound for general resolution shows.

6.1 Ordered DLL on random formulas

Theorem 24. *Let $k \geq 3$ and let $m = \Delta n$ where Δ is greater than the threshold $\theta_k(n)$ and $m = \Delta n$. Suppose that $F \sim \mathcal{F}_m^{k,n}$. Then with probability $1 - o(1)$ in n, the size of the refutation of F produced by ordered DLL is $2^{\Theta(n/\Delta^{1/(k-2)})} n^{\pm\Theta(1)}$. In particular, when $k = 3$, the refutation has size $2^{\Theta(n/\Delta)} n^{\pm\Theta(1)}$.*

The proof of this result has two parts, the upper bound and lower bound, which we split into two subsections. In the upper bound subsection, we first consider a variant of ordered DLL that is easier to analyze.

Upper bound for ordered DLL

We first consider a variant of ordered DLL which is bit more complicated to state but easier to analyze.

Algorithm A. Set $t = 6k\lceil n(n/m)^{1/(k-2)}\rceil$; in particular when $k = 3$ this is $18\lceil n^2/m\rceil$. Run ordered DLL, as long as the variables $x_1, \ldots, x_t$ are not all assigned. When reaching a partial assignment ρ in which $x_1, \ldots, x_t$ are all assigned (and possibly other variables by unit propagation), run the (polynomial-time) algorithm for 2-SAT on the subset $C_2(F, \rho)$ consisting of clauses of size at most 2 in the induced sub-formula $F\lceil_\rho$. The algorithm succeeds (finds a resolution refutation of F) provided that for each such ρ reached in the algorithm the sub-formula $C_2(F, \rho)$ is unsatisfiable.

To analyze Algorithm A we need:

Lemma 25. *Let F be a random 2-CNF formula chosen from $\mathcal{F}^{2,n'}_{2n'}$. Then the probability that F is satisfiable is $o(2^{-n'/9})$.*

Proof. Observe that the expected number of satisfying assignments for a 2-CNF formula with m' clauses and n' variables is $2^{n'}(3/4)^{m'}$ which is $o(2^{-n'/9})$ for $m' > 2.678n'$. (This bound can be reduced below 2 by using the techniques of [KKKS98].) ☐

Theorem 26. *Let Δ be greater than the satisfiability threshold $\theta_k(n)$ and $m = \Delta n$. If $F \sim \mathcal{F}^{k,n}_m$, then, with probability $1 - o(1)$ in n, Algorithm A produces a resolution refutation in time $2^{O(n/\Delta^{1/(k-2)})}n^{O(1)}$; in particular if $k = 3$ Algorithm A produces a resolution refutation in time $2^{O(n/\Delta)}n^{O(1)}$.*

Proof. Let $t = 6k\lceil n(n/m)^{1/(k-2)}\rceil$ and assume without loss of generality that $t < n/10$. Algorithm A clearly runs in time $2^{O(n/\Delta^{1/(k-2)})}n^{O(1)}$. To show that Algorithm A finds a refutation of F with probability $1-o(1)$, it suffices to show to show that with probability $1-o(1)$, $C_2(F, \rho)$ is unsatisfiable for all assignments ρ to $\{x_1, \ldots, x_t\}$. (Note that the restrictions occurring in the algorithm may have additional variables fixed by unit propagation but this can only increase our probability of success).

Fix ρ. Consider the set $\hat{C}_2(F, \rho)$ of clauses of size exactly 2. This size is a binomial random variable $B(m, q)$ where q is equal to the probability, for a random k-clause C, that $C\lceil_\rho$ is a 2-clause:

$$
\begin{aligned}
\Pr[C\lceil_\rho \text{ is a 2-clause}] &= \frac{1}{2^{k-2}} \cdot \Pr[|\{x_1, \ldots, x_t\} \cap v(C)| = k-2] \\
&= \frac{1}{2^{k-2}} \cdot \left(\frac{\binom{n-t}{2}\binom{t}{k-2}}{\binom{n}{k}}\right) \\
&= \frac{1}{2^{k-2}} \cdot \left(\binom{k}{2}\frac{n-t}{n}\frac{n-t-1}{n-1}\frac{t}{n-2}\frac{t-1}{n-3}\cdots\frac{t-k+3}{n-k+1}\right) \\
&> \frac{1}{2^{k-2}}\binom{k}{2}^2\frac{4}{5}\left(\frac{t}{2n}\right)^{k-2} \\
&> \binom{k}{2}\frac{4}{5}\left(\frac{3k}{2}\right)^{k-2}\frac{n}{m} \\
&> 8n/m.
\end{aligned}
$$

Using the binomial tail bound of Proposition 1 (1), it follows that $\Pr[|\hat{C}_2(F, \rho)| \le 2n] \le 2^{-4n}$. By Lemma 25 and the fact that the clauses in $\hat{C}_2(F, \rho)$ are distributed uniformly at random on the remaining $n' = n - t$

variables,

$$\Pr[\hat{C}_2(F,\rho) \text{ is satisfiable} : |\hat{C}_2(F,\rho)| > 2n] = o(2^{-n'/9}).$$

Since there are 2^t choices for ρ, the total failure probability is $2^t \cdot (o(2^{-(n-t)/9}) + 2^{-n})$, which is $o(1)$ since $(n-t)/9 \geq t$ for $t \leq n/10$.

□

Next we consider ordered DLL and prove the upper bound of Theorem 24. At a point in the execution of DLL, say that a variable is *critical* if setting that variable either to 0 or 1 and then applying unit propagation creates the empty clause. Thus, if the splitting rule chooses that variable the current branch will terminate simply by unit propagation.

A point in the execution of DLL corresponds to some restriction ρ. We give a sufficient condition for a variable to be critical in terms of the set $\hat{C}_2(F,\rho)$ of induced 2-clauses on the remaining set of n' variables. Define the standard directed graph $G(F,\rho)$ on $2n'$ vertices, one for each literal, that has directed edges $(\neg x, y)$, and $(\neg y, x)$ corresponding to each 2-clause $(x \vee y)$ in $\hat{C}_2(F,\rho)$. It is easy to see that a sufficient condition for the variable x_i to be critical is that there be directed paths from x_i to $\neg x_i$ and from $\neg x_i$ to x_i, i.e., that x_i and $\neg x_i$ lie in the same strongly connected component.

Lemma 27. *For any $k \geq 3$, there exists a constant c such that if $F \sim \mathcal{F}_m^{k,n}$ and ρ is a fixed restriction of t variables with $n/2 \geq t \geq c\lceil n(n/m)^{1/(k-2)}\rceil$, then with probability at least $1 - 2^{-n}$, for at least half of the $n' = n - t$ unrestricted variables, x_i and $\neg x_i$ belong to the same strongly connected component of $G(F,\rho)$.*

Proof. Clearly, it suffices to show that with probability at least $1 - 2^{-n}$, $G(F,\rho)$ has a strongly connected component of size at least $3n'/2$. Let $C_1, C_2, \ldots, C_d$ be the strongly connected components ordered so that all edges between components go from lower to higher numbered components, and consider the first j such that $|C_1 \cup \ldots \cup C_j| \geq n'/4$. We will show that the probability that $|C_j| < 3n'/2$ is at most 2^{-n}. If $|C_j| < 3n'/2$ then the set $S = C_1 \cup \ldots \cup C_j$ satisfies $n'/4 \leq |S| \leq 7n'/4$ and there is no edge from $\bar{S}$ to S.

So to upper bound the probability that $|C_j| < 3n'/2$ it suffices to upper bound the probability that there is a set S with $n'/4 \leq |S| \leq 7n'/4$ which is *bad* in the sense that there is no edge from $\bar{S}$ to S. Fix S of size s, with $n'/4 \leq s \leq 7n'/4$. The probability that a randomly chosen k-clause C, when restricted by ρ gives an edge from $\bar{S}$ to S is at least $s(n'-s-1)\binom{t}{k-2}/2^k\binom{n}{k} \geq \beta'(t/n)^{k-2} \geq \beta\lceil n/m\rceil$, for some constants $\beta, \beta' > 0$ depending only on k. Hence the probability that none of the m clauses of F gives such an edge is at most $(1-\beta n/m)^m \leq e^{-\beta cn} \leq 2^{-3n}$ for c chosen greater than $3/\beta$. There are at most $2^{2n'}$ such sets S, so the probability that there is a bad set S of size between $n'/4$ and $7n'/4$ is at most 2^{-n}. □

Proof of the upper bound of Theorem 24. Without loss of generality we may assume that $m \geq (4c)^{k-2}n$ where c is the constant of the previous lemma and let $t = cn(n/m)^{1/(k-2)}$, so that $t \leq n/4$.

Fix a restriction ρ of the first t variables. We claim that the probability that there is a branch of the DLL tree consistent with ρ that is still active (not terminated) after the first $4t$ variables are set and the resulting unit propagations are processed is at most 2^{-2t}. Since there are 2^t choices for ρ, this will imply that with probability $1 - 2^{-t}$, every branch of ordered DLL is completed after at most the first $4t$ variables are fixed and all resulting unit propagations are done, and so the tree has at most $n2^{4t}$ nodes (including nodes from unit propagation).

To prove the claim, condition on the size r of the set of critical variables for $F\lceil_\rho$. By lemma 27, the probability that $r < n'/2$ is at most $2^{-n} \leq 2^{-4t}$, so we assume $r \geq n'/2$. The set of critical variables is equally likely to be any r-subset of the $n' = n - t$ unset variables, and so the probability that none of the next

$3t$ variables in order are critical is at most $\binom{n'-3t}{r}/\binom{n'}{r} \leq (1-3t/n')^r \leq e^{-3t/2}$. Hence the probability that some branch consistent with ρ is unfinished after fixing the next $3t$ variables is at most $2^{-4t}+e^{-3t/2} \leq 2^{-2t}$.

Note that in order to obtain the claimed upper bound with probability $1-o(1)$ in n, we can assume without loss of generality that t is at least $\log n$. (If m, the number of clauses, is large enough so that t is less than $\log n$, we can carry out the analysis with fewer clauses since the complexity of ordered DLL decreases monotonically with the number of clauses.) $\square$

Lower bound for ordered DLL

We now complete the proof of Theorem 24 by proving the lower bound.

Proof of lower bound for Theorem 24. Fix $t < n(\frac{1}{4k\Delta})^{1/(k-2)}$ and let S be the first t variables with respect to the given ordering and $L(S)$ be the associated set of $2t$ literals. Let F' denote the set of all clauses of F that contain at least $k-1$ literals from $L(S)$.

Claim. With probability $1-o(1)$ in t, there is a partial assignment τ to $t/2$ of the variables of S that satisfies all clauses in F'.

Assuming the claim, we finish the proof by noting that for each of the at least $2^{t/2}$ restrictions ρ to S that are compatible with τ, all clauses of $F\lceil_\rho$ will have size at least 2. This implies that when applying ordered DLL along the path specified by ρ, no variables outside of S are fixed by unit propagation, and so there is a unique node corresponding to ρ and hence there are at least $2^{t/2}$ nodes in the tree. Without loss of generality, we can assume that t is at least $\log n$, since $t < \log n$, our claimed lower bound is just 1.

So we prove the claim. For each $C \in F$, the probability q that it is in F' is at most $k(\frac{t}{n})^{k-1} \leq t\frac{k}{n}(\frac{t}{n})^{k-2} \leq \frac{t}{4m}$. Construct a 2-CNF F'' of size $|F'|$ by replacing each clause C' of F' by a clause C'' obtained by selecting two literals of $C' \cap L(S)$ uniformly at random. It is easy to see that F'' is a randomly chosen 2-CNF whose number of clauses is binomially distributed according to $B(m,q)$. The tail bound of Proposition 1 (2) implies that $|F''| < t/2$ with probability $1-o(1)$. Conditioned on $|F''| < t/2$, the probability that F'' (and hence F') is satisfiable is $1-o(1)$ because a random 2-CNF formula with $(1-\epsilon)t$ variables on a set of size t is satisfiable with probability $1-o(1)$ [Goe96, CR92]. If F'' is satisfiable, there is a setting τ of at most $|F''| < t/2$ variables of S that satisfies it. This completes the proof of the claim and the theorem. $\square$

6.2 Lower bounds for more general DLL procedures

Having analyzed ordered DLL, we next turn to the problem of proving lower bounds for a wider class of DLL procedures. In the following discussion it will be useful to introduce some additional terminology. Let A be a DLL algorithm and F be a formula, and let $T_A(F)$ denote the DLL tree associated to the execution of A on F. We classify the nodes of $T_A(F)$ into two types: *unit propagation nodes* (those corresponding to a variable in a unit clause) and *branching nodes*.

We will prove a lower bound for a class of algorithms called *oblivious DLL* algorithms. Let B_n denote the full binary tree of depth n and let λ be a labeling of the internal nodes by literals with the property that along each root-leaf path each variable occurs in exactly one literal. The labeling λ specifies an algorithm $A = A_\lambda$ as follows. On input F, A_λ recursively traverses B_n starting from the root. When arriving at node v it has a formula G which is a restriction of F, and it performs a procedure Test(G, v) defined as follows. While G has at least one unit clause but no empty clause, choose a unit clause and fix its literal to true, simplifying G accordingly. If the empty clause is produced, stop, the formula is unsatisfiable. If G ever has no clauses, then stop, the formula is satisfiable. Otherwise, when G has no empty or unit clauses, let v_0 and v_1 denote the left and right children of v. If the literal $\lambda(v)$ is already assigned a value $i \in \{0, 1\}$, move to v_i and run Test(v_i, G). Otherwise, run Test$(v_0, G\lceil_{\lambda(v)=0})$ and Test$(v_1, F\lceil_{\lambda(v)=1})$ and return unsatisfiable if both return unsatisfiable, and satisfiable if at least one of them returns satisfiable.

In the special case that B_n is labeled so that each node of distance i from the root is labeled x_i the above algorithm is ordered DLL. Another case of interest is that of *random DLL* in which the labeling of B_n is chosen at random. In general, we call such algorithms *oblivious* because (except for unit propagation) the choice of splitter does not depend on the function F.

It is worth emphasizing the distinction between the labeled tree (B_n, λ) (which is independent of F) and the DLL tree $T_{A_\lambda}(F)$ associated to an execution of the algorithm on F.

It is not hard to show that if $F \sim \mathcal{F}_m^{k,n}$, then the expected running time of any oblivious algorithm is the same. However, this does not rule out the possibility that some oblivious algorithm may run much faster than ordered DLL on *most* instances, by concentrating the bad behavior on a small set of instances. Here we show that provided that m is big enough, no oblivious algorithm can do much better than ordered DLL on most instances (see Theorem 31 below).

In the lower bound on ordered DLL, we showed that during the first t steps, with high probability unit propagation plays no role. We will prove something like this for oblivious algorithms. The key step is a lemma that implies that for any DLL procedure (oblivious or not) on a random formula, the number of variables fixed by unit propagation along any path in the DLL tree is not much more than the number of branching nodes along the path. For technical reasons, it is easier to consider a generalization of unit propagation that ignores the sign of variables.

The general idea is as follows. Let F be a formula and T a set of variables that have been set so far. We want to describe a natural algorithm that defines a set $\hat{T}$, where $\hat{T}$ will contain T plus the set of additional variables that are set by unit propagation. The algorithm is as follows. Initially $\hat{T} = T$. Given F, let S be the corresponding set system, where each k-clause of F corresponds to a k-set in S (over a universe of size n.) Given F (and hence S), and T, define $S' \subseteq S$ to be those clauses that intersect $\hat{T}$ in exactly $k-1$ elements. Add the elements occurring in S' that are not already in $\hat{T}$ to $\hat{T}$ and continue recursively until S' is empty. Note that the set $\hat{T}$ produced by this algorithm is a minimal set of variables that will be set by setting the variables in T, plus the additional variables set by unit-clause propagation, assuming that we ignore early termination as a result of the formula being set to 0 or 1. The intuition behind the proof of Theorem 11 is that with high probability, the size of $\hat{T}$ will be not much larger than the size of T. The proof of this fact will be a compression argument showing that for each fixed T, if F has the property that $\hat{T}$ is large, then F can be encoded succinctly. Intuitively, if there are a lot of variables that become unit clauses, then these clauses are not random with respect to T, and therefore can be described succinctly.

We need some definitions. Given a k-CNF formula F, a set T of variables is *closed* with respect to F if no clause contains exactly $k-1$ variables of T (either positively or negatively). It is easy to see that the intersection of closed sets is closed, and hence for any set T of variables the set $\hat{T}$ obtained by intersecting all closed sets containing T yields the unique minimal closed set containing T. We call this the *closure of T (with respect to F)*. A clause is said to be *threatened* by (F, T) if it is contained in $\hat{T}$. Note that $\hat{T} = T \cup \cup_D v(D)$ where D ranges over all clauses of F threatened by (F, T). The following fact relates these notions to unit propagation:

Proposition 28. *Let A be any DLL algorithm and F be a k-CNF formula. Suppose v is a node in the DLL tree of $T_A(F)$. Let T be the variables that appear at the branching nodes on the path to v. Then*

1. *The variables at the unit propagation nodes on the path to v are contained in $\hat{T}$.*

2. *The only clauses that can be empty upon reaching v are the clauses threatened by (F, T).*

Proof. For the first part, let $v_0, \ldots, v_j = v$ be the nodes on the path to v and let x_i be the variable whose literal l_i appears at v_i. Applying induction on i, we assume that $\{x_0, \ldots, x_{i-1}\} \subseteq \hat{T}$. Then if v_i is a branching node then $x_i \in T$ and otherwise, if ρ is the restriction defined by the literals $l_0, \ldots, l_{i-1}$, there is a clause C of F such that $C\lceil_\rho$ is the unit clause l_i. If $\hat{T}$ does not contain x_i then C contains exactly $k-1$ variables of $\hat{T}$, which contradicts that $\hat{T}$ is closed.

For the second part, a clause that becomes empty must have all its literals set to 0, which means that all of its variables are in $\hat{T}$. ☐

We next present an algorithm for constructing $\hat{T}$ from T; the details of this algorithm are needed in what follows. Fix an ordering $x_1, \ldots, x_n$ of the variables. The algorithm $Close(F, T)$ takes as input a formula F (viewed as a list $C_1, \ldots, C_m$ of clauses) and subset T of variables with $t = |T|$, and outputs

- A sequence $D_1, \ldots, D_u$ of distinct clauses of F, consisting of the set of clauses threatened by (F, T),
- A sequence $z_1, \ldots, z_{t+u}$ of variables consisting of the variables of $\hat{T}$ (possibly with repetition). Furthermore, $T = \{z_1, \ldots, z_t\}$, listed according to the global variable ordering, and for $i \in [u]$, $z_{t+i} \in v(D_i) \subseteq \{z_1, \ldots, z_{t+i}\}$.
- A sequence $j_1 \leq \cdots \leq j_u$ of integers in the range 1 to $t+u$, where j_i is the least index such that $|\{z_1, \ldots, z_{t+u}\} \cup D_i| = k-1$.

We will build these three lists in a series of iterations, which we divide into two phases. The initialization phase consists of the first t iterations, where we place the variables of T on the list, constructing $z_1, \ldots, z_t$. During the second (main) phase, in iteration $t+i$, we determine z_{t+i}, D_i and j_i. During both phases, we maintain a list of *eligible clauses* of F. Each item on the list is a triple $(D, j(D), y(D))$ where D is a clause that contains at least $k-1$ variables on the list so far, $j(D)$ is the least index for which $|v(D) \cap \{z_1, \ldots, z_{j(D)}\}| = k-1$, and $y(D)$ is the unique variable of $v(D) - \{z_1, \ldots, z_{j(D)}\}$. At iteration j, after adding a variable z_j, the eligible list is updated by identifying all clauses C_s of F that contain z_j and that contain exactly $k-2$ distinct variables from $z_1, \ldots, z_{j-1}$. For each such clause, let $j(C_s) = j$ and let $y(C_s)$ be the unique variable of C_s that does not appear in the list. Order the set of all such triples $(C_s, j, y(C_s))$ according to the index s and append this to the list of eligible clauses.

In the initialization phase, during iteration i, we choose z_i to be the ith variable of T according to the global variable ordering, and then we update the eligible list. During iteration $t+i$ of the main phase, if there are no eligible clauses then the algorithm terminates. Otherwise, remove the first triple (D, j, y) from the eligible list and set $z_{t+i} = y$, $D_i = D$ and $j_i = j$ and update the eligible list.

It is easy to show by induction on i that $\{z_1, \ldots, z_{t+i}\} \subseteq \hat{T}$, and that at termination the list $\{z_1, \ldots, z_{t+u}\}$ is closed and hence is $\hat{T}$. The other properties of the output asserted above are similarly obvious.

We are now ready to state and prove the key lemma.

Lemma 29. *For $k \geq 3$ there is a constant $c(k)$ such that the following holds. Let n, m, t, w be positive integers and let $\Delta = m/n > 1$. Suppose that $t \leq w \leq c(k)n/\Delta^{1/(k-2)}$. Let T be a set of variables of size t. Then for $F \sim \mathcal{F}_m^{k,n}$, the probability that (F, T) threatens at least w clauses is at most 2^{-w}.*

Proof. Fix n, m, w, t and T as in the hypothesis of the lemma. We view an arbitrary formula $F \sim \mathcal{F}_m^{k,n}$ as an ordered sequence $C_1, \ldots, C_m$ of clauses. Thus F is uniformly chosen from $(2^k\binom{n}{k})^m$ possible formulas.

Say that F is *bad* if (F, T) threatens at least w clauses. We will upper bound the probability that F is bad by showing that each bad F can be uniquely "encoded" in such a way that the number of encodings is a 2^{-w} fraction of the number of formulas.

Now suppose that F is bad, and thus the number u of threatened clauses is at least w. In this case, will show how to encode F efficiently. Our encoding will first give enough information in order to to recover the first w clauses, $D_1, \ldots, D_w$, output by the above algorithm, and then we will code more directly the remaining clauses of F. The encoding of $D_1, \ldots, D_w$ refers to some of the output of the above algorithm on F, and will include the following:

- The list $z_{t+1}, \ldots, z_{t+w}$,
- The list j_1,$j_2, \ldots$,j_w,
- For $i \in [w]$, the sequence $d_i(1) < \ldots < d_i(k-2)$ where $d_i(j)$ is the least index such that in $|v(D_i) \cap \{z_1, \ldots, z_{d_i(j)}\}| = j$.
- For each $i \in [w]$, the vector $\alpha_i \in \{+, -\}^k$, where $\alpha_i(j)$ is the sign of the appearance of $z_{d_i(j)}$ in D_i.

It is clear that this information is enough to reconstruct $D_1, \ldots, D_w$. To reconstruct the remaining clauses of F, $C_1, \ldots, C_m$, along with their ordering, we need two more things:

- The list $h_1, h_2, \ldots, h_w$ of indices such that $D_i = C_{h_i}$,
- The ordered list of clauses $(C_i : i \notin \{h_1, \ldots, h_w\})$,

Let us count the number of such encodings. There are at most m^w ways to choose $h_1, \ldots, h_w$. There are $(2^k\binom{n}{k})^{m-w}$ ways to choose the sequence of clauses that are not among the D_i. There are at most n^w ways to choose $z_{t+1}, \ldots, z_{t+w}$. Since the list $j_1, \ldots, j_w$ satisfies $j_1 \leq \cdots \leq j_w$, the number of ways to choose this sequence is at most $\binom{t+2w-1}{w} \leq 2^{t+2w}$. For each i the number of ways to choose $d_i(1), \ldots, d_i(k-2)$ is at most $(t+w)^{k-2}$ and the number of ways to choose α_i is at most 2^k. Multiplying these together and dividing by $(2^k\binom{n}{k})^m$ we get that the probability that (F, T) threatens at least w clauses is at most:

$$\begin{aligned} \frac{m^w n^w 2^{t+2w+kw}(t+w)^{(k-2)w}}{(2^k\binom{n}{k})^w} &\leq \frac{m^w k^w 2^{t+2w}(t+w)^{(k-2)w}}{n^{(k-1)w}} \\ &\leq \left(\frac{8km(2w)^{k-2}}{n^{k-1}}\right)^w, \end{aligned}$$

For some constant $c(k) > 0$, if $w \leq c(k)n(n/m)^{1/(k-2)}$ the above expression is at most 2^{-w}. □

Before we use this to prove the theorem, we derive an immediate corollary which we hope will be useful in analyzing other DLL algorithms. It says that for almost all formulas F every path in any DLL tree for F contains not many more unit propagation nodes than it does branching nodes.

Lemma 30. *For $k \geq 3$ there is a constant $c(k)$ such that the following holds. Let n, m, t be positive integers, $\epsilon > 0$ $w = \lceil (1+\epsilon)\max(t, \log_2\binom{n}{t})\rceil$ and let $\Delta = m/n > 1$. Suppose that $t \leq w \leq c(k)n/\Delta^{1/(k-2)}$. Then for $F \sim \mathcal{F}_m^{k,n}$, the probability that there is a partial assignment of size t that generates at least w unit clauses is $o(1)$ in t.*

Proof. There are only $\binom{n}{t}$ sets of variables T of size t that can be the underlying set of variables of the partial assignment. Furthermore, any unit clause generated by such a partial assignment must be a clause threatened by (F, T). By Lemma 29, any single set T threatens w clauses with probability only 2^{-w}. Summing over all possible sets T yields the desired result. $\square$

Theorem 31. *Let $k \geq 3$ and let n, m be integers and $\Delta = m/n$. Let A be any oblivious DLL algorithm for k-CNF formulas on n variables and suppose $F \sim \mathcal{F}_m^{k,n}$. If $m = \omega(n^{k/2})$ then with probability $1 - o(1)$ in n, the running time of A on F is $2^{\Omega(n/\Delta^{1/(k-2)})}$.*

Proof of Theorem 31. Fix k and let n, m satisfy the hypotheses of the theorem. There are two cases to consider, when F is satisfiable and when F is not satisfiable. Since $m = \omega(n^{k/2})$, the probability that F is satisfiable is exponentially small in n. Thus, we can assume without loss of generality that F is unsatisfiable in what follows, and in particular, every leaf node in any DLL tree for F is labeled by 0. Observe that the conclusion of the theorem is trivial if $m = \Omega(n^{k-1})$ so we may assume that $m = o(n^{k-1})$. Fix an oblivious DLL algorithm and let λ be the associated labeling of B_n. Let $t = c(k)n/\Delta^{1/(k-2)}$ where $c(k)$ is as in Lemma 29. The hypothesis and the assumption $m = o(n^{k-1})$ imply $t = o(\sqrt{n})$ and $t = \omega(1)$.

For a formula F, let $R = R(F)$ be the set of vertices at level t in B_n that are visited by the traversal of B_n in the execution $A_\lambda(F)$. Note that $v \in R$ means that no clause of F becomes empty along the path to v. We will show that the probability that $|R| \leq 2^{t-1}$ is $o(1)$, and since $|R|$ is a lower bound on the running time of $A_\lambda(F)$ this will prove the theorem.

First, for a fixed vertex v at level t in B_n we upper bound $\Pr[v \notin R]$. Let T be the set of variables labeling the nodes in B_n on the path to v. By the definition of the algorithm and by Proposition 28 the variables set by unit propagation along the path to v must be in $\hat{T}$ and any empty clause must be a clause threatened by (F, T). Consider the output $D_1, \ldots, D_u$ and $z_1, \ldots, z_{t+u}$ of the algorithm $Close(F, T)$. We claim that if the variables $z_1, \ldots, z_{t+u}$ are all distinct, then no clause of F becomes empty along the path to v, and since F is unsatisfiable, this implies that $A_\lambda(F)$ reaches v. For this, it suffices to show that for $i \in [u]$, if the variable z_{t+i} is set by unit propagation then it can only be set because D_i becomes a unit clause. So assume for contradiction that z_{t+i} is set because some clause D_j with $j \neq i$ becomes a unit clause, and that this is the first time that this happens. But then since $z_{t+j} \in v(D_j)$, it must have been set before D_j became a unit clause, which contradicts the choice of z_{t+i}.

Thus we have:

$$\begin{aligned}\Pr[v \notin R] &\leq \Pr[z_1, \ldots, z_{t+u} \text{ are not distinct}] \\ &\leq \Pr[u > t] + \Pr[z_1, \ldots, z_{t+u} \text{ are not distinct} : u \leq t].\end{aligned}$$

Now $\Pr[u > t] \leq 2^{-t}$ by Lemma 29. We claim that the second term is bounded above by $3t^2/2n$. To see see this, note that by the definition of the algorithm $Close$ the triple (D_i, j_i, z_{t+i}) is the ith item placed on the eligible clause list. Think of this triple as a random variable (which depends on the random formula F). The key observation is that at the time that we add (D_i, j_i, z_{t+i}) to the eligible list (at the end of iteration j_i) the conditional distribution of z_{t+i} is given by the uniform distribution over the set $\{x_1, \ldots, x_n\} - \{z_1, \ldots, z_{j_i}\}$. Thus the probability that $z_{t+i} \notin \{z_1, \ldots, z_{t+i-1}\}$ is $1 - (t + i - 1 - j_i)/(n - j_i) \geq 1 - (t + i - 1)/n$, and the probability that there is some $i \in [t]$ for which $z_{t+i} \in \{z_1, \ldots, z_{t+i-1}\}$ is at most $\sum_{i=1}^{t}(t + i - 1)/n \leq \frac{3t^2}{2n}$, as claimed.

We conclude that $\Pr[v \notin R] \leq 2^{-t} + \frac{3t^2}{n}$, which is $o(1)$ in n since $t = o(\sqrt{n})$ and $t = \omega(1)$. Therefore, the expected number of vertices $v \notin R$ is at most $o(2^t)$, which under the assumption on t is $o(2^t)$. Thus, by

Markov's inequality, the probability that more than 2^{t-1} vertices are not in R is $o(1)$ and thus $|R| \geq 2^{t-1}$ with probability $1 - o(1)$, as required to complete the proof. □

7 Related Work and Further Research

The question of whether or not resolution is automatizable is still open. In particular, what is the fastest deterministic or probabilistic search algorithm for resolution? The best that is known is presented in section 3, which is essentially due to Clegg, Edmonds and Impagliazzo. It can be shown that this is the best algorithm in the tree-like case, but we know of no negative results of this kind for the general case.

A problem pertinent to section 4 is to prove exponential lower bounds for the weak pigeonhole principle, $\neg PHP_n^m$, where the number of pigeons, m, is large (say n^3). Buss and Pitassi [BP97] showed that there exists a resolution refutation of $\neg PHP_n^m$ when $m \geq 2^{\sqrt{n}\log n}$ of size $2^{\sqrt{n}\log n}$. It has been conjectured that when m is polynomial in n, any resolution refutation of $\neg PHP_n^m$ requires super-polynomial size. Razborov has shown that such a lower bound would imply that proving super-polynomial circuit lower bounds for an explicit function in NP is independent of certain systems of bounded arithmetic. [2]

In a preliminary version of this paper ([BP96]) we asked:

> Can one show that for any 3-CNF formula f, if f has a polynomial-size resolution refutation, then f also has a resolution refutation with maximum clause size $\sqrt{n}$? Such a result would justify the simple and natural deterministic simulation of resolution whereby we exhaustively search for proofs of maximum clause length i, for increasing i.

Based on the Clegg-Edmonds-Impagliazzo algorithm, a version of this clause-width conjecture has recently been proven by Ben-Sasson and Wigderson [BSW99] – in particular that size S resolution refutations can be made $O(\sqrt{n \log S})$-bounded and size S DLL proofs can be converted to $O(\log S)$-bounded resolution refutations. As stated above, this simplifies the algorithm of section 3 and can be used to show that random k-CNF formulas with n variables with clause density $\Delta \geq \theta_k(n)$ require resolution proofs of size $2^{\Omega(n/\Delta^{4/(k-2)+\epsilon})}$ and DLL proofs of size $2^{\Omega(n/\Delta^{2/(k-2)+\epsilon})}$. This size lower bound for general resolution refutations provides a simpler smooth tradeoff between Δ and proof size than our results, but it does not improve the largest densities for which exponential lower bounds for resolution are known to hold almost certainly. The size lower bound for DLL proofs is not as large as our lower bounds in section 6.1 but applies to all DLL procedures, not just oblivious ones.

The problem posed at the end of section 5.2 is whether the hypothesis on s in Lemma 11 needed for property $B_\epsilon(s)$ can be weakened. As mentioned, such a weakening would strengthen the lower bounds on res(F) for random formulas. Furthermore, when combined with the Ben Sasson-Wigderson approach for lower bounding DLL refutation size, this improvement would yield a *tight* $2^{\Omega(n/\Delta^{1/(k-2)+o(1)})}$ lower bound on DLL(F) for F a random formula.[3]

Examining the proof of Lemma 17 we see that a critical place to look for improvement is in the last step of the proof of Proposition 18, where we obtain an upper bound on Pr$[Q(r,q)]$ by multiplying Pr$[B(m,p) \geq q]$ by $\binom{n}{r}$, which corresponds to upper bounding the probability of the union of $\binom{n}{r}$ events by their sum. Can this union bound be refined by a more careful analysis?

[2] Following a proof of subexponential lower bounds for a restricted version of resolution [PR01], such lower bounds have indeed been proved for general resolution by Raz [Raz01a] and simplified and strengthened by Razborov [Raz01b].

[3] Ben-Sasson and Galesi [BSG01] have recently shown just such a lower bound by replacing the property $B_\epsilon(s)$ with a weaker expansion property of such formulas.

The essence of the question is captured by a natural question about random hypergraphs. Given a collection $\mathcal{H}$ of subsets of $[n]$, say that a subset A of $[n]$ is γ-crowded by $\mathcal{H}$ if A contains at least γn members of $\mathcal{H}$. Let $s(\mathcal{H}, \gamma)$ be the largest s for which there is no set of size s that is γ-crowded by $\mathcal{H}$. The question is: if $\mathcal{H}$ is a random collection consisting of Δn subsets of $[n]$ each of size k what is the best lower bound on $s(\mathcal{H}, \gamma)$ that holds with probability $1 - o(1)$. A calculation analogous to that in Lemma 17 (using the union bound) shows that $s(\mathcal{H}, \gamma) \geq \Omega(n/\Delta^{1/(k-2-\frac{1}{\gamma})})$ while it is also not hard to show that that $s(\mathcal{H}, \gamma) \leq O(n/\Delta^{1/(k-2)})$. An argument that the upper bound is tight, could almost certainly be adapted to give an affirmative answer to the problem at the end of section 5.2.

A final open problem is to produce a better algorithm for finding unsatisfiability proofs for random formulas. In particular, is there a polynomial-time algorithm that succeeds in finding a proof of unsatisfiability with high probability for random formulas with cn clauses, for some $c > 0$? We are not aware of *any* algorithm that provably beats the very simple ones we analyzed in section 7, even if we consider more powerful search methods that are not resolution-based.[4]

References

[AS00] D. Achlioptas and G. Sorkin. Optimal myopic algorithms for random 3-SAT. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 590–600, Redondo Beach, CA, November 2000. IEEE.

[ASE92] Noga Alon, J. Spencer, and P. Erdös. *The Probabilistic Method*. John Wiley & Sons, 1992.

[AV79] D. Angluin and L. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.

[BFU93] A. Broder, A. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-CNF formulas. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1993.

[BKPS98] 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.

[BP96] Paul W. Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proceedings 37th Annual Symposium on Foundations of Computer Science*, pages 274–282, Burlington, VT, October 1996. IEEE.

[BP97] S. Buss and T. Pitassi. Resolution and the weak pigeonhole principle. In *Proceedings of Computer Science Logic*, Lecture Notes in Computer Science. Springer-Verlag, 1997.

[BPR97] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *Journal of Symbolic Logic*, 62(3):708–728, September 1997.

[BSG01] E. Ben-Sasson and N. Galesi. Space complexity of random formulae in resolution. In *ProceedingsSixteenth Annual IEEE Conference on Computational Complexity*, pages 42–51, Chicago, IL, June 2001.

[4]Incomplete methods using spectral analysis of graphs associated with these formulas have subsequently been shown to yield polynomial-time proofs of unsatisfiability almost certainly for formulas at much lower densities than those we prove for resolution-based algorithms [GK01, FG01] but the question remains open for constant density.

[BSW99] 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.

[BT88] S. Buss and G. Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–317, 1988.

[CA96] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence*, 81, 1996.

[CEI96] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Gröbner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 174–183, Philadelphia, PA, May 1996.

[CF90] M.T. Chao and J. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics. *Information Science*, 51:289–314, 1990.

[CR77] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1977.

[CR92] 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.

[CS88] V. Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[FG01] J. Friedman and A. Goerdt. Recognizing more unsatisfiable random 3-SAT instances efficiently. In J. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages, and Programming: 28th International Colloquium*, volume 2076 of *Lecture Notes in Computer Science*, pages 310–321, Heraklion, Crete, July 2001. Springer-Verlag.

[FP83] J. Franco and M. Paull. Probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[Fri99] E. Friedgut. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, 12:1017–1054, 1999.

[FS96] 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.

[Fu95] Xudong Fu. *On the complexity of proof systems*. PhD thesis, University of Toronto, 1995.

[GK01] A. Goerdt and M. Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *(STACS) 2001: 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 294–304, Dresden, Germany, February 2001. Springer-Verlag.

[Goe96] A. Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences*, 53:469–486, 1996.

[GPFW97] J. Gu, P. W. Purdom, J. Franco, and B. J. Wah. Algorithms for the Satisfiability (SAT) Problem: A Survey. In *Satisfiability (SAT) Problem*, DIMACS, pages 19–151. American Mathematical Society, 1997.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–305, 1985.

[JSV00] S. Janson, Y. C. Stamatiou, and M. Vamvakari. Bounding the unsatisfiability threshold of random 3-SAT. *Random Structures and Algorithms*, 17(2):103–116, 2000.

[KKKS98] L. M. Kirousis, E. Kranakis, D. Krizanc, and Y. C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures and Algorithms*, 12(3):253–269, 1998.

[KS94] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random formulas. *Science*, 264:1297–1301, May 1994.

[PR01] T. Pitassi and R. Raz. Lower bounds for regular resolution proofs of the weak pigeonhole principle. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 347–355, Hersonissos, Crete, Greece, July 2001.

[Raz01a] R. Raz. Resolution lower bounds for the weak pigeonhole principle. Technical Report TR01-021, Electronic Colloquium in Computation Complexity, `http://www.eccc.uni-trier.de/eccc/`, 2001.

[Raz01b] A. A. Razborov. Improved resolution lower bounds for the weak pigeonhole principle. Technical Report TR01-055, Electronic Colloquium in Computation Complexity, `http://www.eccc.uni-trier.de/eccc/`, 2001.

[SML96] B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.

[Urq87] A. Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, 1987.