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

Rachel Miller, University of Virginia CLAS '09

Distinguished Majors Thesis for Computer Science
University of Virginia 2009

Thesis Adviser Professor abhi shelat, CS Department
Thesis Reader Professor Christian Gromoll, Math Department

This thesis extends collaborative work done in the SUPERB 2008 Program at UC Berkeley with Professor Luca Trevisan and graduate students James Cook and Omid Etesami.

Abstract

One-way functions are easy to compute but hard to invert; their existence is the foundational assumption for modern cryptography. Oded Goldreich's 2000 paper "Candidate One-Way Functions Based on Expander Graphs" [6] proposes a candidate one-way function construction based on any small fixed predicate over d variables and a bipartite expander graph of right-degree d. The function is calculated by taking an n-bit input as the values of the vertices on the left, and then calculating each of the n output bits on the right by applying the predicate to its neighbors.

Inverting Goldreich's one-way function can be expressed as constraints on input bits by the value of each output bit, and so can easily be reduced to a SAT instance. Most modern SAT solvers are based on backtracking algorithms. 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. Cook, Etesami, Miller and Trevisan extended their work to show the function is also secure against myopic backtracking algorithms of higher degree linear functions and against predicates of the form $P_d(x_1, \ldots, x_d) := x_1 \oplus x_2 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ on random graphs.

Alekhnovich et al. also show how to construct satisfiable SAT instances secure against "drunken" backtracking algorithms from unsatisfiable SAT instances. The contribution of this work is to show Goldreich's function is secure against "drunken" backtracking algorithms for linear predicates and predicates of the form $P_d(x_1, \ldots, x_d) := x_1 \oplus x_2 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ on random graphs.

Acknowledgments

This work is closely related to Cook, Etesami, Miller, and Trevisan's "Goldreichs One-Way Function Candidate and Myopic Backtracking Algorithms" [5], to which I made significant contributions. I have included many of the proofs from that paper to help give context for this work.

All sections relating to drunken algorithms are original to this work.

1 Introduction

Goldreich [6] proposed a candidate one-way function construction based on expander graphs. The construction requires a bipartite graph with n nodes per side, and any fixed predicate of degree d, with d constant or d growing slowly as $O(\log n)$. The function is computed by taking n input bits as the left vertices of the graph, and calculating each output bit on the right as the predicate applied to its neighbors on the left. The value of each output bit relies on a fixed number of input bits determined by the graph, and so this function is very fast to compute in parallel. Goldreich suggests that a random predicate and a graph with expansion properties should be used.

Throughout the paper, we will consider random predicates, and graphs with expansion properties, or random graphs which have expansion properties with high probability. We will explore the interesting cases of linear predicates, and predicates of the form $P(x_1, \cdots, x_d) := x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$. This last predicate is inspired by the work of Mossel, Shpilka and Trevisan [7] who construct a small-bias generator using a fixed predicate of the form $P(x_1, \cdots, x_5) := x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$.

1.1 Backtracking Algorithms

The task of inverting Goldreich's function naturally translates into a SAT-instance, where SAT is the boolean satisfiability problem in conjugate normal form. Each output bit can be viewed as a constraint on its corresponding input bits. This constraint can be translated to SAT clauses with size equal to the right-degree of the expander graph. This SAT instance will always be satisfiable since inversion always has a solution.

Because of this natural connection to SAT, an attack of this function as a SAT instance seems plausible. In this paper, we explore "DPLL-style" algorithms, which are the basis for most general SAT solvers and use backtracking. We restrict our study 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 fixing the variable's value by an "assigner" procedure. The recursion ends when a satisfying assignment is found, or if a partial assignment contradicts a constraint given by an output bit.

When any backtracking algorithm of this form runs on an unsatisfiable instance, the transcript of values tried by the algorithm gives a "tree-like resolution proof" of unsatisfiability. For satisfiable instances, subexponential lower bounds require some sort of restriction of the assigner or the scheduler. Otherwise, an attacker unrestricted in complexity could simply select a satisfying assignment in a linear number of steps. Even for attackers with limitations, few lower bounds for solving satisfiable SAT instances exist.

1.2 Previous Work

Analysis of Goldreich's function is motivated by the apparent hardness of inversion. Cook, Etesami, Miller and Trevisan [5] performed an experiment by running MiniSat on SAT instances generated by Goldreich's function based on predicate $P(x_1, \cdots, x_5) := x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$ and a random graph of right-degree 5. MiniSat is one of the best publicly available SAT solvers, but had exponential increases in running time as a function of input length n. The attack with MiniSat appeared to infeasible for moderate input lengths of a few hundred bits.

Alekhnovich, Hirsch and Itsykson [2] consider inversion of Goldreich's function with "myopic" backtracking algorithms. "Myopic" stipulates that the attacker starts without the output it is inverting, but has a constant number of output bits revealed to it for each input it guesses. For myopic algorithms, the scheduler and assigner are restricted to working with the information available to the attacker. Their work proves Goldreich's function on degree-3 linear predicates is secure against myopic backtracking algorithms. They show that with high probability, after assigning a certain number of variables the attacker will have created an unsatisfiable instance with no sub-exponential size tree-like resolution proof of unsatisfiability. It then takes the algorithm an exponential amount of time to recover from the bad partial assignment. However, linear predicates are solvable in subexponential time by Gaussian elimination, so alternative predicates are needed for secure versions of Goldreich's function against broader classes of attackers.

Cook, Etesami, Miller and Trevisan [5] show that Goldreich's function is secure against myopic backtracking algorithms for linear predicates of higher degree, and also for predicates of the form $P(x_1, \cdots, x_d) := x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$. Under a plausible assumption about the many-to-oneness of Goldreich's function, they also show random predicates are secure against myopic backtracking algorithms with high probability.

Alekhnovich et al. [2] also consider the restriction of "drunken" backtracking algorithms. Drunken algorithms have unbounded schedulers, but the assigners' choice of whether to assign first zero or one to the next chosen variable is made randomly with equal probability. Alekhnovich et al. show results for drunken algorithms on carefully designed instances based on hard unsatisfiable SAT instances.

1.3 New Contributions

This work is concerned with proving lower bounds for solving Goldreich's function with drunken backtracking algorithms. We show that linear functions and predicates of the form $P(x_1, \cdots, x_d) := x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \wedge x_d)$ are secure against drunken algorithms. For the same assumption that Cook, Etesami, Miller and Trevisan [5] use about the many-to-oneness of Goldreich's function, we prove random predicates are secure against drunken backtracking algorithms with high probability.

2 Preliminaries

2.1 Goldreich's Function

Goldreich [6] 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 $n \times n$ 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 Drunken Backtracking Algorithms

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

Definition 2.1 (partial assignment) Taken from [3]. *Let* $[n]$ *denote* $\{1, \ldots, n\}$. *A* partial assignment *is a function* $\rho : [n] \to \{0, 1, *\}$ *that maps the input bits of Goldreich's function to values for those bits. 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, a scheduler* $\mathbf{N}$ *and an assigner* $\mathbf{T}$. $\mathbf{N}$ *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 by substituting the opposite truth value to the most recently selected variable that has not had both truth values tried.*

Since there are only a finite number of partial assignments and each partial assignment will only be attained at most once, this procedure terminates.

Drunken backtracking algorithms, after [1], are a special class of backtracking algorithms that have scheduler **N** unrestricted, but then assigner **T** must flip a coin to select 0 or 1 with equal probability. That is, the algorithm has unlimited computation to select which input bit to work on, but then must guess that input bit randomly. We refer to inputs selected randomly by the assigner as "drunkenly selected". Without the *drunken* constraint, there is no way to prevent **T** from immediately selecting the correct value for that bit. Unlike myopic algorithms, drunken algorithms have the scheduler **N** completely unrestricted in both time and in access to the output b.

The work in [2] gives a lower bound for drunken backtracking algorithms on SAT instances constructed from hard unsatisfiable SAT instances. We show lower bounds on drunken algorithms solving Goldreich's function, which can also be generalized to a SAT instance.

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 2.3 (robust predicate) *$P : \{0,1\}^d \to \{0,1\}$ is* h-robust *iff every restriction ρ such that $f|_\rho$ is constant satisfies $d - |\rho| \leq h$ [3, Definition 2.2].*

In other words, the value of the function with any $h+1$ free input variables is *not* fixed. For example, the predicate that sums all its inputs modulo 2 is 0-robust. The predicate that multiplies all its inputs is $d-1$-robust, because any bit that is a 0 fixes its output.

Definition 2.4 (balanced predicate) *$P : \{0,1\}^d \to \{0,1\}$ is* (h, ϵ)-balanced *if, after fixing all variables but $h+1$ of them,*

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

where probability is over the values of the remaining variables.

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 2.5 Cook et al. [5, Lemma 2.6]. *A random predicate on d variables is $(\Theta(\log \frac{d}{\epsilon}), \epsilon)$-balanced with probability $1 - \exp[-\text{poly}(d/\epsilon)]$.*

Corollary 2.6 Cook et al. [5, 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 A be an $m \times n$ matrix with d ones and $n - d$ zeros in each row.

Definition 2.7 (Boundary Element) Taken from [2, Definition 2.1].
For a set of rows I of our $m \times n$ matrix A, we define its boundary ∂I *as the set of all $j \in [n]$ (called* boundary elements*) such that there exists exactly one row $i \in I$ that has a value of 1 in column j.*

In other words, a variable j is a boundary element if it is referenced by exactly one output in a group of outputs.

Definition 2.8 (Expansion) Taken from [2, Definition 2.1].
A is an (r, d, c)-boundary expander *if*

1. $|A_i| \leq d$ *for all* $i \in [m]$, *and*
2. $\forall I \subseteq [m], (|I| \leq r \Rightarrow |\partial I| \geq c|I|)$.

Matrix A is an (r, d, c)-expander *if condition 2 is replaced by*

$2'$ $\forall I \subseteq [m], (|I| \leq r \Rightarrow |\bigcup_{i \in I} A_i| \geq c|I|)$.

Lemma 2.9 Analogous to [2, Lemma 2.1].
Any (r, d, c)-expander is an $(r, d, 2c - d)$-boundary expander.

Assume that A is an (r, d, c)-expander. Consider a set of its rows I with $|I| \leq r$. Since A is an expander $|\bigcup_{i \in I} A_i| \geq c|I|$. On the other hand we may estimate separately the number of boundary and non-boundary variables which will give $|\bigcup_{i \in I} A_i| \leq E + (d|I| - E)/2$, where E is the number of boundary variables. This implies $E + (d|I| - E)/2 \geq c|I|$ and $E \geq (2c - d)|I|$.
Throughout the rest of our paper, we make an assumption about the relationship between the expansion of the matrix A and the balance of the predicate P used in Goldreich's function.

Assumption 2.10 *A is a (r, d, c)-boundary expander and P is an (h, ϵ)-balanced predicate, with $c - h \geq 1$ and $c > 2h$.*

Recall that by Lemma 2.5, a random predicate on d variables is $(\Theta(\log \frac{d}{\epsilon}), \epsilon)$-balanced with high probability. Also, expander graphs exist with $c = \Omega(d)$. Thus, as d increases, c can be made much larger than $2h$. This justifies Assumption 2.10.

3 Previous Results

Now that we have developed the proper language to describe previous results, we will include them here for comparison purposes.

Goldreich's Analysis

Goldreich [6] 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, the lower bounds of Cook, Etesami, Miller and Trevisan [5] are more general.

Myopic Backtracking Algorithms

Definition 3.1 *A* myopic *backtracking algorithm for $f(x) = b$ is a backtracking algorithm where the "scheduler" and the "assigner" procedures 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 the scheduler and assigner procedures are allowed to depend only on these $K|\mathrm{Vars}(\rho)|$ output bits. But notice that both procedures can use the structure of the function f and of the bipartite graph; they have restricted access to only b.

Theorem 3.2 Cook et al. [5, 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*

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

4 Drunken Algorithms use Exponential Time in the Average Case

Theorem 4.1 *Assume* A *is an* $n \times n$ (r, d, c)*-boundary expander with right degree* d *and that* P *is an* h*-robust 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 drunken backtracking algorithm. Choose* $x \in \{0,1\}^n$ *uniformly at random and let* $b = f(x)$.

Then the probability that $\mathcal{A}$ *solves* $f(x) = b$ *in time* $2^{O(r(c-h))}$ *is at most*

$$M 2^{-\frac{cr}{4}}.$$

Applications of Theorem 4.1:

1. Let $P = x_1 \oplus \cdots \oplus x_d$ be a linear predicate. Then $\mathcal{A}$ solves $f(x) = b$ in time $2^{\Omega(rc)}$ except with probability less than $2^{nullity(A) - \frac{rc}{4}}$.

2. 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 $h = \Theta(1)$, $c = d/2 + \Theta(d)$ and $r = \Theta(n/d)$. With high probability, $c > 2h$. Also, Cook et al. [5] show that with high probability $M = 2^{n2^{-\Omega(d)}}$.

   Assuming $c > 2h$ and $M = 2^{n2^{-\Omega(d)}}$, with these parameters, for large enough d Theorem 4.1 shows a drunken algorithm takes time $2^{\Theta(n)}$ with probability $1 - 2^{-Cn}$, where C depends on d.

3. Use a random predicate P and a random graph of right-degree d. Then with high probability, P is h robust, with $h = \Theta(\log d)$ and with $c > 2h$. In this case, Theorem 4.1 says the drunken algorithm takes time $2^{\Theta(n)}$ with probability $1 - M2^{-\frac{rc}{4}}$.

The rest of this section is devoted to proving Theorem 4.1.

To aid this proof, we utilize a construction called the *closure* of a set of input variables, analogous to the definition in [2]. Roughly, the closure is the set of inputs that have short resolution proofs for the current partial assignment. Otherwise, outputs with most neighboring inputs correctly guessed could give very short resolution proofs for their remaining inputs. In this proof, we allow the drunken algorithm to guess all bits in the closure *without* flipping a coin- the attacker gets these bits for free. This can only help the running time of the drunken algorithm. In exchange for this concession, we maintain certain

expansion properties in the graph while the drunken algorithm has drunkenly selected less than $\frac{rc}{4}$ bits.

With high probability, the drunken algorithm will select a globally inconsistent value for an input before this time. In this case, the backtracking algorithm must perform a tree like resolution proof in order to be allowed to backtrack. Cook et al. [5] show this proof will have size $2^{\Omega(r(c-h))}$, so the algorithm must take that many steps before correcting its mistake.

In Section 4.1, we formalize the definition of closure. In Section 4.2, we formally define "clever" drunken algorithms, which are given the input bits in the closure for free. In Section 4.3 we show the result of Cook, Etesami, Miller and Trevisan [5] that backtracking algorithms take exponential time on unsatisfiable formulas. Finally, in Section 4.4 we show that with high probability, the drunken algorithm will create an unsatisfiable formula, completing the proof.

4.1 Closure Operation

We use a definition of taking closure with respect to a set of columns of matrix A.

Definition 4.2 (closure) Analogous to [2, Definition 3.2].
Assume A is an $n \times n$ (r, d, c)-boundary expander. For a set of columns $J \subseteq [n]$ of A, define the following relation on the $[m]$ rows of A:

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

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 $\mathrm{Cl}(J) = G_k$.*

Intuitively, the closure is defined iteratively by collections of less than $r/2$ rows that *do not* satisfy $c/2$ boundary expansion properties. In other words, the closure is the collection of rows that have a small number of unique neighbors.

Definition 4.3 (variables in closure) *The*

Lemma 4.4 *For any sets of columns J_1 and J_2, the closure $\mathrm{Cl}(J_1 \cup J_2)$ can be selected such that $\mathrm{Cl}(J_1 \cup J_2) \supseteq \mathrm{Cl}(J_2)$.*

Any set of rows I_1 satisfying $\left|\partial_A(I_1) \setminus \left[\bigcup_{i \in I} A_i \cup J_1\right]\right| < c/2|I_1|$ must also satisfy $\left|\partial_A(I_1) \setminus \left[\bigcup_{i \in I} A_i \cup (J_1 \cup J_2)\right]\right| < c/2|I_1|$. If the $\mathrm{Cl}(J_1)$ is derived by adding $I_1, I_2, \ldots$, begin the process of deriving $\mathrm{Cl}(J_1 \cup J_2)$ by first adding $I_1, I_2, \ldots$ before continuing with the closure procedure.

Lemma 4.5 *For any set of columns J_1 and $J_2 \subseteq For any sets of columns J_1$ and J_2, the closure $\mathrm{Cl}(J_1 \cup J_2)$ can be selected such that $\mathrm{Cl}(J_1 \cup J_2) \supseteq \mathrm{Cl}(J_2)$.*

Lemma 4.6 Analogous to [2, Lemma 3.5].
If $|J| < \frac{cr}{4}$*, then* $|\mathrm{Cl}(J)| < 2c^{-1}|J|$.

Assume to the contrary that $|J| < cr/4$ but $|\mathrm{Cl}(J)| \geq 2c^{-1}|J|$. Then consider the sequence $I_1, I_2, \ldots, I_t$ appearing in the construction of $\mathrm{Cl}(J)$. These sets must be disjoint, as each set is removed from A after it is created. Denote by $G_t = \bigcup_{k=1}^{t} I_k$ the set of rows derived in t steps. Let T be the first value of t such that $|G_t| \geq 2c^{-1}|J|$. Hence, $|J| \leq c|G_T|/2$. Because A is a (r, d, c)-boundary expander, $|\partial G_T| \geq c|G_T|$, which gives

$$|\partial G_T \setminus J| \geq c|G_T| - |J| \geq c|G_T|/2. \tag{1}$$

On the other hand, for every t, adding I_{t+1} to G_t increases $|\partial G_t \setminus J|$ by at most $|\partial I_{t+1} \setminus \cup_{i \in G_t} A_i \cup J$, which is $< c/2|I_{t+1}|$. This implies

$$|\partial G_T \setminus J| < c|G_T|/2 \tag{2}$$

which contradicts (1).

Lemma 4.7 Analogous to [2, 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.*

Assume to the contrary that $\hat{A}$ has as set of rows I_1 with boundary expansion less than $c/2|I_1|$ in $\hat{A}$. This implies that

$$|\partial_A(I_1) \setminus \left[\bigcup_{i \in I} A_i \cup J \right]| < c/2|I_1|.$$

But then I_1 can be added to $\mathrm{Cl}(J)$, so it was not a set of rows of $\hat{A}$.

Definition 4.8 From [2, 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* $\mathrm{Cl}(\mathrm{Vars}(\rho))$*:*

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

A substitution ρ *is said to be* globally consistent *w.r.t. the function* $f(x) = b$ *if and only if* ρ *can be extended to an assignment that satisfies* $f(x) = b$*.*

Lemma 4.9 Analogous to [2, Lemma 3.6].
Assume that f *employs a* (r, d, c)*-boundary expander and a* h*-robust predicate with* $c > 2h$*. Let* $b \in \{0,1\}^m$ *and* ρ *be a locally consistent partial assignment. Then for any set* $I \subseteq [m]$ *with* $|I| \leq r/2$*,* ρ *can be extended to an assignment* x *which satisfies the subsystem* $(f(x))_I = b_I$*.*

Assume to the contrary that there exists a set I with $|I| \leq r/2$ for which ρ cannot be extended to satisfy $(f(x))_I = b_I$. Choose a minimal such I. Since P is $c/2$-robust, each row in I has at most $c/2$ variables in $\partial_A(I) \setminus \text{Vars}(\rho)$ — otherwise, that row could be removed from I and the remaining rows would be no easier to satisfy. Thus $\text{Cl}(\text{Vars}(\rho)) \supseteq I$ — otherwise, define $I_1 = I \setminus \text{Cl}(\text{Vars}(\rho))$. Since ρ is assumed to be locally consistent, it can be extended to satisfy the constraints in rows $I \subseteq \text{Cl}(\text{Vars}(\rho))$, which is a contradiction.

4.2 Clever Drunken Backtracking Algorithms

Without loss of generality, we allow our algorithm to be a "clever" drunken algorithm in the sense that for input bit added to the partial assignment ρ, the algorithm selects bits in the closure of ρ for free.

More precisely, let $\mathcal{A}$ be a clever drunken algorithm which has guessed input bits in ρ_0. Then $\mathcal{A}$ has already received the bits in $\text{Cl}(\rho_0)$ for free in a way that is globally consistent, if possible. If the scheduler selects bit $r_1 \notin \text{Cl}(\rho_0)$, the value of bit r_1 is drunkenly selected from $\{0, 1\}$ at random, and clever $\mathcal{A}$ selects globally consistent values for any in $\text{Cl}(\rho_0 \cup r_1) \supseteq \text{Cl}(\rho_0)$ for free.

If all of the algorithm's drunkenly selected inputs are globally consistent (extendable to a correct input), we can assume without loss of generality that all of its "free" input bits are globally consistent. If the drunkenly selected input bits are consistent, the algorithm cannot backtrack to change these bits. If it does not select globally consistent choices for its input bits, we can simply ignore the time it takes for the algorithm to correct these values. Additionally, the computation of the drunken algorithm is unbounded, so the algorithm should be able to select globally consistent values for these bits immediately.

The clever property can only reduce the number of backtracking steps taken, and the total amount of time taken by the drunken algorithm.

Lemma 4.10 *After $\frac{rc}{4}$ steps, a clever backtracking algorithm $\mathcal{A}$ has assigned values to at most $\frac{r}{2}$ variables.*

After $\frac{rc}{4}$ steps, $\mathcal{A}$ has drunkenly assigned at most $\frac{rc}{4}$ variables, and may have additional assigned variables in the closure of the drunkenly assigned variables. For each partial assignment ρ_0, when $\mathcal{A}$ drunkenly selects bit $r_1 \notin \text{Cl}(\rho_0)$, $\mathcal{A}$ also selects values for any in $\text{Cl}(\rho_0 \cup r_1) \supseteq \text{Cl}(\rho_0)$. Proceeding inductively, we note that after $\frac{rc}{4}$ steps $\mathcal{A}$ has assigned values only to variables within the closure generated by the $\frac{rc}{4}$ drunkenly selected variables. By Lemma 4.6, this closure contains at most $\frac{r}{2}$ variables.

Lemma 4.11 *For a clever backtracking algorithm $\mathcal{A}$ inverting Goldreich's function on a (r, d, c)-boundary expander, each drunkenly selected bit will be selected from a $(r/2, d, c/2)$-boundary expander.*

By the definition of clever backtracking algorithms, $\mathcal{A}$ selects values for all variables in the closure of selected values. Then apply Lemma 4.7 to give expansion properties for the remaining instance.

4.3 Backtracking Algorithms use Exponential Running Time on Unsatisfiable Formulas

By definition the inversion problem is satisfiable, and so unsatisfiable instances occur only if the algorithm chooses an input bit value that can not be extended to a correct solution. When any backtracking algorithm runs on an unsatisfiable instance, the transcript of values tried by the algorithm gives a "tree-like resolution proof" of unsatisfiability. On unsatisfiable instances, Ben-Sasson and Wigderson [4] give lower bounds for resolution proofs for linear predicates on expander graphs. [5] adapted the Ben-Sasson and Wigderson work to show that any resolution proof of this fact is large for arbitrary predicates on expander graphs. From this, it follows that any drunken backtracking algorithm will take a long time to realize its mistake. The proof of [5] utilizes the robustness h of the predicate in addition to the parameters r and c of the expander graph.

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. Cook et al. [5] find a lower bound on the width of a resolution refutation of an unsatisfiable SAT instance by relating to an h-robust function to be $\frac{(c-h)r}{2}$, and the apply the following lemma from [4, Corollary 3.4]:

Lemma 4.12 *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 $w\Psi$ is the maximal length of a clause in Ψ.*

To do this, fix an $m \times n$ matrix A which is a (r, d, c)-expander, and h-robust functions $g_i : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{Vars}(g_i) \subseteq X_i(A)$. Fix an output vector $b \in \{0,1\}^m$. For a row $i \in [m]$, let $X_i(A) = \{x_{j_1}, \ldots, x_{j_s}\}$, and let Φ_i be the CNF in the variables $X_i(A)$ consisting of all clauses $C = x_{j_1}^{\epsilon_1} \vee \cdots \vee x_{j_s}^{\epsilon_s}$ such that $g_i(x) = b_i \models C$. Let $\Phi = \Phi_1 \wedge \cdots \wedge \Phi_m$.

Given any clause C, define

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

Lemma 4.13 *If $\frac{r}{2} \leq \mu(C) \leq r$, then C has width at least $\frac{(c-h)r}{2}$.*

Let I be a minimal set of rows achieving $A_I x = b_I \models C$, so $\frac{r}{2} \leq |I| \leq r$. Then $|\partial_A(I)| \geq c|I|$.

Assume $|C| < \frac{(c-h)r}{2}$. Then $|C| < (c-h)|I|$ and $|\partial_A(I) \setminus \mathrm{Vars}(C)| > (h)|I|$. Select some $i \in I$ such that $|\partial_A(I) \cap X_i(A) \setminus \mathrm{Vars}(C)| > h$ and set $I' = I \setminus \{i\}$.

$A_{I'} x = b_{I'} \not\models C$, so there is some assignment x such that $A_{I'} x = b_{I'}$ but x does not satisfy C. Since g_i is ℓ-robust, there exists an assignment x' which agrees with x except for variables in $\partial_A(I) \cap X_i(A) \setminus \mathrm{Vars}(C)$, such that $g_i(x_i) = b_i$. But then $A_I x' = b_I$ and x' does not satisfy C, which is contradicts our assumption that $A_I x = b_I \models C$. Thus our assumption that $|C| < \frac{(c-h)r}{2}$ must have been false.

Lemma 4.14 *0. For any $D \in \Phi$, $\mu(D) = 1$.*

1. $\mu(\varnothing) > r$.

2. μ *is subadditive: if* C_2 *is the resolution of* C_0 *and* C_1*, then* $\mu(C_2) \leq \mu(C_0) + \mu(C_1)$.

0 and 2 are easy, and 1 follows from Lemma 4.13.
This theorem is analogous to [3] Theorem 3.1 or [2] Lemma 3.7.

Theorem 4.15 *Any resolution proof that* Φ *is unsatisfiable has width at least* $\frac{(c-h)r}{2}$.

By Lemma 4.14, some clause C must have $\frac{r}{2} \leq \mu(C) \leq r$; apply Lemma 4.13.

Theorem 4.16 Analogous to [2, Lemma 3.9].
If a set of drunkenly selected bits ρ *with* $|Vars(\rho)| \leq cr/4$ *results in an unsatisfiable formula* $\Phi(b)[\rho]$ *then every drunken backtracking algorithm would work* $2^{\Omega(r(c-h))}$ *on* $\Phi(b)[\rho]$.

The computation of a backtracking algorithm as it proves a formula is unsatisfiable can be translated to a tree-like resolution refutation such that the size of the refutation is the working time of the algorithm by [4, Corollary 3.4] given earlier in the section. Thus it is sufficient to show that the minimal tree-like resolution refutation of $\Phi(b)[\rho]$ is large. Denote by $I = Cl(\rho), J = \cup_{i \in I} A_i$. By Lemma 4.6 $|I| \leq r/2$. By Lemma 4.9 ρ can be extended to another partial assignment ρ' on variables x_J, such that ρ' satisfies every equation in $G_I(x) = b_I$. The restricted formula $(G(x) = b)|_{\rho'}$ still encodes an unsatisfiable system $G'(x) = b'$. G' is based off matrix A', where A' results from A by removing rows corresponding to I and variables corresponding to J. By Lemma 4.7, A' is an $(r/2, d, c/2)$-boundary expander. Lemmas 4.13 and 4.12 now imply that the minimal tree-like resolution refutation of the Boolean formula corresponding to the system $G'(x) = b'$ has size $2^{\Omega(r(c-h))}$.

4.4 The Probability of a Correct Guess is Small

Definition 4.17 *Let* $R \subseteq [n]$, $\rho \in \{0,1\}^R$, *and* ι *the output variables that can be calculated from* R.

We say (R_v, ρ_v, ι) *is a* consistent state *if*

$$\Pr[R_v = R \wedge \rho_v = \rho \wedge \iota = b] > 0.$$

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

Lemma 4.18 *Assume* A *is an* $n \times n$ (r, d, c)*-boundary expander with right degree* d *and that* P *is an* h*-robust 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 drunken backtracking algorithm. Choose $x \in \{0,1\}^n$ uniformly at random and let $b = f(x)$. Then the probability that a clever drunken algorithm makes no mistakes in its first $\lfloor \frac{cr}{4} \rfloor$ steps is at most $M(\frac{1}{2})^{\frac{rc}{4}}$, where probability is over choice of x and over the coin flips of the drunken algorithm.

With this Lemma, we can now complete our proof of Theorem 4.1. Choose b randomly from the set of attainable outputs of $f(x)$; more formally, let $x \sim \text{Unif}(\{0,1\}^n)$ and $b = f(x)$.

Define random variable ρ_v to be the first values assigned to the first $\lfloor \frac{cr}{4} \rfloor$ drunkenly selected input bits, and define $R_v = \text{Vars}(\rho_v)$. Recall that without loss of generality, if all drunkenly selected inputs are correct at a given point, we can assume all "free" bits from the closure are globally correct, too.

Here, we try to upper bound the probability that all drunkenly selected inputs are globally consistent.

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

$$Pr[E] = \sum_b \Pr[E|b] \Pr[b] = \mathbf{E}[\Pr[E|b]].$$

We consider each input that maps to b separately. Since all inputs in R_v are selected drunkenly, each input in R_v has a $\frac{1}{2}$ chance of matching a given bit. Thus, for a given input $x \in \{0,1\}^n$, the probability that $x_{R_v} = \rho_v$ is $(\frac{1}{2})^{|R_V|} = (\frac{1}{2})^{\lfloor \frac{cr}{4} \rfloor}$. Then

$$\Pr[E|b] \leq \sum_{y \in f^{-1}(b)} \Pr[\rho_v = y_{R_v}] \leq |f^{-1}(b)|(\frac{1}{2})^{\lfloor \frac{cr}{4} \rfloor}$$

Substituting, we find

$$Pr[E] \leq \mathbf{E}[|f^{-1}(b)|(\frac{1}{2})^{\lfloor \frac{cr}{4} \rfloor}] = M(\frac{1}{2})^{\frac{rc}{4}}.$$

If a globally inconsistent value is chosen on a drunkenly selected input bit, an unsatisfiable instance will occur. In Section 4.3, the running time of backtracking algorithms on unsatisfiable cases is shown to take time $2^{\Omega(r(c-h))}$. For drunkenly selected inputs, the graph will have $(r/2, d, c/2)$-boundary expansion by Lemma 4.11. Then if E does not occur, which happens with probability $\geq 1 - M \cdot (\frac{1}{2})^{\frac{rc}{4}}$, an arbitrary drunken backtracking algorithm will take time $2^{\Omega(r(c-h))}$ to invert Goldreich's function.

References

[1] Dimitris Achlioptas and Gregory B. Sorkin. Optimal myopic algorithms for random 3-SAT. In *FOCS*, pages 590–600, 2000.

[2] Alekhnovich, Hirsch, and Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2004.

[3] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004.

[4] Ben-Sasson and Wigderson. Short proofs are narrow–resolution made simple. *JACM: Journal of the ACM*, 48, 2001.

[5] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *2009 Theory of Cryptography Conference Proceedings*, pages 521–538, 2009.

[6] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

[7] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in nc^0. *Random Struct. Algorithms*, 29(1):56–81, 2006.