An Improved Exponential-time Algorithm for k-SAT

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

Abstract

We propose and analyze a simple new algorithm for finding satisfying assignments of Boolean formulae in conjunctive normal form. The algorithm, ResolveSat, is a randomized variant of the DLL (Davis, Longeman and Loveland) [2] or Davis-Putnam procedure. Rather than applying the DLL procedure to the input formula F, however, ResolveSat enlarges F by adding additional clauses using limited resolution before performing DLL. The basic idea behind our analysis is the same as in [6]: a critical clause for a variable at a satisfying assignment gives rise to a unit clause in the DLL procedure with sufficiently high probability, thus increasing the probability of finding a satisfying assignment. In the current paper, we analyze the effect of multiple critical clauses (obtained through resolution) in producing unit clauses. We show that, for each k, the running time of ResolveSat on a k–CNF formula is significantly better than 2^n, even in the worst case. In particular, we show that the algorithm finds a satisfying assignment of a general 3–CNF in time $O(2^{.446n})$ with high probability; where the best previous algorithm [9, 10] has running time $O(2^{.582n})$. We obtain a better upper bound of $O(2^{(2\ln 2-1)n+o(n)}) = O(2^{0.387n})$ for 3-CNF that have at most one satisfying assignment (unique k-SAT). For each k, the bounds for general k-CNF are the best known for the worst-case complexity of finding a satisfying solution for k-SAT. As in [6], the idea of succinctly encoding satisfying solutions can be applied to obtain lower bounds on circuit size. Here, we exhibit a function f such that any depth–3 AND-OR circuit with bottom fan–in bounded by k requires $\Omega(2^{c_k n/k})$ gates (with $c_k > 1$). This is the first such lower bound with $c_k > 1$.

*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 grant no. A1019602 of the Academy of Sciences of the Czech Republic, and grant INT-9600919/ME 103(1997) under the cooperation of MŠMT, Czech Republic and NSF, USA

‡Research supported by NSF grant CCR-9700239. Department of Mathematics, Rutgers University, New Brunswick, NJ. saks@math.rutgers.edu.This work was done while on sabbatical at University of Washington.

§Bell Laboratories, Lucent Technologies

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 [5] gave the first real improvement by giving a simple algorithm whose running time is $2^{(1-\varepsilon_k)n}$, with $\varepsilon_k > 0$ all k. Algorithms with increasingly better running times (larger values of ε_k) have been analyzed recently. Prior to this paper, the best known algorithm for $k = 3$ was a deterministic algorithm due to Schiermeyer with running time $O(2^{.582n})$ [9, 10]. For $k > 3$, the randomized algorithm in [6] was the fastest known, with a running time of $O(2^{(1-1/k)n})$. In this paper, we present and analyze a very simple randomized algorithm, called ResolveSat, for finding satisfying assignments of k–CNF formula which improves known algorithms for all values of k. In particular, we show that the algorithm finds a satisfying assignment of any satisfiable 3-CNF formula in time $O(2^{.446n})$. The contribution of this paper include a fairly simple algorithm for satisfiability and and an intricate analysis of its running time.

1.2 The Algorithm

First, we need a few definitions. 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 $vars(C)$ for the set of variables appearing in C. If $v \in vars(C)$, 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 $F' = F|a$ 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.

The following simple subroutine takes as input an arbitrary assignment and tries to modify it to a satisfying assignment of formula F by considering the variables one by one, in the order given by permutation π.

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

The algorithm **Search** is obtained by running **Modify** on many random permutations.

```
Search(CNF-formula $F$, integer $I$)
  repeat $I$ times
    $\pi$ = uniformly random permutation of $1, \ldots, n$
    $y$ = uniformly random vector $\in \{T, F\}^n$
    $z$ = Modify$(F, \pi, y)$;
    if $z$ satisfies $F$
      then output($z$); exit;
  end/* end repeat loop */
  output('Unsatisfiable');
```

The algorithm **Search** was analyzed in [6]; we summarize the results in Theorem 1. 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* if $|R(C_1, C_2)| \leq s$. The following function 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 $(F_s)$.
```

In this paper we analyze the following simple combination of **Resolve** and **Search**:

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

1.3 The algorithm Search

The algorithm **Search** was analyzed in [6]. It is easily seen that **Search**(F, I) runs in time $O(I|F|\text{poly}(n))$. It is also clear that **Search**(F, I) always answers "unsatisfiable" if F is unsatisfiable, and the problem of interest is to upper bound the probability that it answer "unsatisfiable" on a satisfiable formula. For a formula F and assignment z write $\tau(F, z)$ to be the probability, over random π and y, that **Modify**(F, π, y) returns the assignment z. Define $\tau(F)$ to be the sum of $\tau(F, z)$ over z that satisfy F i.e., $\tau(F)$ is 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 $|I| \geq n/\tau(F)$. The main result about **Search** in [6] is:

Theorem 1 *For any satisfiable k-CNF formula F on n variables, $\tau(F) \geq 2^{-(1-\frac{1}{k})n}$. Thus the algorithm* **Search** *with $I = n2^{(1-\frac{1}{k})n}$ has error probability $O(e^{-n})$ and runs in time $O(2^{(1-\frac{1}{k})n}\text{poly}(n))$.*

In particular, for $k = 3$ the running time is $O(2^{\frac{2}{3}n}\text{poly}(n))$ which is not as good as the $O(2^{.582n})$ algorithm of [9, 10] mentioned in the introduction, but for $k \geq 4$, Theorem 1 gave the best previously known upper bound.

The analysis of **Search** in [6] proceeds in two steps. First, Theorem 1 for is proved for the case of uniquely satisfiable F. Then an averaging argument is used to prove the result for all satisfiable F. The key idea in [6] is a simple relationship between the structure of the formula and the structure of the space of satisfying solutions expressed in terms of *critical clauses* and *isolated solutions*.

It is also shown in [6] that the succinct description of *isolated solutions* used in proving Theorem 1 for uniquely satisfiable F can also be used to obtain better lower bounds on the size of depth 3 boolean circuits needed to compute certain functions.

1.4 Main results

In this paper we generalize the approach of [6] to analyze **ResolveSat**(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 and can be implemented easily in time $O((n^{2s})|F|\text{poly}(n))$ (this can be improved, but suffices for our purposes). **Search**(F_s, I) runs in time $O(I(|F| + n^s)\text{poly}(n))$. Hence the overall running time of **ResolveSat**(F, s, I) is crudely bounded from above by $I(|F| + n^{2s})\text{poly}(n)$. Provided that $s = o(n/\log n)$, $n^{2s} = 2^{o(n)}$ and we can bound this by $I|F|2^{o(n)}$. 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 $o(n/\log n)$.

The error probability of **ResolveSat**(F, s, I), is just the error probability of **Search**(F_s, I) which, as noted above, is $(1 - \tau(F_s))^I$. So in extending the analysis of **Search** to **ResolveSat**, we want to lower bound $\tau(F_s)$ rather than $\tau(F)$. Trivially, $\tau(F_s) \geq \tau(F)$; the main contribution of this paper is to provide a way to quantify the advantage provided by the multiple critical clauses provided by the resolution preprocessing step. Following the approach of [6], we first upper bound $\tau(F_s)$ for *uniquely satisfiable formulae*. For $k \geq 1$, define:

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

Theorem 2 *Let $k \geq 3$, let $s(n)$ be a slowly growing function. Then for any uniquely 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 uniquely satisfiable k-CNF formula.*

It is not hard to show that $\mu_3 = 4 - 4\ln 2 > 1.226$, and hence for $k = 3$ the running time of the algorithm on any uniquely satisfiable 3-CNF formula is at most $2^{.387n+o(n)}$. It is easily seen that μ_k is an increasing function of k, and for large k, μ_k approaches $\sum_{j=1}^{\infty} \frac{1}{j^2} = (\frac{\pi^2}{6}) \approx 1.644$.

Next we consider general k-CNF formulas. In this case, we are able to extend the result for the uniquely satisfiable case, provided that $k \geq 5$:

Theorem 3 *Let $k \geq 5$, let $s(n)$ be a slowly growing function. 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 $k \geq 5$.*

For the case that $k = 3, 4$, we have an improved constant in the exponent although we have not matched the result we obtained for unique 3-SAT.

Theorem 4 *Let $s = s(n)$ be a slowly growing function. For any satisfiable n variable 3-CNF formula, $\tau(F_s) \geq 2^{-0.446n}$ and so* **ResolveSat**(F, s, I) *with $I = n2^{0.446n}$ has error probability $o(1)$ and running time $2^{0.446n+o(n)}$.*

Theorem 5 *Let $s = s(n)$ be a slowly growing function. For any satisfiable n variable 3-CNF formula, $\tau(F_s) \geq 2^{-0.562n}$ and so* **ResolveSat**(F, s, I) *with $I = n2^{0.562n}$ has error probability $o(1)$ and running time $2^{0.562n+o(n)}$.*

As in [6], the main lemmas used to prove our results can be used to prove lower bounds on the size of depth-3 circuits needed to compute certain functions. Consider the class of such circuits with bottom fan-in bounded by k. Previous techniques [12, 3, 8, 7] do not give lower bounds better than $2^{n/k}$ for $k \geq 3$. In this paper, we obtain a modest improvement over the existing bounds by exhibiting an explicit boolean function for which we prove a $2^{c_k n/k}$ lower bound where $c_k > 1$ is bounded away from 1 for all k.

One motivation for studying lower bounds on unbounded fan-in depth-3 OR-AND-OR circuits is that strong lower bounds in this model (even with bottom fan-in limited by n^{ε}) imply nonlinear bounds on the number of gates required by fan-in 2 logarithmic depth AND-OR-NOT circuits [11]. The computational model, circuits with linearly many fan-in 2 Boolean gates and logarithmic depth, is one of the weakest models that one aspires to separate from NP. Strong lower bounds on even bounded bottom fan-in depth-3 circuits imply nonlinear lower bounds on series-parallel logarithmic depth Boolean circuits. More precisely, a lower bound of $\Omega(2^{n/2})$ on bounded bottom fan-in circuits implies a nonlinear size lower bound on series-parallel depth-3 circuits [11].

2 Upper bounding $\tau(G, z)$

Our main results give lower bounds on $\tau(G)$ which is sum of $\tau(G, z)$ over all satisfying assignments z of G. Here we present a lemma which lower bounds $\tau(G, z)$ in terms of the clause structure of G. This lemma formalizes one of the key ideas in [6].

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*. Let $Forced(G, \pi, y)$ denote the set of variables that are forced. The following easy fact characterizes, for each fixed π, the set of y such that **Modify**(G, π, y) is equal to z.

Proposition 1 *Let z be a satisfying assignment of G.* **Modify**$(G, \pi, y) = z$ *if and only if y and z agree on all variables outside of $Forced(G, \pi, z)$. Thus, for fixed π, the probability with respect to random y that* **Modify**$(G, \pi, y) = z$ *is equal to $2^{|Forced(G,\pi,z)|}/2^n$. Therefore:*

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

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

Now, by the concavity of the exponential function, we can lower bound the last expression by $2^{-n+\mathbf{E}_\pi[|Forced(G,\pi,z)|]}$. Next, we find an alternative expression for the expectation apppearing in the exponent

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, $v \in vars(C)$, 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 $v \in Forced(G, \pi, z)$ (conversely, if $v \in Forced(G, \pi, z)$ then v must appear last in some critical clause for (v, G, z)). Let $\text{Clause}(v, G, z)$ 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 $vars(C)$, and let $P(v, G, z)$ denote the probability that a random permutation π belongs to $\text{Clause}(v, G, z)$, which is equivalent to the probability that $v \in Forced(G, \pi, z)$ for random π. By linearity of expectation, $\mathbf{E}_\pi[|Forced(G, \pi, z)|] = \sum_v P(v, G, z)$. Putting things together we have:

Lemma 1 *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 $\tau(G, z) \geq 2^{-(1-p)n}$.

It is important to emphasize that while the function **Modify** depends on random π and y, the probability represented by $P(v, G, z)$ is independent of y.

3 Unique SAT

3.1 Overview

Using Lemma 1, the proof of Theorem 1 for the case of uniquely satisfiable F is nearly immediate. Indeed, 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 1 implies $\tau(F) = \tau(F, z) \geq 2^{-(1-\frac{1}{k})n}$ as required.

Note that the argument 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, let F contain two critical clauses for the variable x_1 at the all-true satisfying assignment, for example $(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ and $(x_1 \vee \bar{x}_4 \vee \bar{x}_5)$ with all of x_1, x_2, x_3, x_4 and x_5 distinct. In this case, the probability that a unit clause for x_1 occurs is $1/2$, rather than $1/3$ obtained using only a single critical clause. In general, 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, and this will enable us to get a better lower bound on $P(v, F_s, z)$. The main technical contribution of the paper is a technique to account for the advantage obtained by multiple critical clauses.

The argument in the first paragraph actually proves a more general result: that 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 $\tau(F, z) \geq 2^{-(1-\frac{1}{k})n}$. Similarly, our result will hold in a more general situation. We say that a satisfying assignment z of F is *d-isolated (in F)* for integer $d \geq 1$, if there is no other satisfying assignment within Hamming distance d of z. We will prove:

Theorem 6 *Let F be a k-CNF 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 for each k, μ_k is as defined before Theorem 2 and $\epsilon_k(d)$ is a function that tends to 0 as d gets large.

If z uniquely satisfies F, then z is d-isolated for any d, and so Theorem 2 follows.

In order to prove this result, we will need to be able to make a precise statement about the structure of critical clauses for (v, F_s, z). This structure will be described in terms of a rooted tree with some of the nodes labeled by variables. We'll 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. The depth of the tree is the minimum depth of any leaf. 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. If A is a set of nodes, write $L(A)$ for the set of variables that appear as labels of nodes of A.

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 $vars(C(A)) \subseteq L(A) \cup \{v\}$.

A critical clause tree for (v, G, z) is a convenient way of representing the set of multiple critical clauses for (v, G, z) so that one can account for the probability of forcing v. We prove two lemmas that together imply Theorem 6 and hence Theorem 2.

Lemma 2 *Let F be a k-CNF formula 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 depth d and maximum degree $k-1$.*

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

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

where for each k, $\epsilon_k(d)$ tends to 0 as d gets large.

Combining these two lemmas with Lemma 1 immediately yields Theorem 6. So it remains to prove these two lemmas.

3.2 Existence of a deep, bounded-degree critical clause tree

In this subsection we prove Lemma 2. Fix a k-CNF formula F, and d-isolated assignment z; we will assume without loss of generality that $z = 1^n$. Let v be an arbitrary variable of F. For a set of variables U, $z \oplus U$ denotes the assignment obtained from z by complementing the variables of U.

We "grow" a tree of depth d 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 $|P_i| \leq d$, $z \oplus L(P_i)$ does not satisfy F; choose a clause C_i that is not satisfied by $z \oplus L(P_i)$. For each variable w of $vars(C_i) - L(P_i)$, give b_i a child labeled w. If $vars(C_i) - L(P_i)$ 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 of depth d and maximum degree $k-1$. The total number of nodes in the final tree is bounded above by k^d, and hence any clause whose variables all appear as node labels in the tree has size at most s.

We will prove by induction on i that T_i 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 $L(A) \cup \{v\}$. 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 $z \oplus v$, which is 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_i. Assume that T_i is obtained by extending the path P_i of T_{i-1}. Let A be a cut of T_i, and let $A' = A - N_i$. Note that for each $a \in P_i - \{v\}$, the set $A_a = A' \cup a$ is a cut of T_i and a cut of T_{i-1}. Hence, by the induction hypothesis, there is a critical clause $C(A_a)$ for (v, F_s, z) such that $vars(C(A_a)) \subseteq L(A_a)$. If for some a, $vars(C(A_a)) \subseteq L(A')$, then we may take $C(A_a)$ to be $C(A)$. Similarly, if for some $a' \neq a$, $L(C_a) \subseteq vars(C(A_{a'}))$, then $L(a) \in L(A_{a'})$, implying that $L(A_a) \subseteq L(A')$ and that we can again take $C(A_a)$ to be $C(A')$. Now, we assume that for each $a \in P_i - \{v\}$, a is labeled, $L(a) \notin L(A')$ and $L(a) \in vars(C(A_a)) \subseteq L(A' \cup \{a\})$ but $L(a') \notin vars(C(A_a))$ for $a' \in P_i - \{v\}, a' \neq a$. It follows that $C(A_a)$ can be written as $B_a \vee \overline{L(a)} \vee v$, where B_a is a clause consisting entirely of negated variables, disjoint from $L(P_i)$.

Now consider the clause C_i that is used to construct T_i from T_{i-1}. We can write $C_i = v \vee C_i' \vee (\bigvee_{i=1}^{t} r_i)$ or $C_i = C_i' \vee (\bigvee_{i=1}^{t} r_i)$ where C_i' is a clause consisting of the negation of variables of $L(N_i)$, and $r_1, r_2, \ldots, r_t$ are variables in $L(P_i) - \{v\}$. Let a_j denote the unique node with label r_j.

Let $D_0 = c_i$ and for $1 \leq j \leq t$ define the clause $D_j = v \vee C_i' \vee (\bigvee_{h=j+1}^{t} r_h) \vee (\bigvee_{h=1}^{j} B_{a_h})$ (where a vacuous disjunction is regarded as *False*). Then for each $1 \leq j \leq t$, D_{j-1} and $C(A_{a_j})$ are resolvable on variable

r_j, and $D_j = R(D_{j-1}, C(A_{a_j}))$. By induction on j, since each of the D_j have size most s, each is in F_s. Then D_t is the desired critical clause $C(A)$.

3.3 Lower Bounding $P(v, G, z)$

We now proceed to the proof of Lemma 3. We are given a critical clause tree for (v, G, z) of depth d and maximum degree $k-1$ 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.

For the analysis, it will be useful to view the permutation π as a random variable on the following probability space. A *placement* of the variables is a function α that maps each variable to $(0, 1)$. We take the set of placements as our sample space and consider the uniform distribution, i.e., the values $\alpha(u)$ are independent and uniformly distributed on $(0, 1)$. 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. 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 this probability space. Events are defined by (measurable) sets 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 $w \in L(A)$. 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 $w \in L(A)$. 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:

Proposition 2 *If T is a critical clause tree for (v, G, z), then $P(v, G, z) \geq Q_T$.*

So to prove the lemma, it suffices to lower bound Q_T in the case that T is a depth d tree of maximum degree $k-1$.

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. We have the following recursive lower bound on $Q_T(r)$:

Lemma 4 *Let T be an admissible tree with more than one node with root labelled by v and $r \in (0, 1)$. Let $T_1, T_2, \ldots, T_t$ be the admissible subtrees rooted at the labeled children of the root of T. Then:*

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

where an empty product is interpreted as 1.

We apply this lemma in the case that T is a tree of degree at most $k-1$ and depth at least d. For fixed k and r, define the sequence $(Q_{k,d}(r) : d \geq 0)$ recursively by $Q_{k,0}(r) = 0$, and $Q_{k,d}(r) = (r + (1-r)Q_{k,d-1}(r))^{k-1}$. Also define $Q_{k,d} = \int_0^1 Q_{k,d}(r)dr$. Lemma 4 together with induction on d yields:

Lemma 5 *If T is an admissible tree of degree at most $k-1$ and depth at least d then for all $r \in (0, 1)$:*
$Q_T(r) \geq Q_{k,d}(r)$ and $Q_T \geq Q_{k,d}$.

Thus, we can complete the proof of Lemma 3 by establishing a suitable lower bound on $\int_0^1 Q_{k,d}(r)dr$.

3.4 Evaluating the Integral

In order to lower bound $Q_{k,d}$, we will first show that for each k and r, the sequence $(Q_{k,d}(r) : d \geq 0)$ converges. Then, by choosing d sufficiently large, we can closely approximate this limit behavior. For fixed k and $r \in (0, 1)$, define the function $f_k^r(x) = (r + (1-r)x)^{k-1}$. Note that $Q_{k,d}(r) = f_k^r(Q_{k,d-1}(r))$. Let us define $R_k(r)$ to be the smallest nonnegative real root of the equation $f_k^r(x) - x = 0$; $R_k(r)$ is well defined since 1 is a root of $f_k^r(x) - x = 0$. Let $R_k = \int_0^1 R_k(r)dr$. It is easy to show that $R_k(1) = 1$, and we define r_k to be the least r such that $R_k(r) = 1$.

By elementary analytic arguments one can show:

Proposition 3 *Let $k \geq 3$.*

1. *$R_k(r)$ is continuous and strictly increasing on the interval $[0, r_k] = 1$ and $R_k(r) = 1$ on the interval $[r_k, 1]$*

2. *For each fixed d, $Q_{k,d}(r)$ is a continuous, nondecreasing function of r on $[0, 1]$, with $Q_{k,d}(0) = 0$ and $Q_{k,d}(1) = 1$.*

3. *For each fixed r, $Q_{k,d}(r)$ is nondecreasing in d, and strictly increasing if $r \in (0, 1)$.*

4. *For each fixed r, $(Q_{k,d}(r) : d \geq 0)$ converges to $R_k(r)$.*

5. *$(Q_{k,d} : d \geq 0)$ converges to R_k.*

We will show that $R_k = \frac{\mu_k}{k-1}$, where μ_k is as defined before the statement of Theorem 2. Then Proposition 2 and Lemma 5 and Proposition 3 imply Lemma 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 $k > 3$, we invert the function $R_k(r)$ and then integrate the inverse function. As we noted, $R_k(r)$ is a continuous strictly increasing function on $[0, r_k]$ and is 1 on $[r_k, 1]$. Therefore we can define a unique function $S_k(t)$ on the interval $[0, 1)$ which is an inverse for $R_k(r)$ on the interval $[0, r_k]$, and $\int_0^1 R_k(r)dr = \int_0^1 (1 - S_k(t))dt$ (draw a picture!).

Now $S_k(t)$ is the unique value of r satisfying $t = (r + (1-r)t)^{k-1}$, which is:

$$S_k(t) = \frac{t^{\frac{1}{k-1}} - t}{1 - t}$$

.

We now have

$$R_k = \int_0^1 (1 - S_k(t))dt = \frac{\mu_k}{k-1},$$

completing the proof.

4 General k-SAT

We now proceed to the analysis of general k-CNF formulae. Theorem 6 applies to any formulae that has a sufficiently isolated satisfying assignment, but a satisfiable formula need not have such an assignment. Recall that isolation comes into the building of the critical clause tree: we need d-isolation to guarantee that the tree will extend to depth d. 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 Sections 1.3 and 1.4, upper bounding the running time of **Search**(F_s, I) is accomplished by upper bounding $\tau(F_s)$, the probability that **Modify**(F_s, π, y) returns a satisfying assignment.

The method for bounding $\tau(G)$ outlined in Section 2 and applied in the uniquely satisfiable case focused on the probability $\tau(G, z)$ of accepting a particular assignment. We will need a more general 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 6 *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 disjoint subcubes so that $z \in B_z$ for each $z \in A$.*

If G is a satisfiable formula, we apply this lemma in the case that A is the set $S(G)$ of satisfying assignments of the formula G. 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 write $\tau(G|B_z)$ for the sum of $\tau(G, w|B_z)$ over all satisfying assignments w. We then have

$$\begin{aligned} \tau(G) &= \sum_{z \in S(G)} \tau(G|B_z)\mathbf{Prob}[y \in B_z] \\ &\geq \sum_{z \in S(G)} \tau(G, z|B_z)\mathbf{Prob}[y \in B_z], \end{aligned}$$

from which we conclude:

Proposition 4 *For any satisfiable formula G, $\tau(G) \geq \min_{z \in S(G)} \tau(G, z|B_z)$.*

So, to lower bound $\tau(G)$, we take a generic satisfying assignment z and lower bound the probability $\tau(G, z|B_z)$ that **Modify**(G, π, y) returns z given that $y \in B_z$.

Let $D = D(z)$ be the set of variables which define the subcube B_z (i.e., the set of variables which are constant over that subcube) and let $N = N(z)$ be the remaining variables. The variables in $D(z)$ are referred to as the *defining* variables of z and those in $N(z)$ are referred to as *nondefining*.

Now, to lower bound $\tau(G, z|B_z)$, we first try to generalize the argument leading to Lemma 1. Given that $y \in B_z$, we have that y agrees with z on the defining variables, so **Modify**(G, π, y) returns z if and only if the non-defining variables are set according to z. Write $Forced_z(G, \pi, y)$ for the set of non-defining variables in $Forced(G, \pi, y)$. Then Proposition 1 can be generalized to conclude:

$$\tau(G, z|B_z) = 2^{-|N(z)|}\mathbf{E}[2^{|Forced_z(G,\pi,z)|}].$$

Continuing the argument, one finally obtains the following generalization of Lemma 1:if the average of $P(v, G, z)$ over defining variables is at least p then $\tau(G, z|B_z) \geq 2^{-(1-p)|N(z)|}$.

Next we look to generalize the lower bound on $P(v, G, z)$ in Theorem 6. What we would like is to get the same $\frac{\mu_k}{k-1} + \epsilon_k(d)$ lower bound. The difficulty comes when we try to construct the critical clause tree of depth d. Recall, that to extend the tree from a given leaf b_i we used a clause C_i that was not satisfied by $z \oplus L(P_i)$, and the existence of such a clause was guaranteed by the fact that $z \oplus L(P_i)$ does not satisfy the formula (because of the d-isolation of z). Now, however, such a clause C_i need not exist because we don't know that z is isolated so $z \oplus L(P_i)$ might satisfy G. But we do know that if $L(P_i)$ consists only of nondefining variables then $z \oplus L(P_i)$ does not satisfy G since z is the only satisfying assignment in B_z. So suppose we 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, and so by the above observation, it is always possible to expand any other leaf. Thus we conclude the following counterpart to lemma 2

Lemma 7 *Let F be a k-CNF formula and z an arbitrary satisfying assignment, and let B_z be as defined above. If v is any nondefining variable, and d is any integer, and $s \geq k^d$, 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 are leaves, (ii) any leaf that labeled by a nondefining variable is at depth d.*

We say that such a tree is a *depth d tree with respect to the set $N(z)$*. Such trees are nice, but not good enough to directly get a result such as Lemma 3. For instance, it could be that the tree consists of a root together with $k-1$ children all labeled by defining nodes. In this case, we get $1/k$ as the probability Q_T, as compared to $\mu_k/(k-1)$. This is the worst case since there must be at least one critical clause for each nondefining variable. This is weak, but does give a good lower bound on $P(v, G, z)$ in the case that $D(z)$ is a reasonably large fraction of n. Note that $P(v, G, z)$ is the probability that the variable $v \in \mathit{Forced}(G, \pi, z)$ for random π.

Lemma 8 *Let F be a boolean formula and z a satisfying assignment. Then:*

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

In particular if $|D(z)|/n \geq \frac{k}{(k-1)^2}\mu_k - \frac{1}{k-1}$ then this gives the desired $2^{(1-\frac{\mu_k}{k-1})n}$ lower bound. However, if $D(z)$ is a small fraction of n (so $N(z)$ is close to n) the above bound will not be good enough to improve the results of [6]. So, we need a way to handle the "bad case" which, roughly speaking, is the case that $D(z)$ is not too big a fraction of n and there are many shallow leaves labeled by defining variables.

So, we need to back up somewhat and generalize an earlier part of the argument. Let's return to the statement of Proposition 1 which we generalized above. The next step leading to Lemma 1 was to use concavity to bring the expectation inside the exponential. This step still provides a lower bound, but in this case it gives too much away. Intuitively here's why. For a random variable X, the inequality $\mathbf{E}[2^X] \geq 2^{\mathbf{E}[X]}$ is tight if X is constant and becomes very loose if X has a small but non-negligible probability of being very large. Now consider the random variable of interest $\mathit{Forced}_z(G, \pi, z)$. If we consider the "bad case" in the development sketched above, we see that permutations for which the defining variables appear early in the permutation will tend to force many more nondefining variables than permutations for which the defining variables appear later. This creates exactly the situation where the concavity bound is loose.

So we proceed as follows. Suppose we identify a set Γ of placements having "fairly large" probability with the property that the average of $|\mathit{Forced}_z(G, \pi, z)|$ conditioned on $\alpha \in \Gamma$ is much larger than the overall average. We then have

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

where $\mathbf{E}_\Gamma$ denotes the conditional expectation given that $\alpha \in \Gamma$. Following the argument as in Section 2, letting $P_\Gamma(v, G, z)$ denote the probability that $\pi \in \mathrm{Clause}(v, G, z)$ given that $\alpha \in \Gamma$, and repeating the argument in Section 2 we obtain the following generalization of Lemma 1:

Lemma 9 *Let Γ be a (measurable) subset of placements. For any satisfying assignment z of the CNF formula G if $P_\Gamma(v, G, z) \geq p$ for all non-defining variables v then $\tau(G, z|B_z) \geq 2^{-(1-p)|N(z)|}\mathbf{Prob}[\alpha \in \Gamma]$.*

One choice of Γ is to take all α such that the defining variables all appear before the nondefining variables. For such a Γ, the difficulty with shallow defining variables in the critical tree disappears, indeed, we can completely eliminate the defining variables by fixing them to their values since we know they will appear before all of the nondefining ones. So we get the same lower bound on $P(v, F_s, z)$ as in the uniquely satisfiable case. But now, in bounding $\tau(F_s, z|B_z)$ we must multiply by $\mathbf{Prob}[\alpha \in \Gamma]$, and for this choice of Γ the best lower bound is of the form $(\frac{c_o|D(z)|}{n})^{|D(z)|}$.

Lemma 10 *Let F be a k-CNF formula and z a satisfying assignment. Let $d \geq 1$ and $s \geq k^d$. Then:*

$$\tau(F_s, z|B_z) \geq 2^{-(1-\frac{\mu_k}{k-1}+\epsilon_k(d))n}(\frac{c_0|D(z)|}{n})^{|D(z)|},$$

where c_0 is a constant and for each k, $\epsilon_k(d)$ is a function that tends to 0 as d gets large.

If $|D(z)| = o(n)$ the expression for $\mathbf{Prob}[\alpha \in \Gamma]$ is at least $2^{-o(n)}$ and the bound in this lemma gives what we want. But for larger D, the $\mathbf{Prob}[\alpha \in \Gamma]$ significantly reduces the quality of the bound.

So we can handle the case that $D(z)$ is small and also the case that $D(z)$ is large. To handle intermediate ranges of $D(z)$, we will aim to choose Γ so that the defining elements *tend* to occur earlier than the nondefining ones, but so that Γ still has reasonably high probability.

To this end, define a *distribution function* to be a non-decreasing (finitely) piecewise differentiable function H on

$[0,1]$ such that $H(0) = 0$ and $H(1) = 1$. We say that a placement α is *H-good* with respect to the set D of variables provided that, for each r, the number of all defining variables that are mapped to $[0,r]$ is at least $\lfloor H(r)|D|\rfloor$. For fixed D, we define $\Gamma_H(D)$ to be the set of all placements that are H-good. We define a constant β_H and a sequence $\gamma_H(k)$ associated with the function H (below h is the derivative $H'(r)$.)

$$\begin{aligned}\beta_H &= \int_0^1 h(r)\log_2(h(r))dr \\ \gamma_H(k) &= \int_0^1 H(r)^{k-1}dr\end{aligned}$$

The negative of β_H is known as *differential entropy* where H is the distribution function of the random variable r. It is related to the discrete entropy $E(\Delta)$ of the "quantized" random variable r by $\sum_i \Delta h(r_i)\log(h(r_i)) = \sum_i \Delta h(r_i)\log(\Delta h(r_i)) - \log\Delta$. The quantity to the left converges to β_H as Δ goes to zero. $E(\Delta) = -\sum_i \Delta h(r_i)\log(\Delta h(r_i))$ is the discrete entropy of the quantized random variable. Since i takes at most $1/\Delta$ values, we have $0 \le E(\Delta) \le \log\frac{1}{\Delta}$. From these observations, we conclude $\beta_H \ge 0$.

Observe that for $H(r) = r + (1-r)R_k(r) = R_k(r)^{\frac{1}{k-1}}$, $\gamma_H(k) = \frac{\mu_k}{k-1}$ as derived in Section 3.4.

It is not hard to derive a lower bound on the probability that a random placement is in $\Gamma_H(D)$:

Lemma 11 *Let H be a distribution function and let h be its derivative. Then*

$$\mathbf{Prob}[\alpha \in \Gamma_H(D)] \ge 2^{-\beta_H|D|-o(|D|)}.$$

Next, we want to lower bound $P_\Gamma(v,G,z)$. Recalling the definition of $R_k(r)$ we obtain the following analog of Lemma 3:

Lemma 12 *Let F be a formula, z a satisfying assignment and let D and N be complementary sets of variables. Suppose that $v \in N$ and that there is a critical clause tree that is degree $k-1$ and depth d with respect to N. Let $H(r)$ be a distribution function satisfying $H(r) \ge r + (1-r)R_k(r)$ for all $r \in [0,1]$. Then*

$$P_\Gamma(v,G,z) \ge \gamma_H(k) - O(k^{2d}/|D|) - \epsilon_k(d),$$

where $\epsilon_k(d)$ tends to 0 as d gets large.

One natural candidate for H is to match the restriction placed on it in the hypothesis, that is, $H(r) = r + (1-r)R_k(r)$. This choice corresponds to the case that the critical clause tree is infinitely deep with all nodes labelled by distinct variables. For this H, we express β_H as a function of H (rather than its derivative) and establish that β_H decreases monotonically with k.

Lemma 13 *For $H(r) = r + (1-r)R_k(r)$,*

$$\beta_H = \int_0^1 \log_2 \frac{(1-H^{k-1})^2}{1+(k-2)H^{k-1}-(k-1)H^{k-2}}dH$$

With some additional calculations, we can also show the following

Lemma 14 *β_H decreases monotonically with k.*

Suppose that $D(z)$ is much larger than $s = k^d$ and that d is an increasing function of n. Let $\Delta = |D(z)|/n$. Then Lemmas 9, 11 and 12 imply

$$\tau(v, F_s|B_z) \ge 2^{-((1-\gamma_H)-(1-\gamma_H-\beta_H)\Delta-\delta)n},$$

where δ can be made arbitrarily small. Define $\chi = (1-\gamma_H-\beta_H)$ to be the coefficient of Δ in this express. Using numerical calculation, we show that $\chi \ge 0$ for $k=5$. Since $\gamma_H(k) = R_k = \frac{\mu_k}{k-1}$ for our choice of H, it follows that $\gamma_H(k)$ decreases monotonically with k. Since β_H decreases monotonically with k (Lemma 14), we have $\chi \ge 0$ for $k \ge 5$. Hence, the right hand side is always at least $2^{-(1-\gamma_H)n+o(n)}$. Summarizing this, and combining with Lemma 10 we get the following:

Lemma 15 *Let F be a k-CNF formula, z a satisfying assignment, and $N(z)$ and $D(z)$ be as above. Let d be an increasing function of n, say $\log\log n$.*

1. *If $D(z) = o(n)$ then:*
$$\tau(F_s, z) \ge 2^{-(1-\gamma_H(k))n+o(n)}$$
2. *If $D(z) = \Omega(n)$, $H(r) = r + (1-r)R_k(r)$ and $(1-\gamma_H-\beta_H(k)) \le 0$, we have:*
$$\tau(v, F_s|B_z) \ge 2^{-((1-\gamma_H)-(1-\gamma_H-\beta_H)-o(1))n}$$
3. *If $D(z) = \Omega(n)$, $H(r) = r + (1-r)R_k(r)$ and $k \ge 5$*
$$\tau(F_s, z|B_z) \ge 2^{-(1-\gamma_H(k))n+o(n)}$$

With numerical calculations, we obtain the following results, where c is the constant appearing in an upper bound of 2^{cn} on the running time of the algorithm.

k	β_H	R_k	χ	c
3	1.115	0.614	-0.729	0.533
4	0.666	0.445	-0.111	0.581
5	0.478	0.350	0.172	0.649
6	0.373	0.288	0.339	0.711

For $k = 3, 4$, however, we get appreciably better results. In those cases, using computer search. we obtain the improved upper bounds on the running time of $2^{0.446n}$ for $k = 3$ and $2^{0.562n}$ for $k = 4$. The details will appear in the final paper.

This discussion completes a sketch of the proofs of Theorems 3, 4 and 5.

5 Lower Bounds for Depth-3 Circuits

Efficient coding of suffciently isolated satisfying solutions implies that if a k-CNF accepts only inputs that are sufficiently apart in the Hamming space, then it cannot accept too many inputs. Such limitation on the set of points accepted by a k–CNF can be exploited to prove a lower bound on the size of certain depth-3 circuits. Let Σ_k^3 denote the class of depth–3 circuits with an OR gate at the top and bottom fan-in k. A depth–2 subcircuit of a Σ_k^3 circuit is a k–CNF.

Let Ecc be a binary error–correcting code with minimum distance d_n is a slowly increasing function of n. Let E be the function where $E(x) = 1$ if x is a codeword of Ecc, and 0 otherwise.

Lemma 16 *No k–CNF can accept more than $2^{(1-\frac{\mu_k}{k-1}+o(1))n}$ codewords of Ecc while rejecting all non-codewords.*

Now, we combine this lemma with a simple observation: If f is a Boolean function computed by a Σ_k^3 circuit C of size s, then there must be some depth–2 subcircuit C' so that C' accepts $|f^{-1}(1)|/s$ points with $f(x) = 1$ while rejecting all points with $f(x) = 0$. Also, one can construct error correcting codes with distance $\Omega(d_n)$ and $2^{n-o(n)}$ codewords.

Corollary 1 *For any $k \geq 3$, any Σ_k^3 circuit computing Ecc must have size at least $2^{\frac{\mu_k}{k-1}n-o(n)}$.*

Corollary 2 *Any Σ_3^3 circuit computing E must have size at least $2^{0.613n}$.*

Corollary 3 *For any $k \geq 3$, any Σ_k^3 circuit computing E must have size at least $2^{\frac{\mu_k}{k-1}n}$.*

Acknowledgments: We thank Professor Ed Bender for his help in evaluation the integral in Section 3.4. We also thank Yuming Zhang for his helpful comments and corrections.

References

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

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

[3] 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.

[4] 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.

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

[6] Paturi, R., Pudlák, P., and Zane, F., (1997), Statisfiability Coding Lemma, *in* Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, pp 566–574, October 1997.

[7] 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

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

[9] 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.

[10] Schiermeyer, I. (1996), Pure Literal Look Ahead: An $O(1.497^n)$ 3-Satisfiability Algorithm, Preprint.

[11] 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.

[12] 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.

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