Exponential Lower Bounds for a DPLL Attack against a One-Way Function Based on Expander Graphs

Rachel Miller, University of Virginia
Professor Luca Trevisan in EECS Department, UC Berkeley
James Cook, UC Berkeley
Omid Etesami, UC Berkeley
August 1, 2008

This paper was written as a culmination of my work in the SUPERB 2008 Program at UC Berkeley, but is a work in progress. Please contact me at rachelmiller@virginia.edu for the most recent copy.

Abstract

Oded Goldreich’s 2000 paper “Candidate One-Way Functions Based on Expander Graphs” [4] describes a function that employs a fixed random predicate and an expander graph. Goldreich conjectures that this function is difficult to invert, but this difficulty does not seem to stem from any standard assumption in Complexity Theory. The task of inverting Goldreich’s function reduces naturally to a SAT instance. We adapt the work of Alekhnovich, Hirsch and Itsykson [1] to show that any myopic DPLL algorithm takes on average exponential time to invert the function. [1] shows this when the predicate is $x_1 \oplus x_2 \oplus x_3$; we show it for higher-degree linear predicates, and for random predicates under a plausible assumption about Goldreich’s function.

DPLL is for Davis, Putnam, Logemann and Loveland, and many modern SAT solvers fit in the DPLL framework. For unsatisfiable instances, DPLL algorithms are subject to exponential lower bounds of tree-like resolution proofs. However, few lower bounds exist for satisfiable instances such as this. “Myopic” stipulates that the heuristic guiding the backtracking can only read a small part of the function’s output at a time; without any restriction, the heuristic could immediately guide the algorithm to the correct solution.
1 Introduction

We utilize the potential one-way function developed in Goldreich’s paper [4]. Each output bit relies on a fixed number of input bits determined by an expander graph. Goldreich notes that this function seems to be exponentially difficult to solve in some measure of the expansion.

Inversion of the function naturally translates into a SAT-instance, where SAT is the boolean satisfiability problem in conjugate normal form. The clauses in the SAT instance will all have size of the degree of the expander graph. As inversion will always have a solution, its corresponding SAT instance will always be satisfiable. Lower bounds for unsatisfiable cases are equivalent to tree-like resolution proofs, but few bounds exist for satisfiable cases. [1] gives exponential lower bounds on average for inverting linear degree-3 predicates. Like their paper, we assume a Myopic Algorithm, which can only view a limited amount of SAT-clauses per step. We follow their work, and create exponential lower bounds on average for inverting linear functions of any degree.

We further extend the work to accommodate functions of varying robustness. Robustness is a measure of how many bits of input must be given fixed truth values before the output might have a fixed truth value. Linear functions are fully robust; all bits of a linear function must be fixed before the output is fixed. By accommodating for functions of various robustness, we create lower bounds for a function based off a short, random predicate, rather than linear functions. Thus, we probe lower exponential bounds for inverting Goldreich’s predicate.

Section 2 formally describes Goldreich’s function and myopic algorithms. It also describes properties of random predicates and contains information about expansion. Section 3 describes exponential lower bounds for average case of inverting Goldreich’s function; this contains the bulk of our novel work. Section 4 closely follows [1] to give exponential lower bounds on unsatisfiable inversions. We use the results of Section 4 for our proofs in Section 3.

2 Preliminaries

2.1 Goldreich’s Function

Goldreich constructs a collection of functions \( \{ f_{n,m} : \{0,1\}^n \rightarrow \{0,1\}^m \} \) \( n,m \in \mathbb{N} \). The function will employ a predicate \( P : \{0,1\}^d \rightarrow \{0,1\} \) for a constant \( d \); Goldreich suggests using a random predicate. This construction also uses a collection of \( d \)-subsets, \( S_1, \ldots, S_m \subset \{1, \ldots, n\} \), which should satisfy certain expansion properties. In our paper, we commonly refer to the \( m \times n \) matrix \( A \) which has its rows comprised of \( S_1, \ldots, S_m \). For \( x = x_1 \cdots x_n \in \{0,1\}^n \) and \( S \subset \{1, \ldots, n\} \), where \( S = \{i_1, i_2, \ldots, i_t\} \) and \( i_j < i_{j+1} \), Goldreich denotes by \( x_S \) the projection of \( x \) on \( S \). Thus, \( x_S = x_{i_1} x_{i_2} \cdots x_{i_t} \). For a fixed predicate \( P \) and fixed \( S_1, \ldots, S_m \) with expansion, Goldreich defines

\[
f_n \overset{\text{def}}{=} P(x_{S_1})P(x_{S_2}) \cdots P(x_{S_m})
\]
For a fixed \( y \in \{0, 1\}^m \), we define a \( d \)-CNF formula \( \Phi_y(x) \) which is logically equivalent to the statement \( f(x) = y \). The \( i \)-th bit of \( y \) translates to a set of at most \( 2^d \) clauses that enforce the constraint \( P(x_{S_i}) = y_i \). The problem of inverting \( f \) is thus reduced to finding a solution to the SAT instance \( \Phi_y \) for some \( y \).

### 2.2 DPLL Algorithms

DPLL Algorithms (for Davis, Putnam, Logemann and Loveland) form the basis of nearly all efficient and complete SAT solvers. Generally, DPLL algorithms are all backtracking algorithms. They select a boolean variable, substitute a truth value for that variable, and recursively checking if the resulting formula is satisfiable. If the resulting formula is unsatisfiable, the algorithm “backtracks” and tries the opposite truth value for that variable.

DPLL Algorithms can be said to have some Method A of selecting a variable, and then some second Method B for selecting the truth value for that variable. Algorithms are also allowed to use logical manipulations and substitutions in between steps that don’t change the satisfiability of the formula, such as pure literal elimination and deciding the values of variables in unit clauses.

If \( P=NP \) and Method B is not constricted, the algorithm can simply choose the correct value for each variable, and so will quickly terminate. Thus, proving exponential bounds for an algorithm with an unrestricted Method B would be equivalent to showing \( P\neq NP \). To show exponential lower bounds for the average case, we must restrict Method B in some way, and prove the lower bounds for this restricted DPLL Algorithm.

Myopic Algorithms restrict both Methods A and B with respect to which clauses of the formula they can read. Method A can read \( K = n^{1-\epsilon} \) clauses per substitution (for some \( \epsilon > 0 \)), the formula with negation signs removed, and the number of occurrences of each literal. Method B can use information obtained by Method A. Information revealed can be used in subsequent recursive calls, but not in different recursive branches of the DPLL tree.

### 2.3 Random Predicates

**Definition 2.1** (partial assignment). Taken from [2]. A partial assignment is a function \( \rho : [n] \to \{0, 1, *\} \). Its size is defined to be \( |\rho| = |\rho^{-1}(\{0, 1\})| \). Given \( f : \{0, 1\}^n \to \{0, 1\}^m \), the restriction of \( f \) by \( \rho \), denoted \( f|_{\rho} \), is the function obtained by fixing the variables in \( \rho^{-1}(\{0, 1\}) \) and allowing the rest to vary.

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 2.2** (robust predicate). \( p : \{0, 1\}^d \to \{0, 1\} \) is \( h \)-robust iff every restriction \( \rho \) 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 2.3** (balanced predicate). \( p : \{0, 1\}^d \to \{0, 1\} \) is \((h, \epsilon)\)-balanced if, after fixing all variables but \(h + 1\) of them,

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

Special case: predicates of the form \(x_1 \oplus \cdots \oplus x_{d-2} \oplus (x_{d-1} \land x_d)\) are \((2, 0)\)-balanced and \((1, \frac{1}{2})\)-balanced. The predicate that sums all its inputs is \((0, 0)\)-balanced.

**Lemma 2.4.** A random predicate on \(d\) variables is \((\Theta(\log \frac{d}{\epsilon}), \epsilon)\)-balanced with probability \(1 - \exp[-\text{poly}(d/\epsilon)]\).

**Proof.** A random predicate is not \((h, \epsilon)\)-balanced with probability

\[
\leq 2^{d-h-1} \binom{d}{h+1} \Pr[|x_1 + \ldots + x_{2^{h+1}} - 2^h| > 2^h \epsilon]
\]

(Chernoff’s bound)

\[
\leq 2^{d-h} \binom{d}{h+1} \exp\left[\frac{-2^h \epsilon^2}{2^{h+1}}\right]
\]

\[
\leq 2^{d-h} \binom{d}{h+1} \exp[-2^h \epsilon^2]
\]

\[
\leq 2^{d-h} d^{h+1} \exp[-2^h \epsilon^2]
\]

\[= \exp[(h + 1) \ln d + (d - h) \ln 2 - \epsilon 2^h].\]

Finally, take \(h = \Theta(\log \frac{d}{\epsilon})\).

**Corollary 2.5.** 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 \(n - d\) zeroes in each row. For \(i \in [m]\), let \(J_i = \{j : A_{ij} = 1\}\).

Goldreich utilizes the following expansion:

**Definition 2.6** (Goldreich’s Expansion). In [4], the expansion of \(A\) is defined to be

\[
\max_k \min_{I \subseteq [m] \mid |I| = k} \left| \bigcup_{i \in I} J_i \right| - k.
\]

Goldreich notes that the hardness of inverting his function seems to be exponential in its expansion.

**Definition 2.7** (Boundary Element). Taken from [1, Definition 2.1]. For a set of rows \(I\) of our \(m \times n\) matrix \(A\), we define its boundary \(\partial I\) as the set of all \(j \in [n]\) (called boundary elements) such that there exists exactly one row \(i \in I\) that contains \(j\).
Definition 2.8 (Expansion). Taken from [1, 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' \(\forall I \subseteq [m], (|I| \leq r \Rightarrow |\bigcup_{i \in I} A_i| \geq c|I|)\).

Throughout the rest of our paper, we assume

\[ e - h \geq 1 \]  

(2)

and also assume

\[ c > 4h/3. \]  

(3)

Recall that Corollary 2.5 gives random predicate on \(d\) variables is \(\Theta(\log d)\)-robust with probability \(1 - \exp[-poly(d)]\). Expander graphs exist with \(c\) arbitrarily close to \(d - 1\). Thus, as \(d\) increases, \(c\) can be made much larger than \(4h/3\).

Lemma 2.9. Analogous to [1, Lemma 2.1].

Any \((r, d, c)\)-expander is an \((r, d, 2c - d)\)-boundary expander.

Proof. Assume that \(A\) is an \((r, s, c)\)-expander. Consider a set of its rows \(I\) with \(|I| \leq r\). Since \(A\) is an expander \(\big|\bigcup_{i \in I} A_i\big| \geq c|I|\). On the other hand we may estimate separately the number of boundary and non-boundary variables which will give \(\big|\bigcup_{i \in I} A_i\big| \leq E + (d|I| - E)/2\), 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 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 use a definition of taking closure with respect to a set of columns of matrix \(A\).

Definition 2.10 \((h\text{-closure})\). Analogous to [1, Definition 3.2].

For a set of columns \(J \subseteq [n]\) define the following relation on \(2^{[m]}\):

\[ I \vdash^h J \iff I \cap I_1 = \emptyset \land |I_1| \leq \frac{r}{2} \land \left| \partial_A(I_1) \setminus \left[ \bigcup_{i \in I} A_i \cup J \right] \right| < 3c/4|I_1|. \]

Given a set \(J \subseteq [n]\), define the \(h\)-closure of \(J\), \(\text{Cl}^h(J)\), as follows. Let \(G_0 = \emptyset\). Having defined \(G_k\), choose a non-empty \(I_k\) such that \(G_k \vdash^h I_k\), and set \(G_{k+1} = G_k \cup I_k\). Remove equations \(I_k\) from matrix \(A\). (Fix an ordering on \(2^{[m]}\) to ensure a deterministic choice of \(I_k\).) When \(k\) is large enough that no non-empty \(I_k\) can be found, set \(\text{Cl}^h(J) = G_k\).
Lemma 2.11. Analogous to [1, Lemma 3.5].
If $|J| < \frac{r}{2}$, then $|C_l(J)| < 2c^{-1}|J|$.

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

$$|\partial C_t \setminus J| \geq c|C_t| - |J| > 3c|C_t|/4. \quad (4)$$

However, for each $I_{t+1}$ added to $C_t$, only $3c/4$ new elements may be added to $\partial C_t \setminus J$. This implies

$$|\partial C_t \setminus J| \leq 3c|C_t|/4 \quad (5)$$

which contradicts 4.

Lemma 2.12. Analogous to [1, Lemma 3.4].
Assume that $A$ is an arbitrary matrix and $J$ is a set of its columns. Let $I^* = C_l(J), J^* = \bigcup_{i \in C_l^*(J)} A_i$. Denote by $\tilde{A}$ the matrix that results from $A$ by removing the rows corresponding to $I^*$ and columns to $J^*$. If $\tilde{A}$ is non-empty than it is an $(r/2,d,3c/4)$-boundary expander.

Proof. If $\tilde{A}$ is non-empty, there must exist non-empty subsets of $\tilde{A}$ with size $\leq r/2$. Such subsets $I_k$ must contradict

$$|\partial A(I_k) \setminus \left[ \bigcup_{i \in I} A_i \cup J \right]| < 3c|I_k|/4 \quad (6)$$

by the definition of closure. This satisfies the definition of $(r/2,d,3c/4)$-boundary expansion.

Definition 2.13. Analogous to [1, Definition 3.4].
A substitution $\rho$ is said to be locally consistent w.r.t. the function $G(x) = b$ if and only if $\rho$ can be extended to an assignment on $X$ which satisfies the equations corresponding to $Cl(\rho)$:

$$G_{Cl(\rho)}x = b_{Cl(\rho)}$$

Lemma 2.14. Analogous to [1, Lemma 3.6].
Assume that $G$ employs a $(r,d,c)$-boundary expander. Let $b \in \{0,1\}^m$ and $\rho$ be a locally consistent partial assignment. Then for any set $I \subseteq [m]$ with $|I| \leq r/2$, $\rho$ can be extended to an assignment $x$ which satisfies the subsystem $G_I(x) = b_I$.

Proof. Assume for the contradiction that there exist sets $I$ for which $\rho$ cannot be extended to satisfy $G_I(x) = b_I$. Choose the minimal such $I$. Then for each row in $I$, no row may have more than $h$ boundary variables, otherwise one could remove an equation with $h + 1$ boundary variables in $\partial A(I) \setminus \text{Vars}(\rho)$ from $I$. But $h < 3c/4$ by assumption 3. Thus, $\text{Cl}(\rho) \supset I$, which contradicts Definition 2.13.
3 Myopic Algorithms use Exponential Time in the Average Case

We show that given the value $y = f(x)$ for a random $x \in \{0, 1\}$, with high probability a Myopic DPLL algorithm will take exponential time to find any inverse of $y$. We assume that the predicate $P$ is balanced in the sense of Definition 2.3, and that $A$ is a boundary expander. The proof strategy shows that after a fixed number of steps, the deterministic myopic algorithm will have selected locally consistent truth values for a set of variables. However, it can only have selected one of many possible locally consistent values- and most of those many locally consistent values are wrong for any single extension of the output seen by the algorithm. Thus, with high probability, the algorithm will have selected globally inconsistent values that lead to an unsatisfiable problem. We then show that any resolution proof showing this new problem is unsatisfiable has size $2^{\Omega(r(c-h))}$, so the algorithm must take that many steps before correcting its mistake.

We use Clever Myopic Algorithms as defined in [1]. Without loss of generality, we assume a myopic algorithm with the following properties:

- whenever the set of variables $x_j$ are revealed, the algorithm can also read all clauses in $Cl(J)$ for free and reveal the corresponding occurrences, where $J$ is the set of all $j$;
- it never asks for the number of occurrences of a literal (the algorithm can compute this number itself: the number of occurrences outside unit clauses does not depend on the substitutions that the algorithm has made; all unit clauses belong to $Cl(J)$;
- Method A always selects one of the revealed variables;
- never makes stupid guesses: whenever it reveals the clauses $\vec{C}$ and chooses the variable $x_j$ for branching it makes the right assignment $x_j = \epsilon$ in the case when $\vec{C}$ semantically imply $x_j = \epsilon$ (this assumption can only save running time).

**Proposition 3.1.** Analogous to [1, 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 2.11 will result in at most $r/2$ bits of $b$. \hfill \Box

**Proposition 3.2.** Analogous to [1, 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 2.14. 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 $t < \frac{cr}{4dK}$)
Lemma 3.4. \[ \text{strictly less than} \ 2 \] for \( |\text{Cl}(\text{Vars}(\rho_t)) \cup \{x_j\}| \leq r/2 \) for the newly chosen \( x_j \).

Now choose \( b \) randomly from the set of attainable outputs of \( f(x) \); more formally, let \( x \sim \text{Unif}\{(0,1)^n\} \) and \( b = f(x) \). 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 \( \lfloor \frac{cr}{\dim} \rfloor \) steps of the Algorithm. By Proposition 3.2, 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 \( \lfloor \frac{cr}{\dim} \rfloor \)-th vertex \( v \) in the leftmost branch of its DPLL tree. By Proposition 3.1, the algorithm has revealed at most \( r/2 \) bits of \( b \).

Denote by \( I_v \subset [m] \) the set of revealed bits, and by \( R_v \) the set of the assigned variables, with \( |R_v| = \lfloor \frac{cr}{\dim} \rfloor \). The idea of this proof is to show out of the many possible locally consistent choices for \( R_v \), only very few will be able to satisfy a given value of \( b \). Denote by \( \rho_v \) the partial assignment to the variables in \( R_v \) made by the algorithm. Consider the following event

\[ E = \{ \rho_v \in (f^{-1}(b))_{R_v} \} \] (7)

Recall that this is over our probability space for \( b \) is over the weighted set of attainable outputs of \( f \). This event holds if and only if there exists some extension of \( \rho_v \) that is globally consistent with \( b \). For \( I \subset [m], R \subset [n], b_{I,v} = \bar{c} \in f(\text{Unif}\{(0,1)^n\})_{t}, \rho \in \{0,1\}^R \) we want to estimate the conditional probability

\[ \Pr[E | I_v = I, R_v = R, b_{I,v} = \bar{c}, \rho_v = \rho]. \] (8)

If we show that this condition probability is small for all choices of \( I, R, \bar{c}, \) and \( \rho \), it follows that the probability of \( E \) is small. Thus, it will be likely that \( \rho_v \), though locally consistent, can not be extended to satisfy \( b \), and an unsatisfiable instance will occur. In Section 4, we explore the running time of DPLL algorithms on unsatisfiable cases to show if \( E \) does not occur, the algorithm will take time \( 2^{O(r(c-h))} \).

Lemma 3.3. Analogous to [1, Lemma 3.10].

Assume that an \( m \times n \) matrix \( A \) is an \( (r, d, c) \)-boundary expander, \( X = \{x_1, \ldots, x_n\} \) is a set of variables, \( \hat{X} \subseteq S, |\hat{X}| < r, b \in f(\text{Unif}\{(0,1)^n\})_m, \) and \( \mathcal{L} = \{\ell_1, \ldots, \ell_k\} \) (where \( k < r \)) is the tuple of constraints corresponding to outputs 1, \ldots, \( k \). Denote by \( L \) the set of assignments to the variables in \( \hat{X} \) that can be extended on \( \hat{X} \) to satisfy \( \mathcal{L} \). For \( d = 3 \), if \( L \) is not empty, then \( \text{dim}(L) \geq |\hat{X}|/(3 \cdot 2^3 + 3 \cdot 2) \). More generally for \( d > 3 \), for \( L \) not empty we have \( \text{dim}(L) \geq \frac{f(\hat{X})}{r + 2^d - 1} \), with \( f \) the greatest integer strictly less than \( 2c - d - h \).

Lemma 3.4. Assume that an \( m \times n \) matrix \( A \) is an \( (r, d, c) \)-boundary expander, \( X = \{x_1, \ldots, x_n\} \) is a set of variables, \( \hat{X} \subseteq S, |\hat{X}| < r, b \in f(\text{Unif}\{(0,1)^n\})_m, \) and \( \mathcal{L} = \{\ell_1, \ldots, \ell_k\} \) (where \( k < r \)) is the tuple of constraints corresponding to outputs 1, \ldots, \( k \). Then for any \( x \in \{0,1\}^|\hat{X}| \),

\[ \Pr[X|\hat{X} = x] \mathcal{L} \leq 2^{-s} \left( \frac{1 + 2r}{1 - 2r} \right)^{|\mathcal{L}|}; \]
if \( d = 3 \) we can take \( s = |\tilde{X}|/(3 \cdot 2^3 + 3 \cdot 2) \), and in general we can take \( s = f|\tilde{X}|/(f + d(d - 1)) \), where \( f \) is the greatest integer strictly less than \( 2c - d - h \).

Assuming \( f \) is nearly one-to-one, in the sense that \( \forall y \in \{0,1\}^m \) \( |f^{-1}(y)| \leq M \), we have

\[
\Pr[E|I_v = I, R_v = R, b_{I_v} = \bar{c}, \rho_v = \rho] \\
\leq \sum_{x \in f^{-1}(b)} \Pr[\rho_v = x] |I_v = I, R_v = R, b_{I_v} = \bar{c}, \rho_v = \rho| \\
\leq M2^{-s} \left( \frac{1 + 2\epsilon}{1 - 2\epsilon} \right)^{|L|}.
\]

### 3.1 Proof of Lemmas 3.3 and 3.4

We want to show that a large number of \( \tilde{X} \)-values are locally consistent with \( L \), the output seen by the algorithm after \( \lfloor \frac{2d}{c} \rfloor \) steps. We do this by showing that there \( \exists g \subset \tilde{X} \) s.t. any value selected for \( g \) can be extended to a locally consistent value for \( \tilde{X} \), and that the size of \( g \) is large. Further, we use Lemma 3.4 to show that no value of \( g \) is much more likely than any other value, and so each has low probability.

In Sections 3.1.1 and 3.1.2, we pick truth values for our set \( g \subset \tilde{X} \) and prove that the remaining system in \( L \) is always still satisfiable.

#### 3.1.1 Degree 3 Case

Though [1] already offers a proof that \( |L| \) is large for the case of degree-3, theirs is only applicable in the case of a linear predicate; our analysis gives a weaker bound but can be used for any degree-3 predicate.

**Lemma 3.5.** Assume \( A \) is an \((r,3,c)\)-expander where \( c > \frac{15}{8} \). Then for every \( \tilde{X} \subseteq [n] \) there exists a \( g \subseteq U \) such that \( |g| \geq |\tilde{X}|/(3 \cdot 2^3 + 3 \cdot 2) \), and \( \forall I \subseteq [m] \), \( |I| \leq r \Rightarrow |\partial I \setminus g| \geq 1 \).

**Proof.** Given \( \tilde{X} \subseteq [n] \), select the largest possible \( g \subseteq \tilde{X} \) such that no two elements of \( g \) are within distance 4 of each other: that is, for any distinct \( i_0, i_1 \in g \), \( i_0 \notin \Gamma^3(\{i_1\}) \). Every element \( i \in g \) excludes at most \( |\Gamma^4(\{i\})| - 1 \leq 3 \cdot 2^3 + 3 \cdot 2 \) other elements from being in \( g \), so \( g \) has size at least \( |\tilde{X}|/(3 \cdot 2^3 + 3 \cdot 2) \).

Consider any \( I \subseteq [m] \) with \( |I| \leq r \) and assume for a contradiction that \( \partial I \setminus g = \emptyset \). Partition \( I \) as \( I = S \cup T \), where \( S = I \cap \Gamma(g) \) and \( T = I \setminus \Gamma(g) \). Notice that by the construction of \( g \), no two elements of \( S \) are within distance 2 of each other: that is, for distinct \( i_0, i_1 \in S \), \( i_0 \notin \Gamma^2(\{i_1\}) \).

Let \( B = \Gamma(S) \setminus g \subseteq [n] \). Let \( j \in B \) then for any \( i_0, i_1 \in S \cap \Gamma(\{j\}) \), \( i_0 \in \Gamma^2(i_1) \), so \( i_0 = i_1 \). Thus \( |S \cap \Gamma(\{j\})| = 1 \), so we have

\[
|B| = |S|.
\]

By assumption, \( \partial I \subseteq g \), so \( B \cap \partial I = \emptyset \), so for every \( j \in B \) there exist distinct \( i_0, i_1 \in I \cap \Gamma(\{j\}) \). At most one of \( i_0 \) and \( i_1 \) is in \( S \), so \( \Gamma(\{j\}) \cap T \neq \emptyset \): so

\[
|B| = |S|.
\]
Now, \[ |\Gamma(I) \setminus g| \geq c|I| - |S| \]
\[ > \frac{15}{8}(|S| + |T|) - |S| \]
\[ = |S| + \frac{3}{2}|T| + (3|T| - |S|) \]
Combining (9) and (10), we have \(|S| \leq 3|T|\).

3.1.2 Higher Degree Case

Let \( g \subset \hat{X} \) be the set of inputs that have been fixed.

**Lemma 3.6.** If each output in has at most \( f \) of its \( d \) inputs fixed, where \( f = \lceil 2c - d - h \rceil - 1 \), then \( \forall I \subseteq \mathcal{L}, |\partial I \setminus g| > h|I| \).

**Proof.** Consider any set of \( a \) outputs. Let \( \varphi = \frac{|\Gamma(g,a)|}{|a|} \); \( \varphi \) is the average number of fixed inputs over the outputs in \( a \). By Lemma 2.9, we have at least \( (2c - d)|a| \) boundary input nodes connected to the \( a \)-outputs. At most \( \varphi|a| \) of these boundary-inputs have been fixed, and so we have at least \( (2c - d)|a| - \varphi|a| \) boundary-inputs outside of \( g \). Thus, \( |\partial a \setminus g| \geq (2c - d)|a| - \varphi|a| > h|a| \), and \( \varphi < 2c - d - h \). Let \( f \) be the maximum value for \( \varphi \); \( f \) will be the greatest integer strictly less than \( 2c - d - h \). Thus, \( f = \lceil 2c - d - h \rceil - 1 \).

**Lemma 3.7.** If \( \forall I \subseteq \mathcal{L}, |\partial I \setminus g| > h|I| \), then \( \mathcal{L} \) is satisfiable.

**Proof.** We make our proof by contradiction; assume \( \mathcal{L} \) is unsatisfiable. 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 3.8.** We can find \( g \) with \( |g| \geq \frac{f|\hat{X}|}{f + 4d_n(4d_n - 1)} \), such that no output has more than \( f \) inputs in \( g \).

**Proof.** Construct \( g \) using the following algorithm. We will
- \( g \leftarrow \emptyset \).
\[ n_i \leftarrow \begin{cases} f & i \in \hat{X} \\ 0 & i \notin \hat{X} \end{cases} \]

\[ \text{while } \exists i, n_i > 0, \]

- **Invariant:** If an output has \( f - a \) inputs in \( g \), then for every input \( i \) connected to it, \( n_i \leq a \).

- \( g \leftarrow g \cup \{i\} \).

- \( n_i \leftarrow n_i - f \).

- \( \forall j, \text{if dist}(i, j) = 2, \text{then } n_j \leftarrow n_j - 1 \).

We start with \( f|\hat{X} | \) counters, and remove \( f + \text{left}(d_{\text{right}} - 1) \) counters at every step, so in the end,

\[ |g| \geq \frac{f|\hat{X} |}{f + \text{left}(d_{\text{right}} - 1)} \]

We have now proved Lemma 3.3. We can find a set \( g \subset \hat{X} \) with \( |g| \geq f|\hat{X} | \) such that for any fixed value of \( g \), \( L \) is still satisfiable, by Lemmas 3.6, 3.7, and 3.8. Thus, \( \dim(L) \geq \frac{f|\hat{X} |}{f + \text{left}(d_{\text{right}} - 1)} \), where \( L \) is the set of assignments to the variables in \( \hat{X} \) that can be extended on \( X \) to satisfy \( L \).

### 3.1.3 Proof of Lemma 3.4

We use this proof to show that no value of \( g \) is much more likely than any other to be extendable to satisfy a fixed output. Further, \( g \) is large, so no value of \( \hat{X} \) may be too likely. Otherwise, the Myopic Algorithm could simply pick the most likely value of \( \hat{X} \).

**Lemma 3.9.** Assume \( p \) is \((h, \epsilon)\)-balanced. Let \( \mathcal{L} \subseteq [m] \) and \( g \subseteq [n] \). If every \( I \subseteq \mathcal{L} \) has at least \( h + 1 \) boundary elements outside \( g \), then

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

**Proof.** Find a sequence \( \mathcal{L} = L_1, L_1|{i+1}|, \ldots, L_0 = \emptyset \) such that \( L_{i+1} = L_i \cup \{i+1\} \), and \( \forall i, (\Gamma(i+1)) \backslash (\Gamma(L_{i-1}) \cup g) \geq h + 1 \).

Then

\[ \frac{\Pr[x_g = g_1 | L_{i+1}]}{\Pr[x_g = g_2 | L_{i+1}]} = \frac{\Pr[L_{i+1} | x_g = g_1] \Pr[x_g = g_2]}{\Pr[L_{i+1} | x_g = g_2] \Pr[x_g = g_1]} \]

\[ = \frac{\Pr[L_i | x_{gi} = g_1] \Pr[i+1 | L_i, x_{gi} = g_1]}{\Pr[L_i | x_{gi} = g_2] \Pr[i+1 | L_i, x_{gi} = g_2]} \]

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

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

11
The Lemma follows when we observe that
\[
\frac{\text{Pr}[x|g = g_1|L_0]}{\text{Pr}[x|g = g_2|L_0]} = 1.
\]

Take \( g_1 \) such that \( \text{Pr}[x|g = g_1|L] \) is as small as possible. There are \( 2^{|g|} \) possible values for \( g_1 \), so \( \text{Pr}[x|g = g_1|L] \leq 2^{-|g|} \). So by Lemma 3.9, for any \( g_2 \),
\[
\text{Pr}[x|g = g_2|L] = \text{Pr}[x|g = g_1|L] \frac{\text{Pr}[x|g = g_2|L]}{\text{Pr}[x|g = g_1|L]} \leq 2^{-|g|} \left( \frac{1}{2} + \epsilon \right)^{|L|}.
\]

### 3.2 Linear Case for \( d > 3 \)

The work in [1] gives exponential lower bounds for the average case of inverting a degree-3 linear predicate. We add to their work by giving exponential lower bounds for inverting linear predicates of any degree.

Recall we have chosen \( |\hat{X}| = \lceil c^r \frac{4d}{dK} \rceil \). For \( L \) is the set of locally consistent assignments to the variables in \( \hat{X} \). By Lemma 3.3, we have \( \dim(L) \geq f |\hat{X}| \geq f \lceil c^r \frac{4d}{dK} \rceil \geq f \lceil c^r \frac{4d}{dK} \rceil \frac{2c-d-h}{2d} \). We can find expander graphs with \( c \) arbitrarily close to \( d-1 \), and so \( \dim(L) \in \Omega(\frac{r}{dK}) \).

For \( I \) the set of revealed bits in \( b \), as in [1, Lemma 3.10], let
\[
(\hat{b})_i = \begin{cases} 
\epsilon_i & i \in I \\
 b_i & \text{otherwise}
\end{cases}
\] (11)

When \( I_v = I \) and \( b_I = \epsilon, \hat{b} \) has the distribution of \( b \). [1] notes that the vector \( \hat{b} \) is independent from the event \( E_1 = [I_v = I, b_I = \epsilon, \rho_v = \rho] \), because to determine whether \( E_1 \) holds is only dependent on the bits \( b_I \). Like [1], we assume an expander graph of full rank. Thus, \( Ax = b \) must be an injective transformation. Thus,
\[
\text{Pr}[E] = \text{Pr}[A^{-1}b] = \rho |I_v = I, b_I = \epsilon, \rho_v = \rho]
= \text{Pr}[(A^{-1}b)_R = \rho |I_v = I, b_I = \epsilon, \rho_v = \rho]
\leq 2^{-\dim L} \leq 2^{-\lceil c^r \frac{4d}{dK} \rceil \frac{2c-d-1}{2d}}.
\]

If \( E \) does not happen, we will prove in Lemma 4.5 it will take \( 2^{\Omega(r(c-h))} \) for the algorithm to refute the resulting unsatisfiable condition.

### 4 DPLL Algorithms use Exponential Running Time on Unsatisfiable Formulas

In Section 3, we showed that with high probability a myopic DPLL algorithm will choose a partial assignment to \( x \) that cannot be extended to satisfy \( f(x) = y \): that
Proof. Let \( \Phi(x) \) be a minimal set of rows achieving \( |I| \). Let \( A \) be the \( m \times n \) matrix of \( \{0,1\}\)-values \( A_{ji}(x) \). Fix an output vector \( b \in \{0,1\}^m \). For a row \( i \in [m] \), let \( X_i = \{ x_j : x_j \in \text{Vars}(g_i) \} \). Let \( \Phi \) be the CNF consisting of all clauses \( C = \bigwedge_{i \in I} x_{j_i} \) such that \( g_i(x) = b_i \). Let \( \mu = \min_{I : A_i = b} |I| \).

Lemma 4.1. The size of any tree-like resolution refutation of a formula \( \Psi \) is at least \( 2^{w-w\Psi} \), where \( w \) is the minimal width of a resolution refutation of \( \Psi \), and \( w\Psi \) is the maximal length of a clause in \( \Psi \).

Our setup and proof strategy are similar to those found in [2] and [1]. [2] measures robustness in terms of \( \ell \), where \( \ell = d - h \). Our result is identical to Theorem 3.1 in [2], except that our hypothesis is weaker since our formula has less clauses, and our resulting width is \( (c+\ell-d)r \) instead of \( (c+\ell-d)\).

By our assumption in Equation 2, the resulting width is \( \geq r/2 \).

Assume \( |I| < \frac{r}{2} \). Then \( |I| < (c-h)\mu(|I|) \) and \( |\partial(A)\cap\partial(A)| > (h)\mu(|I|) \). Select some \( i \in I \) such that \( |\partial(A)\cap\partial(A)| > (c-h)\mu(|I|) \). Let \( A_{i}x = b_{i} \), so there is some assignment \( x \) such that \( A_{i}x = b_{i} \) but \( x \) does not satisfy \( C \). Since \( g_i \) is \( \ell \)-robust, there exists an assignment \( x' \) which agrees with \( x \) except for variables in \( \partial(A)\cap\partial(A) \), such that \( g_i(x') = b_i \). But then \( A_{i}x' = b_{i} \) and \( x' \) does not satisfy \( C \), which is contradictory to our assumption that \( A_{i}x = b_{i} \). Thus our assumption that \( |I| < \frac{r}{2} \) must have been false.

Lemma 4.3. For any \( D \in \Phi \), \( \mu(D) = 1 \).

1. \( \mu(\emptyset) > 1 \).

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

Proof. This theorem is analogous to [2] Theorem 3.1 or [1] Lemma 3.7.
Theorem 4.4. Any resolution proof that $\Phi$ is unsatisfiable has width at least \( \frac{(c-h)r}{2} \).

Proof. By Lemma 4.3, some clause $C$ must have $\frac{r}{2} \leq \mu(C) \leq r$; apply Lemma 4.2. 

Theorem 4.5. Analogous to [1, Lemma 3.9].

If a locally consistent substitution $\rho$ such that $|\text{vars}(\rho)| \leq cr/4$ results in an unsatisfiable formula $\Phi(b)[\rho]$ then every generalized myopic DPLL would work $2^{O(r(c-h))}$ on $\Phi(b)[\rho]$.

Proof. The state of a DPLL 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. Thus it is sufficient to show that the minimal tree-like resolution refutation of $\Phi(b)[\rho]$ is large. Denote by $I = C^h(\rho)$, $J = \cup_{i \in I} A_i$. By Lemma 4 $|I| \leq r/2$. By Lemma 2.14 $\rho$ can be extended to another partial assignment $\rho'$ on variables $x_I$, such that $\rho'$ 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 2.12, $A'$ is an $(r/2, d, 3c/4)$-boundary expander. Lemmas 4.2 and 4.1 now imply that the minimal tree-like resolution refutation of the Boolean formula corresponding to the system $G'(x) = b'$ has size $2^{O(r(c-h))}$. 

5 Conclusion

Goldreich has already shown that inverting his function using a simple backtracking algorithm is hard for most predicates [4]. Our work adds to the evidence that Goldreich’s function is one-way by showing inversion using a specific family of DPLL algorithms is hard. Specifically, we give exponential lower bounds for average case inversion of Goldreich’s function by myopic algorithms. This includes Goldreich functions for a wide array of predicates; most random predicates satisfy our necessary balance and robustness conditions.

More generally, our research adds to the small amount of work about inverting satisfiable SAT-instances. Previous work in [1] gives exponential lower bounds for inverting degree-3 linear predicates. We add exponential lower bounds for solving linear predicates in any degree. By accommodating functions of various robustness and balance, we show exponential lower bounds hold for inverting most random predicates. We also added further analysis of Goldreich’s function, giving insight on a very new type of combinatorial function.

Our current work relies on the conjecture that most random predicates are few-to-one. We hope to prove this conjecture in future work. Predicates that are many-to-one are easier to invert than those that are few-to-one; the more inputs that map to a given output, the easier it should be to find one of them. Other future work could explore additional attacks against Goldreich’s function for further evidence that the function is in face one-way.
References


