Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms

James Cook[1,⋆], Omid Etesami[1,⋆⋆], Rachel Miller[2,⋆⋆⋆], and Luca Trevisan[1,†]

[1] Computer Science Division, U.C. Berkeley
{jcook,etesami,luca}@cs.berkeley.edu
[2] University of Virginia
rachel.an.miller@gmail.com

Abstract. Goldreich (ECCC 2000) proposed a candidate one-way function construction which is parameterized by the choice of a small predicate (over $d = O(1)$ variables) and of a bipartite expanding graph of right-degree d. The function is computed by labeling the n vertices on the left with the bits of the input, labeling each of the n vertices on the right with the value of the predicate applied to the neighbors, and outputting the n-bit string of labels of the vertices on the right.

Inverting Goldreich's one-way function is equivalent to finding solutions to a certain constraint satisfaction problem (which easily reduces to SAT) having a "planted solution," and so the use of SAT solvers constitutes a natural class of attacks.

We perform an experimental analysis using MiniSat, which is one of the best publicly available algorithms for SAT. Our experiment shows that the running time required to invert the function grows exponentially with the length of the input, and that such an attack becomes infeasible already with small input length (a few hundred bits).

Motivated by these encouraging experiments, we initiate a rigorous study of the limitations of back-tracking based SAT solvers as attacks against Goldreich's function. Results by Alekhnovich, Hirsch and Itsykson imply that Goldreich's function is secure against "myopic" backtracking algorithms (an interesting subclass) if the 3-ary parity predicate $P(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ is used. One must, however, use non-linear predicates in the construction, which otherwise succumbs to a trivial attack via Gaussian elimination.

We generalized the work of Alekhnovich et al. to handle a more general class of predicates, and we present a lower bound for the construction that uses the predicate $P_d(x_1, \ldots, x_d) := x_1 \oplus x_2 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ and a random graph.

⋆ Work supported by the National Science Foundation under grant No. CCF-0729137 and by the National Sciences and Engineering Research Council of Canada under a PGS award.

⋆⋆ Work supported by the National Science Foundation under grant No. CCF-0729137.

⋆⋆⋆ Work done at U.C. Berkeley, supported by an NSF SUPERB fellowship.

† Work supported by the National Science Foundation under grant No. CCF-0729137 and by the BSF under grant 2002246.

O. Reingold (Ed.): TCC 2009, LNCS 5444, pp. 521–538, 2009.

1 Introduction

Goldreich [11] proposed in 2000 a candidate one-way function construction based on expanding graphs. His construction is parameterized by the choice of a bipartite graph with n vertices per side and right-degree d (where d is either a constant independent of n, or grows very moderately as $O(\log n)$) and of a boolean predicate $P : \{0,1\}^d \to \{0,1\}$. To compute the function, on input $x \in \{0,1\}^n$ we label the vertices on the left by the bits of x, and we label each vertex on the right by the value of P applied to the label of the neighbors. The output of the function is the sequence of n labels of the vertices on the right.

Goldreich's Function and Cryptography in NC_0. A function is computable in NC_0 if every bit of the output depends only on a constant number of bits of the input. One can see any NC_0-computable function as a generalization of Goldreich's function in which the graph is allowed to be arbitrary, subject to having bounded right-degree, and in which different predicates can be used for different bits of the output.

Cryan and Miltersen [7] first raised the question of whether cryptographic primitives (their work focused on pseudorandom generators) can be computed in NC_0. Mossel, Shpilka and Trevisan [13] construct, for arbitrarily large constant c, a function $f : \{0,1\}^n \to \{0,1\}^{cn}$ based on a bipartite graph of right-degree 5 and the fixed predicate $P(x_1, \cdots, x_5) := x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$, and show that the function computes a small-bias generator. Such a construction may in fact be a pseudorandom generator, and hence a one-way function.[1]

Applebaum, Ishai and Kushilevtiz [4,5] show that, under standard assumptions, there are one-way functions and pseudorandom generators that can be computed in NC_0; their one-way function is computable with right-degree 3.[2] In their construction, the graph encodes the computation of a log-space machine computing a one-way function that is used as a primitive.

In this paper, we are interested in the security of Goldreich's original proposal, implemented using a random graph and a fixed predicate.

Goldreich's Function and DPLL Algorithms. Inverting Goldreich's one-way function (and, indeed, inverting any one-way function that is computable in NC_0) can be seen as the task of finding a solution to a constraint satisfaction problem with a planted solution. A plausible line of attack against such a construction is thus to employ a general-purpose SAT solver to solve the constraint satisfaction problem. We performed an experimental study using MiniSat, which is one of the best publicly available SAT solvers, and has solved instances with several thousand variables. Using a random graph of right-degree 5, and the

[1] The graph used in this construction, however, is not a random graph or a strong expander graph of right-degree 5, so this is not an instantiation of Goldreich's proposal.

[2] This is the best possible, because it is easy to show that no function based on a bipartite graph of right-degree 2 can be one-way, by reducing the problem of finding the inverse to a 2SAT instance.

predicate $(x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5))$, we observed an exponential increase of the running time as a function of the input length, and an attack with MiniSat appears infeasible already for moderate input lengths (a few hundred bits). See Appendix A.

Our goal in this paper is to provide a rigorous justification for these experimental results, and to show that "DPLL-style" algorithms based on backtracking (such as most general SAT solvers) cannot break Goldreich's construction in sub-exponential time. We restrict ourselves to algorithms that instantiate variables one at a time, in an order chosen adaptively by a "scheduler" procedure, and then recurse on the instance obtained by by fixing the variable to zero and then to the instance obtained by fixing the variable to one, or viceversa (the scheduler decides which assignment to try first). The recursion stops if the current partial assignment contradicts one of the constraints in the instance, or if we find a satisfying assignment.

When such an algorithm runs on an unsatisfiable instance, then a transcript of the algorithm gives a "tree-like resolution proof" of unsatisfiability; a number of techniques are known to prove exponential lower bounds on the size of tree-like resolutions proofs of unsatisfiability, and so such proofs give lower bounds to the running time of any such algorithm, regardless of how the scheduler is designed.

When dealing with *satisfiable* instances, however, one cannot prove lower bounds without putting some restriction on the scheduler. (If unrestricted in complexity, the scheduler could compute a satisfying assignment, and then assign the variables accordingly, making the algorithm converge in a linear number of steps.)

The Lower Bound of Alekhnovich et al. Alekhnovich, Hirsch and Itsykson [3] consider two such restrictions: they consider (i) "myopic" algorithms in which the scheduler chooses which variable to assign based on only a bounded number of variables and clauses of the current formula, and (ii) "drunken" algorithms in which the order of variables is chosen arbitrarily by the scheduler, but the choice of whether to assign first zero or one to the next chosen variable is made randomly with equal probability. The result of the second type is proven for carefully designed instances, and it remains an open question to prove a lower bound for drunken algorithms on a random satisfiable constraint satisfaction problem. Lower bounds of the first type are proven for random instances, and they are proved via a reduction to the problem of certifying unsatisfiability: Alekhnovich et al. show that a myopic algorithm, with high probability, after assigning a certain number of variables will be left with an instance that is unsatisfiable, but for which there is no sub-exponential size tree-like resolution proof of unsatisfiability. Hence the algorithm will take an exponential amount of time before it realizes it has chosen a bad partial assignment.

Our Results. The result of Alekhnovich et al. applies to myopic algorithms for random instances of 3XOR with a planted solution, and provided a lower bound for myopic DPLL inversion algorithms for the instantiation of Goldreich's proposal using the 3XOR predicate.

Unfortunately, the use of 3XOR as a predicate in Goldreich's construction leads to a total break via Gaussian elimination, so our goal is to extend the result of Alekhnovich et al. to a setting in which we have either a random predicate or the predicate $(x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d))$ which is inspired by the work of Mossell et al.

In order to extend the work of Alekhnovich et al. to the setting of Goldreich's one way functions, we need to make the following changes:

- The proof in [3] uses the fact that all constraints have arity 3. It is not difficult to adapt it to handle linear constraints of larger constant arity, by relying on the strong expansion properties which are true of random constraint graphs.
- The proof in [3] uses the linearity of the constraints. We show that it is sufficient for the predicate to be such that it remains nearly balanced even after many variables have been fixed to arbitrary values. For example, a d-ary parity remains perfectly balanced even after $d-1$ variables are assigned arbitrary variables. The predicate $(x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d))$ remains perfectly balanced even after $d-3$ variables are assigned arbitrary variables, and a random predicate remains ϵ-close to balanced after any $d - O(\log d/\epsilon)$ variables are fixed to arbitrary values. (Those parameters are sufficient for our proof to go through.)
- The proof in [3] assumes that there is a unique solution, and this is not true in our setting. We show that the proof carries over if one assumes that the total number of pairs x, y such that $f(x) = f(y)$ is at most $2^{(1+\epsilon)n}$ for small ϵ. We are able to show that such a condition is satisfied by the predicate $(x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d))$ and by the choice of a highly-expanding graph, with $\epsilon = 2^{-\Omega(d)}$. We believe that the same result holds with high probability if we choose a random d-ary predicate, but we have not been able to prove it.

With such results, we are able to show an exponential lower bound for myopic algorithms in a construction that uses a random graph and the predicate $(x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d))$. If we consider the construction that uses a random graph and a random predicate, then we have a conditional exponential lower bound under the assumption that the resulting function is nearly injective.

Goldreich's Analysis. Goldreich [11] considered the following algorithm for computing x given $y = f(x)$. The algorithm proceeds in n steps, revealing the output bits one at a time. Let R_i be the set of inputs connected to the first i outputs. Then in the ith step, the algorithm computes the list L_i of all strings in $\{0,1\}^{R_i}$ which are consistent with the first i bits of y. Goldreich proves that if the graph satisfies an expansion condition, then for a random input x, the expected size of one of the sets L_i is exponentially large.

Since Goldreich's algorithm is forced to consider all consistent assignments to the bits in each set R_i, it takes no less time than a (myopic) backtracking algorithm that chooses the input bits in the same order, and possibly much more time. For this reason, our new lower bounds are more general.

Open Questions. We believe that there is motivation for further experimental and rigorous analysis of Goldreich's construction.

The main limitation of the present work is the somewhat artificial setup of myopic algorithms, which fails to capture certain natural "global" heuristics used in SAT solvers. Since the algorithm is required to work only with partial information on the object given as an input, negative results for myopic algorithms are similar in spirit (but very different technically) to results on "space bounded cryptography." It would be very interesting to have a lower bounds for drunken algorithms, which are restricted in a way that is more computational than information-theoretic. As a first step, it would be interesting to show that drunken algorithms take exponential time to find planted solutions in a random 3XOR instance.

It would also be interesting to show that no "variation of Gaussian elimination" can invert Goldreich's function when non-linear predicates are used. Unfortunately it is not clear how to even formalize such a statement.

2 Preliminaries

2.1 Goldreich's Function

Goldreich [11] constructs a function $f : \{0,1\}^n \to \{0,1\}^n$ parameterized by a d-ary predicate P and a bipartite graph $G = (V, E)$ connecting n *input nodes* u_i on the left to n *output nodes* v_i on the right. The output nodes all have degree d. To compute the function, on input $x \in \{0,1\}^n$, we label the input nodes with the bits of x, and label each output node by the value of P applied to the labels of its neighbors. The output of the function is the sequence of n labels of the output nodes. For example, if the neighborhood of v_i is $\{u_{j_1}, u_{j_2}, \ldots, u_{j_d}\}$, then

$$(f(x))_i = P(x_{j_1}, x_{j_2}, \ldots, x_{j_d}).$$

We denote by A the $n \times n$ matrix adjacency matrix of G, whose columns correspond to input nodes and whose rows correspond to output nodes:

$$A_{ij} = \begin{cases} 1 & (u_j, v_i) \in E \\ 0 & (u_j, v_i) \notin E \end{cases}.$$

Goldreich suggests using a random predicate P, and a graph G with expansion properties.

2.2 Myopic Backtracking Algorithms

We consider the class of algorithms that might invert Goldreich's function by backtracking.

First, we need a notion of a *partial truth assignment.*

Definition 2.1 (partial assignment). Taken from [2]. *A* partial assignment *is a function* $\rho : [n] \to \{0, 1, *\}$. *Its set of* fixed variables *is* $\mathrm{Vars}(\rho) = \rho^{-1}(\{0,1\})$.

Its size *is defined to be* $|\rho| = |\mathrm{Vars}(\rho)|$. *Given* $f : \{0,1\}^n \to \{0,1\}^n$, *the* restriction of f by ρ, *denoted* $f|_\rho$, *is the function obtained by fixing the variables in* $\mathrm{Vars}(\rho)$ *and allowing the rest to vary.*

Definition 2.2. *A* backtracking algorithm *for solving an equation* $f(x) = b$ *for* x *is defined by two procedures* $\mathbf{N}$ *and* $\mathbf{T}$. $\mathbf{N}$ *takes a partial assignment* ρ *and returns the index of a new variable* $\mathbf{N}(\rho) \in [n]$ *to assign, and* $\mathbf{T}$ *chooses a truth value* $\mathbf{T}(\rho) \in \{0,1\}$ *for* $x_{\mathbf{N}(\rho)}$. *More precisely, the algorithm has the form:*

- *Initialize* ρ *to the empty truth assignment* $(*, *, \ldots, *)$.
- *While not all variables in* ρ *are fixed,*
  - $j \leftarrow \mathbf{N}(\rho)$.
  - *Update* ρ *by assigning* x_j *the truth value* $\mathbf{T}(\rho)$.
  - *If there is row* i *such that* $f(\rho)_i$ *is determined by* ρ *but* $f(\rho)_i \neq b_i$ *then backtrack.*

We study a special class of backtracking algorithms which we call *myopic* backtracking algorithms, after [1].

Definition 2.3. *A* myopic *backtracking algorithm for* $f(x) = b$ *is a backtracking algorithm where procedures* $\mathbf{N}$ *and* $\mathbf{T}$ *are restricted in that they are not allowed to see all the output bits in vector* b. *More precisely, myopic backtracking algorithm of parameter* K *have the following properties:*

- *In the beginning of the algorithm, the algorithm does not have the value of any of* b.
- *At each step of fixing a new variable, the algorithm is allowed to ask the value of* K *output bits corresponding to* K *equations chosen by the algorithm.*
- *When we backtrack from a step we have already taken, we lose the value of the output bits that were revealed to us at that step.*

Thus, in the middle of the algorithm, when the partial assignment is ρ, the algorithm sees the values of $K|\mathrm{Vars}(\rho)|$ output bits, and the outputs of procedures $\mathbf{N}$ and $\mathbf{T}$ are allowed to depend only on these $K|\mathrm{Vars}(\rho)|$ output bits. But notice that procedures $\mathbf{N}$ and $\mathbf{T}$ can use the structure of the function f; they have restricted access to only b.

Notice that in the above definitions, there is no restriction on the computational complexity of procedures $\mathbf{N}$ and $\mathbf{T}$. Therefore without the *myopic* constraint, there is no way to prevent $\mathbf{T}$ from guiding the algorithm immediately towards the correct solution.

The work in [3] gives a lower bound for myopic backtracking algorithms for SAT instances. They translate a system of linear equations $Ax = b$ into a CNF formula. Similarly, for inverting Goldreich's function $f(x) = b$ for a fixed $b \in \{0,1\}^n$, we can define a d-CNF formula $\Phi_b(x)$ which is logically equivalent to the statement $f(x) = b$. The i-th bit of b translates to a set of at most 2^d clauses that enforce the constraint $P(x_{S_i}) = b_i$. Then the problem of inverting f can be reduced to finding a solution to the SAT instance Φ_b. Notice that (myopic) backtracking algorithms for solving Φ_b are similar to (myopic) backtracking algorithms for solving $f(x) = b$.

In [3] the authors consider a notion of myopic backtracking algorithms that is slightly more powerful, called myopic DPLL algorithms after [12,8,1], which might get more information about b using two new rules called Unit Clause Propagation and Pure Literal Elimination. It can be seen that when the equations of $f(x) = b$ are linear, these two rules do not give an advantage to the backtracking algorithm. However, the same reduction from DPLL to ordinary backtracking does not apply to the more general case $f(x) = b$ which we consider. Therefore, in this paper we restrict ourselves to backtracking algorithms.

2.3 Random Predicates

We follow Goldreich's suggestion in choosing $P : \{0,1\}^d \to \{0,1\}$ uniformly at random. Here we define two useful properties that most random predicates have.

Definition 2.4 (robust predicate). $P : \{0,1\}^d \to \{0,1\}$ *is* h-robust *iff every restriction* ρ *such that* $f|_\rho$ *is constant satisfies* $d - |\rho| \leq h$ *[2, Definition 2.2]. For example, the predicate that sums all its inputs modulo* 2 *is* 0*-robust.*

Definition 2.5 (balanced predicate). $P : \{0,1\}^d \to \{0,1\}$ *is* (h, ϵ)-balanced *if, after fixing all variables but* $h + 1$ *of them,*

$$|\Pr[P(x) = 0] - \frac{1}{2}| \leq \epsilon.$$

For example, predicates of the form $P_d(x) = x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ are $(2, 0)$-balanced and $(1, \frac{1}{4})$-balanced. The predicate that sums all its inputs is $(0, 0)$-balanced.

Lemma 2.6. *A random predicate on* d *variables is* $(\Theta(\log \frac{d}{\epsilon}), \epsilon)$*-balanced with probability* $1 - \exp[-\text{poly}(d/\epsilon)]$.

(We omit the proof to save space.)

Corollary 2.7. *A random predicate on* d *variables is* $\Theta(\log d)$*-robust with probability* $1 - \exp[-\text{poly}(d)]$.

2.4 Expansion Properties

Let G be a bipartite graph with n nodes on each side and right-degree d. Equivalently, let A be an $n \times n$ matrix with d ones and $n - d$ zeros in each row.

Definition 2.8 (Boundary and Neighborhood). Taken from [3, Definition 2.1]. *Let* I *be a set of output nodes. Its* boundary*, denoted* ∂I*, is the set of all nodes* $j \in U$ *such that there is exactly one edge from* j *to* I*. The* neighborhood *of* I*,* $\Gamma(I) \subseteq U$ *is the set of all nodes connected to* I*.*

Definition 2.9 (Expansion). Taken from [3, Definition 2.1]. G *(or* A*) is an* (r, d, c)-boundary expander *if for all* $I \subseteq V, (|I| \leq r \Rightarrow |\partial I| \geq c|I|)$*.* G *(or* A*) is an* (r, d, c)-expander *if* $\forall I \subseteq V, (|I| \leq r \Rightarrow |\Gamma(I)| \geq c|I|)$*.*

Lemma 2.10. Analogous to [3, Lemma 2.1]. *Every (r, d, c)-expander is an $(r, d, 2c - d)$-boundary expander.*

Throughout our paper, we will use c to denote neighborhood expansion, and c' to denote boundary expansion, with $c' = 2c - d$.

2.5 Closure Operation

We define the *closure* of a set of input nodes, or columns of A.

Definition 2.11 (closure). Analogous to [3, Definition 3.2]. *For a set of columns $J \subseteq [n]$, define the following relation on $2^{[n]}$:*

$$I \vdash_J I_1 \iff I \cap I_1 = \varnothing \wedge |I_1| \leq \frac{r}{2} \wedge \left| \partial(I_1) \setminus \left[\bigcup_{i \in I} A_i \cup J \right] \right| < c/2|I_1|.$$

Define the closure of J, $\mathrm{Cl}(J)$, *as follows. Let $G_0 = \varnothing$. Having defined G_k, choose a non-empty I_k such that $G_k \vdash_J I_k$, set $G_{k+1} = G_k \cup I_k$, and remove equations I_k from matrix A. (Fix an ordering on $2^{[n]}$ to ensure a deterministic choice of I_k.) When k is large enough that no non-empty I_k can be found, set $\mathrm{Cl}(J) = G_k$.*

We omit the proofs in this section, since similar facts are proved in Section 3 of [3].

Lemma 2.12. Analogous to [3, Lemma 3.5]. *If $|J| < \frac{cr}{4}$, then $|\mathrm{Cl}(J)| < 2c^{-1}|J|$.*

Lemma 2.13. Analogous to [3, Lemma 3.4]. *Assume that A is an arbitrary matrix and J is a set of its columns. Denote by $\hat{A}$ the matrix that results from A by removing the rows in $\mathrm{Cl}(J)$ and the columns in $\bigcup_{i \in \mathrm{Cl}(J)} A_i$. If $\hat{A}$ is non-empty then it is an $(r/2, d, c/2)$-boundary expander.*

Definition 2.14. From [3, Definition 3.4]. *A substitution ρ is said to be* locally consistent *w.r.t. the function $f(x) = b$ if and only if ρ can be extended to an assignment on X which satisfies the equations corresponding to $\mathrm{Cl}(\mathrm{Vars}(\rho))$:*

$$(f(x))_{\mathrm{Cl}(\mathrm{Vars}(\rho))} = b_{\mathrm{Cl}(\mathrm{Vars}(\rho))}$$

Lemma 2.15. Analogous to [3, Lemma 3.6]. *Assume that f employs a (r, d, c)-boundary expander and a h-robust predicate with $c > 2h$. Let $b \in \{0, 1\}^n$ and ρ be a locally consistent partial assignment. Then for any set $I \subseteq [n]$ with $|I| \leq r/2$, ρ can be extended to an assignment x which satisfies the subsystem $(f(x))_I = b_I$.*

3 Myopic Algorithms Use Exponential Time in the Average Case

Theorem 3.1. *Assume A is an $n \times n$ (r, d, c)-boundary expander with left and right degree d and that P is an (h, ϵ)-balanced predicate. Let f be Goldreich's*

function for A and P, and assume f is M-to-one-on-average, in the sense that the number of pairs (x, y) such that $f(x) = f(y)$ is at most $M2^n$. Let $\mathcal{A}$ be any myopic backtracking algorithm. Choose $x \in \{0,1\}^n$ uniformly at random and let $b = f(x)$. Let $F = \lceil 2c-d-h \rceil - 1$, and $s = F/(F+d(d-1))$. Then the probability that $\mathcal{A}$ solves $f(x) = b$ in time $2^{O(r(c-h))}$ is at most

$$M2^{-s\lfloor \frac{cr}{4dK} \rfloor} \left(\frac{1+2\epsilon}{1-2\epsilon} \right)^{r/2}. \tag{1}$$

(We can relax the degree requirement to say that A has right degree d_{right}, and the nodes in every set of $s\lfloor \frac{cr}{4dK} \rfloor$ input nodes have average degree at most d_{left}, where in this case $s = F/(F + d_{\text{left}}(d_{\text{right}} - 1))$.)

Applications of Theorem 3.1

1. Use the predicate $P_d(x) = x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ and a random graph of right-degree d. Then $h = \Theta(1)$, $\epsilon = 0$, $c = d/2 + \Theta(d)$ and $r = \Theta(n/d)$. This gives $F = \Theta(d)$ and $s = \Theta(1/d)$. In Section 4, we show that with high probability $M = 2^{n2^{-\Omega(d)}}$. Furthermore, the average degree of any set of $s\lfloor \frac{cr}{4dK} \rfloor$ input nodes is at most $3d$ with high probability. With these parameters, Theorem 3.1 says that for constant K, the myopic algorithm takes time $2^{\Theta(n)}$ with probability $1 - 2^{-Cn}$, where C depends on d and K.
2. Use a random predicate P and a random graph of right-degree d. Then with high probability, P is (h, ϵ) balanced, with $h = \Theta(\log d)$ and $\epsilon = 1/\text{poly}(d)$. Conditioned on the assumption $M = 2^{nO(d^{-C_0})}$ for $C_0 > 2$, Theorem 3.1 says as before that for constant K, the myopic algorithm takes time $2^{\Theta(n)}$ with probability $1 - 2^{-C_1 n}$, where C_1 depends on d and K.

The rest of this section is devoted to proving Theorem 3.1. First in Section 3.1 we show how it is possible to assume that after a fixed number of steps, the partial truth assignment ρ made by the algorithm will be locally consistent. Then in Section 3.2 we show that the algorithm can only have selected one of many possible locally consistent partial truth assignments – and for any fixed $b \in \{0,1\}^n$, most of these partial assignments will be wrong. Thus, with high probability, the algorithm will have selected globally inconsistent values that lead to an unsatisfiable formula. We then show in Section 3.3 that any resolution proof showing that this new formula is unsatisfiable has size $2^{\Omega(r(c-h))}$, so the algorithm must take that many steps before correcting its mistake.

3.1 Clever Myopic Algorithms

Without loss of generality, we allow our algorithm to be a "clever" myopic algorithm in the sense that, as defined in [3], it satisfies these two properties.

1. Let J be the set of indices of all variables x_j that appear in equations whose output bit b_i has been revealed. Then the algorithm may also read all clauses in $\text{Cl}(J)$ for free and reveal the corresponding new variables.

2. The algorithm never makes stupid guesses: whenever the equations corresponding to the revealed output bits b_i determine the value of a variable x_j, the algorithm will never make the wrong assignment for x_j.

Property 2 can only reduce the number of backtracking steps taken. Property 1 is justified by the following proposition.

Proposition 3.2. Analogous to [3, Proposition 3.1]. *After the first* $\lfloor \frac{cr}{4dK} \rfloor$ *steps a clever myopic algorithm reads at most* $r/2$ *bits of* b.

Proof. At each step, the algorithm makes K clause-queries, asking for dK variable entries. This sums to at most $dK(cr/4dK) = cr/4$ variables, which by Lemma 2.12 will result in at most $r/2$ bits of b. □

Once we have assumed that our algorithm is clever, the following proposition shows that we can further assume the algorithm only makes locally consistent assignments in its first $\lfloor cr/4dK \rfloor$ steps.

Proposition 3.3. Analogous to [3, Proposition 3.2]. *During the first* $\lfloor \frac{cr}{4dK} \rfloor$ *steps the current partial assignment made by a clever myopic algorithm is locally consistent, and so the algorithm will not backtrack.*

Proof. This statement follows by repeated application of Lemma 2.15. Note that clever myopic algorithms are required to select a locally consistent choice of variables if one is available. The proof is accomplished through induction. Initially, the partial assignment is empty, and so is locally consistent. For each step t (with $t < \frac{cr}{4dK}$) with a locally consistent partial assignment ρ_t, a clever myopic algorithm will extend this assignment to ρ_{t+1} which is also locally consistent if possible. By Lemma 2.15 it can always do so as long as $|\mathrm{Cl}(\mathrm{Vars}(\rho_t)) \cup \{x_j\}| \leq r/2$ for the newly chosen x_j. □

3.2 The Probability of a Correct Guess Is Small

Now choose b randomly from the set of attainable outputs of $f(x)$; more formally, let $x \sim \mathrm{Unif}(\{0,1\}^n)$ and $b = f(x)$. Initially, the value of b should be hidden from the algorithm. Whenever the algorithm reveals a clause corresponding to the i^{th} row of A, the i^{th}-bit of b should be revealed to the algorithm. We consider the situation after $\lfloor \frac{cr}{4dK} \rfloor$ steps of the algorithm. By Proposition 3.3, the current partial assignment must be locally consistent, and no backtracking will have occurred. Thus, at this point in time we observe the algorithm in the $\lfloor \frac{cr}{4dK} \rfloor$-th vertex v in the leftmost branch of its backtracking tree. By Proposition 3.2, the algorithm has revealed at most $r/2$ bits of b.

Define random variable $I_v \subseteq [n]$ to be the set of output bits revealed after $\lfloor \frac{cr}{4dK} \rfloor$ steps. Similarly define random variable ρ_v to be the the partial truth assignment given by the algorithm at that time. Define $R_v = \mathrm{Vars}(\rho_v)$. Hence $|R_v| = \lfloor \frac{cr}{4dK} \rfloor$.

Definition 3.4. *Let $I \subseteq [n]$, $R \subseteq [n]$, $\iota \in \{0,1\}^I$ and $\rho \in \{0,1\}^R$. We say (I, R, ι, ρ) is a* consistent state *if*

$$\Pr[I_v = I \wedge R_v = R \wedge b_{I_v} = \iota \wedge \rho_v = \rho] > 0.$$

Put another way, (I, R, ι, ρ) is a consistent state iff there exists some $x \in \{0,1\}^n$ such that after $\lfloor \frac{cr}{4dK} \rfloor$ steps, I is exactly the set of revealed bits, R is exactly the set of assigned variables, ρ is the values assigned those variables, and $b_I = \iota$.

Our first attempt at a proof is to show that there are many choices for ρ_v that are locally consistent. Intuitively, if the number of those possible choices for ρ is large compared to M, we expect the algorithm to make the wrong choice with high probability. This line of reasoning would need a result of the following form.

Lemma 3.5. Analogous to [3, Lemma 3.10]. *Assume that an $n \times n$ matrix A is an (r, d, c)-boundary expander and P is an (h, ϵ)-balanced predicate, and let f be Goldreich's function corresponding to A and P. Let $\hat{X} \subseteq [n]$ be a set of input variables with $|\hat{X}| < r$. Choose x uniformly at random from $\{0,1\}^n$ and let $b = f(x)$. Let $Y \subseteq [n]$ be a set of output variables, $|Y| < r$. For $i \in Y$ let ℓ_i be the constraint $(f(x))_i = b_i$, and let $\mathcal{L} = \{\ell_i : i \in Y\}$. Denote by L the set of partial assignments who assign values to the variables in $\hat{X}$ and can be extended to complete truth assignments that satisfy $\mathcal{L}$.*

Let s be defined as in Theorem 3.1. Then $\log |L| \geq s|\hat{X}|$.

In fact, our proof requires the following stronger lemma instead of Lemma 3.5, which states that conditioned on what the algorithm has seen, the minimum entropy of x_{R_v} is high.

Lemma 3.6. *Let x, $\hat{X}$, $\mathcal{L}$, and s be as in Lemma 3.5. Then for any $\hat{x} \in \{0,1\}^{|\hat{X}|}$,*

$$\Pr[x_{|\hat{X}|} = \hat{x} | \mathcal{L}] \leq 2^{-s|\hat{X}|} \left(\frac{1+2\epsilon}{1-2\epsilon} \right)^{|\mathcal{L}|}.$$

We postpone the proof of Lemma 3.6 (and do not prove Lemma 3.5, since we do not use it). We are now prepared to complete the proof of the main theorem.

Proof (Theorem 3.1). Our goal is to bound the probability of the following event:

$$E = \{\rho_v \in (f^{-1}(b))_{R_v}\}.$$

We first condition on the state of the algorithm, considering all consistent states in the sense of Definition 3.4:

$$\begin{aligned}
&\Pr[E] \\
=&\sum_{(I,R,\iota,\rho) \text{ consistent}} \Pr[E | I_v = I \wedge R_v = R \wedge b_{I_v} = \iota \wedge \rho_v = \rho] \cdot \\
&\qquad \Pr[I_v = I \wedge R_v = R \wedge b_{I_v} = \iota \wedge \rho_v = \rho] \\
=&\mathbf{E}[\Pr[E | I_v, R_v, b_{I_v}, \rho_v]].
\end{aligned}$$

Since the algorithm is deterministic and only observes the bits in b_{I_v}, the event $[I_v = I \wedge R_v = R \wedge \rho_v = \rho]$ is implied by the event $[b_{I_v} = \iota]$ – put another way, if bits of b outside the set of observed bits I_v are changed, the behavior of the algorithm will not be affected, so the values of I_v, R_v and ρ_v will not change. This gives us:

$$\begin{aligned}
&\Pr[E] \\
=&\mathbf{E}[\Pr[E|b_{I_v}]] \\
=&\mathbf{E}[\Pr[\rho_v \in (f^{-1}(b))_{R_v}|b_{I_v}]] \\
\leq&\mathbf{E}\left[\max_{\rho_v^* \in \{0,1\}^{R_v}} |f^{-1}(b)| \Pr[\rho_v^* = x_v|b_{I_v}]\right] \\
\leq&\mathbf{E}[|f^{-1}(b)|] \max_{I_v, b_{I_v}, \rho_v^*} \Pr[\rho_v^* = x_v|b_{I_v}] \\
\leq&M2^{-s}\left(\frac{1+2\epsilon}{1-2\epsilon}\right)^{r/2}.
\end{aligned}$$

In the last step, we applied Lemma 3.6, replacing $\hat{X}$ by R_v, $\mathcal{L}$ by I_v and $\hat{x}$ by ρ. Note that $|I_v| < r/2$, so $|\mathcal{L}| < r/2$ in the hypothesis of the lemma.

We have shown that it will be likely that ρ_v, though locally consistent, can not be extended to satisfy b, and an unsatisfiable instance will occur. In Section 3.3, we explore the running time of backtracking algorithms on unsatisfiable cases to show if E does not occur, the algorithm will take time $2^{\Omega(r(c-h))}$.

Proof of Lemma 3.6

Lemma 3.7. *Fix any $g \subseteq \hat{X}$. If each output in $\mathcal{L}$ has at most $F = \lceil 2c-d-h \rceil - 1$ of its d inputs in g, then $\forall I \subseteq \mathcal{L},\ |\partial I \setminus g| > h|I|$.*

Proof. Consider any subset $I \subseteq \mathcal{L}$. By Lemma 2.10, $|\partial I| \geq (2c-d)|I|$, so $|\partial I \setminus g| \geq (2c-d-F)|I| > h|I|$.

Lemma 3.8. *Fix any $g \subseteq \hat{X}$. If $\forall I \subseteq \mathcal{L},\ |\partial I \setminus g| > h|I|$, then any partial assignment $\rho : g \to \{0,1\}$ can be extended to a complete assignment that satisfies $\mathcal{L}$.*

Proof. We make our proof by contradiction; assume ρ cannot be extended to satisfy the equations in $\mathcal{L}$. Let k be a minimal set of unsatisfiable equations. We assume our predicate is h-robust. $\forall I \subseteq \mathcal{L},\ |\partial I \setminus g| > h|I|$ implies that some equation in I must have at least $h+1$ boundary elements outside of g. However, no equation in k should have more than h boundary variables; otherwise, those $h+1$ boundary variables could be set to a value that satisfies that equation, and it should not be in the minimal set k.

Lemma 3.9. *Let s and F be as in Theorem 3.1. We can find $g \subseteq \hat{X}$ with $|g| \geq s|\hat{X}|$, such that no output has more than F inputs in g.*

Proof. Construct g using the following algorithm:

- $g \leftarrow \varnothing$.
- $n_i \leftarrow \begin{cases} F & i \in \hat{X} \\ 0 & i \notin \hat{X} \end{cases}$.
- **while** $\exists i,\ n_i > 0$,
  - *Invariant:* If an output has $F - a$ inputs in g, then for every input i connected to it, $n_i \leq a$.
  - $g \leftarrow g \cup \{i\}$.
  - $n_i \leftarrow n_i - F$.
  - $\forall j$, if $\mathrm{dist}(i, j) = 2$, then $n_j \leftarrow n_j - 1$.

We start with $F|\hat{X}|$ counters, and remove on average $F + d_{\text{left}}(d_{\text{right}} - 1)$ counters at every step. (Recall that output nodes have degree d_{right}, and as long as $|g| \leq s\lfloor \frac{cr}{4dK} \rfloor$, the average degree of g is at most d_{left}.) In the end,

$$|g| \geq \frac{F|\hat{X}|}{F + d_{\text{left}}(d_{\text{right}} - 1)}.$$

□

Proof (Lemma 3.6). Choose $g \subseteq \hat{X}$ with $|g| \geq s|\hat{X}|$ as in Lemma 3.9. By Lemma 3.7, every subset of $\mathcal{L}$ has a row with at least $h + 1$ boundary variables that are not in g. Therefore we can order the rows of $\mathcal{L}$ as $\ell_1, \ldots, \ell_{|\mathcal{L}|}$ such that setting $L_i = \{\ell_1, \ldots, \ell_i\}$, for all i we have $|\Gamma(\ell_i) \setminus (\Gamma(L_{i-1}) \cup g)| \geq h + 1$. Then

$$\begin{aligned} \frac{\Pr[x_{|g} = g_1 | L_{i+1}]}{\Pr[x_{|g} = g_2 | L_{i+1}]} &= \frac{\Pr[L_{i+1} | x_{|g} = g_1] \Pr[x_{|g} = g_2]}{\Pr[L_{i+1} | x_{|g} = g_2] \Pr[x_{|g} = g_1]} \\ &= \frac{\Pr[L_i | x_{|g} = g_1] \Pr[\ell_{i+1} | L_i, x_{|g} = g_1]}{\Pr[L_i | x_{|g} = g_2] \Pr[\ell_{i+1} | L_i, x_{|g} = g_2]} \end{aligned}$$

(Use the fact that the predicate is (h, ϵ)-balanced.)

$$\leq \left(\frac{\frac{1}{2} + \epsilon}{\frac{1}{2} - \epsilon} \right) \frac{\Pr[L_i | x_{|g} = g_1]}{\Pr[L_i | x_{|g} = g_2]}.$$

Observe that

$$\frac{\Pr[x_{|g} = g_1 | L_0]}{\Pr[x_{|g} = g_2 | L_0]} = 1.$$

It follows that

$$\frac{\Pr[x_{|g} = g_1 | \mathcal{L}]}{\Pr[x_{|g} = g_2 | \mathcal{L}]} \leq \left(\frac{1 + 2\epsilon}{1 - 2\epsilon} \right)^{|\mathcal{L}|}.$$

Take $g_1 \in \{0,1\}^g$ that minimizes $\Pr[x_{|g} = g_1 | \mathcal{L}]$. There are $2^{|g|}$ possible values for g_1, so $\Pr[x_{|g} = g_1 | \mathcal{L}] \leq 2^{-|g|} \leq 2^{-s|\hat{X}|}$. For any $\hat{x} \in \{0,1\}^{\hat{X}}$,

$$\begin{aligned} \Pr[x_{|\hat{X}} = \hat{x} | \mathcal{L}] &\leq \Pr[x_{|g} = \hat{x}_{|g} | \mathcal{L}] \\ &= \Pr[x_{|g} = g_1 | \mathcal{L}] \frac{\Pr[x_{|g} = \hat{x}_{|g} | \mathcal{L}]}{\Pr[x_{|g} = g_1 | \mathcal{L}]} \leq 2^{-s|\hat{X}|} \left(\frac{\frac{1}{2} + \epsilon}{\frac{1}{2} - \epsilon} \right)^{|\mathcal{L}|}. \end{aligned}$$

□

3.3 Backtracking Algorithms Use Exponential Running Time on Unsatisfiable Formulas

In Section 3.2, we showed that with high probability a myopic backtracking algorithm will choose a partial assignment to x that cannot be extended to satisfy $f(x) = b$. We now prove that once this happens, the algorithm must run for exponential time:

Theorem 3.10. Analogous to [3, Lemma 3.9]. *Let f be Goldreich's function for predicate P and graph G, where G is an $n \times n$ (r, d, c)-boundary expander with right-degree d and P is h-robust with $h < c/2$. Fix $b \in \{0,1\}^n$. Let ρ be a locally consistent partial assignment such that $|Vars(\rho)| \leq cr/4$. If ρ cannot be extended to any input x satisfying $f(x) = b$, then every backtracking algorithm that makes the partial assignment ρ will run for time $2^{\Omega(r(c-h))}$.*

We will make use of the following lemma from [6, Corollary 3.4]. The *width* of a resolution proof is the greatest width of any clause that occurs in it, and the width of a clause is the number of variables in it.

Lemma 3.11. *The size of any tree-like resolution refutation of a formula Ψ is at least 2^{w-w_Ψ}, where w is the minimal width of a resolution refutation of Ψ, and w_Ψ is the maximal width of a clause in Ψ.*

Our setup and proof strategy are similar to those found in [2, Section 3] and [3]. [2] measures robustness in terms of ℓ, where $\ell = d - h$.

Proof (Theorem 3.10). Let $I = \mathrm{Cl}(\rho)$ and $J = \Gamma(I)$. By Lemma 2.12 $|I| \leq r/2$. By Lemma 2.15, ρ can be extended to another partial assignment ρ' on variables x_J, such that ρ' satisfies every equation in $(f(x))_I = b_I$. The restricted formula $(f(x) = b)|_{\rho'}$ still encodes an unsatisfiable system $f'(x) = b'$. The underlying graph G' of f' is produced from G by removing every output node in I and every input node in J. By Lemma 2.13, G' is an $(r/2, d, c/2)$-boundary expander.

We can express the equation $f'(x) = b'$ using a CNF formula Φ, by representing each equation $(f'(x))_i = b'_i$ by at most 2^d clauses. The computation of a backtracking algorithm as it discovers that $f'(x) = b'$ is unsatisfiable can be translated to a tree-like resolution refutation of the formula Φ, such that the size of the refutation is the working time of the algorithm. Thus it is sufficient to show that every tree-like resolution refutation of Φ is large.

We say a set of equations $(f'(x))_I = b'_I$ *semantically implies* a clause C iff every truth assignment satisfying $(f(x))_I = b_I$ also satisfies C. Following [2, Section 3], we define the *measure* of C to be

$$\mu(C) = \min_{I:(f'(x))_I = b'_I \models C} |I|.$$

We omit the proofs of the following facts; similar facts are proved in [2].

1. For any $D \in \Phi$, $\mu(D) = 1$.
2. $\mu(\varnothing) > r$.

3. μ is subadditive: if C_2 is the resolution of C_0 and C_1, then $\mu(C_2) \leq \mu(C_0) + \mu(C_1)$.
4. For any clause C, if $\frac{r}{2} \leq \mu(C) \leq r$, then C has width at least $\frac{(c/2-h)r}{4}$.

1, 2 and 3 together imply that any resulation proof will result in a clause C whose width is between $\frac{r}{2}$ and r. By 4, C has width at least $\frac{(c/2-h)r}{4}$, so by Lemma 3.11, the resolution proof has size $2^{\Omega(r(c-h))}$. □

4 The Size of Pre-images of Goldreich's Function

In this section we prove that Goldreich's function has pre-images sufficiently small for Theorem 3.1 to work.

Theorem 4.1. *For every degree d, let $P_d(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$. Choose a random graph for Goldreich's function by connecting each output to d inputs chosen uniformly at random (with replacement). Then*

$$\mathbf{E}[\#(x, y) : f(x) = f(y)] = 2^{(1+2^{-\Omega(d)})n},$$

where the expectation is over the choice of graph.

Proof. For $x, y \in \{0,1\}^n$ and $i, j \in \{0,1\}$, let $n_{ij}(x, y)$ be the number of indices k where $x_k = i$ and $y_k = j$. We have $n_{00}(x,y)+n_{01}(x,y)+n_{10}(x,y)+n_{11}(x,y) = n$.

Since the input indices to the predicate are selected uniformly at random, the probability that a single output bit will be equal for inputs x and y is only a function of $\alpha_{ij} \stackrel{\text{def}}{=} n_{ij}(x,y)/n$. We call this function the *probability of equality*, $\mathrm{PE}(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11})$.

Then

$$\begin{aligned}
&\mathbf{E}[\#(x, y) : f(x) = f(y)] \\
=&\sum_{x,y \in \{0,1\}^n} \Pr[f(x) = f(y)] \\
=&\sum_{n_{00}+n_{01}+n_{10}+n_{11}=n} \binom{n}{n_{00}, n_{01}, n_{10}, n_{11}} \Pr[f(x) = f(y) | n_{ij}] \\
\leq& n^4 \max_{\alpha_{00}+\alpha_{01}+\alpha_{10}+\alpha_{11}=1} \binom{n}{n\alpha_{00}, n\alpha_{01}, n\alpha_{10}, n\alpha_{11}} \mathrm{PE}(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11})^n \\
=&\max_{\alpha_{00}+\alpha_{01}+\alpha_{10}+\alpha_{11}=1} (2^{H(\alpha_{00},\alpha_{01},\alpha_{10},\alpha_{11})} \mathrm{PE}(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}))^{n(1+o(1))}
\end{aligned}$$

where $H(\alpha_{ij})$ is the base-2 entropy of the distribution defined by α_{ij}. Thus, it suffices to show that there is a constant $\epsilon > 0$ such that for sufficiently large d,

$$\forall \alpha_{ij} \; H(\alpha_{ij}) + \log_2 \mathrm{PE}(\alpha_{ij}) \leq 1 + 2^{-\epsilon d}.$$

It can be shown that for the predicate P_d which we have defined,

$$\mathrm{PE}(\alpha_{ij}) \leq \frac{1 + (\alpha_{00} + \alpha_{11} - \alpha_{01} - \alpha_{10})^{d-2}}{2}.$$

Now, let $p = \alpha_{00} + \alpha_{11}$ and $q = \alpha_{01} + \alpha_{10} = 1 - p$. Forcing $\alpha_{00} = \alpha_{11}$ and $\alpha_{01} = \alpha_{10}$ can only increase $H(\alpha_{ij})$, so without loss of generality, we assume $\alpha_{00} = \alpha_{11} = p/2$ and $\alpha_{01} = \alpha_{10} = q/2$, and prove

$$\forall p \in [0,1]: \; H(p/2, p/2, q/2, q/2) + \log_2\left(\frac{1 + (p-q)^{d-2}}{2}\right) \leq 1 + 2^{-\epsilon d},$$

or equivalently,

$$\forall q \in [0, \tfrac{1}{2}]: \; H(q) + \log_2(1 + (1-2q)^{d-2}) \leq 1 + 2^{-\epsilon d}.$$

We consider four cases for the value of q. (We will choose positive constants $\epsilon, \epsilon_1, \epsilon_2, \epsilon_3$ suitably as we go along.)

Case 1: $q > \epsilon_1$

$$H(q) + \log_2(1 + (1-2q)^{d-2}) \leq 1 + (1-2\epsilon_1)^{d-2} \log_2 e \leq 1 + 2^{-\epsilon d},$$

for $\epsilon < -\log_2(1 - 2\epsilon_1)$ and sufficiently large d.

For the remaining three cases, q is small. Using the Taylor expansion of $\log_2$ around 2, we get

$$\log_2(1 + (1-2q)^{d-2}) \leq 1 + \frac{(1-2q)^{d-2} - 1}{2 \ln 2} \leq 1 + \frac{e^{-2qd} - 1}{2 \ln 2}.$$

Case 2: $\epsilon_1 \geq q > \epsilon_2/d$

$$H(q) + \log_2(1 + (1-2q)^{d-2}) \leq H(\epsilon_1) + 1 + \frac{e^{-2\epsilon_2} - 1}{2 \ln 2} \leq 1$$

if we choose ϵ_1 small enough that $H(\epsilon_1) < \frac{1 - e^{-2\epsilon_2}}{2 \ln 2}$.

For the remaining two cases we fix ϵ_2, say $\epsilon_2 = \frac{1}{2}$. Now, $qd \leq \frac{1}{2}$, and we have the approximation

$$H(q) + 1 + \frac{e^{-2qd} - 1}{2 \ln 2} \leq (q \log_2(1/q) + 2q) + 1 - \frac{qd}{2 \ln 2} = q(\log_2(1/q) - \Theta(d)) + 1.$$

Case 3: $\epsilon_2/d \geq q > 2^{-\epsilon_3 d}$

For $\epsilon_3 < \frac{1}{\ln 2}$ and sufficiently large d: $\log_2(1/q) - \Theta(d) < 0$.

Case 4: $2^{-\epsilon_3 d} \geq q$

For $\epsilon < \epsilon_3$ and sufficiently large d: $q \log(1/q) \leq \epsilon_3 d 2^{-\epsilon_3 d} \leq 2^{-\epsilon d}$.

This completes our proof. □

References

1. Achlioptas, D., Sorkin, G.B.: Optimal myopic algorithms for random 3-SAT. In: FOCS, pp. 590–600 (2000)
2. Alekhnovich, M., Ben-Sasson, E., Razborov, A.A., Wigderson, A.: Pseudorandom generators in propositional proof complexity. SIAM Journal on Computing 34(1), 67–88 (2004)
3. Alekhnovich, M., Hirsch, E.A., Itsykson, D.: Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. J. Autom. Reasoning 35, 51–72 (2005)
4. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC0. SIAM J. on Computing 36(4), 845–888 (2006)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in NC^0 . In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 260–271. Springer, Heidelberg (2006)
6. Ben-Sasson, Wigderson: Short proofs are narrow–resolution made simple. J. ACM: Journal of the ACM 48 (2001)
7. Cryan, M., Miltersen, P.B.: On pseudorandom generators in NC. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, p. 272. Springer, Heidelberg (2001)
8. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM 5, 394–397 (1962)
9. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 61–75. Springer, Heidelberg (2005)
10. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
11. Goldreich, O.: Candidate one-way functions based on expander graphs. Electronic Colloquium on Computational Complexity (ECCC) 7(90) (2000)
12. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM 7, 201–215 (1960)
13. Mossel, E., Shpilka, A., Trevisan, L.: On ϵ-biased generators in NC^0. Random Structures and Algorithms 29(1), 56–81 (2006)

A MiniSat Experiment

Inverting Goldreich's function can be seen as the task of solving a constraint satisfaction problem with a planted solution. This suggests the use of a general-purpose SAT solver to solve the constraint satisfaction problem. We performed an experiment using MiniSat version 2.0 beta [10,9], which is one of the best publicly available SAT solvers. We always use the degree-five predicate $P_5(x) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$. For each trial, we choose a new random graph of right-degree 5. MiniSat requires a boolean formula in conjuctive normal form as input, so we represent each constraint $P(x_{j_1}, x_{j_2}, x_{j_3}, x_{j_4}, x_{j_5}) = v_i$ by 16 clauses: one for each truth assignment to $x_{j_1}, \cdots, x_{j_5}$ that would violate the constraint.

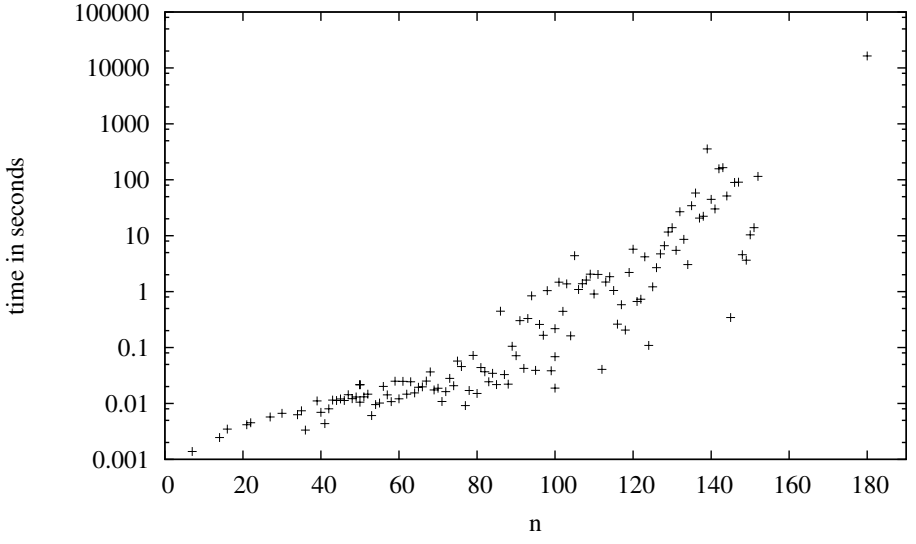

Fig. 1. Number of seconds taken by MiniSat to invert Goldreich's function for different values of n. We use the degree-five predicate $P_5(x) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$ and a random bipartite graph of right-degree five.

We ran MiniSat on a Lenovo T61 laptop with 2GB of RAM and a 2.00GHz Intel T7300 Core Duo CPU. Fig. 1 plots the number of seconds taken to find a solution versus the input size n. The graph is plotted on a logarithmic scale. The time appears to grow exponentially in n.