An Improved Exponential-time Algorithm for k-SAT

Ramamohan Paturi* Pavel Pudlák† Michael E. Saks‡ Francis Zane§

May 31, 2004

Abstract

We propose and analyze a simple new randomized algorithm, called ResolveSat, for finding satisfying assignments of Boolean formulas in conjunctive normal form. The algorithm consists of two stages: a preprocessing stage in which resolution is applied to enlarge the set of clauses of the formula, followed by a search stage that uses a simple randomized greedy procedure to look for a satisfying assignment. Currently, this is the fastest known probabilistic algorithm for k–CNF satisfiability for $k \geq 4$ (with a running time of $O(2^{0.5625n})$ for 4–CNF). In addition, it is the fastest known probabilistic algorithm for k–CNF, $k \geq 3$, that have at most one satisfying assignment (unique k-SAT) (with a running time $O(2^{(2\ln 2-1)n+o(n)}) = O(2^{0.386\ldots n})$ in the case of 3–CNF). The analysis of the algorithm also gives an upper bound on the number of the codewords of a code defined by a k-CNF. This is applied to prove a lower bounds on depth 3 circuits accepting codes with nonconstant distance. In particular we prove a lower bound $\Omega(2^{1.282\ldots\sqrt{n}})$ for an explicitly given boolean function of n variables. This is the first such lower bound that is asymptotically bigger than $2^{\sqrt{n}+o(\sqrt{n})}$.

1 Introduction

1.1 k-CNF satisfiability

Given that the problem of deciding whether a given k–CNF formula is satisfiable is NP-complete for $k \geq 3$, it is natural to look for algorithms that improve significantly on the worst case running time of the naive exhaustive search algorithm, which is $\mathrm{poly}(n)2^n$ for a formula on n variables. Monien and Speckenmeyer [15] gave the first real improvement by giving a simple algorithm whose running time is $O(2^{(1-\varepsilon_k)n})$, with $\varepsilon_k > 0$ for all k. In a sequence of results, algorithms with increasingly better running times (larger values of ε_k) have been proposed and analyzed. Most of these concentrated on the case of 3–CNF. Prior to the introduction of probabilistic algorithms, the fastest algorithm in this field was Kullmann's [12], which achieved $O(2^{0.589n})$ for 3–CNF satisfiability.

All of these algorithms are backtrack search algorithms of a type originally proposed by Davis, Logemann and Loveland [4] (which are sometimes called Davis-Putnam procedures). Such algorithms search for a satisfying assignment by assigning values to variables one by one (in some order), backtracking if a clause is made false. In 1997 Paturi, Pudlák and Zane [17] proposed a randomized algorithm for k–SAT. This

*University of California, San Diego; This research is supported by NSF grant CCR-9734911 from Theory of Computing Program

†Mathematical Institute ČSAV, Žitna 25, Praha 1, Czech Republic. Supported by grants no. A1019901 of the Academy of Sciences of the Czech Republic and no. LN0056 of the Ministry of Education of the Czech Republic

‡Research supported by NSF grant CCR-9700239. Department of Mathematics–Hill Center, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854. saks@math.rutgers.edu.This work was done while on sabbatical at University of Washington.

§Bell Laboratories, Lucent Technologies

algorithm is very simple and can be viewed as doing a very limited form of backtracking: if the algorithm reaches a point where it must backtrack twice in succesion then it restarts. The algorithm is not faster than the previous ones for $k = 3$, but it is for larger values of k.

In this paper we present an algorithm that is a significant improvement of Paturi, Pudlák and Zane's algorithm. When it was announced in the preliminary version of this paper [16], it was the fastest algorithm for all values of $k \geq 3$. Soon after, Schöning [22] discovered another algorithm, based on random walk on the boolean cube. His algorithm beats ours for $k = 3$, but is still worse for $k \geq 4$. Further improvements of his algorithm were found quite recently [11]. Both Paturi, Pudlák and Zane's algorithm and Schöning's algorithms have been derandomized, see [17, 3], but the deterministic versions perform significantly worse. Other work has investigated related problems, such as solving satisfiability of general CNF formulae, proving bounds on parameters such as formula length and number of literals [9].

Our algorithm is very simple. Given a k-CNF formula, we first generate clauses that can be obtained by Resolution without exceeding a certain clause length. Then we take a random order of variables and gradually assign values to them in this order. If a variable currently considered occurs in a unit clause, we assign the value so that the clause is satisfied. If it occurs in contradictory unit clauses we start over. At each step we also check if the formula is satisfied and if it is satisfied, we accept. This subroutine is repeated until a satisfying assignment is found, or a given time limit is exceeded. (The algorithm is defined precisely below.)

First we analyze the algorithm on formulas that have a unique satisfying assignment (more precisely, an assignment that has large Hamming distance from all others). As for the previous algorithm of Paturi, Pudlák and Zane, this property of the set of satisfying assignments causes frequent occurrence of unit clauses in the process, thus the probability that a satisfying assignment is found is high. The preprocessing by Resolution has the effect that there are more clauses that can be reduced to a unit clause, thus there are more variables the value of which is forced. The resulting bound is better than the running time for Schöning's algorithm, so our algorithm outperforms Schöning algorithm for formulas with a unique satisfying assignment.

If a satisfying assignment does have close neighbors, it is not true that values of many variables must be forced when the algorithm happens to guess this satisfying assignment. Hence the analysis of general k–CNFs is based on the idea that the lower probability that a particular assignment is found is compensated by having more satisfying assignments. The analysis is much more complicated and we do not get as good bounds as for the case of uniquely satisfiable formulas. Namely, the bounds are worse for $k = 3$ and 4, for $k > 4$ we get asymptotically the same bound. Our bounds for unique–k–SAT and k–SAT, for $k = 3, 4, 5, 6$ are shown in Table 1.1, where we also compare them with the corresponding bounds of [22, 11].

In a preliminary version of the paper, which appeared before Schöning did his work, we presented analysis to show that the exponent for the general $k = 3$ case was at most $2^{0.446n}$. In contrast with the analysis for the cases $k >= 4$, this analysis is somewhat tedious and we have omitted it from this version for a few reasons. Firstly, the bounds we obtain from this analysis are not as good as those obtained in [11] and [22]. Secondly, we believe that it should be possible to show for $k = 3$ and $k = 4$ that the bounds for the unique SAT case extend to the general case (as we do for $k \geq 5$), and thereby show that the algorithm we give here has better worst case bounds than those of [11] and [22]. Our existing analysis to obtain numerical bounds does not seem useful towards this goal.

Our main aim was to find an asymptotically fastest algorithm, however we believe that, due to its simplicity, the algorithm is interesting also for practical applications. The preprocessing phase in which new clauses are generated causes larger space complexity of this algorithm if compared with [17, 22]. But in practical applications it is better to give up a little of the time efficiency and use a stricter bound on the size of clauses. It turns out that the biggest marginal gain in time efficiency is obtained for k–CNF's when we use only resolution up to clauses of length $2k - 1$, and performing further resolution yields diminishing returns.

k	unique-k–SAT	general k–SAT	[22]	[11]
3	0.386...	0.521...	0.415...	0.410...
4	0.554...	0.56249...	0.584...	
5	0.650...		0.678...	
6	0.711...		0.736...	

Table 1: The exponent c in the bound $2^{cn-o(n)}$ of our algorithm for unique-k–SAT, for k–SAT and the corresponding bounds for Schöning's algorithm [22] and its improved version for 3–SAT [11].

Furthermore, one can use other ideas to improve the practical performance such as assigning the value to all unit clauses that appear, or resolving after each single assignment to a variable. These improvements have not been analyzed theoretically yet, but an algorithm that is based on a combination of ours and Schöning's has been tested on a number of benchmarks with quite promising results [10].

1.2 Lower bounds on depth 3 circuits

When starting this research our objective was to prove better lower bounds for bounded depth circuits. It is quite surprising that our techniques can be used both for upper bounds on the running time of algorithms and for lower bounds on the size of certain circuits. In particular, the techniques provide interesting information on the sets of satisfying assignments of k-CNF formulas. We prove an upper bound on the number of code words of an error correcting code accepted by a k-CNF formula. As in [17], this implies lower bounds on the size of circuits computing such explicit codes. Our bounds pertain to the restricted circuit classes Σ^3, the set of unbounded fan-in depth-3 circuits whose output gate is an OR gate, and Σ^3_k, the set of Σ^3 circuits with bottom fan-in (ie, the fan-in of gates closest to the inputs) at most k. Circuits of this form have been studied extensively. One motivation for studying such circuits is that sufficiently strong lower bounds on Σ^3 circuits would resolve other long-standing questions. Namely, using a technique of Valiant [23], it can be shown that a lower bound of $2^{n/\log\log n}$ on $\Sigma^3_{n^\epsilon}$ circuits would imply nonlinear lower bounds on the size of constant fan-in, logarithmic depth circuits.

Previously, the strongest lower bounds on the size of such circuits for explicitly functions were proven for the parity function: Σ^3 circuits of size $\Theta(n^{1/4}2^{\sqrt{n}})$ and Σ^3_k circuits of size $2^{n/k+o(n)}$ are necessary and sufficient to compute parity [17]. Here, we obtain better bounds $2^{c\sqrt{n}-o(n)}$ for the membership function for an error correcting code, where $c = \pi/\sqrt{6} = 1.282\ldots$. This is the first lower bound for an explicitly given boolean function with the constant in the exponent $c > 1$. Although these lower bounds are only slightly better, they have the significant property that $\frac{f \log_2 S}{n} > 1$ where f is the bottom fan-in of the circuit and S is the size of the circuit. This is interesting, because proving $\frac{f \log_2 S}{n} > \omega(1)$ would give a nonlinear lower bound for series-parallel circuits [23]. Earlier lower bound techniques [6, 20, 7] do not seem to be able to give such lower bounds. Note that one can get easily an explicit boolean function that has the same asymptotic complexity for both Σ^3 and Π^3 circuits by combining these functions with their negations. It is also interesting to note that we get a polynomial gap between the Σ^3 and Π^3 complexities of some error correcting codes, since all linear codes over GF_2 have complexity at most $poly(n)2^{\sqrt{n}}$.

2 Definitions and statements of the main results

For our purposes, a CNF boolean formula $F(x_1, x_2, \ldots, x_n)$ is viewed as both a boolean function and a set of clauses. We say that F is a k-CNF if all the clauses have size at most k. For a clause C, we write $var(C)$

for the set of variables appearing in C. If $v \in var(C)$, the *orientation* of v is positive if the literal v is in C and is negative if $\bar{v}$ is in C. Recall that if F is a CNF boolean formula on variables $(x_1, x_2, \ldots, x_n)$ and a is a partial assignment of the variables, the *restriction* of F by a is defined to be the formula $F' = F\lceil_a$ on the set of variables that are not set by a, obtained by treating each clause C of F as follows: if C is set to 1 by a then delete C, and otherwise replace C by the clause C' obtained by deleting any literals of C that are set to 0 by a. Finally, by a *unit clause*, we mean a clause that contains exactly one literal.

To define our algorithm **ResolveSat** we first define some subroutines. The following simple function takes as input an arbitrary assignment y, a CNF formula F, and a permutation π and produces an assignment u, which is obtained by considering the variables of y in the order given by π and modifying their values in an attempt to satisfy F.

Function **Modify**(CNF formula $G(x_1, x_2, \ldots, x_n)$,
  permutation π of $\{1, 2, \ldots, n\}$, assignment y) $\longrightarrow$ (assignment u)
 $G_0 = G$.
 for $i = 1$ **to** n
  if G_{i-1} contains the unit clause $x_{\pi(i)}$
   then $u_{\pi(i)} = 1$
  else if G_{i-1} contains the unit clause $\bar{x}_{\pi(i)}$
   then $u_{\pi(i)} = 0$
  else $u_{\pi(i)} = y_{\pi(i)}$
  $G_i = G_{i-1}\lceil_{x_{\pi(i)}=u_{\pi(i)}}$
 end /* end for loop */
 return u;

The algorithm **Search** is obtained by running **Modify**(G, π, y) on many pairs (π, y) where π is a random permutation and y is a random assignment.

Search(CNF-formula F, integer I)
 repeat I times
  π = uniformly random permutation of $1, \ldots, n$
  y = uniformly random vector $\in \{0, 1\}^n$
  u = **Modify**(F, π, y);
  if u satisfies F
   then output(u); **exit**;
 end/* end repeat loop */
 output('Unsatisfiable');

The algorithm we investigate here is obtained by combining **Search** with a preprocessing step consisting of *bounded resolution*. We recall the definition of resolution. If C_1 and C_2 are two clauses we say that C_1 and C_2 *conflict* on variable v if one of them contains v and the other contains $\bar{v}$. C_1 and C_2 is a *resolvable pair* if they conflict on exactly one variable v. For such a pair their *resolvent*, denoted $R(C_1, C_2)$ is the clause $C = D_1 \vee D_2$ where D_1 and D_2 are obtained by deleting v and $\bar{v}$ from C_1 and C_2. It is easy to see that any assignment satisfying C_1 and C_2 also satisfies C. Hence if F is a satisfiable CNF formula containing the resolvable pair C_1, C_2 then the formula $F' = F \wedge R(C_1, C_2)$ has the same satisfying assignments as F. We say that the resolvable pair C_1, C_2 is *s-bounded* [5] if $|C_1|, |C_2| \leq s$ and $|R(C_1, C_2)| \leq s$. The following subroutine extends a formula F to a formula F_s by applying as many steps of s-bounded resolution as possible.

Resolve(CNF Formula F, integer s)
  $F_s = F$.
  while F_s has an s-bounded resolvable pair C_1, C_2
    with $R(C_1, C_2) \notin F_s$
   $F_s = F_s \wedge R(C_1, C_2)$.
  return (F_s).

Our algorithm for k-SAT is the following simple combination of **Resolve** and **Search**:

ResolveSat(CNF-formula F, integer s, positive integer I)
  $F_s =$ **Resolve**(F, s).
  Search(F_s, I).

The running time of **ResolveSat**(F, s, I) can be bounded as follows. **Resolve**(F, s) adds at most $O(n^s)$ clauses to F by comparing pairs of clauses, so a naive implementation runs in time $n^{3s}\text{poly}(n)$ (this time bound can be improved, but this will not affect the asymptotics of our main results). **Search**(F_s, I) runs in time $I(|F| + n^s)\text{poly}(n)$. Hence the overall running time of **ResolveSat**(F, s, I) is crudely bounded from above by $(n^{3s} + I(|F| + n^s))\text{poly}(n)$. If $s = o(n/\log n)$, we can bound the overall running time by $I|F|2^{o(n)}$ since $n^s = 2^{o(n)}$. For our purposes, it will be sufficient to choose s either to be some large constant or to be a *slowly growing* function of n, by which we mean that $s(n)$ tends to infinity with n but is $o(\log n)$.

The algorithm **Search**(F, I) always answers "unsatisfiable" if F is unsatisfiable. Thus the only problem is to upper bound the error probability in the case that F is satisfiable. Define $\tau(F)$ to be the probability that **Modify**(F, π, y) finds some satisfying assignment. Then for a satisfiable F the error probability of **Search**(F, I) is equal to $(1 - \tau(F))^I \leq e^{-I\tau(F)}$, which is at most e^{-n} provided that $I \geq n/\tau(F)$. Hence it suffices to give good upper bounds on $\tau(F)$.

To state our results on the algorithm **ResolveSat** we need to define certain constants μ_k for $k \geq 2$:

$$\mu_k = \sum_{j=1}^{\infty} \frac{1}{j(j + \frac{1}{k-1})}.$$

It is straightforward to show that $\mu_3 = 4 - 4\ln 2 > 1.226$ using Taylor's series expansion of $\ln 2$. Using standards facts, it is easy to show that μ_k is an increasing function of k with the limit $\sum_{j=1}^{\infty} \frac{1}{j^2} = (\frac{\pi^2}{6}) = 1.644\ldots$[1]

Our results on the algorithm **ResolveSat** are summarized in the following three theorems.

Theorem 1 **(i)** *Let $k \geq 5$, let $s(n)$ be a function going to infinity. Then for any satisfiable k-CNF formula F on n variables,*

$$\tau(F_s) \geq 2^{-(1-\frac{\mu_k}{k-1})n - o(n)}$$

Hence, **ResolveSat**(F, s, I) *with $I = 2^{(1-\frac{\mu_k}{k-1})n + o(n)}$ has error probability $o(1)$ and running time $2^{(1-\frac{\mu_k}{k-1})n + o(n)}$ on any satisfiable k-CNF formula, provided that $s(n)$ goes to infinity sufficiently slowly.*

(ii) *For $k \geq 3$, we get the same bounds provided that F is* uniquely satisfiable.

[1] Alternatively, the bound can be expressed in terms of standard functions as follows: $\mu_k/(k-1) = \gamma + \Gamma'(k/(k-1))/\Gamma(k/(k-1))$, where $\gamma = 0.577\ldots$ is the Euler constant and Γ is the standard Gamma function.

We shall compare this bound with the bound in Schöning's algorithm [22]. The exponent in [22] is $1 + \log_2(1 - \frac{1}{k})$. Since

$$-(k-1)\log_2(1-\frac{1}{k}) < \lim_{k\to\infty} -(k-1)\log_2(1-\frac{1}{k}) = \frac{1}{\ln 2} = 1.442\ldots < 1.470\ldots = \mu_7 \le \mu_k,$$

for $k \ge 7$, we have $1 + \log_2(1 - \frac{1}{k}) > 1 - \frac{\mu_k}{k-1}$ for $k \ge 7$. The inequality for $k < 7$ can be checked by direct computations, see Table 1.1.

Theorem 1 is proved by first considering the uniquely satisfiable case and then relating the general case to the uniquely satisfiable case. When $k \ge 5$, our analysis shows that the asymptotics of the general case is no worse than that of the uniquely satisfiable case. When $k = 3$ or $k = 4$, our analysis gives somewhat worse bounds for the general case than for the uniquely satisfiable case.

Theorem 2 *Let $s = s(n)$ be a slowly growing function. For any satisfiable n variable 3-CNF formula, $\tau(F_s) \ge 2^{-0.521n}$ and so* **ResolveSat**(F, s, I) *with $I = n2^{0.521n}$ has error probability $o(1)$ and running time $2^{0.521n+o(n)}$.*

Theorem 3 *Let $s = s(n)$ be a slowly growing function. For any satisfiable n variable 4-CNF formula, $\tau(F_s) \ge 2^{-0.5625n}$ and so* **ResolveSat**(F, s, I) *with $I = n2^{0.5625n}$ has error probability $o(1)$ and running time $2^{0.5625n+o(n)}$.*

Let z be a satisfying assignment to a formula F. For integer $d \ge 1$, we say that z is *d-isolated in F* if there is no other satisfying assignment within Hamming distance d of z. Define

$$\epsilon_k^{(d)} = \frac{3}{(d-1)(k-2)+2}. \tag{1}$$

Theorem 4 *Let n, k, d be integers with $k \ge 3, d \ge 1$. Let F be a CNF formula on n variables and suppose that F has at most L clauses of size larger than k. Then the number of d-isolated satisfying assignments of F is at most*

$$2^{(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n} + Ln^d 2^{n-k}.$$

We believe that the gap between the bounds for the general case and the uniquely satisfiable case when $k \in \{3, 4\}$ is due to a weakness in our analysis, and we conjecture that the asymptotic bounds for the uniquely satisfiable case hold in general for all k. If true, the conjecture would imply that our algorithm is also faster than any other known algorithm in the $k = 3$ case.

Recall that a set $E \subseteq \{0,1\}^n$ is an *error correcting code* with minimum distance d if $e_1, e_2 \in E, e_1 \ne e_2$ implies that the Hamming distance between e_1 and e_2 is at least d. Elements of E are referred to as *codewords*. Abusing notation slightly, let $E(x)$ be the function which is 1 iff $x \in E$. Our lower bounds on depth 3 circuits will apply to any code E with minimum distance $d > \log n$ and at least $2^{n-(n/\log n)}$ codewords. These are fairly lenient choices of parameters, and constructions of codes with these properties are well known. For example, a BCH code with designed distance $\log n$ has at least $2^{n-\log^2 n}$ codewords, see e.g. [14]. We prove:

Theorem 5 *Let E be an error correcting code of minimum distance $d > \log n$ and at least $2^{n-(n/\log n)}$ codewords. If C is a Σ_k^3 circuit computing E with $k \ge 3$, then C has at least $2^{\frac{\mu_k}{k-1}n - \frac{4n}{\log n}}$ gates.*

For example, we obtain a lower bound of $2^{0.612n}$ on the size of Σ_3^3 circuits computing E. In addition, we obtain

Theorem 6 *Let E be an error correcting code of minimum distance $d > \log n$ and at least $2^{n-(\sqrt{n}/\log n)}$ codewords. If C is a Σ^3 circuit computing E, then C has at least $2^{(\frac{\pi}{\sqrt{6}} - \frac{5}{\log n})\sqrt{n}}$ gates.*

Note that since $\frac{\pi}{\sqrt{6}} > 1.282$, this proves a lower bound of $2^{1.282\sqrt{n}}$ on the size of such circuits.

The rest of the paper is organized as follows. In Section 3 we prove our result for unique SAT, in Section 4 we prove our results on general SAT and in Section 5 we prove Theorem 4 and the lower bounds on depth 3 circuits.

3 Unique SAT

3.1 Overview

In this section we prove part (ii) of Theorem 1 that establishes the bound on the running time for our algorithm for the case of uniquely satisfiable formulas. Let G be an arbitrary formula and z an assignment. Define $\tau(G, z)$ to be the probability with respect to random π and y, that $\mathbf{Modify}(G, \pi, y)$ returns z. For our main results we want to lower bound $\tau(G)$, which is the sum of $\tau(G, z)$ over all satisfying assignments z of G.

The following is a more precise statement of the result for unique SAT (Theorem 1 (ii)).

Theorem 7 *Let F be a k-CNF, $k \geq 3$ formula on n-variables and suppose that z is a d-isolated satisfying assignment of F. Then for $s \geq k^d$,*

$$\tau(F_s, z) \geq 2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n},$$

where $\epsilon_k^{(d)}$ is as defined in (1)

To deduce Theorem 1(ii) from this, note that when F is uniquely satisfiable, its satisfying assignment is d-isolated for any $d \geq 1$. Suppose $s(n)$ tends to ∞ with n and define $d = d(n) = \lfloor \log_k s(n) \rfloor$. Then $\epsilon_k^{(d(n))} = o(1)$ and so the bound of Theorem 7 implies that of Theorem 1(ii).

To analyze $\tau(F_s, z)$, we initially ignore the fact that F_s is obtained from F by s-bounded resolution and analyze $\tau(G, z)$ for an arbitrary formula G.

Consider the run of **Modify**(G, π, y). Recall that each variable x_i is assigned so as to satisfy some unit clause, or is set to y_i. A variable whose assignment is determined by a unit clause is said to be *forced* (with respect to π and y). Let $\mathrm{Forced}(G, \pi, y)$ denote the set of variables that are forced. We first observe:

Lemma 1 *Let z be a satisfying assignment of G, and let π be a permutation of $\{1, \ldots, n\}$ and y be any assignment to the variables. Then* **Modify**$(G, \pi, y) = z$ *if and only if y and z agree on all variables outside of* $\mathrm{Forced}(G, \pi, z)$.

Proof. If y agrees with z on all variables outside of $\mathrm{Forced}(G, \pi, z)$, then a simple induction on i shows that in the execution of $\mathbf{Modify}(G, \pi, y)$, the output bit $u_{\pi(i)}$ is z_i. If the variable $x_{\pi(i)}$ is not in $\mathrm{Forced}(G, \pi, z)$ then the $u_{\pi(i)}$ is set to $y_{\pi(i)} = z_{\pi(i)}$. If variable $x_{\pi(i)}$ is in $\mathrm{Forced}(G, \pi, z)$ then, by induction, for $j < i$, $u_{\pi(j)} = z_{\pi(j)}$ and so the clauses of G force output bit $u_{\pi(i)}$ to be $z_{\pi(i)}$.

Now suppose y disagrees with z on some variable outside of $\mathrm{Forced}(G, \pi, z)$ and let i be the first index such that $y_{\pi(i)} \neq z_{\pi(i)}$. Then it is easy to see that the output bit $u_{\pi(i)}$ will equal $y_{\pi(i)}$ and hence **Modify**$(G, \pi, y) \neq z$. ∎

Thus, for fixed π, the number of y for which $\mathbf{Modify}(G, \pi, y)$ outputs z is $2^{|\mathrm{Forced}(G,\pi,z)|}$, and we have:

Lemma 2 *Let z be a satisfying assignment of the formula G. Then*

$$\tau(G,z) = \frac{1}{2^n n!}\sum_{\pi} 2^{|\mathrm{Forced}(G,\pi,z)|} = 2^{-n}\mathbf{E}_{\pi}[2^{|\mathrm{Forced}(G,\pi,z)|}].$$

where $\mathbf{E}_\pi$ denotes expectation with respect to random π.

By the convexity of the exponential function, the last expression can be lower bounded by

$$2^{-n+\mathbf{E}_{\pi}[|\mathrm{Forced}(G,\pi,z)|]}.$$

We next find an alternative expression for $\mathbf{E}_\pi[|\mathrm{Forced}(G,\pi,z)|]$. If v is a variable of formula G and z is a satisfying assignment we say that a clause C is *critical* for (v,G,z) if C is in G, $v \in var(C)$, and under the assignment z, the only true literal in C is the one corresponding to v. Suppose that C is critical for (v,G,z), and that π is a permutation such that v appears last among the variables of C. Then, in the run **Modify**(G,π,z), by the time v is assigned, all of the other literals in C have been falsified and so $v \in \mathrm{Forced}(G,\pi,z)$ (conversely, if $v \in \mathrm{Forced}(G,\pi,z)$ then v must appear last in some critical clause for (v,G,z)). Let $\mathrm{Last}(v,G,z)$ be the set of permutations π of the variables such that for at least one critical clause C for (v,G,z), v appears last among all variables in $var(C)$, and let

$$P(v,G,z)$$

denote the probability that a random permutation π belongs to $\mathrm{Last}(v,G,z)$, which is equivalent to the probability that $v \in \mathrm{Forced}(G,\pi,z)$ for random π. By linearity of expectation,

$$\mathbf{E}_\pi[|\mathrm{Forced}(G,\pi,z)|] = \sum_v P(v,G,z).$$

Putting things together we have:

Lemma 3 *For any satisfying assignment z of the CNF formula G:*

$$\tau(G,z) \geq 2^{-n+\sum_v P(v,G,z)}.$$

In particular, if $P(v,G,z) \geq p$ for all variables v then $\tau(G,z) \geq 2^{-(1-p)n}$.

Hence, to lower bound $\tau(G,z)$ it suffices to lower bound $P(v,G,z)$. It is important to emphasize that while the function **Modify** depends on random π and y, the probability represented by $P(v,G,z)$ depends only on π and is independent of y.

To motivate the proof below we first consider what happens if the preprocessing by Resolution is omitted (which is essentially the algorithm studied in [17]). Let z be the unique satisfying assignment for the k-CNF F. Then for each variable v, F must contain at least one critical clause C_v for (v,F,z), otherwise the assignment obtained from z by complementing the value in position v is also a satisfying assignment. Since C_v has at most k variables, we conclude that for a random permutation π, v appears last in C_v with probability at least $1/k$, i.e., $P(v,F,z) \geq 1/k$ and so Lemma 3 implies

$$\tau(F) = \tau(F,z) \geq 2^{-(1-\frac{1}{k})n}.$$

In fact, this argument actually proves a more general result: if F is a k-CNF, and z is an isolated satisfying assignment, in the sense that no assignment differing from z in only one position satisfies F, then $\tau(F,z) \geq$

$2^{-(1-\frac{1}{k})n}$. Our aim is to improve the bound by showing that $\frac{1}{k}$ can be replaced by $\frac{\mu_k}{k-1}$ assuming that z is isolated to a nonconstant Hamming distance (recall that $\mu_k > 1$).

Note that the argument above uses only the fact that each variable has at least one critical clause; if there are more critical clauses for each variable we could get a better lower bound. For example, suppose $z = 1^n$ is an isolated satisfying assignment for the formula F, and that F contains the two clauses $(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ and $(x_1 \vee \bar{x}_4 \vee \bar{x}_5)$ which are both critical for (x_1, F, z). In this case, the probability that a random permutation of the variables puts x_1 last in some critical clause is at least $7/15$, rather than $1/3$ obtained using only a single critical clause.

In general, even if F is uniquely satisfiable, F need contain only one critical clause per variable, so we can't hope for a general improvement of this kind. However, what we will show is that for appropriately chosen s, F_s contains many critical clauses for each variable. As an example, consider a formula F which contains clauses $C_1 = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ and $C_2 = (x_2 \vee \bar{x}_4 \vee \bar{x}_5)$, which are critical for x_1, x_2, respectively. Resolution on the variable x_2 will produce the clause $C = (x_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5)$, which is also a critical clause for x_1. Thus F_4 contains both C_1 and C, and the probability that x_1 will be forced increases.

Considering only critical clauses, however, may not produce any additional effect. If instead of C_2 the unique critical clause for x_2 is, say, $C_3 = (\bar{x_1} \vee x_2 \vee \bar{x_4})$, then C_1 and C_3 are not resolvable, because they conflict on two variables x_1 and x_2 (resolving with C_1 over x_2 would produce a tautological clause $(x_1 \vee \bar{x_1} \vee \bar{x_3} \vee \bar{x_4})$ which is useless). To get new critical clauses for $z = 1^n$ we have to use essentially the fact that z is d-isolated with $d > 1$. In particular, 2-isolation implies that 001^{n-2} is not a satisfying assignment, therefore there is a clause that is not satisfied by this assignment. Such a clause can be, e.g., $C_4 = (x_1 \vee x_2 \vee \bar{x_4})$, which is resolvable with C_1 and produces the critical clause $(x_1 \vee \bar{x_3} \vee \bar{x_4})$. It is clear that any clause of the formula that is not satisfied by 001^{n-2} is either resolvable with C_1, or it is a critical clause for variable x_1 that does not contain x_2, thus it is different from C_1. Hence we always get another critical clause for x_1. Then we get another clause by considering the non-satisfying assignment 0101^{n-3}. If $d > 2$ we can continue and find more critical clauses etc.

To describe the process of adding new critical clauses for a variable v, we shall use labelled rooted trees. The root will be labelled by v, the children of the root will correspond to the literals of a chosen critical clause for v. The next nodes will be given by clauses that are resolvable with this clause etc. The formal description is below.

We need some definitions. The *degree* of a node in a rooted tree is the number of its children. The *depth* of a node is its distance from the root.

A tree is of *uniform depth* d if all of its leaves are at depth d. A subset A of nodes is a *cut* if it does not include the root, and every path from the root to a leaf includes a node of A. (This definition is not quite what one might expect, e.g., the set of all leaves is a cut by this definition, while the root isn't). If A is a set of nodes, write $var(A)$ for the set of variables that appear as labels of nodes of A.

Definition 1 *A rooted tree is said to be* admissible *with respect to a given set of boolean variables if it has the following properties:*

- *the root is labeled by a variable;*
- *each node in the tree is either labeled by a variable or unlabeled;*
- *for any path P from the root to a leaf, no two nodes have the same label. In other words, if node a is an ancestor of node b and both are labeled, then they have different labels.*

A tree is said to be a critical clause tree *for variable v, formula G and satisfying assignment z if it is admissible and in addition satisfies:*

- *the root label is v;*
- *for any cut A of the tree, G has a critical clause $C(A)$ for (v, G, z) such that $var(C(A)) \subseteq var(A) \cup \{v\}$.*

In the definition of critical clause tree, the portion of the tree below an unlabeled node is essentially irrelevant (except for the third admissibility condition). More precisely, we may replace the subtree from an unlabeled node by any subtree consisting only of unlabelled nodes and all the conditions will be preserved. It would be natural to impose the simplifying assumption that unlabelled nodes are leaves, but we prefer not to do this since in our results on critical clause trees, the statements and analysis are simplified if we consider trees of uniform depth. Hence, in section 3.2, where we describe a construction of a uniform depth critical clause tree, unlabeled nodes of depth less than d are given a single unlabeled child.

We will show that if F is uniquely satisfiable, then for some appropriately large s, there is a "large" critical clause tree for (v, F_s, z). This critical clause tree represents multiple critical clauses for (v, F_s, z). This will enable us to derive a better lower bound on the probability that v is forced.

We shall split the proof of Theorem 7 into two parts stated below as two lemmas.

Lemma 4 *Let F be a k-CNF formula, $k \geq 3$, and z be a d-isolated satisfying assignment of F. If v is any variable then for any $s \geq k^d$, there exists a critical clause tree for (v, F_s, z) of maximum degree $k-1$ and uniform depth d.*

The second lemma asserts that the existence of a sufficiently deep critical clause tree for (v, G, z) of bounded degree implies a lower bound on $P(v, G, z)$.

Lemma 5 *Let G be a formula, z a satisfying assignment, and v a variable. If there is a critical clause tree for (v, G, z) with maximum degree $k-1$ and uniform depth d, then:*

$$P(v, G, z) \geq \frac{\mu_k}{k-1} - \epsilon_k^{(d)}.$$

Combining these two lemmas with Lemma 3 immediately yields Theorem 7 and hence also Theorem 1 (ii). So it remains to prove these two lemmas.

3.2 Proof of Lemma 4

We need one additional piece of notation: for a set of variables U, $z \oplus U$ denotes the assignment obtained from z by complementing the variables of U.

Fix a k-CNF formula F, and d-isolated assignment z; we will assume without loss of generality that $z = 1^n$. Furthermore, let a variable v be fixed. Note that every critical clause of F for v consists of v and a set of negated variables. We will construct a critical clause tree for (v, F_s, z). We "grow" the tree by the following process. Start with a tree T_0 consisting of one node labeled v. We construct a sequence of trees $T_1, T_2, \ldots$ as follows. Having constructed T_{i-1}, if all leaves have depth d stop. Otherwise, choose a leaf b_i of depth less than d, and let P_i be the set of nodes appearing on the path from b_i to the root (including b_i and the root). Since z is d-isolated, and $|P_i| \leq d$, $z \oplus var(P_i)$ does not satisfy F; choose a clause C_i that is not satisfied by $z \oplus var(P_i)$. For each variable w of $var(C_i) - var(P_i)$, give b_i a child labeled w. If $var(C_i) - var(P_i)$ is empty, give b an unlabeled child. Let N_i denote the set of nodes corresponding to the children of b_i.

Clearly the above procedure terminates with an admissible tree T of maximum degree $k-1$ of uniform depth d. The total number of nodes in the final tree is bounded above by $\sum_{i=0}^{d}(k-1)^i \leq k^d$, and hence any clause whose variables all appear as node labels in the tree has size at most s.

It remains to prove that T is a critical clause tree for (v, F_s, z), i.e., that for any cut A, F_s contains a critical clause $C(A)$ for (v, F_s, z) whose variable set is contained in $var(A) \cup \{v\}$. To this end, we will prove by induction on i that each T_i is a critical clause tree. The result is vacuously true for T_0, which has no cuts. For T_1, the tree has only one cut. T_1 is constructed from a clause C which is not satisfied by $z \oplus v$, i.e., a critical clause for v at z.

Now suppose $i \geq 2$ and the result holds for T_{i-1}. T_i consists of T_{i-1} together with the node set N. Let A be a cut of T_i, and let $A' = A - N$. Let the nodes of P_i be denoted $v = a_0, a_1, \ldots, a_t = b_i$ where $t < d$, let $v, r_1, \ldots, r_t$ be the labels of these nodes. For each $1 \leq j \leq t$, the set $A_j = A' \cup \{a_j\}$ is a cut of T_{i-1}. Hence, by the induction hypothesis, there is a critical clause $C(A_j)$ for (v, F_s, z) such that $var(C(A_j)) \subseteq var(A_j) \cup \{v\}$. Consider the following two cases:

1. for some $j \in \{1, \ldots, t\}$, $var(A_j) \subseteq var(A')$;

2. for all $j \in \{1, \ldots, t\}$, $var(A_j) \not\subseteq var(A')$, i.e., $r_j \notin var(A')$.

In the first case we can choose $C(A)$ to be one of the clauses $C(A_j)$ such that $var(A_j) \subseteq var(A')$.

In the second case r_l does not occur in $C(A_j)$ for every $l \neq j$, since otherwise r_l would be in $var(A')$, hence $var(A_l) \subseteq var(A')$. Thus

$$r_j \in var(C(A_j)) \subseteq (var(A') \setminus \{r_1, \ldots, r_t\}) \cup \{r_j\}. \tag{2}$$

Now consider the clause C_i of F that is used to construct T_i from T_{i-1}. We can write C_i in the form $R \vee U$ where R is a clause consisting of negated variables from $var(N)$ and U is a clause consisting of positive variables from $v, r_1, \ldots, r_t$.

We now prove, by reverse induction on $j \in \{0, \ldots, t\}$ that there is a clause D_j in F_s of the form $D_j = R \vee S_j \vee U_j$ where R is defined in the last paragraph, S_j is a clause consisting of negations of variables from A' and U_j is a clause consisting of (positive occurrences of) variables from $\{v, r_1, \ldots, r_j\}$. D_0 must then be of the form $R \vee S_0 \vee U_0$ where U_0 is empty or $U_0 = v$. It can't be that U_0 is empty, since then D_0 would consist only of negated variables, and such a clause can not be in F_s since $z = 1^n$ satisfies F_s. Thus we may take $C(A)$ to be D_0.

To prove the existence of D_j, for the basis we take $D_t = C_i$, with S_t being the empty clause and $U_t = U$. For $j < t$, assuming we have constructed D_{j+1}, if r_{j+1} does not appear in U_{j+1} then take $D_j = D_{j+1}$. Otherwise the clause D_{j+1} can be resolved with $C(A_{j+1})$ to eliminate r_{j+1} and yield D_j, since, by (2), r_{j+1} is the unique conflict variable of the two clauses. As each of the D_j has size at most s, each is in F_s.

This finishes the proof of Lemma 4.

3.3 Proof of Lemma 5

We now proceed to the proof of Lemma 5. We are given a critical clause tree for (v, G, z) of maximum degree $k - 1$ and uniform depth d and we want to deduce a lower bound on $P(v, G, z)$, the probability with respect to a random permutation π, that v appears last in some critical clause. More precisely we will estimate from below the probability that this happens for some critical clause tree for v. The critical clauses of the tree are given by cuts of the tree, so we need to estimate the probability that for some cut the variables on the cut are before the variable v. This looks like a big mess, because the same variables may occur on different places of the tree, so the events for different cuts may be dependent. But we shall show that the correlation is always positive, hence the worst case is when all nodes of the tree are labeled by different variables.

Let us consider an example. Suppose the following is a critical clause tree for v at z of depth 2.

$$T = \quad \begin{array}{ccccccc} & & & v & & & \\ & x_2 & & & & x_5 & \\ x_1 & & x_3 & & x_4 & & x_6 \end{array} \tag{3}$$

Such a tree is guaranteed if z is 2-isolated, which is always true if z is the unique satisfying assignment (and $n \geq 2$). The tree T is special, because all nodes are labeled by different variables (thus we shall refer to nodes by their labels), but we shall show below that it is the worst case. This tree has four cuts, which are combinations of one of the two cuts in the tree below x_2 and of one of the two cuts in the tree below x_5. The root variable v is forced by a clause given by a cut, if all variables of the cut precede v. Hence the probability that v is forced, when we have T, is equal to the probability that

((x_1 and x_3 are before v) or x_2 is before v) and ((x_4 and x_6 are before v) or x_5 is before v).

Since all variables are different, the events that they are before v are independent. Let us denote by p_i the probability that x_i is before v. Then the above probability can be expressed as

$$(1-(1-p_1p_3)(1-p_2))(1-(1-p_4p_6)(1-p_5)). \tag{4}$$

To compute this probability, we first choose the position of v, then the probabilities p_i are determined and are all the same. In general we get the formula for this probability by recurrence.

For the analysis, it will be useful to view the permutation π as a random variable on the following continuous probability space. A *placement* of the variables is a function α that maps each variable to the open interval $(0,1)$, thus a placement is a point in $(0,1)^{\{x_1,\ldots,x_n\}} \cong (0,1)^n$. We take the set of placements as our sample space and consider the uniform probability distribution, i.e., we select α so that the values $\alpha(u)$ are independent and uniformly distributed on $(0,1)$. An event A in this probability space is a (Lebesgue measurable) subset of $(0,1)^n$ and the probability of an event is the Lebesgue measure of the subset.

Given a placement α we define a permutation $\pi = \pi_\alpha$ obtained by ranking the variables according to their α values, breaking ties by some (any) arbitrary rule. In other words, variables with smaller α value will appear before variables with larger α value. Since α is one-to-one with probability 1, π is uniformly distributed over all permutations.

Henceforth, all probabilities we compute are with respect to the probability space of placements.

For an admissible tree T with root labeled by variable v, we define the event Cut_T to consist of all placements α such that for some cut A of T, $\alpha(w) < \alpha(v)$ for all $w \in var(A)$. We define $\mathrm{Cut}_T(r)$ for $r \in [0,1]$ to consist of all α such that for some cut A of T, $\alpha(w) < r$ for all $w \in var(A)$. We define Q_T (resp. $Q_T(r)$) to be the probability Cut_T (resp. $\mathrm{Cut}_T(r)$) occurs. From the definitions we have:

Lemma 6 *If T is a critical clause tree for (v, G, z), then $P(v, G, z) \geq Q_T$.*

So to prove Lemma 5, it suffices to lower bound Q_T in the case that T is a tree of maximum degree $k-1$ and uniform depth d. Now it is easy to see that $Q_T(r)$ is just equal to the conditional probability of Cut_T given that $\alpha(v) = r$ (here we need to use the fact that in an admissible tree, no other node has the same label as the root). Hence:

$$Q_T = \int_0^1 Q_T(r)dr.$$

We say that T is trivial if it consists of one node; in this case $Q_T = Q_T(r) = 0$ for all r. Otherwise, for each child a of the root, the subtree $T(a)$ rooted at a is admissible if and only if a is labeled.

The following lemma gives a simple recursive lower bound on $Q_T(r)$. In anticipation of what we will need in the next section, we state the lemma in a more general form then is needed in this section: we consider distributions over placements where each variable is placed independently in $[0,1]$ but where the distribution used to place each variable may be arbitrary.

Lemma 7 *Let T be an admissible tree with more than one node with root labelled by v and let $r \in [0, 1]$. Let $T_1, T_2, \ldots, T_t$ be the (admissible) subtrees rooted at those children of the root of T that are labeled; let v_i be the label of the root of T_i and let q_i be the probability that $\alpha(v_i) \leq r$. Then:*

$$Q_T(r) \geq \prod_{i=1}^{t} (q_i + (1 - q_i) Q_{T_i}(r)),$$

where an empty product is interpreted as 1.

Proof. If none of the children of the root of T is labeled, then T has a cut A such that $var(A) = \emptyset$ and so $\mathrm{Cut}_T(r)$ occurs with probability 1. Otherwise, let v_i be the variable labeling the root of T_i. The event $\mathrm{Cut}_T(r)$ can be written as the intersection of the events $\{K_i : 1 \leq i \leq t\}$ where $K_i = \mathrm{Cut}_{T_i}(r) \vee (\alpha(v_i) < r)$.

By the admissibility of T_i, v_i does not appear elsewhere in T_i, so the events Cut_{T_i} and $\alpha(v_i) < r$ are independent. Thus $\mathbf{Prob}[K_i] = q_i + (1 - q_i) Q_{T_i}(r)$, and we need that $\mathbf{Prob}[\bigwedge_{1 \leq i \leq t} K_i] \geq \prod_{1 \leq i \leq t} \mathbf{Prob}[K_i]$. This follows from standard correlation inequalities: For a placement α, let $W = W_r(\alpha)$ be the subset of variables w such that $\alpha(w) < r$. Then each event K_i depends only on W. Let $\mathcal{W}_i$ denote the set of all subsets U of variables such that $W = U$ implies K_i. We have (i) The set W is selected according to a distribution where the events $\{s \in W : s \in S\}$ are mutually independent and (ii) each $\mathcal{W}_i$ is an increasing family of subsets, i.e., if $U \in \mathcal{W}_i$ then so is any superset of U. Harris' inequality ([8], special case of the FKG inequality, cf. Theorem 3.2 from Chapter 6 of [1]) says that (i) and (ii) are sufficient to conclude:

$$\mathbf{Prob}[\bigwedge_{1 \leq i \leq t} K_i] = \mathbf{Prob}[W \in \cap_{1 \leq i \leq t} \mathcal{W}_i] \geq \prod_{1 \leq i \leq t} \mathbf{Prob}[W \in \mathcal{W}_i] = \prod_{1 \leq i \leq t} \mathbf{Prob}[K_i],$$

which completes the proof of Lemma 7. (Actually, Harris' inequality appears in the references only for the case $t = 2$; the well known general case follows by a routine induction on t). ■

In this section, we will only need the case that all of the q_i are equal to r. We apply this lemma in the case that T is a tree of degree at most $k - 1$ and uniform depth d. We need some definitions. Fix $k \geq 3$ and $r \in [0, 1]$.

- Let $f_k(x; r) = (r + (1 - r)x)^{k-1}$.
- Define the sequence $(Q_k^{(d)}(r) : d \geq 0)$ by the recurrence $Q_k^{(0)}(r) = 0$ and $Q_k^{(d)}(r) = f_k(Q_k^{(d-1)}(r); r)$.
- Define $Q_k^{(d)} = \int_0^1 Q_k^{(d)}(r) dr$.

Lemma 7 together with induction on d yields:

$$Q_T(r) \geq Q_k^{(d)}(r),$$

hence:

$$Q_T \geq Q_k^{(d)}.$$

Finally we will prove:

Lemma 8 *Let $k \geq 3$. Then for each $d \geq 1$,*

$$Q_k^{(d)} \geq \frac{\mu_k}{k - 1} - \epsilon_k^{(d)}.$$

Lemmas 6 and 8, and the inequalities above give $P(v, G, z) \geq Q_T \geq \frac{\mu_k}{k-1} - \epsilon_k^{(d)}$, as required for Lemma 5. So it remains to prove this lemma.

Proof. In order to lower bound $Q_k^{(d)}$, we will show that for each k and r, the sequence $(Q_k^{(d)}(r) : d \geq 0)$ converges and also give bounds on the rate of convergence. We need the following definitions.

- For $r \in [0, 1]$ define $R_k(r)$ to be the smallest nonnegative x that satisfies $f_k(x; r) = x$. $R_k(r)$ is well defined and has range $[0, 1]$, since for each r, $f_k(x; r) - x$ is a polynomial in x, having 1 as a root.
- $R_k = \int_0^1 R_k(r)\, dr$.
- For $t \in [0, 1]$, define $S_k(t)$ as

$$S_k(t) = \frac{t^{\frac{1}{k-1}} - t}{1-t}, t \in [0, 1), S_k(1) = \frac{k-2}{k-1}.$$

  It turns out that $S_k(.)$ is the inverse of the function $R_k(.)$.

We shall show that $R_k(r)$ (resp. R_k) is the limit of $Q_k^{(d)}(r)$ (resp. $Q_k^{(d)}$) for $d \to \infty$. So the meaning of $R_k(r)$ is the probability that a variable v placed on r will be forced by a critical clause of a critical clause tree of v assuming the tree is of infinite depth and all labels are different (the worst case). R_k is the same probability over all placements of v. Of course, we shall use only finite trees, therefore we need an estimate on the convergence.

We first establish:

Claim 1 *Fix $k \geq 3$.*

- *For $r \in [\frac{k-2}{k-1}, 1]$, $R_k(r) = 1$.*
- *On the interval $[0, \frac{k-2}{k-1}]$, $R_k(.)$ is a strictly increasing continuous map onto $[0, 1]$, and its inverse is $S_k(.)$.*

Proof. Fix r and let $\epsilon = 1 - R_k(r)$, so $\epsilon \in [0, 1]$. Substituting into $R_k(r) = f_k(R_k(r); r)$, yields $g(\epsilon; r) = 1$ where $g(\epsilon; r) = (1 - (1 - r)\epsilon)^{k-1} + \epsilon$. If $\epsilon > 0$, then $g(\epsilon; r) > 1$ for $r \in (\frac{k-2}{k-1}, 1]$, a contradiction. Thus for $r \in [\frac{k-2}{k-1}, 1]$, we have $\epsilon = 0$ and thus $R_k(r) = 1$.

Now, for fixed $0 < r < \frac{k-2}{k-1}$, $g(\epsilon; r)$ is a continuous function of ϵ. Since $g(0; r) - 1 = 0$, $g'(0; r) < 0$ and $g(1; r) - 1 > 0$, we conclude that for all $r < \frac{k-2}{k-1}$ there is a positive ϵ^* satisfying $g(\epsilon^*; r) = 0$, which implies that $R_k(r) < 1$.

Thus for $r \in [0, \frac{k-2}{k-1})$, $R_k(r) \in [0, 1)$. Then the equation $f_k(R_k(r); r) = R_k(r)$ can be solved for r to obtain $r = S_k(R_k(r))$. By elementary calculus, $S_k(t)$ is a continuous increasing function of t mapping $[0, 1)$ onto $[0, \frac{k-2}{k-1})$, from which it follows that its inverse $R_k(r)$ is a continuous increasing function mapping $[0, \frac{k-2}{k-1})$ to $[0, 1)$. ■

Next we define the sequences $\{\Delta_k^{(d)}(r) : d \geq 0\}$ and $\{\Delta_k^{(d)} : d \geq 0\}$ by:

- $\Delta_k^{(d)}(r) = R_k(r) - Q_k^{(d)}(r)$,
- $\Delta_k^{(d)} = R_k - Q_k^{(d)}$.

Claim 2 *Let $k \geq 3$. For all $d \geq 1$,*

$$0 \leq \Delta_k^{(d)} \leq \epsilon_k^{(d)}.$$

Proof For simplicity of notation, we omit the subscript k. For each $r \in [0,1]$, $\Delta^{(d)}(r) \geq 0$ since $Q^{(d)}(r)$ is nondecreasing in d and $R(r) = Q^{(\infty)}(r)$. It then follows $\Delta^{(d)} = \int_0^1 \Delta^{(d)}(r) \geq 0$. Now

$$\Delta^{(d)}(r) = R(r) - Q^{(d)}(r) = R(r) - f(Q^{(d-1)}(r); r) = R(r) - f(R(r) - \Delta^{(d-1)}(r); r). \tag{5}$$

To upper bound $\Delta^{(d)}(r)$ we show that for any $\gamma \in [0, R(r)]$,

$$0 \leq R(r) - f(R(r) - \gamma; r) \leq \gamma \frac{(k-1)(1-r)R(r)}{r + (1-r)R(r)} \tag{6}$$

The first inequality follows since for fixed $r \in (0,1)$, $f(x;r)$ is an increasing function of x and $f(R(r);r) = R(r)$. For the second inequality, we use $x^{k-1} - y^{k-1} = (x-y)\sum_{i=0}^{k-2} x^i y^{k-2-i}$ (which can be proved by induction on k) to show:

$$\begin{aligned}
R(r) - f(R(r) - \gamma; r) &= f(R(r); r) - f(R(r) - \gamma; r) \\
&= (r + (1-r)R(r))^{k-1} - (r + (1-r)(R(r) - \gamma))^{k-1} \\
&= \gamma(1-r)\sum_{j=0}^{k-2}(r + (1-r)R(r))^{k-2-j}(r + (1-r)(R(r) - \gamma))^j \\
&\leq \gamma(1-r)(r + (1-r)R(r))^{k-2})(k-1) \\
&= \gamma \frac{(1-r)R(r)}{r + (1-r)R(r)}(k-1).
\end{aligned}$$

Taking $\gamma = \Delta^{(d-1)}(r)$ and using induction on d with (5) and (6) yields:

$$\Delta^{(d)}(r) \leq \left(\frac{(k-1)(1-r)R(r)}{r + (1-r)R(r)}\right)^d.$$

Hence:

$$\Delta^{(d)} \leq \int_0^1 \left(\frac{(k-1)(1-r)R(r)}{r + (1-r)R(r)}\right)^d dr.$$

We split the range of integration into two intervals. For $r \in [\frac{k-2}{k-1}, 1]$ we have $R(r) = 1$ and the integral over this range is easily calculated to be $\frac{1}{(d+1)(k-1)}$.

For $r \in [0, \frac{k-2}{k-1}]$, $R(r)$ maps the interval bijectively to $[0,1]$, and $r = S(R(r))$. Make the substitution $u = R(r)^{1/(k-1)}$, so $r(u) = \frac{u - u^{k-1}}{1 - u^{k-1}}$ and $\frac{(1-r)R(r)}{r + (1-r)R(r)} = \frac{(1-u)u^{k-2}}{1 - u^{k-1}}$. Also $\frac{dr}{du} = \frac{1-(k-1)u^{k-2} + (k-2)u^{k-1}}{(1-u^{k-1})^2}$ which, for $u \in [0,1]$, is nonnegative and

$$\frac{1-(k-1)u^{k-2} + (k-2)u^{k-1}}{(1-u^{k-1})^2} = \frac{(1 + 2u + 3u^2 + \ldots + (k-2)u^{k-3})(1-u)^2}{(1 + u + u^2 + \ldots + u^{k-2})(1-u)(1-u^{k-1})} =$$

$$\frac{(1 + 2u + 3u^2 + \ldots + (k-2)u^{k-3})(1-u)}{(1 + u + u^2 + \ldots + u^{k-2})(1-u^{k-1})} \leq$$

$$\frac{((k-1) + (k-1)u + (k-1)u^2 + \ldots + (k-1)u^{k-3} + (k-1)^{k-2})(1-u)}{(1 + u + u^2 + \ldots + u^{k-2})(1-u^{k-1})} = \frac{(k-1)(1-u)}{1-u^{k-1}}.$$

This gives:

$$\begin{aligned}
\int_0^{\frac{k-2}{k-1}} \left(\frac{(k-1)(1-r)R(r)}{r+(1-r)R(r)}\right)^d dr &\leq \int_0^1 (k-1)^{d+1} \frac{(1-u)^{d+1}u^{(k-2)d}}{(1-u^{k-1})^{d+1}} du \\
&= \int_0^1 \frac{u^{(k-2)d}}{(\frac{1}{k-1}(1+u+\cdots+u^{k-2}))^{d+1}} du \\
&\leq \int_0^1 \frac{u^{(k-2)d}}{u^{(\frac{k-2}{2})(d+1)}} du \\
&= \frac{2}{kd-2d-k+4} = \frac{2}{(k-2)(d-1)+2}.
\end{aligned}$$

The next-to-last inequality follows from the arithmetic-geometric mean inequality. Summing the two integrals over the two ranges gives the desired upper bound $\Delta^{(d)} \leq \frac{3}{(k-2)(d-1)+2}$, which finishes the proof of Claim 2. ∎

Since $Q_k^{(d)} = R_k - \Delta_k^{(d)}$, to finish the proof of the lemma it remains to show $R_k = \frac{\mu_k}{k-1}$ for $k \geq 3$.

For $k = 3$, we can explicitly solve for $R_3(r)$ to get

$$R_3(r) = \begin{cases} (\frac{r}{1-r})^2 & r < 1/2 \\ 1 & r \geq 1/2 \end{cases}$$

Integrating this from 0 to 1 yields $R_3 = 2 - 2\ln 2 \geq 0.6137$.

For general $k \geq 3$, we evaluate $\int_0^1 R_k(r)dr$ by a change of variables. As we noted, $R_k(r) = 1$ on $[\frac{k-2}{k-1}, 1]$, and is continuous and strictly increasing function on $[0, \frac{k-2}{k-1}]$ with inverse given by $S_k(t)$. One can then see (either geometrically, or by a change of variables) that $\int_0^1 R_k(r)dr = \int_0^1 (1 - S_k(t))dt$. We have:

$$1 - S_k(t) \quad = \quad 1 - \frac{t^{\frac{1}{k-1}} - t}{1-t} \quad = \quad \sum_{i=0}^{\infty} t^i - t^{i+\frac{1}{k-1}}$$

As $t^i - t^{i+\frac{1}{k-1}} \geq 0$ on $[0, 1]$, for all $i \geq 0$, the monotone convergence theorem allows us to evaluate the integral from 0 to 1 of this sum term by term:

$$\begin{aligned}
R_k &= \int_0^1 (1 - S_k(t))dt &= \sum_{i=0}^{\infty} \frac{1}{i+1} - \frac{1}{i+1+\frac{1}{k-1}} \\
&= \sum_{j=1}^{\infty} \frac{1}{j} - \frac{1}{j+\frac{1}{k-1}} &= \frac{1}{k-1}\sum_{j=1}^{\infty} \frac{1}{j(j+\frac{1}{k-1})} \quad = \quad \frac{\mu_k}{k-1}.
\end{aligned}$$

This finishes the proof of Lemma 5, hence also of Theorem 7.

4 General k-SAT

4.1 Overview

We now proceed to the analysis of general k-CNF formulas. Theorem 7 applies to any formula that has a sufficiently isolated satisfying assignment, but a satisfiable formula need not have such an assignment. Intuitively, though, if F has few satisfying assignments, then it should be close to the unique-SAT case, and if it has many satisfying assignments, then finding one should be easy. Our aim is to formalize this intuition.

As described in the Section 2, upper bounding the running time of **Search**(F_s, I) is accomplished by lower bounding $\tau(F_s)$, the probability that **Modify**(F_s, π, y) returns a satisfying assignment. In the uniquely satisfiable case, we proved such a lower bound, and we will prove a similar lower bound in the general case. However, several new technical difficulties arise.

As we did in the uniquely satisfiable case, we initially consider a general formula G and only later let $G = F_s$ for some formula F and positive integer s. The method for bounding $\tau(G)$ applied in the uniquely satisfiable case focused on the probability $\tau(G, z)$ of accepting a particular assignment z, where z was d-isolated. We need to refine this approach. We start with a simple combinatorial lemma. If a is a partial assignment to the variables $\{x_1, \ldots, x_n\}$, the *subcube defined by* a is the set of all assignments that extend a.

Lemma 9 *Let A be a nonempty set of assignments (i.e., points in $\{0,1\}^n$). Then $\{0,1\}^n$ can be partitioned into a family $(B_z : z \in A)$ of distinct disjoint subcubes so that $z \in B_z$ for each $z \in A$.*

Proof. If $|A| = 1$, the result is trivial. Otherwise, there are at least two assignments, and one variable u which occurs both as 0 and 1 in S. Split $\{0,1\}^n$ into two subcubes, one with $u = 0$ and one with $u = 1$, and recursively partition each. ■

If G is a satisfiable formula, we apply this lemma in the case that A is the set $sat(G)$ of satisfying assignments of the formula G, and we fix any such collection $\{B_z : z \in sat(G)\}$ of disjoint subcubes. We will analyze the probability $\tau(G)$ that **Modify**(G, π, y) finds some satisfying assignment by conditioning according to the subcube B_z that contains y. For satisfying assignments w and z write $\tau(G, w|B_z)$ for the probability that **Modify**(G, π, y) returns w given $y \in B_z$. We have:

$$\tau(G) \geq \sum_{z \in sat(G)} \tau(G, z|B_z)\mathbf{Prob}[y \in B_z] \geq \quad \min_{z \in sat(G)} \tau(G, z|B_z)$$

So, to lower bound $\tau(G)$, it suffices to consider a single fixed satisfying assignment z and the subcube $B = B_z$ that contains it, and then to give a lower bound on the conditional probability $\tau(G, z|B)$ that **Modify**(G, π, y) returns z given $y \in B$. In the analysis, the only property of B we will need is that it is a subcube containing z and no other satisfying assignment of G.

For the rest of this section z represents a fixed satisfying assignment of the formula G and B is a subcube containing z and no other satisfying assignment of G. (Later we will take G to be F_s for some formula F and positive integer s.) D denotes the set of variables that defines the subcube B (i.e., the set of variables which are constant over that subcube) and let N be the remaining variables. The variables in D are referred to as the *defining* variables of B and those in N are referred to as *nondefining*.

To lower bound $\tau(G, z|B)$, we try to generalize the argument leading to Lemma 3. Since $y \in B$, y agrees with z on the defining variables, so **Modify**(G, π, y) returns z if and only if the nondefining variables are set according to z. Writing $\mathrm{Forced}_z(G, \pi, y)$ for the set of nondefining variables in $\mathrm{Forced}(G, \pi, y)$ (i.e., $\mathrm{Forced}_z(G, \pi, y) = N \cap \mathrm{Forced}(G, \pi, y)$) we have the following generalization of Lemma 2:

$$\tau(G, z|B) = 2^{-|N|}\mathbf{E}[2^{|\mathrm{Forced}_z(G,\pi,z)|}]. \tag{7}$$

Continuing the argument, one obtains the following generalization of Lemma 3: if the average of $P(v, G, z)$ over nondefining variables is at least p then

$$\tau(G, z|B) \geq 2^{-(1-p)|N|}. \tag{8}$$

One might hope that restricting to the cube B in which z is the unique satisfying assignment, would reduce the analysis of the general case to the uniquely satisfiable case, where we proved $P(v, G, z) \geq$

$\frac{\mu_k}{k-1} - o(1)$. It is generally not true that $P(v, G, z) = P(v, G', z)$ where G' is obtained by fixing all of the defining variables to their values in B. Consider the following example. Let $C = \{v, x_1, x_2\}$ be a clause, let z be an assignment in which $v = 1, x_1 = x_2 = 0$ and let B be the subcube defined by $x_1 = 0$. If we substitute $x_1 = 0$ in the formula, C will be reduced to $\{v, x_2\}$. Hence, if we apply the algorithm to the reduced formula, v will be forced whenever x_2 precedes v. But if we apply our algorithm to the original formula, this does not suffice, we need that both x_1 and x_2 precede v. (It is important to keep in mind that the algorithm does not know z or the cube B and so it cannot first substitute for defining variables of B.)

So we still have some work to do. We do have that for a nondefining variable v, $P(v, G, z) \geq \frac{1}{k}$. This is because z is the unique satisfying assignment in B, hence flipping v produces a non-satisfying assignment. This implies that there is a critical clause for v at z, and v is last in that critical clause with probability at least $\frac{1}{k}$. Hence we get from (7):

Lemma 10 *Let G be a k-CNF formula, z a satisfying assignment and let B be a subcube of $\{0,1\}^n$ that contains z and no other satisfying assignment. Then:*

$$\tau(G, z|B) \geq 2^{-(1-\frac{1}{k})|N|}$$

Let us note, in passing, that this is enough to prove the bound for the Paturi, Pudlák and Zane algorithm [17]:

$$\tau(G) \geq \min_{z \in sat(G)} \tau(G, z|B) \geq 2^{-(1-\frac{1}{k})|N|} \geq 2^{-(1-\frac{1}{k})n}.$$

We shall not need this here. What we shall need, however, is the following observation. If the number of nondefining variables is strictly less than n, then the conditional probability $\tau(G, z|B)$ is strictly larger than $2^{-(1-\frac{1}{k})n}$. In particular, if $|D| \geq \delta n$ for some constant $\delta \geq 0$ then Lemma 10 gives $\tau(G) \geq 2^{-(1-\frac{1}{k})(1-\delta)n}$, thus we get a smaller constant than n in the exponent.

Thus, this observation suffices to get the bounds we want provided that the subcube under consideration is small enough (has enough defining variables), even without considering the gain from resolution. If the subcube under consideration has relatively few defining variables, we need to consider the critical clauses produced by resolution. We would like to use the critical clause tree of uniform depth d but there is a new problem. Recall the proof of Lemma 4, where to extend the tree from a given leaf b_i we used a clause C_i that was not satisfied by $z \oplus var(P_i)$, and the existence of such a clause was guaranteed by the fact that $z \oplus var(P_i)$ does not satisfy the formula (because of the d-isolation of z). Now, however, such a clause C_i need not exist—since z need not be d-isolated, $z \oplus var(P_i)$ might satisfy G (because it might be outside of the subcube B). But we do know that if $var(P_i)$ consists only of nondefining variables then $z \oplus var(P_i)$ does not satisfy G since z is the only satisfying assignment in B. We now modify the rule for building the tree to say that we never try to expand the tree from a leaf labeled by a defining variable. In this way, we maintain the property that defining variables appear only at leaves. By the above observation, it is always possible to expand any other leaf. Thus we deduce the following counterpart to lemma 4.

Lemma 11 *Let F be a k-CNF formula, z an arbitrary satisfying assignment, and let B be a subcube of $\{0,1\}^n$ containing z and no other satisfying assignment of F. If v is any nondefining variable of B, and d is any positive integer, and $s \geq k^d$, then there exists a critical clause tree for (v, F_s, z) of maximum degree $k-1$ such that*

(i) the only nodes labeled by defining variables of B are leaves,

(ii) any leaf labeled by a nondefining variable of B is at depth d.

A tree satisfying the conclusion of the lemma is said to be of *uniform depth d with respect to the set N*. Such trees are nice, but not good enough to give directly a result such as Lemma 5. For instance, it could be that the tree consists of a root together with $k-1$ children all labeled by defining variables. (The $k-1$ children of the root always exist; we have used this fact in Lemma 10.) In this case, we get only $1/k$ as the probability Q_T, leading to a bound no better than that given in Lemma 10. As we noted, this bound gives a good improvement on the unique-sat analysis if $|D|$ is a large enough fraction of n. The “bad case” for this bound is when D is not too big a fraction of n and there are many shallow leaves labeled by defining variables.

Thus the lower bound of (8) does not seem strong enough to derive running times for the general case better than the $2^{(1-1/k)n}$ bounds of [17]. So we need to return to (7). There we had $\tau(G, z|B) = 2^{-|N|}\mathbf{E}[2^{|\mathrm{Forced}_z(G,\pi,z)|}]$ and the next step was to use convexity to say $\mathbf{E}[2^{|\mathrm{Forced}_z(G,\pi,z)|}] \geq 2^{\mathbf{E}[|\mathrm{Forced}_z(G,\pi,z)|]}$. The problem is that this step gave too much away. For a random variable X, the inequality $\mathbf{E}[2^X] \geq 2^{E[X]}$ is tight if X is constant and is very loose if X has a significant probability of being much larger than the mean. This may well be the case for the random variable $|\mathrm{Forced}_z(G,\pi,z)|$. Since $|\mathrm{Forced}_z(G,\pi,z)|$ only counts nondefining variables that are forced by critical clauses, it can only increase if the order π is replaced by an order π' having the same relative order of the nondefining variables and of the defining variables as π but whose defining variables are pushed closer to the beginning of the order. Thus, $|\mathrm{Forced}_z(G,\pi,z)|$ will have a higher average when π is restricted to orderings with the defining variables placed first.

This leads to the key idea for handling the general case. We will improve our lower bound on $\mathbf{E}[2^{|\mathrm{Forced}_z(G,\pi,z)|}]$ by identifying a set Γ of placements having reasonably large probability in which the defining variables tend to appear early, so that the conditional expectation $|\mathrm{Forced}_z(G,\pi,z)|$, given that $\alpha \in \Gamma$ is much larger than the overall average. We have the trivial lower bound:

$$\mathbf{E}[2^{|\mathrm{Forced}_z(G,\pi,z)|}] \geq \mathbf{E}_\Gamma[2^{|\mathrm{Forced}_z(G,\pi,z)|}]\mathbf{Prob}[\alpha \in \Gamma],$$

where $\mathbf{E}_\Gamma$ denotes the conditional expectation given that $\alpha \in \Gamma$. Letting $P_\Gamma(v, G, z)$ denote the conditional probability that $\pi \in \mathrm{Last}(v, G, z)$ given $\alpha \in \Gamma$ we obtain a straightforward generalization of Lemma 3.

Lemma 12 *Let Γ be a (Lebesgue measurable) subset of placements. For any satisfying assignment z of the CNF formula G if $P_\Gamma(v, G, z) \geq p$ for all nondefining variables v of B, then*

$$\tau(G, z|B) \geq 2^{-(1-p)|N|}\mathbf{Prob}[\alpha \in \Gamma].$$

The potential advantage of this lemma over (8) is that the quantity p which was a lower bound on $P(v, G, z)$ is now a lower bound on $P_\Gamma(v, G, z)$ which may be much larger. On the other hand, this potential increase in $2^{-(1-p)|N|}$ is attained at the cost of the new factor $\mathbf{Prob}[\alpha \in \Gamma]$, which is why we need Γ to have ”large” probability.

So we want to choose a suitable Γ. There are various ways that one might try to define such a biased set Γ of placements. Here we will begin by analyzing a simple and natural choice. This choice enables us to prove bounds on the running time for our algorithm for general k-SAT that match the bounds for the running time for unique k-SAT for all $k \geq 5$ and nearly match the bounds for $k = 4$. For $k = 3$, however, there is a substantial gap between the general 3-SAT bounds obtained from the simple choice of Γ and the unique 3-SAT bounds.

To motivate our initial choice of Γ, let us start by considering an example. Suppose that instead of getting a tree such as T, see (3), we get only the following one

$$T' = \quad \begin{array}{ccccc} & & v & & \\ x_2 & & & x_5 & \\ & & x_4 & & x_6 \end{array}$$

because x_2 is a defining variable. Hence, instead of the bound (4) for the tree T, we get only

$$p_2(1 - (1 - p_4 p_6)(1 - p_5)).$$

This truncated subtree under x_2 contributes less than we need towards forcing v. However, if we condition on x_2 coming earlier in the ordering, then we can compensate for this lost contribution. More precisely, we take a suitable subset of placements Γ such that the probability that x_2 occurs before v is higher and then we get higher probability that v is forced.

With this motivation, we now define the set Γ. Γ depends on the set D of defining variables of the subcube B and also on a parameter $\theta \in [0, 1]$ that we will choose later. $\Gamma = \Gamma(D, \theta)$ is the placements is defined to be the set of placements α satisfying $\alpha(v) \leq \theta$ for all $v \in D$.

We want to use this choice of Γ in Lemma 12 to lower bound $\tau(G, z|B)$. $\mathbf{Prob}[\alpha \in \Gamma] = \theta^{|D|}$ so to apply the lemma, we need only give a lower bound on $P_\Gamma(v, G, z)$ that holds for all nondefining variables v. For this we need a counterpart to Lemma 6. For an admissible tree T, let $Q_{T,\Gamma}(r) = \mathbf{Prob}_\Gamma[\mathrm{Cut}_T(r)]$ and $Q_{T,\Gamma} = \mathbf{Prob}_\Gamma[\mathrm{Cut}_T]$. Exactly as before we have:

$$P_\Gamma(v, G, z) \geq Q_{T,\Gamma} = \int_0^1 Q_{T,\Gamma}(r)dr.$$

In the previous section we used Lemma 7 and induction to show that for a critical clause tree of uniform depth d, $Q_T \geq Q_k^{(d)}$. Now the tree T we are dealing with is only uniform depth d with respect to the set N of nondefining variables. This complicates the induction. As suggested in the example above, the conditioning Γ is used to compensate for this. In the following lemma we prove a lower bound for $Q_{T,\Gamma}$. For this bound we need to fix the parameter θ to be $(k-2)/(k-1)$.

Lemma 13 *Let F, z, B, D, N be as above. Suppose $v \in N$ and that T is a critical clause tree of degree $k-1$ and of uniform depth d with respect to N. Let $\Gamma = \Gamma(D, \frac{k-2}{k-1})$.*

Then for each $r \in [0, 1]$

$$Q_{T,\Gamma}(r) \geq Q_k^d(r)$$

and therefore

$$Q_{T,\Gamma} \geq Q_k^d.$$

Proof. It suffices to fix r and prove the first inequality. We use induction on d. If $d = 0$, the conclusion is true since $Q_k^{(0)}(r) = 0$. So assume $d \geq 1$ and that T is of uniform depth d with respect to N. Let $T_1, \ldots, T_t$ be the subtrees rooted at labeled children of the root (where $t \leq k-1$), let v_i be the label of T_i and let $v_1, \ldots, v_s$ be those labels in D. This implies that the trees T_i for $i \leq s$ consist of the single node labeled v_i.

We apply Lemma 7, which was stated for the general case that the placement of variables is done according to arbitrary independent distributions. For $i \in \{1, \ldots, s\}$, the probability that $\alpha(v_i) \leq r$ conditioned on Γ is $\min\{1, (k-1)r/(k-2)\}$. Then by Lemma 7 and the induction hypothesis:

$$\begin{aligned}
Q_{T,\Gamma}(r) &\geq (\min\{1, r(k-1)/(k-2)\})^s \prod_{i=s+1}^{t} [r + (1-r)Q_{T_i,\Gamma}(r)] \\
&\geq (\min\{1, r(k-1)/(k-2)\})^s (r + (1-r)Q_k^{d-1}(r))^{t-s} \\
&\geq (\min\{1, r(k-1)/(k-2)\})^s Q_k^d(r)^{(t-s)/(k-1)}.
\end{aligned}$$

To complete the proof we now show that that $\min\{1, r(k-1)/(k-2)\} \geq R_k(r)^{1/(k-1)}$ (where $R_k(r)$ is as defined in the proof of Lemma 8). Since $R_k(r) \geq Q_k^d(r)$ (Claim 2 within the proof of Lemma 8), we can then lower bound the last expression of the above chain of inequalities by $Q_k^d(r))^{t/(k-1)} \geq Q_k^d$.

To prove $\min\{1, r(k-1)/(k-2)\} \geq R_k(r)^{1/(k-1)}$ we assume, $r \leq (k-2)/(k-1)$ since otherwise the result is trivial. Since $R_k(r)$ is the smallest non-negative x satisfying $f_k(x; r) - x = 0$ where $f_k(x; r) =$

$(r + (1-r)x)^{(k-1)}$, it is sufficient to show that there is an $x \in [0, (r(k-1)/(k-2))^{k-1}]$ satisfying $f_k(x; r) - x = 0$.

Since $f_k(x;r) - x$ is continuous and is positive at $x = 0$ we can infer the existence of the desired x if we show that $f_k(x;r) - x < 0$ at $x = (r(k-1)/(k-2))^{k-1}$, and this inequality is equivalent to:

$$1 + (1-r)r^{k-2}((k-1)/(k-2))^{k-1} \leq (k-1)/(k-2).$$

The expression on the left hand side achieves its maximum value of $(k-1)/(k-2)$ at $r = (k-2)/(k-1)$ on the interval $[0, (k-2)/(k-1)]$.

Thus $\min\{1, r(k-1)/(k-2)\} \geq R_k(r)^{1/(k-1)}$, as required to prove the lemma. ■

We deduce the following lower bound on $\tau(F_s, z, B)$.

Corollary 14 *Let F be an n-variable k-CNF formula with $k \geq 3$. Let $s = s(n)$ be any function tending to ∞ with n. Let z, B, D, N be as above and let $\delta = |D|/n$. Then*

$$\tau(F_s, z|B) \geq 2^{-(1-\frac{\mu_k}{k-1})n} \left(\frac{(k-2)2^{1-\frac{\mu_k}{k-1}}}{k-1} \right)^{\delta n} 2^{o(n)}.$$

Proof. Let $d = \lceil \log_k s \rceil$. By lemma 11, there is a critical clause tree T for (v, F_s, z) of maximum degree $k-1$ that is uniform of depth d with respect to N. By lemma 13,

$$\tau_\Gamma(v, G, z) \geq Q_{T,\Gamma} \geq Q_k^d.$$

Using lemma 12 we obtain:

$$\begin{aligned} \tau(F_s, z|B) &\geq 2^{-(1-Q_k^d)(1-\delta)n} \left(\frac{k-2}{k-1} \right)^{\delta n} && (9) \\ &\geq 2^{-(1-\frac{\mu_k}{k-1})n} \left(\frac{(k-2)2^{1-\frac{\mu_k}{k-1}}}{k-1} \right)^{\delta n} 2^{o(n)}. && (10) \end{aligned}$$

The last inequality comes from lemma 8 which says that $\frac{\mu_k}{k-1} - Q_k^d \leq \epsilon_k^{(d)}$, which tends to 0 as d gets large. ■

We can now complete the analysis of the algorithm in the case that $k \geq 5$.

Proof of Theorem 1(i). Recall that we have an n-variable k-CNF formula F and $s = s(n)$ is a function tending to infinity with n. We want to bound $\tau(F_s)$ from below. As shown following Lemma 9, it suffices to give a bound $\tau(F_s, z|B)$ that holds for every choice of a satisfying assignment z and subcube B that contains z and no other satisfying assignment. If δ is the fraction of defining variables of B then corollary 14 implies:

$$\tau(F_s, z|B) \geq 2^{-(1-\frac{\mu_k}{k-1})n} \left(\frac{(k-2)2^{1-\frac{\mu_k}{k-1}}}{k-1} \right)^{\delta n} 2^{o(n)}.$$

Now for $k \geq 5$, the quantity $\frac{(k-2)2^{1-\frac{\mu_k}{k-1}}}{k-1}$ is more than 1, which is easy to check using the fact that for all k, $\mu_k \leq \sum_{j \geq 1} 1/j^2 = \pi^2/6$. Thus the above lower bound on $\tau(F_s, z|B)$ is smallest when $\delta = 0$, and taking $\delta = 0$ gives the lower bound claimed by the theorem. ■

Proofs of Theorems 2 and 3. When $k = 3$ and $k = 4$, the quantity

$$\frac{(k-2)2^{1-\frac{\mu_k}{k-1}}}{k-1}$$

is less than 1, which means that the bound on $\tau(F_s, z|B)$ is minimized when $\delta = 1$. But we can do better than just subsitute $\delta = 1$ into this bound. Lemma 10 gives the alternative bound:

$$\tau(F_s, z|B) \geq 2^{-(1-1/k)(1-\delta)n}.$$

Since this bound increases with δ and the other decreases with δ, the minimum of the maximum of the two of them occurs when δ is chosen to make them equal. This occurs when we choose:

$$\delta = \frac{\frac{\mu_k}{k-1} - \frac{1}{k}}{\frac{\mu_k}{k-1} - \frac{1}{k} + \log_2 \frac{k-1}{k-2}}.$$

Substituting this value of δ into either bound gives

$$\tau(F_s, z|B) \geq 2^{-\zeta_k n},$$

where

$$\zeta_k = \frac{(1-\frac{1}{k})\log_2 \frac{k-1}{k-2}}{\frac{\mu_k}{k-1} - \frac{1}{k} + \log_2 \frac{k-1}{k-2}}.$$

When $k = 4$, one can calculate $\zeta_4 \leq 0.5625$. We conclude that $\tau(F_s) \geq 2^{-0.5625n}$ which proves Theorem 3. Note that ζ_4 is only slightly larger than $1 - \frac{\mu_4}{3} \approx .555$ which is what we'd need to match the results for unique 4-SAT.

When $k = 3$, one can calculate $\zeta_3 \leq 0.521$. This gives $\tau(F_s) \geq 2^{-0.521n}$ which is still pretty far from the lower bound $2^{-0.386n}$ for unique 3-SAT. ■

We know that it is possible to improve the bound for the case of $k = 3$ using a more refined choice of Γ, but we do not know the best bound on the running time of our algorithm. In particular, we do no know, whether it runs faster than that of [11] and whether it runs asymptotically the same time as for the unique 3-SAT.

5 Lower Bounds for Depth-3 Circuits

In this section we prove Theorems 4, 5 and 6.

In proving lower bounds on the size of Σ_k^3 and Σ^3 circuits needed to compute specific boolean functions, we follow the general approach used in [17]. In that paper, the key observation was that a lower bound on the probability that $\mathbf{Modify}(G, \pi, y)$ returns a satisfying assignment places an upper bound on the number of isolated solutions that a k-CNF formula can have. This, in turn, is used to prove a lower bound on the size of depth 3 circuits. Here we use Theorem 7 to prove the stronger bound of Theorem 4.

Proof of Theorem 4. First consider the case that $L = 0$, which means that F is a k-CNF. By Theorem 7, if $s = k^d$ then for each d-isolated satisfying assignment z to F, the probability $\tau(F_s, z)$ that **Modify**(F_s, π, y) outputs z is at least $2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}$. Since **Modify**$(G, \pi, y)$ outputs at most one assignment, we have $1 \geq \sum_z \tau(F_s, z) \geq I_d(F)2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}$. The upper bound on $I_d(F)$ follows.

Now let F be a general CNF. Write $F = F_{small} \wedge F_{large}$ where F_{large} consists of the clauses of size more than k. For each clause C of F_{large} let Z_C be the set of points that do not satisfy C and let $Z = \bigcup_{C \in F_{large}} Z_C$. Note that $F^{-1}(1) = F^{-1}_{small}(1) - Z$. If $x \in I_d(F)$ then
(i) $x \in I_d(F_{small})$, or
(ii) there is a point $x' \in Z$ that is within d of x.

The number of points satisfying (i) is at most $2^{(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}$ by the $L = 0$ case. To upper bound the number of points satisfying (ii), note that each point $x' \in Z$ has at most n^d points within distance d of it. Since for a large clause C, $|Z_C| = 2^{n-|C|} < 2^{n-k}$, we have that the number of points satisfying (ii) is at most $L2^{n-k}n^d$ as required. ■

Before proving Theorem 4 we review some simple facts about depth 3 circuits. We may assume that every Σ^3 circuit is constructed in a levelled fashion: the inputs are connected to OR gates, these OR gates are connected to AND gates, and these AND gates are connected to the output OR gate. A circuit which does not have this property can be easily converted into this form. Because the gates have unbounded fan-in, two gates of the same type which are connected can simply be collapsed into one gate. If an input x_i is directly connected to an AND gate or to the output gate, we modify the circuit to maintain this levelled property by introducing new gates of fan-in one at the levels which it skips. Since this process introduces at most $O(n)$ gates, it has negligible impact on the exponential lower bounds we prove, and we may assume that all circuits are levelled.

Call the AND gates of the circuit *CNF gates*, since they compute functions which are ANDs of ORs of inputs, and call the circuit rooted at such a gate a *CNF subcircuit*. Similarly, call the non-output OR gates of the circuit *clause gates*. As usual, given a circuit $\mathcal{C}$ or a formula F, we use $\mathcal{C}(x)$ and $F(x)$ to denote the Boolean functions they compute.

Theorem 4 also implies an upper bound on the number of d-isolated points accepted by any Σ^3 circuit in terms of the number of gates:

Lemma 15 *Let $\mathcal{C}$ be an Σ^3 circuit on n variables and let k, d be positive integers with $k \geq 3$. Let Q be the number of CNF gates and R be the number of clause gates having fan-in more than k. Then*

$$I_d(\mathcal{C}) \leq 2^n \left(Q2^{(-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n} + R2^{-k}n^d \right).$$

Proof. Let $G_1, \ldots, G_Q$ be the CNF gates of $\mathcal{C}$. Each d-isolated point of $F^{-1}(1)$ is accepted by at least one G_i and is still d-isolated in $G^{-1}(1)$. As in the proof above, divide each G_i into $G_{i,small} \wedge G_{i,large}$. The number of d-isolated inputs that are accepted by some $G_{i,small}$ is

$$\leq \sum_{i=1}^{Q} I_d(G_{i,small}) \leq Q2^{(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}.$$

The number of the others is $\leq R2^{n-k}n^d$, where $R = |\bigcup_i G_{i,large}|$. ■

An immediate consequence of Lemma 15 is:

Corollary 16 *Let F be boolean function with n variables and let $k \geq 3$. Any Σ^3_k circuit computing F has at least*

$$I_d(F)2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}$$

CNF gates.

This gives immediately our first lower bound on depth 3 circuits.

Proof of Theorem 5. Recall that we want to prove a lower bound on the smallest Σ_k^3 circuit computing membership in an error correcting code E with $|E| \geq 2^{n-(n/\log n)}$ and distance greater than $\log n$. For $d = \lfloor \log n \rfloor$, every element of E is d-isolated and so by Corollary 16, any circuit computing $\mathcal{C}$ has at least $|E| 2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}$ gates. Since $\epsilon_k^{(d)} = \frac{3}{(d-1)(k-2)+2} \leq \frac{3}{(d-1)+2} \leq \frac{3}{d+1} \leq \frac{3}{\log n}$ we conclude that $\mathcal{C}$ has at least $2^{\frac{\mu_k}{k-1}n - \frac{4n}{\log n}}$ gates. ■

Finally we consider lower bounds on Σ^3 circuits with no restriction on fan-in. Again we first prove a lower bound on the number of satisfying assignments and then derive a lower bound on the number of gates in the circuit.

Corollary 17 *Let F be a boolean function with n variables. Any Σ^3 circuit computing F has at least*

$$I_d(F) 2^{-n+\min\{(\frac{\mu_k}{k-1}-\epsilon_k^{(d)})n, k-d\log n\}}$$

gates.

Proof. By Lemma 15, for any circuit $\mathcal{C}$, $I_d(F) \leq 2^n(Q+R)\max\{2^{(-\frac{\mu_k}{k-1}+\epsilon_k^{(d)})n}, 2^{-k+d\log n}\}$. Since $Q+R$ lower bounds the number of gates of $\mathcal{C}$, the corollary follows. ■

Proof of Theorem 6. Again we are looking at the membership function for the code E. In applying Corollary 17, we are free to choose k to make $\min\{(\frac{\mu_k}{k-1} - \epsilon_k^{(d)})n, k - d\log n\}$ as large as possible.

First we note:

$$\frac{\pi^2}{6} - \mu_k = \sum_{j=1}^{\infty} \frac{1}{j^2} - \frac{1}{j(j+\frac{1}{k-1})} = \sum_{j=1}^{\infty} \frac{\frac{1}{j(k-1)}}{j(j+\frac{1}{k-1})} \leq \sum_{j=1}^{\infty} \frac{\frac{1}{k-1}}{j(j+\frac{1}{k-1})} = \frac{\mu_k}{k-1}$$

and so $\frac{\mu_k}{k-1} \geq \frac{\pi^2}{6k}$

As for $\epsilon_k^{(d)}$, as long as k and d are both growing functions of n, then for sufficiently large n, $\epsilon_k^{(d)} = \frac{3}{(d-1)(k-2)+2} \leq \frac{4}{dk}$.

Since $d = \log n$, the first term in the min is at least $\frac{n}{k}(\frac{\pi^2}{6} - \frac{4}{\log n})$. and the second is at least $k - \log^2 n$. Choose $k = \lfloor \sqrt{\frac{\pi^2 n}{6}} \rfloor$. Then both terms in the min are at least $\sqrt{\frac{\pi^2 n}{6}} - \frac{4\sqrt{n}}{\log n}$. Since $I_d(E) = |E| \geq 2^{n-\sqrt{n}/\log n}$, Corollary 17 gives a circuit size lower bound

$$2^{\sqrt{\frac{\pi^2 n}{6}} - \frac{5\sqrt{n}}{\log n}}$$

proving the theorem. ■

6 Acknowledgements

We thank Professor Ed Bender for his help in evaluation the integral in Section 3.3. We also thank Yuming Zhang for helpful suggestions. We are grateful to the reviewers for their excellent reviews and constructive suggestions.

References

[1] Alon, N., Spencer, J., and Erdös, P., (1992), "The Probabilistic Method", John Wiley & Sons, Inc.

[2] Cover, T. and Thomas, J. (1991), "Elements of Information Theory", New York: Wiley.

[3] Dantsin, E., Goerdt, A., Hirsch, E. A., Kannan, R., Kleinberg, J., and Papadimitriou, C., and Raghavan, P., and Schöning, U.", A deterministic $(2 - \frac{2}{k+1})^n$ algorithm for k-SAT based on local search, *Theoretical Computer Science*, vol. 289, No.1, pp. 69–83, 2002.

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

[5] Galil, Z. (1975) On the validity and complexity of bounded resolution. *in* "Proceedings of the 7th ACM Symposium on Theory of Computing," pp. 72–82, May 1975.

[6] Håstad, J., (1986), Almost Optimal Lower Bounds for Small Depth Circuits, *in* " Proceedings of the 18th ACM Symposium on Theory of Computing", pp. 6–20.

[7] Håstad, J., Jukna, S., and Pudlák, P., (1993), Top–Down Lower Bounds for Depth 3 Circuits, "Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science", pp. 124–129.

[8] Harris, T.E. (1960), A lower bound for the critical probability in a certain percolation process, *Proc. Camb. Phil. Soc.*, 56: 13–20.

[9] Hirsch, E. (2000) New worst-case upper bounds for SAT. *Journal of Automated Reasoning*, 24(4):397–420.

[10] Hirsch, E. and Kojevnikov, A. Unit Walk: A new SAT solver that uses local search guided by unit clause elimination, "Proceedings of Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2002)", pp.35–42

[11] Hofmeister, T., Schöning, U., Schuler, R. and Watanabe, O. A probabilistic 3–SAT algorithm further improved. STACS 2002, LNCS 2285, Springer 2002, pp. 192–202.

[12] Kullmann, O. New methods for 3-SAT decision and worst-case analysis. (1999) *Theoretical Computer Science*, 223(1-2):1–72, July 1999.

[13] Kullmann, O. and Luckhardt, H. (1998) Algorithms for SAT/TAUT decision based on various measures. Preprint, 81 pages `http://www.cs.utoronto.ca/ kullmann/`.

[14] MacWilliams, F.J. and Sloane, N.J., The Theory of Error-Correcting Codes, North-Holland, 1977.

[15] Monien, B. and Speckenmeyer, E., (1985), Solving Satisfiability In Less Than 2^n Steps, Discrete Applied Mathematics **10**, pp. 287–295.

[16] Paturi, R., Pudlák, P., Saks, M. and Zane, F., An Improved Exponential-time Algorithm for k-SAT, *in* Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (1998), pp. 628–637

[17] Paturi, R., Pudlák, P., and Zane, F., Satisfiability Coding Lemma, *in* Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (1997), pp 566–574; Chicago Journ. of Theor. Comput. Science, December 1999, `http://cjtcs.cs.uchicago.edu/`

[18] Paturi, R., Saks, M.E., and Zane F., (1997), Exponential Lower Bounds on Depth 3 Boolean Circuits, *in* Proceedings of the 29th Annual ACM Symposium on Theory of Computing", pp. 86–91

[19] Price, K. and Storn, R. (1997), Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization **11**, pp. 341–359.

[20] Razborov, A.A. (1986), Lower Bounds on the Size of Bounded Depth Networks over a Complete Basis with Logical Addition, *Matematicheskie Zametki* **41** pp. 598–607 (in Russian). English Translation in*Mathematical Notes of the Academy of Sciences of the USSR* **41**, pp. 333–338.

[21] Schiermeyer, I. (1993), Solving 3-Satisfiability in less than 1.579^n Steps, *in* Selected papers from CSL '92, LNCS Vol. 702, pp. 379–394.

[22] Schöning, U. (1999), A probabilistic algorithm for k-SAT and constraint satisfaction problems. 40th Annual Symposium on Foundations of Computer Science (FOCS '99), pp. 410–414. IEEE.

[23] Valiant, L.G., (1977), Graph–theoretic arguments in low–level complexity, in *Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science*, Springer–Verlag, Lecture Notes in Computer Science, vol. 53, pp. 162–176.

[24] Yao, A. C–C. (1985), Separating the Polynomial Hierarchy by Oracles, *in* "Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science", pp. 1–10.

[25] Zhang, W. (1996), Number of models and satisfiability of sets of clauses, Theoretical Computer Science **155**, pp. 277–288.