The Complexity of Satisfiability of Small Depth Circuits

Chris Calabro, Russell Impagliazzo, Ramamohan Paturi
University of California, San Diego

June 10, 2009

Abstract

We consider the satisfiability problem for circuits of limited size and/or depth. Say that an algorithm solving a Boolean satisfiability problem on n variables is *improved* iff it takes time $O(2^{cn})$ for some constant $c < 1$, i.e. iff it is exponentially better than a brute force search. We show an improved algorithm for the satisfiability problem for circuits of constant depth and linear size. If improved upper bounds are not possible for a variant where the size is somewhat more than linear or the depth grows, can we provide evidence regarding the hardness of the problem? (note to authors: Did we actually discuss this question in greater depth in the paper? Maybe this should be reworded.)

For each d and c, we give a randomized algorithm solving the satisfiability problem for depth d circuits with n variables and at most cn gates in time $2^{(1-\delta)n}$ where $\delta \geq 1/O(c^{2^{d-2}-1} \lg^{3\cdot 2^{d-2}-2} c)$, 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 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 general 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 to these problems [Sch99, PPZ99, PPSZ05, Sch05, DW05, CIP06], 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 consider the exact complexity of the satisfiability problem for circuits of limited size and depth, which seems significantly harder than k-SAT. We give what we believe are the first improved algorithms for the satisfiability problem for circuits of constant depth $d > 2$ (AC^0 type). (There is some work that improves the performance of SAT algorithms in terms of the circuit size parameter, but these algorithms are no better than exhaustive search once the circuit size gets larger than around $4n$ or so.) 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, the randomization procedure described above also yields the best quantum algorithm for this case, with running time $O(2^{\frac{1-1/O(\lg c)}{2}n})$. 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 $\text{poly}(m)2^{cn}$ 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 *Strongly 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 a more general circuit can compute more complicated functions, it may become 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 polytime 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 polytime 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. In this paper, we 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 $2^{-(1-\delta)n}$.

2 Improved upper bounds for the satisfiability of bounded depth circuits of linear size

2.1 Definitions

A *literal* is a variable or its negation. 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 $d-1$.

We define the *savings* for an algorithm A whose input ϕ contains n boolean variables is the supremum of δ such that A has run time $\leq \text{poly}(|\phi|)2^{(1-\delta)n}$. We can think of the savings as the fraction of variables 'saved' over the run time of an exhaustive search algorithm.

2.2 High level description

Assume F has n variables and at most cn gates. To test whether F is satisfiable, we first reduce the fan-in of each bottom gate to some k by branching on a k-subset of the edges coming into any bottom gate with fan-in $> k$. We branch until either we've halved the original number of variables, in which case we simply use exhaustive search to solve, or each bottom gate has fan-in $\leq k$, in which case we choose a random restriction of $(1-p)n$ variables for some p. This may leave some bottom level gates with > 1 free variable. We clean up these gates by branching on all possible assignments to them. By choosing k, p appropriately, w.h.p. there will still be $\Omega(pn)$ free variables, but the bottom level gates will each have at most 1 free variable. So we can collapse the circuit to depth $d-1$ and recurse.

The random restriction technique used in this paper is a simpler version of the technique used in [IPS97] to obtain nonlinear lower bounds of the number of edges of bounded depth threshold circuits.

2.3 Detailed description

We describe our algorithm in several subroutines:

- The functions $\text{main}_{d,c}(F), A_{d,c}(F), A'_{d,c}$ all test the satisfiability of F when F has depth d and at most cn gates. $\text{main}_{d,c}$ is a wrapper function that sets the global variable n_0 to the number of variables in the initial call. The function $A_{d,c}$ is a wrapper for the function $A'_{d,c}$. The initial invocation $A'(d, 2c)(F)$ from the function $A_{d,c}$ is such that the ratio of gates to variables in F is at most c, but A' will set variables, increasing the ratio. If the ration ever exceeds $2c$, A' will resort to exhaustive search since this means at least half of the variables must have been set since the most recent call to $A_{d,c}$.

- $A_{d,c,k}(F)$ will test the satisfiability of F when F has depth d, at most cn gates, and bottom fan-in $\leq k$.
- find_restriction(F,p) finds a set of variables W in F 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 ≤ 1 free variable.
- PPZ is the k-SAT solver (which is the same as a depth 2 circuit solver) from [PPZ99] whose run time has savings $1/k$. We also assume this algorithm handles depth 1 circuits in poly time.

Below, the choices of k, p, c' are unspecified, and are left for the analysis section.

If h is an AND of literals, then $F|h$ 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 $F|h$ is simply false. If h is an OR of literals, $F|h$ 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 $\neg h$ is the OR (AND) of the negations of those literals.

```
main_{d,c}(F) // wrapper function, F has depth d, ≤ cn gates
  n_0 ← |var(F)|
  return A_{d,c}(F)

A_{d,c}(F) // wrapper function, F has depth d, ≤ cn gates
  return A'_{d,2c}(F)


A'_{d,c}(F)
  if the gates to variables ratio is > c // solve by brute force
    for each a ∈ 2^{var(F)}
      if F(a), return 1
    return 0
  choose k // as some function of d, c
  if ∃ bottom gate g in F of fan-in > k // branch
    let h be a subgate of g of size k
    if A'_{d,c}(F|h), return 1
    if A'_{d,c}(F|¬h), return 1
    return 0
  return A_{d,c,k}(F)


A_{d,c,k}(F) // F has depth d, ≤ cn gates, bottom fan-in ≤ k
  if d ≤ 2, return PPZ(F)
  choose p, c' // as functions of d, c, k
  W ← find_restriction(F, p)
  for each a ∈ 2^W
    // the bottom level gates of F|a are trivial
    F' ← F|a but collapsing the bottom level
    if A_{d-1,c'}(F'), return 1
  return 0
```

```
find_restriction(F, p) // F has n variables
  B ← {bottom gates of F}
  do 2n_0 times
    U ← random subset of var(F) of size (1 − p)n
    G ← {g ∈ B | |var(g) − U| > 1}
    V ← var(G)
    if |V| ≤ (1/2)pn
      return U ∪ V
  die // algorithm fails
```

2.4 Run time analysis

Let $a_{d,c}, a_{d,c,k}$ be the savings (see Section 2.1 for the definition) in the run time for $A_{d,c}, A_{d,c,k}$ *not counting* the time spent in find_restriction in any subcall. We assume $c \geq 2$ and $\forall d \geq 2, k \geq 4$ $a_{d,c}, a_{d,c,k} \leq \frac{1}{4}$.

First we analyze $A_{d,c}$. Let T be the call tree of the invocation $A'_{d,2c}(F)$, where leaves are either calls to $A_{d,2c,k}$ or exhaustive searches and the internal nodes are calls to $A'_{d,2c}$. Each branch either eliminates a gate or k variables. So each path has at most cn branches of the first type and at most $r \leq \frac{n}{k}$ of the second type, where $n = |\mathrm{var}(F)|$.

The amount of time T_0 spent on all the leaves that use exhaustive search is at most $\mathrm{poly}(n)2^{\frac{n}{2}}$ times

$$\leq \sum_{r=0}^{\frac{n}{k}} \binom{cn+r}{r} \leq \sum_{r=0}^{\frac{n}{k}} \binom{2cn}{r} \leq 2^{h(\frac{1}{2ck})2cn+1},$$

where h is the binary entropy function. Since $\forall \epsilon \in (0, \frac{1}{2}]$ $h(\epsilon) \leq \epsilon \lg(\frac{4}{\epsilon})$,

$$h\Big(\frac{1}{2ck}\Big)2c \leq \frac{1}{2ck}\lg(8ck)2c = \frac{\lg c + \lg k + 3}{k},$$

which is at most $\frac{1}{4}$ for sufficiently large c if we choose

$$k \geq 5 \lg c. \tag{1}$$

So $T_0 \leq \mathrm{poly}(n)2^{\frac{3}{4}n}$.

The amount of time T_1 spent on all the leaves that call $A_{d,2c,k}$, *not counting* time spent in find_restriction in any subcall, is at most $\mathrm{poly}(n)$ times

$$\begin{aligned}&\sum_{r=0}^{\frac{n}{k}} \binom{cn+r}{r} 2^{(1-a_{d,2c,k})(n-kr)}\\ \leq &2^{(1-a_{d,2c,k})n} \sum_{r=0}^{2cn} \binom{2cn}{r} 2^{-(1-a_{d,2c,k})kr}.\end{aligned}$$

The summation is $(1 + 2^{-(1-a_{d,2c,k})k})^{2cn} \leq 2^{2^{-\frac{k}{2}+2}cn}$ since $a_{d,2cn,k} \leq \frac{1}{2}$. We want $2^{-\frac{k}{2}+2}c \leq \frac{1}{2}a_{d,2c,k}$, which happens if we choose

$$k \geq 2 \lg \frac{c}{a_{d,2c,k}} + 6. \tag{2}$$

Both conditions (1),(2) are implied by

$$k \geq 5 \lg \frac{c}{a_{d,2c,k}} \tag{3}$$

since $c \geq 2, a_{d,2c,k} \leq \frac{1}{4}$. So $T_1 \leq \text{poly}(n) 2^{(1-\frac{1}{2}a_{d,2c,k})n}$ provided k can be chosen to satisfy (3). Since we assume $a_{d,2c,k} \leq \frac{1}{2}$, T_0 is no more than this bound, and we have the following.

Lemma 1. *$\forall d, c \geq 2$ if $k \geq 5 \lg \frac{c}{a_{d,2c,k}}$, then $a_{d,c} \geq \frac{1}{2} a_{d,2c,k}$.*

Now we analyze find_restriction. 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 at most p^2. So

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

We want $E(|V|) \leq \frac{1}{4} pn$, and this happens if we choose

$$p = \frac{1}{2ck^3}. \tag{4}$$

By Markov's inequality,

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

So the probability that any call to find_restriction dies is at most 2^{-2n_0} and the time spent in each call is at most $\text{poly}(n_0)$.

Now consider the loop of $A_{d,c,k}$, which leaves $f \in [\frac{1}{2} pn, pn]$ variables free and sets the rest. The gates to variables ratio for F' is at most $\frac{cn}{\frac{1}{2} pn} = 4c^2 k^3$. So set

$$c' = 4c^2 k^3. \tag{5}$$

The running time of $A_{d,c,k}$, *not counting* the time spent in find_restriction, is then at most $\text{poly}(n)$ times

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

and we have the following.

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

Lemma 3. *$\forall d, c \geq 2$,*

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

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 (3) 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 2,

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

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

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

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 savings for $\text{main}_{d,c}$ is at least*

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

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

PROOF: This is a corollary from the previous lemma together with the following observations. The total time spent in find_restriction is at most $\text{poly}(n_0)$ times the number of calls to $A_{d',c',k'}$ for all values of d', c', k', so it suffices to upper bound the time spent elsewhere. Also, from the union bound, and since the number of these calls is at most 2^{n_0}, the probability of dying in some call to find_restriction is at most 2^{-n_0}. ■

2.5 Fully randomizing the algorithm

We now sketch how to make the algorithm run in polytime and succeed with probability $\geq 2^{-(1-\delta)n}$ where $\delta \geq 1/O(c^{2^{d-2}-1} \lg^{3 \cdot 2^{d-2}-2} c)$. The 'exhaustive search' in $A'_{d,c}$ should be changed to simply choose a random assignment a and return $F(a)$. Also, instead of calling both $A'_{d,c}(F|h)$ and $A'_{d,c}(F|\neg h)$, we instead call just one of them randomly. In particular, call the one that eliminates k variables with some probability q and the other with probability $1 - q$. The iteration over all $a \in 2^W$ in $A_{d,c,k}$ should be changed to choose a random $a \in 2^W$.

It can be shown that choosing $q = \frac{a_{d,c,k}}{4c}$ allows us to use almost exactly the same analysis as before, only instead of upper bounding run time, now we lower bound success probability. Details can be found in the Appendix.

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.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size–depth tradeoffs for threshold circuits. *SIAM J. Comput.*, 26(3):693–707, 1997.

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

[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.

Appendix

3 Polytime version of depth d, linear size circuit algorithm

3.1 High level description

Assume F has n variables and at most cn gates. To test whether F is satisfiable, we first reduce the fan-in of each bottom gate to some k by repeatedly selecting a k-subset h of the edges coming into any bottom gate g with fan-in $> k$ and setting h to either true or false. One of these settings will eliminate k variables 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 guessing until either we've halved the original number of variables, in which case we simply guess an assignment to the remaining variables, or each bottom gate has fan-in at most k, in which case we choose a random restriction of $(1 - p)n$ variables for some p. This may leave some bottom level gates with > 1 free variable. We clean up these gates by randomly setting all the variables in them. By choosing k, p appropriately, with probability $\geq \frac{1}{2}$ there will still be $\Omega(pn)$ free variables, but the bottom level gates will each have at most 1 free variable. So we can collapse the circuit to depth $d - 1$ and recurse.

This procedure takes polynomial time and has exponentially small probability s of finding a satisfying assignment. By iterating s^{-1} times, we increase the probability of success to a constant.

3.2 Detailed description

We describe our algorithm in several subroutines:

- $A_{d,c}(F)$ tests the satisfiability of F when F has depth d and at most cn gates. F initially has gates to variables ratio of at most c, but it may set variables, increasing the ratio. If it ever exceeds $2c$, $A_{d,c}$ will simply guess an assignment to the remaining variables since this means at least half of the variables must have been set.

- $A_{d,c,k}(F)$ will test the satisfiability of F when F has depth d, at most cn gates, and bottom fan-in at most k.

- find_restriction(F, p) finds, with probability $\geq \frac{1}{2}$, a set of variables W in F 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 1 free variable.

- PPZ_iterate is 1 iteration of the k-SAT solver (which is the same as a depth 2 circuit solver) from [PPZ99] whose run time has savings $1/k$ - *i.e.*, PPZ_iterate takes polytime and has success probability $\geq 2^{-(1-\frac{1}{k})n}$. More explicitly, it chooses a random permutation of the variables, then assigns them one at a time, in that order, uniformly randomly, unless the current variable appears in a unit clause C, in which case it is set as C demands, and simplifies the formula after each iteration by removing true clauses and false literals. We also assume this algorithm handles depth 1 circuits in polytime.

Our algorithm description is not the most efficient, compromises were made to simplify the proof. For example, it is easy to construct an equivalent algorithm containing only 2 subroutines, but the analysis would be 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 $1-q$.

$A_{d,c}(F)$ // F has depth d, at most cn gates
  choose k, q // as some function of d, c
  while $\exists$ bottom gate g in F of fan-in $> k$
    if the gates to variables ratio is $> c$ // guess assignment
      $a \leftarrow$ random assignment to var(F)
      return $F(a)$
    let h be a subgate of g of size k
    $b \leftarrow \begin{cases} 1 & \text{with probability } 1-q \\ 0 & \text{with probability } q \end{cases}$
    $b' \leftarrow \begin{cases} 1 & \text{if } h \text{ is an AND gate} \\ 0 & \text{if } h \text{ is an OR gate} \end{cases}$
    $F \leftarrow F|(h = b \text{ XOR } b')$
  return $A_{d,2c,k}(F)$

$A_{d,c,k}(F)$ // F has depth d, at most cn gates, bottom fan-in $\leq k$
  if $d \leq 2$, return PPZ_iterate(F)
  choose p, c' // as functions of d, c, k
  $W \leftarrow$ find_restriction(F, p)
  $a \leftarrow$ random assignment to W
  // the bottom level gates of $F|a$ are trivial
  $F' \leftarrow F|a$ but collapsing the bottom level
  return $A_{d-1,c'}(F')$

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

3.3 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. 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 4. *$\forall d, c \geq 2$ 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 $A_{d,c}(F)$ 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), will lead $A_{d,c}(F)$ to find a. So the probability that $A_{d,c}(F)$ finds a given that each call to find_restriction succeeds, is

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

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

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

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

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

■

Now we analyze find_restriction. 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

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

We want $E(|V|) \leq \frac{1}{4}pn$, and this happens if we choose

$$p = \frac{1}{2ck^3}. \tag{6}$$

By Markov's inequality,

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

So the probability that any call to find_restriction dies is $\leq \frac{1}{2}$.

Now consider $A_{d,c,k}(F)$, which leaves $f \in [\frac{1}{2}pn, pn]$ variables free and sets the rest. The gates to variables ratio for F' is $\leq \frac{cn}{\frac{1}{2}pn} = 4c^2k^3$. So set

$$c' = 4c^2k^3. \tag{7}$$

The probability that $A_{d,c,k}(F)$ finds a, 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 we have the following.

Lemma 5. *If we set* $p = \frac{1}{2ck^3}, c' = 4c^2k^3$, *then* $a_{d,c,k} \geq \frac{1}{4ck^3} a_{d-1,4c^2k^3}$.

Lemma 6. $\forall d, c \geq 2$,

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

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 4 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 4, 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 5,

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

To use lemma 4, 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

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

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 2. $\forall d, c \geq 2$, *the probability that* $A_{d,c}(F)$ *finds a 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. ■