The Complexity of Satisfiability of Small Depth Circuits

Chris Calabro, Russell Impagliazzo[⋆], and Ramamohan Paturi[⋆⋆]

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404, USA

Abstract. Say that an algorithm solving a Boolean satisfiability problem x on n variables is *improved* if it takes time $\text{poly}(|x|)2^{cn}$ for some constant $c < 1$, *i.e.*, if it is exponentially better than a brute force search. We show an improved randomized algorithm for the satisfiability problem for circuits of constant depth d and a linear number of gates cn: for each d and c, the running time is $2^{(1-\delta)n}$ where the improvement $\delta \geq 1/O(c^{2^{d-2}-1} \lg^{3\cdot 2^{d-2}-2} c)$, and the constant in the big-Oh depends only on d. The algorithm can be adjusted for use with Grover's algorithm to achieve a run time of $2^{\frac{1-\delta}{2}n}$ on a quantum computer.

1 Introduction

All **NP**-complete problems are equivalent as far as the existence of polynomial time algorithms is concerned. However, the exact complexities of these problems vary widely. There are frequently algorithms for **NP**-complete problems that achieve substantial improvement over exhaustive search. This raises the questions: Which problems have such improved algorithms? How much can we improve? Can we provide evidence that no improvement over some known algorithm is possible? Work addressing such questions, both from the algorithmic and complexity theoretic sides, has become known as *exact complexity*, and it is related to the field of parameterized complexity. While significant work has been done, both areas are still fairly new and leave open many problems. In particular, the answers and techniques seem to rely on the exact **NP**-complete problem in question, and there are few unifying techniques. (This is in some ways similar to the situation for the exact approximation ratios achievable for different **NP**-complete problems, which also is problem dependent. However, the use of probabilistically checkable proofs, and the unique games conjecture and

[⋆] This research is supported by supported by the Simonyi Fund, the Bell Company Fellowship and the Fund for Math, and NSF grants DMS-083573, CNS-0716790 and CCF-0832797.

[⋆⋆] This research is supported by NSF grant CCF-0947262 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

related conjectures, provide very general tools for understanding approximability for a wide variety of problems. We are still looking for similar tools for exact complexity.)

From the viewpoint of exact complexity, the most studied and best understood problems are probably the restricted versions of the satisfiability problem (SAT), in particular, k-SAT, a restriction of SAT to k-CNFs, and CNF-SAT, a restriction to general CNFs. There has been a sequence of highly nontrivial and interesting algorithmic approaches [Sch99,PPZ99,PPSZ05,Sch05,DW05,CIP06] to these problems, where the best known constant factor improvements in the exponent are of the form $1-1/O(k)$ for k-SAT and $1-1/O(\lg c)$ for CNF-SAT with at most cn clauses. Also, a sequence of papers ([IPZ01,IP01,CIKP08,CIP06]) has shown many nontrivial relationships between the exact complexities of these problems, and helped characterize their hardest instances (under the assumption that they are indeed exponentially hard.) For what other circuit/formula models can we expect to show *improved* exponential-time (*i.e.*, $O(2^{cn})$-time for $c < 1$) algorithms for the satisfiability problem?

1.1 Linear-size Bounded Depth Circuits

In this paper, we give what we believe is the first improved algorithm for the satisfiability problem for circuits of constant depth and linear size (AC^0 type), which seems significantly harder than k-SAT. (Note that it is trivially possible to give an improved algorithm in terms of the circuit size parameter m if the circuit has fan-in 2, e.g. by the standard reduction to 3-SAT and then applying a 3-SAT solver, but this is no better than exhaustive search once m gets larger than around $4n$ or so. Sergey Nurik [Nur09] has recently communicated a somewhat improved bound along these lines.) For each $c, d > 0$, we give a constant $\delta > 0$ and a randomized algorithm that works in $2^{(1-\delta)n}$ time and solves the satisfiability problem for depth d, size at most cn circuits. (Here, it is significant that circuit size is measured by gates rather than wires.)

For $d = 2$, our algorithm becomes deterministic and matches the current best bound [CIP06], since our algorithm and analysis are generalizations of the ones there. However, randomizing the algorithm also yields the best quantum algorithm for this case, with running time $2^{\frac{1-1/O(\lg c)}{2}n}$. For $d = 3$, this gives $\delta \geq 1/O(c \lg^4 c)$, which, as far as the authors know, is the first improvement achieved for this problem.

There are a few motivations to consider linear-size circuits. One is the question of ideal block cipher design. Block ciphers are carefully constructed to maximize efficiency for a given level of security. Particularly, since we want ciphers to be usable by low-power devices, and to be implemented at the network level, it is often very important to have efficient hardware implementations that make maximum use of parallelism. A typical cipher computes for a small number of “rounds”, where in each round, very simple operations are performed (e.g., substitutions from a look-up table called an S box, permutations of the bit positions, or bit-wise $\oplus$ operations). These operations are almost always AC^0 type or even simpler. It is also considered vital to have key sizes that are as small as possible, and an algorithm that breaks the cryptosystem in significantly less time than exhaustive search over keys is considered worrisome. So this raises the question: Can we have an ideal block cipher family (one per key size), *i.e.*, so that the number of rounds remains constant, each round being implementable in constant depth with a linear number of gates, and security is almost that of exhaustive search over keys? Our results rule out such ideal block ciphers, and so give a partial explanation for why the number of rounds needs to increase in new generations of block ciphers. (Block ciphers require average-case security, not worst-case, but worst-case algorithms obviously also rule out average-case security. Our values of δ are vanishingly small for the sizes and depths of real cryptosystems, so our results cannot be used for cryptanalysis of existing block ciphers.)

Another motivation is that linear-size circuits are perhaps the most general class of circuits for which we can expect to show improved upper bounds on their exact complexity. To explain this statement, we need the following notation. Let $s_k = \inf\{c | \exists$ a randomized algorithm for k-SAT with time complexity $\mathrm{poly}(m)2^{cn}$ for k-CNF formulas of size m over n variables$\}$. Let **ETH** denote the Exponential-Time Hypothesis: $s_3 > 0$. We know that the sequence $\{s_k\}$ has a limit and let s_∞ denote this limit. [IP01] proposed the open question whether $s_\infty = 1$, which we will call the *Strong Exponential-Time Hypothesis* (**SETH**). The best known upper bounds for s_k are all of the form $1 - 1/O(k)$, which makes the conjecture **SETH** plausible.

Here is the connection between **SETH** and the complexity of satisfiability of linear-size circuits: Since one can embed k-CNFs for any k into any non-linear size circuit model (in particular, nonconstant density CNF) [CIP06], improved upper bounds for the satisfiability problem for nonlinear-size circuits would imply $s_\infty < 1$. Thus, we are primarily left with the question of the complexity of the satisfiability problem for linear-size circuits if **SETH** holds. The following partial converse shows a further connection between **SETH** and improved bounds for the satisfiability of linear-size circuits. If $s_\infty < 1$, one can easily show using the depth-reduction technique of Valiant [Val77] (see also [Cal08]) that the satisfiability problem for cn-size series-parallel circuits has an improved upper bound of $2^{\delta(c)n}$ where $\delta(c) < 1$.

Yet another motivation is that improved algorithms for SAT for a circuit model $\mathcal{C}$ may reveal structural properties of the solution space of circuits in $\mathcal{C}$. These structural properties may in turn be helpful in proving stronger lower bounds on the size of circuits which are disjunctions of circuits in $\mathcal{C}$. In fact, [PSZ00,PPZ99,PPSZ05,IPZ01] exploit this connection to provide the best known lower bounds of the form $2^{\Delta(k)n/k}$ where $\Delta(k) > 1$ for depth-3 unbounded fan-in circuits with bounded bottom fan-in k. This connection between the hardness of the satisfiability problem for a circuit model and lower bounds of a related circuit model is not surprising since, as a more general circuit can compute more complicated functions, it may be more difficult to invert, *i.e.*, check the satisfiability of these functions.

1.2 Extension to Quantum Computing Model

Since Grover's quantum search algorithm [Gro96] provides a quadratic speed-up, the baseline in the quantum model for improved algorithms is $2^{n/2}$. In other words, a quantum algorithm is an improvement for the satisfiability problem if the constant factor in the exponent in the running time is strictly less than 1/2. However, it is not clear that every improved algorithm in the classical model can benefit from a quadratic speed-up in the quantum model. It is known that the class of algorithms that are exponential iterations of probabilistic polynomial time algorithms can obtain quadratic speed-up using Grover's technique.

More precisely, [Gro96,BBHT96] show that a probabilistic algorithm running in time t and with success probability p can be transformed into a quantum algorithm with running time $O(t/\sqrt{p})$ and with constant success probability. The quadratic speed-up provided by quantum algorithms prompts the following question: Given an algorithm A with exponential running time t, can we transform it into an exponential iteration of a polynomial time algorithm B with success probability approximately $1/t$? Such a transformation would prime A for use in Grover's algorithm and we could reap the full benefit of its quadratic speedup in the quantum model. We will show that our algorithm for the satisfiability of bounded-depth linear-size circuits can be sped up quadratically in the quantum model, *i.e.*, that our algorithm, which runs in time $2^{(1-\delta)n}$ with constant success probability, can be sped up by transforming it into a probabilistic polynomial time algorithm that succeeds with probability at least $2^{-(1-\delta)n}$.

For simplicity, we will only describe this probabilistic version of the algorithm. To convert this back into a backtracking algorithm, simply try both branches deterministically. Note that either way, the subroutine find_restriction would still use randomness, though the authors strongly suspect that it can be derandomized. For completeness, we include the backtracking version (without analysis, which would be essentially the same, just more verbose) at the end in figure 2.

2 The Algorithm

2.1 Definitions

The *inputs* (*outputs*) of a dag are those nodes with indegree 0 (outdegree 0). A *circuit* F is a dag where each input is labeled with a literal, each non-input (called a *gate*) is labeled AND or OR, and there is exactly 1 output. A *subgate* of size k of a gate g is a gate h (not necessarily in F) with the same label as g and with k of g's inputs. The *depth* d of F is the number of edges in a longest path in F. The *ith level* of F is the set of gates a distance of i from the output, e.g. the output is at level 0 and the bottom gates are at level $d-1$.

2.2 High Level Description

The overall algorithm consists of four subroutines, $A_{d,c}, A_{d,c,k}$, find_restriction and PPZ_main. We first provide a high level description of the key routines $A_{d,c}$

and $A_{d,c,k}$ which mutually call each other. We then provide an intuitive explanation as to how the algorithm reduces the average number of variables whose values need to be guessed. In the next subsection, we provide a detailed description of the subroutines accounting for all the parameters. Figure 1 provides a complete description of the routines except for PPZ_main.

The argument F of $A_{d,c}$ and $A_{d,c,k}$ denotes the current circuit after it is simplified as a result of variable assignments. We use the argument V of $A_{d,c}$ and $A_{d,c,k}$ to keep track of the unassigned variables, which is a superset of the remaining variables $\mathrm{var}(F)$. In particular, each time we simplify the circuit by assigning to a group of variables, we will remove them from V. Subsequently simplifying the circuit may result in the removal of other variables from F so that $\mathrm{var}(F) \subseteq V$ may become a proper containment. Because of these simplifying steps, $|V|$ is a measure of the algorithm's progress that will be easier to keep track of than $|\mathrm{var}(F)|$. In order to describe the progress made at various points in $A_{d,c}$ and $A_{d,c,k}$, we use the identifier V to denote the set of unassigned variables at that point whereas we use n to denote the number of unassigned variables at the beginning of an invocation of the routines.

$A_{d,c}$ starts with a circuit F of depth d and a set of unassigned variables V satisfying the condition, $\mathrm{var}(F) \subseteq V$ and $|F| \leq cn$ where $n = |V|$. It reduces the the fan-in of each bottom gate to k by repeatedly selecting a subgate h of size k of any bottom gate g with fan-in greater than k and setting h to either true or false. One of these settings will eliminate k variables from V and the other will eliminate a gate. The one that eliminates k variables we choose with probability q and the other with probability $1-q$. We continue setting the subgates of bottom gates until we reach one of two cases. In the first case, we've at least halved the number of unassigned variables compared to the number n at the invocation of $A_{d,c}$. In this case, we simply guess an assignment to the unassigned variables.

In the second case we have that each bottom gate has fan-in less than or equal to k and that the number of gates is at most $2c$ times the number $|V|$ of unassigned variables. In this case the control passes to the routine $A_{d,2c,k}$.

$A_{d,c,k}$ takes as input a set V of unassigned variables and a circuit F of depth d, bottom fan-in restricted to k satisfying the condition $\mathrm{var}(F) \subseteq V$ and $|F| \leq cn$ where $n = |V|$. It chooses a random restriction (by invoking the routine find_restriction) to all but a p fraction of the variables of V for some p. This may leave some bottom level gates with more than one unassigned variable. We clean up these gates by randomly setting all the variables in them. By choosing k, p appropriately, with probability at least $\frac{1}{2}$ there will still be $\geq \frac{1}{2}pn$ unassigned variables but the bottom level gates will each have at most one unassigned variable. So we can collapse the circuit to depth $d-1$ and recurse. If the circuit is already at depth 2, $A_{d,c,k}$ applies PPZ_main, which applies one iteration of the PPZ solver [PPZ99], which takes polynomial time and finds a satisfying assignment with probability at least $2^{-(1-\frac{1}{k})n}$ if one exists.

To see why the algorithm succeeds better than random guessing, observe that the algorithm either preserves a constant fraction of the unassigned variables by the time the circuit reaches depth 2 or sets a large number of variables correctly according to a satisfying assignment without having to guess each one of them independently. If the algorithm produces a circuit of depth 2 with a constant fraction of unassigned variables, PPZ_main guarantees that at most $(1-\frac{1}{k})$ fraction of the unassigned variables need to be looked at on average to find a satisfying assignment. If the algorithm terminates earlier in $A_{d,c}$ when the number of unassigned variables gets halved, it must be the case that at least half of the variables are assigned in $A_{d,c}$ by setting subgates of size k. The variables of the subgates are set with probability q and each such setting results in an assignment to k variables where k is sufficiently large compared to $\lg\frac{1}{q}$ thus saving a number of bits.

The overall algorithm takes polynomial time and has exponentially small probability s of finding a satisfying assignment given that there is one. By iterating s^{-1} times, we increase the probability of success to a constant.

2.3 Detailed Description

We describe our algorithm in several subroutines:

- $A_{d,c}(F,V)$ seeks a solution of F when F has depth d with V as the set of unassigned variables such that $\mathrm{var}(F)\subseteq V$, $|V|=n$, and $|F|\leq cn$. Although initially $\frac{|F|}{|V|}\leq c$, the algorithm may set variables, increasing the ratio. If it ever exceeds $2c$, $A_{d,c}$ will simply guess an assignment to the remaining variables.
- $A_{d,c,k}(F,V)$ seeks a solution of F when F has depth d, $\mathrm{var}(F)\subseteq V$, $|V|=n$, $|F|\leq cn$, and F has bottom fan-in at most k.
- find_restriction(F,V,p) finds, with probability at least $\frac{1}{2}$, a set of variables $W\subseteq V$ whose complement has size in the interval $[\frac{1}{2}pn,pn]$ and such that if the variables of W are assigned, then each bottom level gate has at most one unassigned variable.
- PPZ_main is 1 iteration of the k-SAT solver (which is the same as a depth 2 circuit solver) from [PPZ99] which takes polynomial time and has success probability $\geq 2^{-(1-\frac{1}{k})n}$. More specifically, PPZ_main assigns the variables, one at a time, in a random order. If a variable about to be assigned appears in a unit clause C (a clause of size 1), then it is assigned so as to satisfy C, otherwise it is assigned uniformly randomly. Note that this algorithm solves depth 1 circuits in polynomial time and with success probability 1.

Our algorithm description is not the most succinct. For example, one could construct an equivalent algorithm containing only one subroutine by eliminating tail recursion, but this would make the analysis obtuse. Below, the choices of k,p,q,c' are unspecified, and are left for the analysis section.

If h is an AND of literals, then $F|(h=1)$ sets those literals to true and simplifies the circuit by removing true children of AND gates, false children of OR gates, replacing empty AND gates by true, replacing empty OR gates by false; unless h contains contradictory literals, in which case $F|(h=1)$ is simply false. If h is an OR of literals, $F|(h=1)$ removes any gate of which h is a subgate and then performs a similar simplification. Also if h is an AND (OR) of literals, then $F|(h=0)$ can be treated as $F|(h'=1)$ where h' is the OR (AND) of the negations of those literals. The purpose of all these definitions is so that below, in $A_{d,c}$, the line $F \leftarrow F|(h = b \text{ XOR } b')$ sets h in the way that eliminates k variables with probability q and the other way with probability $1-q$. See figure 1.

```
$A_{d,c}(F,V)$ // $F$ has depth $d$, $\mathrm{var}(F) \subseteq V, |V| = n$, $|F| \leq cn$
  choose $k, q$ // as some function of $d, c$
  while $\exists$ bottom gate $g$ in $F$ of fan-in $> k$
    let $h$ be a subgate of $g$ of size $k$
    $b \leftarrow \begin{cases} 1 & \text{with probability } 1-q \\ 0 & \text{with probability } q \end{cases}$
    $b' \leftarrow \begin{cases} 1 & \text{if } h \text{ is an AND gate} \\ 0 & \text{if } h \text{ is an OR gate} \end{cases}$
    $F \leftarrow F|(h = b \text{ XOR } b')$
    if $b = 0$, $V \leftarrow V - \mathrm{var}(h)$
    if $|F| > 2c|V|$ // guess assignment
      choose $a \in_u 2^{\mathrm{var}(F)}$
      if $F(a) = 1$, return $a$
      return "probably not satisfiable"
  return $A_{d,2c,k}(F,V)$

$A_{d,c,k}(F,V)$ // $F$ has depth $d$, $F$ also has bottom fan-in $\leq k$, $\mathrm{var}(F) \subseteq V, |V| = n$
  if $d \leq 2$, return PPZ_main($F$)
  choose $p, c'$ // as functions of $d, c, k$
  $W \leftarrow$ find_restriction($F, V, p$)
  choose $a \in_u 2^W$
  // the bottom level gates of $F|a$ are trivial
  $F' \leftarrow F|a$ but collapsing the bottom level
  return $A_{d-1,c'}(F', V - W)$

find_restriction($F, V, p$) // $|V| = n$
  $B \leftarrow \{$bottom gates of $F\}$
  $U \leftarrow$ random subset of $V$ of size $(1-p)n$
  $G \leftarrow \{g \in B \mid |\mathrm{var}(g) - U| > 1\}$
  $U' \leftarrow \mathrm{var}(G)$
  if $|U'| \leq \frac{1}{2}pn$, return $U \cup U'$
  else die // algorithm fails
```

Fig. 1. Linear size, constant depth circuit solver

3 Run Time Analysis

Suppose $A_{d,c}, A_{d,c,k}$ succeed with probability $\geq 2^{-(1-a_{d,c})n}, 2^{-(1-a_{d,c,k})n}$, respectively, given that find_restriction succeeds on each call to it – we will eliminate this assumption later. Here n is the number of variables in V at the time $A_{d,c}$ or $A_{d,c,k}$ are invoked. We assume $c \geq 2$ and $\forall d \geq 2, k \geq 4\ a_{d,c}, a_{d,c,k} \leq \frac{1}{4}$.

Lemma 1. *$\forall d, c \geq 2$ if $k \geq 4 \lg \frac{4c}{a_{d,2c,k}}$, then $a_{d,c} \geq \frac{1}{2} a_{d,2c,k}$.*

Proof. Each iteration of the while loop of $A_{d,c}(F, V)$ eliminates (1) a gate or (2) k variables. (1) occurs $\leq cn$ times and (2) occurs $r \leq \frac{n}{k}$ times. Let a be a solution to F. Exactly one sequence of random choices, say with r choices of type (2), can lead $A_{d,c}(F, V)$ to find a. So the probability that $A_{d,c}(F, V)$ finds a solution given that each call to find_restriction succeeds, is

$$\geq q^r (1-q)^{cn} \min\{2^{-(1-a_{d,2c,k})(n-kr)}, 2^{-\frac{n}{2}}\}.$$

(Note carefully that we are *not* asserting that with at least this probability a is found. This is because PPZ_main may return another solution.)

To lower bound $q^r(1-q)^{cn} 2^{-(1-a_{d,2c,k})(n-kr)}$, take the logarithm, divide by n, and set $r' = \frac{r}{n} \in [0, \frac{1}{k}]$ to get

$$\begin{aligned}
&r' \lg q + c \lg(1-q) - (1 - a_{d,2c,k}) + (1 - a_{d,2c,k}) k r' \\
=& -(1 - a_{d,2c,k}) + c \lg(1-q) + r'(\lg q + (1 - a_{d,2c,k})k) \\
\geq& -(1 - a_{d,2c,k}) - 2cq + r'(\lg q + (1 - a_{d,2c,k})k) \\
&\quad (\text{taking } n = 4, x = nq \text{ in the fact } \forall n \geq 1, x \in [0,1] \left(1 - \frac{x}{n}\right)^{n-1} \geq e^{-x}) \\
\geq& -(1 - a_{d,2c,k}) - 2cq + r'\Big(\lg q + \frac{1}{2}k\Big) \quad \text{since } a_{d,2c,k} \leq \frac{1}{2},
\end{aligned}$$

which is $\geq -(1 - \frac{1}{2} a_{d,2c,k})$ if we set $q = \frac{a_{d,2c,k}}{4c}, k \geq -2 \lg q$.

To lower bound $q^r(1-q)^{cn} 2^{-\frac{n}{2}}$, note that if we choose $k \geq -4 \lg q$, then

$$\begin{aligned}
&r' \lg q + c \lg(1-q) - \frac{1}{2} \\
\geq& \frac{1}{k} \lg q - 2cq - \frac{1}{2} \\
\geq& -\frac{1}{4} - \frac{1}{2} a_{d,2c,k} - \frac{1}{2} \\
\geq& -\Big(1 - \frac{1}{2} a_{d,2c,k}\Big) \quad \text{since } a_{d,2c,k} \leq \frac{1}{4}.
\end{aligned}$$

Lemma 2. *If $|V| = n$, $|F| \leq cn$, F has bottom fan-in $\leq k$, and we choose $p = \frac{1}{2ck^3}$, then the probability that* find_restriction(F, V, p) *dies is $\leq \frac{1}{2}$.*

Proof. Let $g \in B$ and $X = |\text{var}(g) - U|$. g has a fan-in $k' \leq k$ and so X is hypergeometric with parameters n, k', pn. We claim that $Pr(g \in G) = Pr(X \geq$

$2) \leq \binom{k'}{2}p^2$. To see this, note that the sample points where $X \geq 2$ can be partitioned according to the positions among the k' variables of g of the first 2 free variables (*i.e.*, not in U), and the probability of any of these $\binom{k'}{2}$ events is $\leq p^2$. So

$$E(|U'|) \leq E(|G|)k \leq \binom{k}{2}p^2|B|k \leq \frac{1}{2}k^3p^2cn \leq \frac{1}{4}pn.$$

By Markov's inequality,

$$Pr\Big(|U'| > \frac{1}{2}pn\Big) \leq \frac{E(|U'|)}{\frac{1}{2}pn} \leq \frac{1}{2}.$$

So the probability that find_restriction(F, V, p) dies is $\leq \frac{1}{2}$.

Lemma 3. *If we set* $p = \frac{1}{2ck^3}, c' = 4c^2k^3$, *then* $a_{d,c,k} \geq \frac{1}{4ck^3}a_{d-1,4c^2k^3}$.

Proof. $A_{d,c,k}(F,V)$ leaves $f \in [\frac{1}{2}pn, pn]$ variables of V free and sets the rest. So $\frac{|F'|}{|V-W|} \leq \frac{cn}{\frac{1}{2}pn} = 4c^2k^3 = c'$. (The proof of this lemma would have been confounded if we had used var(F) to keep track of variables instead of V.) The probability that $A_{d,c,k}(F,V)$ finds a solution, assuming each call to find_restriction succeeds, is then

$$2^{-(n-f+(1-a_{d-1,c'})f)} = 2^{-(n-a_{d-1,c'}f)} \geq 2^{-(1-\frac{1}{2}pa_{d-1,c'})n},$$

and the lemma follows.

Lemma 4. $\forall d, c \geq 2$,

$$a_{d,c} \geq 1/O(c^{2^{d-2}-1}\lg^{3\cdot 2^{d-2}-2} c),$$

where the constant in the big-Oh depends only on d.

Proof. We use induction to show that for each d, k can be chosen to be $O(\lg c)$ so as to satisfy the hypothesis of lemma 1 and $a_{d,c} \geq 1/O(c^{f(d)}\lg^{g(d)} c)$ for some functions f, g where the constants in the big-Ohs depend only on d.

$a_{2,c,k} = \frac{1}{k}$. From lemma 1, we need to choose k such that $k \geq O(\lg \frac{c}{a_{2,2c,k}}) = O(\lg c + \lg k)$. So $k = O(\lg c)$ suffices, and we conclude that $a_{2,c} \geq 1/O(\lg c)$. So $f(2) = 0, g(2) = 1$. This completes the base case.

From the inductive hypothesis and lemma 3,

$$\begin{aligned} a_{d,c,k} &\geq 1/O(ck^3(c^2k^3)^{f(d-1)}\lg^{g(d-1)}(c^2k^3)) \\ &= 1/O(c^{2f(d-1)+1}k^{3f(d-1)+3}\lg^{g(d-1)}(ck)). \end{aligned}$$

To use lemma 1, we need to choose k such that

$$\begin{aligned} k &\geq O(\lg(c^{2f(d-1)+2}k^{3f(d-1)+3}\lg^{g(d-1)}(ck))) \\ &= O(\lg c + \lg k). \end{aligned}$$

So $k = O(\lg c)$ suffices, and we conclude that

$$a_{d,c} \geq 1/O(c^{2f(d-1)+1} \lg^{3f(d-1)+3+g(d-1)} c).$$

So we have the recurrence

$$\begin{aligned} f(d) &= 2f(d-1) + 1 & f(2) &= 0 \\ g(d) &= g(d-1) + 3f(d-1) + 3 & g(2) &= 1 \end{aligned}$$

which has solution $f(d) = 2^{d-2} - 1, g(d) = 3 \cdot 2^{d-2} - 2$.

Theorem 1. *$\forall d, c \geq 2$, the probability that $A_{d,c}(F, V)$ finds some solution is $\geq 2^{-(1-\alpha)n}$ where*

$$\alpha \geq 1/O(c^{2^{d-2}-1} \lg^{3 \cdot 2^{d-2}-2} c),$$

where the constant in the big-Oh depends only on d.

Proof. This is a corollary from the previous lemma together with the following: find_restriction is called $\leq d$ times, each with success probability $\geq \frac{1}{2}$, so the probability that in every call it succeeds is $\geq 2^{-d}$, a penalty that can be absorbed into the big-Oh in the theorem statement.

Again, we are not asserting that every solution is found with this probability, just that some is, and this asymmetric property is inherited from the PPZ algorithm.

4 Open Problems

Can find_restriction be derandomized without too much performance penalty? Can we find a nontrivial algorithm for the case where d grows very slowly with n?

Acknowledgments: We would like to thank Mike Saks for useful discussions.

References

[BBHT96] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching, May 1996.

[Cal08] Chris Calabro. A lower bound on the size of series-parallel graphs dense in long paths. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(110), 2008.

[CIKP08] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. *J. Comput. Syst. Sci.*, 74(3):386–393, 2008.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for sat. In *CCC '06: Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, pages 252–260, Washington, DC, USA, 2006. IEEE Computer Society.

[DW05] Evgeny Dantsin and Alexander Wolpert. An improved upper bound for sat. In Fahiem Bacchus and Toby Walsh, editors, *SAT*, volume 3569 of *Lecture Notes in Computer Science*, pages 400–407. Springer, 2005.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219, 1996.

[IP01] Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[Nur09] Sergey Nurik, 2009. Personal communication, to appear in ECCC.

[PPSZ05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

[PPZ99] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago Journal of Theoretical Computer Science*, 1999(115), December 1999.

[PSZ00] R. Paturi, M.E. Saks, and F. Zane. Exponential lower bounds for depth 3 boolean circuits. *Computational Complexity*, 9(1):1–15, November 2000. Preliminary version in *29th annual ACM Symposium on Theory of Computing*, 96–91, 1997.

[Sch99] U. Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *FOCS*, pages 410–414, 1999.

[Sch05] Rainer Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, 54(1):40–44, 2005.

[Val77] L. Valiant. Graph-theoretic arguments in low level complexity. In *Proceedings of the 6th Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 53*, 1977.

```
$A_{d,c}(F, V)$
  if $|F| \geq 2c|V|$ // solve by brute force
    for each $a \in 2^{\mathrm{var}(F)}$
      if $F(a)$, return 1
    return 0
  choose $k$ // as some function of $d, c$
  if $\exists$ bottom gate $g$ in $F$ of fan-in $> k$ // branch
    let $h$ be a subgate of $g$ of size $k$
    $(V_0, V_1) \leftarrow \begin{cases} (V, V - \mathrm{var}(h)) & \text{if } h \text{ is an AND gate} \\ (V - \mathrm{var}(h), V) & \text{if } h \text{ is an OR gate} \end{cases}$
    if $A_{d,c}(F|(h = 1), V_1)$, return 1
    if $A_{d,c}(F|(h = 0), V_0)$, return 1
    return 0
  return $A_{d,2c,k}(F, V)$
$A_{d,c,k}(F, V)$
  if $d \leq 2$, return PPZ$(F)$
  // PPZ solves $k$-SAT in time poly$(|F|)2^{(1-\frac{1}{k})n}$
  // with exponentially small error probability
  choose $p, c'$ // as functions of $d, c, k$
  $W \leftarrow$ find_restriction$(F, V, p)$    // same subroutine as in figure 1
  for each $a \in 2^W$
    // the bottom level gates of $F|a$ are trivial
    $F' \leftarrow F|a$ but collapsing the bottom level
    if $A_{d-1,c'}(F', V - W)$, return 1
  return 0
```

Fig. 2. Linear size, constant depth circuit solver, backtracking version