Pseudorandomness against Depth-2 Circuits and Analysis of Goldreich's Candidate One-Way Function

Seyed Omid Etesami

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-180
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-180.html

December 30, 2010

Pseudorandomness against Depth-2 Circuits and Analysis of Goldreich's Candidate One-Way Function

by

Seyed Omid Etesami

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2010

Pseudorandomness against Depth-2 Circuits and Analysis of Goldreich's Candidate One-Way Function

Abstract

Pseudorandomness against Depth-2 Circuits and Analysis of Goldreich's Candidate One-Way Function

by

Seyed Omid Etesami

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Luca Trevisan, Chair

In the first part of this thesis, we consider the construction of unconditional pseudorandom generators whose outputs look random to depth-2 boolean circuits. We prove the existence of a $poly(n, m)$-time computable pseudorandom generator which "$1/poly(n, m)$-fools" DNFs with n variables and m terms, and has seed length $O(\log^2 nm \cdot \log\log nm)$. Previously, the best pseudorandom generator for depth-2 circuits had seed length $O(\log^3 nm)$, and was due to Bazzi (FOCS 2007).

It follows from our proof that a $1/m^{\tilde{O}(\log mn)}$-biased distribution $1/poly(nm)$-fools DNFs with m terms and n variables. For inverse polynomial distinguishing probability this is nearly tight because we show that for every m, δ there is a $1/m^{\Omega(\log 1/\delta)}$-biased distribution X and a DNF ϕ with m terms such that ϕ is not δ-fooled by X.

For the case of *read-once* DNFs, we show that seed length $O(\log mn \cdot \log 1/\delta)$ suffices, which is an improvement for large δ.

It also follows from our proof that a $1/m^{O(\log 1/\delta)}$-biased distribution δ-fools all read-once DNFs with m terms. We show that this result too is nearly tight, by constructing a $1/m^{\tilde{\Omega}(\log 1/\delta)}$-biased distribution that does not δ-fool a certain m-term read-once DNF.

In the second part of this thesis, we consider Goldreich's (ECCC 2000) proposed candidate one-way function construction which is parameterized by the choice of a small predicate (over d 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 a solution for a certain constraint satisfaction problem with a "planted solution." Such a problem easily reduces to SAT, and so the use of SAT solvers constitutes a natural class of attacks.

We initiate a rigorous study of the limitations of backtracking 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. However, the construction must use non-linear predicates; otherwise inversion succumbs to a trivial attack via Gaussian elimination.

We generalize the work of Alekhnovich et al. to handle more general classes of predicates, and we present a lower bound for the construction that uses random predicates or predicates of the form $P(x_1,\ldots,x_d) = x_1 \oplus x_2 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1},\ldots,x_d)$, and a random graph.

We also study how far Goldreich's function is from an injective function. We give upper bounds of the form $2^{2^{-\Omega(d)}n}$ on the average size of preimages of Goldreich's function when the graph G is random, and the predicate P is random or $P = x_1 \oplus x_2 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1},\ldots,x_d)$ for $d-h = \Omega(d)$.

To my parents.

Contents

List of Figures

Acknowledgments

I have greatly benefited from Mohammad Ghodsi and Ruzbeh Tusserkani who introduced me to computer science and mathematics while I was in high school, as well as Saeed Akbari and Amin Shokrollahi who helped me later during my undergraduate studies at Sharif University of Technology. I am indebted to them for inspiring my interests and for wishing me well ever since.

I am greatly indebted to Luca Trevisan, my advisor for my great years at UC Berkeley. Thanks for helping me in all the different stages of my study, guiding me through my journey, suggesting to me directions to look at, while at the same time giving me the space and freedom to be independent.

I also want to thank all my other teachers at Berkeley, especially the members of the friendly theory group. I am greatly thankful to my qualifying exam and dissertation committee members Christos Papadimitriou, Alistair Sinclair, and David Aldous who patiently listened to my ideas and offered helpful feedback. I am grateful to Elchanan Mossel for being a great advisor and collaborator, and for the many courses he taught me. Thanks to Satish Rao for guiding me through my teaching experience at Berkeley.

I would like to thank Jennifer Chayes, Christian Borgs, Mohammad Mahdian, Nicole Immorlica, and other members of the theory group at Microsoft Research for hosting me for the two valuable summers of 2005 and 2007.

I am indebted to James Cook, Rachel Miller, Anindya De, and Madhur Tulsiani for the collaborations and conversations that led to several of the results discussed in this thesis.

Thanks to each of the students in Berkeley's Electrical Engineering and Computer Science department who shared this experience with me, especially Amin Aminzadeh, Arashali Amini, James Cook, Nima Noorshams, Brighten Godfrey, Anindya De, Madhur Tulsiani, Grant Schoenebeck, Siu Man Chan, Siu On Chan, Thomas Watson, Andrej Bogdanov, Henry Lin, Alex Dimakis, Slav Petrov, Daniel Preda, Gregory Valiant, Kamalika Chaudhuri, Bonnie Kirkpatrick, Mani Narayanan, Alexandre Stauffer, Thomas Vidick, Lorenzo Orrechia, Sam Riesenfeld, Alexandra Kolla, Boriska Toth, Kunal Talwar, and James Lee. Thanks also to Aria Rafaat, Brenna Moloney, Ali Ghazizadeh, and Nazanin Shahrokni for being great friends for me during my time at Berkeley.

Last but most importantly, I am grateful to my family. I have no way to thank them for their invaluable presence in my life and their constant love and support.

Chapter 1

Introduction

This thesis considers two fundamental notions of theoretical computer science, namely pseudorandom generators and one-way functions. Pseudorandom generators are algorithms that generate a long string of bits that look random from a short string of bits that are truly random. One-way functions are functions that can be computed efficiently, but cannot be inverted efficiently.

Do there exist *simple constructions of pseduorandom generators and one-way functions*? We analyze a fairly simple class of pseudorandom generators, and show that they can, using fewer random bits compared to previous constructions, generate bits that look random to depth-2 boolean circuits. We also analyze a fairly simple candidate one-way function, and show that certain classes of backtracking algorithms cannot invert it. Both of these constructions existed before, and it is the analysis that is new.

1.1 One-Way Functions and Applications

A *one-way function* is a function f that is easy to compute, but hard to invert on average.

- The function f is often considered "easy to compute" if f can be computed in polynomial-time.
- The function f is often considered "hard to invert on average" if the probability that any polynomial-time algorithm, given the image $f(x)$ of a random x, can find a preimage of $f(x)$ is negligible; i.e. the probability diminishes faster than the inverse of any polynomial in the size of x.

One-way functions are the basic building block for many cryptographic protocols. The security of these protocols rests on the computational difficulty of inverting the function actually used for the one-way function building block. This is in accord with the general paradigm in modern cryptography of basing security on the computational limits of the adversary.

The existence of one-way functions implies the existence of many other cryptographic primitives including

- pseudorandom generators [28];
- pseudorandom function families [26];
- private-key encryption schemes secure against adaptive chosen-ciphertext attack (see for example [25, Section 5.4.4.3]);
- message authentication codes [26];
- digital signature schemes [53, 47, 35];
- bit commitment schemes [46].

The existence of one-way functions implies P $\neq$ NP. However, it is not known whether P $\neq$ NP implies existence of one-way functions. Yet, it is widely conjectured that one-way functions exist. In fact, there are various candidate functions that are proposed to be used as one-way functions. Here is a classification of some of the major candidates:

Based on number-theoretic problems : There are candidate functions whose security depends on the hardness of computational number theory problems such as factoring composite integers and finding the discrete logarithm modulo a prime number.

Involving NP-complete problems : There are candidate functions involving such NP-complete problems as the subset-sum problem or the decoding of a linear code.

Levin's universal function : Levin has constructed a function which is one-way if any one-way function exists [36].

Levin's function is theoretically interesting, but is not as practical since its running time has an exponential loss in the size of the smallest program that computes a one-way function.

The candidate functions involving NP-complete problems, though more practical, could have been theoretically greater were they based on the worst-case hardness of these NP-completeness problems, rather than based on the less certain average-case hardness of the NP-complete problems.

Finally the candidate functions based on number-theoretic problems are likely to be affected by possible breakthroughs in number theory. Indeed, the quantum algorithms for factoring composite integers and finding the discrete logarithm show that surprises within number theory are possible.

This last remark suggests that constructing one-way functions out of a simple combinatorial problem, where there is not much algebraic underlying structure as in number theory, has advantages. Oded Goldreich [24] has proposed one such simple candidate one-way function that because of simplicity is also very practical by being easy to compute.

1.1.1 Goldreich's Candidate One-Way Function

Goldreich [24] suggested in 2000 a one-way function based on bipartite expander graphs. Let G be a bipartite graph with n nodes on the left-hand side and n nodes on the right-hand side. Assume that each node on the right-hand side has degree d; that is, it is connected to d nodes on the left-hand side.

The candidate function is obtained from the graph G and a d-ary predicate $P : \{0,1\}^d \to \{0,1\}$. We call the function $f = f_{P,G}$. The function maps an n-bit string $x \in \{0,1\}^n$ to an n-bit string $f_{G,P}(x) \in \{0,1\}^n$. Each node on the left-hand side corresponds to a bit of x, and each node on the right-hand side corresponds to a bit of $f(x)$. If a right-hand side node v is connected to the left-hand side nodes $u_1, \ldots, u_d$, then $f(x)_v = P(u_1, \ldots, u_d)$; that is, the output bit associated to each right-hand side node is the evaluation of the predicate P to the input bits corresponding to the neighbors of that right-hand side node.

Each bit of Goldreich's function depends on a constant number of input bits (when $d = O(1)$). A function with this property is said to be in NC_0. Functions in NC_0 are more general in that different predicates may be used for different output bits of the function. Applebaum, Ishai and Kushilevtiz [9, 10] show that, under standard assumptions, one can construct one-way functions and pseudorandom generators that can be computed in NC_0. However, Goldreich's function is still interesting to study because of its simple construction.

It is obvious that for Goldreich's function $f_{G,P}$ to be one-way, the predicate P should be nonlinear. Goldreich suggests to choose P as some random predicate, where d is a constant, or d grows moderately like logarithmic in n. Furthermore, Goldreich suggests that the graph G be an expander; that is, any subset S of right-hand nodes has a large neighborhood, say a neighborhood proportional to the size of S. There are known constructions of expander graphs that can be used.

The natural question is:

$$\boxed{\textit{Is Goldreich's candidate function one-way?}}$$

The difficulty of inverting Goldreich's function does not seem to follow from well-known assumptions. In this thesis, we will consider the complexity of inverting Goldreich's function on its own, and show that certain classes of backtracking algorithms take exponential time to invert the function.

1.2 Lower Bounds for Satisfiability Algorithms and Proof Complexity

Inverting Goldreich's one-way function 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 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 been used to solve instances with several thousand variables. Using a random graph of right-degree 5, and the 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 very modest input lengths (a few hundred bits). See Section 6.5.

One of our goals in this thesis is to provide a rigorous justification for these experimental results, and to show that 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 fixing the variable to zero and then to the instance obtained by fixing the variable to one, or vice versa (the scheduler decides which assignment to try first). A recursive branch stops if the current partial assignment contradicts one of the constraints in the instance. The program terminates when we find a satisfying assignment.

When such an algorithm runs on an unsatisfiable instance, a transcript of the algorithm's substitutions gives a "*tree-like resolution proof*" of unsatisfiability. The resolution rule in propositional logic says that the truth of the two clauses $x_i \vee C$ and $\bar{x}_i \vee D$ implies that $C \vee D$ is also true. A resolution proof for the unsatisfiability of a SAT formula consists of beginning with all the clauses in the SAT formula, applying the resolution rule iteratively to obtain clauses that are implied from the original SAT formula, until finally the empty false clause is implied. The resolution proof is considered tree-like if each clause that has been implied using a resolution rule is later used in only one resolution rule.

A number of techniques are known to prove exponential lower bounds on the size of tree-like resolution proofs of unsatisfiability. In particular, one technique is to use the expansion properties of the constraint graph of the satisfiability formula to show that any resolution proof for the unsatisfiability of the formula will have to apply the resolution rule to a clause of large width (that is a clause with many variables). Next one applies a theorem of Ben-Sasson and Wigderson [13] that says that necessity of a clause of large width indicates large tree-like resolution proofs.

In this way, one can get lower bounds for the running time of any backtracking algorithm on unsatisfiable instances, 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 simply compute a satisfying assignment and assign the variables accordingly, giving an algorithm that converges in a linear number of steps.)

Alekhnovich, Hirsch and Itsykson [4] consider two such restrictions: they consider (i) "myopic" algorithms in which the scheduler chooses which variable to assign based on only a limited view 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. They show that after a myopic algorithm assigns a certain number of variables, with high probability it is left with an instance that is unsatisfiable, but which has no sub-exponential size tree-like resolution proof of unsatisfiability. Hence, with high probability the algorithm must take an exponential amount of time to discover it has chosen a bad partial assignment.

The result of Alekhnovich et al. for myopic algorithms is only for linear constraint satisfaction problems (and their result for drunken algorithms were proven for carefully designed instances unrelated to Goldreich's function). But as was mentioned, Goldreich's function could be a candidate one-way function only for non-linear predicates. Therefore in this thesis we consider non-linear pedicates P, and study the performance of restricted classes of backtracking algorithms in solving constraint satisfaction problems corresponding to inverting Goldreich's function for predicate P.

1.3 Pseudorandom Generators

A *pseudorandom generator* is a function that takes a short string of truly random bits, called the seed, and generates a longer string of bits that only "look random". There are five parameters related to a pseudorandom generator:

What is the seed length? One common goal is to reduce the seed length, the number of truly random bits the generator uses as input.

What is the length of the generated output? The whole point of a pseudorandom generator is to generate more output bits than the seed length.

To whom does the generated output look random? Since the generated output has entropy less than the number of generated bits, one cannot hope that the generated output looks random to a computationally unbounded observer. Therefore, each pseudorandom generator targets a specific complexity class of algorithms to which its generated output is designed to look random.

How random does the generated output look? This is determined by the probability with which an algorithm from the targeted complexity class can distinguish the generated output from truly random bits.

How computationally efficient is the generator? One can often obtain inefficient pseudorandom generators with very good seed-length using the probabilistic method. However, most applications of pseudorandom generators require that the generator be efficient and explicit.

There are at least two major applications for pseudorandom generators:

Cryptography: Assume you want to send a message using a secret key. You can feed the secret key as seed to a pseudorandom generator which outputs a pseudorandom string of the same length as the message. You can then send your message by XOR-ing it with the pseudorandom string, using the string as a one-time pad.

Derandomization and deterministic approximate counting: Consider an algorithm from the complexity class to which the pseudorandom generator is targeted. If the seed-length is very small, say logarithmic, one can iterate over all the possible seeds. This way, one can deterministically learn the behavior of the algorithm on the outputs of the generator. Since we know that the algorithm has almost the same behavior on truly random inputs, we can deterministically learn the behavior of the algorithm on random inputs. This means we can deterministically learn the behavior of a randomized algorithm (the derandomization application) or the fraction of inputs towards which the algorithm behaves in a specific way (the deterministic approximate counting application).

There are two types of constructions of pseudorandom generators: those conditioned on some complexity assumption, and those which are unconditional.

1.3.1 Conditional constructions

One important conditional construction is that of [28]: Assuming that one-way functions exist, there exist polynomial time pseudorandom generators whose output size is polynomially larger than the seed-length. The output can be distinguished from truly random outputs by the class of polynomial time algorithms only with probability that dimnishes faster than inverse polynomial in the output size. This type of pseudorandom generator is useful for cryptographic applications.

There are other constructions that get smaller seed length, and are more useful for derandomization. A series of work starting with the work of Nisan and Wigderson [51] uses hard functions to construct pseudorandom generators of logarithmic seed-length. In particular, assuming that there exist functions $f : \{0,1\}^n \to \{0,1\}^n$ computable (uniformly) in exponential time $2^{O(n)}$, but not computable by boolean circuits of size $2^{\epsilon n}$ for some constant $\epsilon > 0$, then all probabilistic polynomial-time algorithms can be derandomized [29].

It turns out that this dependency on hard functions to construct pseudorandom generators is not a coincidence. By a result of Impagliazzo and Kabanets [33], one can prove superpolynomial lower bounds on circuit size if one can derandomize the randomized polynomial time algorithm for polynomial identity testing.

1.3.2 Unconditional constructions

There has been some success in constructing pseudorandom generators with relatively small, for example polylogarithmic, seed-length whose outputs look random to the following two restricted models of computation:

Space-bounded computation: The current best polynomial-time pseudorandom generator against $O(\log n)$-spaced computation is due to Nisan [50], and has seed-length $O(\log^2 n)$.

Bounded-depth boolean circuits: Nisan, using hardness of parity for constant-depth circuits [27], gave a polynomial-time pseudorandom generator with seed of length $O(\log^{2d+6} n)$ against depth-d boolean circuits of size n [48]. The simplest case is that of depth-2 circuits, which are conjunctive normal form (CNF) formulas and disjunctive normal form (DNF) formulas. Luby, Velickovic, and Wigderson[38] improved the seed-length to $O(\log^4 n)$ for depth-2 circuits.

Next, Bazzi [12] showed that k-wise independent distributions, that is distributions on binary strings where the projection on any k bits is uniform and independent, look random to depth-2 circuits for $k = O(\log^2 n)$. This improved the seed-length for depth-2 circuits to $O(\log^3 n)$.

In this thesis, instead of k-wise independent distributions, we consider *small-bias distributions* [44], i.e. distributions on binary strings where the XOR of any nonempty subset of the bits has a small-bias (that is, is equal to 0 and 1 with almost equal probability). We ask:

How random do small-bias distributions look to depth-2 circuits?

We will show that a pseudorandom generator can use a seed of length $O(\log^{2+o(1)} n)$ to create a small-bias distribution that looks random to depth-2 circuits. We will also show that one can reduce the seed-length to logarithmic for "read-once" depth-2 circuits (i.e. depth-2 circuits where the graph of the dependencies of the wires of the circuit is a tree) if one wants to only get arbitrary small but constant distinguishing probability.

Our results are obtained by "sandwiching" the function computed by the circuit between two approximating low-weight polynomials. We use in particular results on the concentration of Fourier weight in functions computed by depth-2 circuits.

Finally, we obtain some almost tight bounds showing the limits of pseudorandomness of small-bias distributions against depth-2 circuits.

1.4 Our Results

1.4.1 Pseudorandomness against Depth-2 Circuits

An ϵ-biased distribution on n-bit strings is a distribution on n-bit strings where the XOR of any subset of the bits is 1 with probability between $1/2 - \epsilon$ and $1/2 + \epsilon$. In time $\text{poly}(n, 1/\epsilon)$ and using a seed of length $O(\log n + \log(1/\epsilon))$, one can generate an ϵ-biased n-bit string [44].

We study the pseudorandomness of small-bias distributions against depth-2 circuits. We assume without loss of generality that the depth-2 circuit is a DNF formula (instead of a CNF formula). We say that a distribution on n-bit strings δ-fools a DNF formula if the probability that the formula is satisfied on an n-bit string sampled from that distribution is at most δ different from the probability that the formula is satisfied on a uniformly random n-bit string.

We show that

- An ϵ-biased distribution δ-fools read-once DNF formulas with m terms for $\epsilon = m^{-O(\log(1/\delta))}$.

	DNF Family	Seed length
[48]	general DNFs	$O(\log^{10}(mn/\delta))$
[38]	general DNFs	$O(\log^{4}(mn/\delta))$
[12]	general DNFs	$O(\log n \cdot \log^{2}(m/\delta))$
This work	general DNFs	$O(\log n + \log^{2}(m/\delta) \cdot \log\log(m/\delta))$
[37]	width-w DNFs	$O(\log n + w2^{w} \cdot \log(1/\delta))$
This work	width-w DNFs	$O(\log n + w \log w \cdot \log(m/\delta))$
[11]	read-once DNFs	$O(\log n \cdot \log m \cdot \log(1/\delta))$
This work	read-once DNFs	$O(\log n + \log m \cdot \log(1/\delta))$

Figure 1.1: Pseudorandom generators to δ-fool DNFs with m terms and n variables

- An ϵ-biased distribution δ-fools width-w DNF formulas with m terms for $\epsilon = w^{-O(w\log(m/\delta))}$.

- An ϵ-biased distribution δ-fools general DNF formulas with m terms for $\epsilon = (\log(m/\delta))^{-O(\log^2(m/\delta))}$.

Figure 1.4.1 shows the improvements in the seed-length that we get from using, as in this thesis, ϵ-bias distributions. We also prove some lower bounds on the pseudorandomness of small-bias distributions against depth-2 circuits:

- There exists an ϵ-bias distribution with $\epsilon = m^{-O(\log(1/\delta)/\log\log(1/\delta))}$ that does not δ-fool a read-once DNF formula with m terms and $n = m\log m$ variables. (This shows that the upper bound above we had for read-once DNF formulas is almost tight.)

- There exists an ϵ-bias distribution with $\epsilon = m^{-O(\log(1/\delta))}$ that does not δ-fool a general DNF formula with m terms and $n = O(\log m \log(1/\delta))$ variables. (This shows that the upper bound above we had for general DNF formulas is almost tight when m and $1/\delta$ are polynomially related.)

1.4.2 Analysis of Goldreich's Candidate One-Way Function

We study Goldreich's function $f = f_{G,P}$ [24] derived from the bipartite graph G and predicate P. Let the bipartite graph have n left nodes and n right nodes, and have right-degree d, so that $P : \{0,1\}^d \to \{0,1\}$ is a d-ary predicate and $f : \{0,1\}^n \to \{0,1\}^n$ maps n bits to n bits.

First of all, we study the size of the preimages of Goldreich's function $f = f_{G,P}$ for random graphs G of right-degree d. We show that the function is not far from being injective. More specifically, we show that with high probability we have

$$\mathop{\mathbf{E}}_{x\sim\text{Unif}\{0,1\}^n}[|f^{-1}(f(x))|] \leq 2^{2^{-\Omega(d)}n},$$

for random predicates P and also for predicates of the form $P(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1}, \ldots, x_d)$ for $d - h = \Omega(d)$.

Next, we consider the following process. We begin with a random $x \in \{0,1\}^n$, we compute $b = f_{G,P}(x)$, and ask a backtracking algorithm to find, given b, an $x' \in \{0,1\}^n$ such that $f(x') = b$. We prove a lower bound on the running time and success probability of the backtracking algorithm if the algorithm is restricted to be (s,t)-myopic, which means the algorithm should guess the value of more than s bits of x' before it is allowed to read more than t bits of b. In other words, a myopic backtracking algorithm has restricted access to b. Our lower bound requires that

- The function $f_{G,P}$ has bounded average preimage size. (Notice that as mentioned above, we can prove this for random graphs G.)
- The predicate P is balanced. More specifically, the value of the predicate is undetermined and unbiased even after all but $h + 1$ of the arguments of the predicate are fixed. (This property is satisfied by random predicates P and predicates of the form $P(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1}, \ldots, x_d)$).
- The graph G is a very good expander, so that the graph is also a good "boundary"-expander (in the sense of Definition 6.14). (A random graph is with high probability a very good expander.)

In particular, we prove:

> If $s \geq 2^{-\Theta(d)}n$ and $t \leq n/\Theta(d)$, then an (s,t)-myopic algorithm will have the chance of inverting Goldreich's function in less than exponential time for random graphs G and random predicate P only with probability at most $2^{-s/\Theta(d)}$.

Part I

Pseudorandomness against Depth-2 Circuits

Chapter 2

Preliminaries

2.1 Boolean Fourier Analysis

We start by reviewing some basic Fourier analysis.

Definition 2.1 (Characters of $\{0,1\}^n$) *The characters of $\{0,1\}^n$ are all functions from $\{0,1\}^n$ to $\{-1,1\}$ of the form*

$$\chi_S(x) = \prod_{i \in S} (-1)^{x_i} \text{ where } S \subseteq [n].$$

It is easy to see that the following identities are true.

- For any character χ, $||\chi||_2 = \mathbb{E}_{x \in U_n}[\chi^2(x)] = 1$.
- For two distinct characters, χ and χ', $\langle \chi, \chi' \rangle = \mathbb{E}_{x \in U_n}[\chi(x)\chi'(x)] = 0$.

Note that there are 2^n characters and hence they form an orthonormal basis for the functions mapping $\{0,1\}^n$ to $\mathbb{R}$. Therefore, every function f can be expressed as a linear combination of these characters, called the Fourier expansion.

Definition 2.2 (Fourier expansion) *The Fourier expansion of $f : \{0,1\}^n \to \mathbb{R}$ is denoted by*

$$f(x) = \sum_S \hat{f}(S)\chi_S(x).$$

In the above, $\hat{f}(S)$ is called the Fourier coefficient corresponding to the set S.

It is easy to check that the following identity (known as Parseval-Plancherel identity) is true:

$$\sum_S \hat{f}^2(S) = \mathop{\mathbb{E}}_{x \in U_n}[f^2(x)]$$

Definition 2.3 (ℓ_1 norm in the Fourier domain) *We use the following notation for the Fourier ℓ_1 norm of f and a minor variant of it:*

$$\|f\|_1 \ := \ \sum_S \left|\hat{f}(S)\right| \qquad \textit{and} \qquad \|f\|_1^{\neq\emptyset} \ := \ \sum_{S\neq\emptyset} \left|\hat{f}(S)\right|$$

It is easy to observe the following properties of ℓ_1 norm of functions over the Fourier domain.

Observation 2.4 *If $f, g : \{0,1\}^n \to \mathbb{R}$, then $\|f+g\|_1 \leq \|f\|_1 + \|g\|_1$ and $\|fg\|_1 \leq \|f\|_1\|g\|_1$*

Observation 2.5 *If $\phi : \{0,1\}^n \to \{0,1\}$ is an AND of some subset of literals (i.e., variables or their negations), then $\|\phi\|_1 = 1$.*

2.2 Small-bias Distributions

First, we make the pseudorandomness of a probability distribution to a function precise:

Definition 2.6 (Functions fooled by distributions) *We say a probability distribution X over $\{0,1\}^n$ ϵ-fools a real function $f : \{0,1\}^n \to \mathbb{R}$ if*

$$|\,\mathbb{E}[f(X)] - \mathbb{E}[f(U_n)]| \leq \epsilon.$$

We will now define our pseudorandom objects of study:

Definition 2.7 (ϵ-biased distributions) *We say a probability distribution X over $\{0,1\}^n$ is ϵ-biased if it ϵ-fools the character functions χ_S.*

It is important that we can generate ϵ-biased distributions efficiently:

Proposition 2.8 (Efficient construction of ϵ-biased sets [44, 6]) *A subset $B \subseteq \{0,1\}^n$ is called an ϵ-biased set if the uniform distribution with support B is ϵ-biased. There exist ϵ-biased sets of size $O(n^2/\epsilon^2)$ such that a random element from the set can be sampled using a seed of length $2\log(n/\epsilon) + O(1)$, in time $\mathrm{poly}(n, \log(1/\epsilon))$.*

We shall study the pseudorandomness of ϵ-biased distributions to functions, particularly boolean functions computable by DNF formulas.

2.3 DNF Formula

Definition 2.9 (DNF and CNF formulas) *A* literal *is a boolean variable or the negation of a boolean variable. A* term *is the AND of several literals. A* DNF formula *is the OR of several terms. A* clause *is the OR of several literals. A* CNF formula *is the AND of several clauses.*

Any depth-2 boolean circuit is a DNF or CNF formula. Since a CNF formula is the negation of a DNF formula, we assume without loss of generality, that we want to fool a DNF formula.

We will be interested in two special types of DNF forumals:

Definition 2.10 *A* read-once *DNF formula is a DNF formula where every variable appears in at most one term.*

A width-w *DNF formula is a DNF formula where every term has at most w literals.*

2.4 Sandwiching

In this section, we state a characterization of functions that can be fooled well by ϵ-biased probability distributions. Bazzi [12] independently derived both this characterization and the characterization of functions that can be fooled by k-wise independent distributions.

The first observation is that if f has a small Fourier ℓ_1 norm, then it is fooled by small ϵ-biased sets:

Lemma 2.11 *Every function $f : \{0,1\}^n \to \mathbb{R}$ is $\epsilon\|f\|_1^{\neq\emptyset}$-fooled by any ϵ-biased probability distribution.*

Proof: Let X be sampled from an ϵ-biased distribution. We have

$$\begin{aligned}
|\mathbb{E}[f(X)] - \mathbb{E}[f(U_n)]| &= \left|\mathbb{E}\left[\sum_S \hat{f}(S)\chi_S(X)\right] - \hat{f}(\emptyset)\right| \\
&= \left|\sum_{S\neq\emptyset} \hat{f}(S)\,\mathbb{E}[\chi_S(X)]\right| \\
&\leq \epsilon \sum_{S\neq\emptyset} |\hat{f}(S)| \;=\; \epsilon\|f\|_1^{\neq\emptyset}.
\end{aligned}$$

■

We can strengthen Lemma 2.11 as follows.

Proposition 2.12 (Sandwich bound) *Suppose $f, f_\ell, f_u : \{0,1\}^n \to \mathbb{R}$ are three functions such that for every $x \in \{0,1\}^n$ we have $f_\ell(x) \leq f(x) \leq f_u(x)$. Furthermore, assume $\mathbb{E}[f(U_n)] - \mathbb{E}[f_\ell(U_n)] \leq \delta$ and $\mathbb{E}[f_u(U_n)] - \mathbb{E}[f(U_n)] \leq \delta$. Let $l = \max(\|f_\ell(x)\|_1^{\neq\emptyset}, \|f_u(x)\|_1^{\neq\emptyset})$. Then any ϵ-biased probability distribution $(\delta + \epsilon l)$-fools f.*

Proof: Let X be an ϵ-biased random variable. We have

$$\begin{aligned}
\mathbb{E}[f(X)] &\leq \mathbb{E}[f_u(X)] \\
&\leq \mathbb{E}[f_u(U_n)] + \epsilon\|f_u\|_1^{\neq\emptyset} \\
&\leq \mathbb{E}[f(U_n)] + \delta + \epsilon\|f_u\|_1^{\neq\emptyset}.
\end{aligned}$$

Similarly we have $\mathbb{E}[f(X)] \geq \mathbb{E}[f(U_n)] - \delta - \epsilon\|f_\ell\|_1^{\neq\emptyset}$. Thus the result follows. ■

The following result shows that the condition of Proposition 2.12 is not only a sufficient condition for being fooled by ϵ-biased distributions but also a necessary condition.

Proposition 2.13 (Inverse of the sandwich bound) *Suppose $f : \{0,1\}^n \to \mathbb{R}$ is ϵ'-fooled by any ϵ-biased set. Then there exist functions $f_\ell, f_u : \{0,1\}^n \to \mathbb{R}$ and $\delta, l \in \mathbb{R} \geq 0$ with the following properties:*

- *For every $x \in \{0,1\}^n$ we have $f_\ell(x) \leq f(x) \leq f_u(x)$.*
- *$\mathbb{E}[f(x)] - \mathbb{E}[f_\ell(x)] \leq \delta$ and $\mathbb{E}[f_u(x)] - \mathbb{E}[f(x)] \leq \delta$,*
- *$\|f_\ell(x)\|_1^{\neq\emptyset} \leq l$, $\|f_u(x)\|_1^{\neq\emptyset} \leq l$, and $\delta + \epsilon l \leq \epsilon'$.*

Proof: Consider the following linear program in variables p_x:

$$\begin{array}{ll} \min & \sum_x f(x)p_x \\ & \sum_x p_x = 1 \\ \forall S \neq \emptyset & \sum_x p_x \chi_S(x) \geq -\epsilon \\ \forall S \neq \emptyset & \sum_x p_x \chi_S(x) \leq \epsilon \\ \forall x & p_x \geq 0 \end{array}$$

where $x \in \{0,1\}^n$ and $S \subseteq \{1,\ldots,n\}$. The constraints specify that p_x is the probability distribution of an ϵ-biased random variable. Since f is ϵ'-fooled by ϵ-biased sets, the optimum value of the LP is $\geq \mathbb{E}[f(U_n)] - \epsilon'$.

We now write the dual of the above LP:

$$\begin{array}{ll} \max & z - \epsilon \sum_{S\neq\emptyset}(y_S^+ + y_S^-) \\ \forall x & z + \sum_{S\neq\emptyset} \chi_S(x)(y_S^+ - y_S^-) \leq f(x) \\ \forall S \neq \emptyset & y_S^+, y_S^- \geq 0 \end{array}$$

which is equivalent to

$$\begin{array}{ll} \max & z - \epsilon \sum_{S\neq\emptyset} |y_S| \\ \forall x & z + \sum_{S\neq\emptyset} \chi_S(x) y_S \leq f(x) \end{array}$$

Since the optimum value of the primal is $\geq \mathbb{E}[f(U_n)] - \epsilon'$, there exists a feasible set of values z^* and y_S^* for the above optimization program such that $z^* - \epsilon \sum_{S\neq\emptyset} |y_S^*| \geq \mathbb{E}[f(U_n)] - \epsilon'$. Let $f_\ell(x) = z^* + \sum_{S\neq\emptyset} y_S^* \chi_S(x)$. Clearly $\mathbb{E}[f_\ell(U_n)] = z^*$ and $\sum_{S\neq\emptyset} |y_S^*| = \|f_\ell\|_1^{\neq\emptyset}$; Set $\delta = \mathbb{E}[f(U_n)] - z^*$ and $l = \sum_{S\neq\emptyset} |y_S^*|$. It is easy to check that f_ℓ, δ, l so defined satisfies all the constraints. Similarly, one can consider a different primal where the objective is to maximize $\sum_x f(x)p_x$ and then use its dual to define f_u which satisfies the aforementioned conditions. ∎

Chapter 3

Fooling DNF Formulas using Small-Bias Distributions

3.1 Fooling Read-Once DNF Formulas

In this section, we show that ϵ-biased sets can fool read-once DNFs. In particular, we show the following theorem.

Theorem 3.1 *Let ϕ be a read-once DNF formula with m terms. For $1 \leq k \leq m$, ϵ-biased distributions $O(2^{-\Omega(k)} + \epsilon m^k)$-fool ϕ. In particular, we can δ-fool ϕ by an ϵ-biased distribution, for $\epsilon = m^{-O(\log(1/\delta))}$.*

If we plug in the construction from Proposition 2.8, we get a pseudorandom generator which δ-fools a read-once DNF with n variables and m terms and has seed length $O(\log n + \log m \cdot \log(1/\delta))$. Before going into the proof of Theorem 3.1, we recall the inclusion-exclusion principle.

Let $A_1, \ldots, A_m$ be m arbitrary events in a probability space. The principle of inclusion and exclusion asserts that

$$\Pr[A_1 \cup \cdots \cup A_m] = \sum_{j=1}^{m} (-1)^{j-1} T_j,$$

where

$$T_j = \sum_{S \subseteq [m], |S|=j} \Pr\left[\bigcap_{i \in S} A_i\right].$$

Moreover, the partial sum $\sum_{j=1}^{r} (-1)^{j-1} T_j$ is an upper bound for $\Pr[A_1 \cup \cdots \cup A_m]$ for odd values of r, and a lower bound for $\Pr[A_1 \cup \cdots \cup A_m]$ for even values of r.

We now return to the proof of Theorem 3.1. The proof follows that of Theorem 2 in [23].

Proof: [of Theorem 3.1] Let $\phi = C_1 \vee \cdots \vee C_m$ be the read-once formula. For $1 \leq i \leq m$, let A_i denote the event that term C_i is satisfied. We divide the analysis into two cases depending on whether $\sum_{i=1}^{m} \Pr[A_i] \leq k/(2\mathrm{e})$ or not.

<u>Case 1</u>: $\sum_{i=1}^{m} \Pr[A_i] \leq k/(2\mathrm{e})$.

Let T_k denote the kth term of the inclusion-exclusion formula. Since the terms are disjoint, we have

$$T_k = \sum_{S \subseteq [m], |S|=k} \prod_{i \in S} \Pr[A_i].$$

We now observe that $T_k \leq 2^{-k}$. Indeed, subject to the restriction $\sum_{i=1}^{m} \Pr[A_i] = \alpha$ and $\Pr[A_i] \geq 0$, a convexity based argument implies that T_k is maximized when all the $\Pr[A_i]$'s are equal implying that $T_k \leq \binom{m}{k}(2\mathrm{e}m/k)^{-k} \leq 2^{-k}$.

Consider the rth approximation to ϕ, obtained by inclusion-exclusion:

$$\phi_r(x) = \sum_{j=1}^{r} (-1)^{j-1} \sum_{S \subseteq [m], |S|=j} \bigwedge_{l \in S} C_l(x),$$

where $\bigwedge$ is the AND function. The functions ϕ_{k-1} and ϕ_k sandwich ϕ and we shall use them in applying Proposition 2.12. To verify the conditions, we note that the function $\bigwedge_{l \in S} C_l(x)$ is an AND of AND terms, therefore $\| \bigwedge_{l \in S} C_l(x) \|_1^{\neq \emptyset} = O(1)$, and hence $\|\phi_r\|_1^{\neq \emptyset} = O(m^r)$. We also have $|\mathbb{E}[f_k(U_n)] - \mathbb{E}[f_{k-1}(U_n)]| = T_k \leq 2^{-k}$. and hence, by Proposition 2.12, ϕ is $O(2^{-k} + \epsilon m^k)$-fooled by ϵ-biased distributions.

<u>Case 2</u>: $\sum_{i=1}^{m} \Pr[A_i] > k/(2\mathrm{e})$.

Consider the first m' where $\sum_{i=1}^{m'} \Pr[A_i] \geq k/(2\mathrm{e})$. Define $\phi' = C_1 \vee \cdots \vee C_{m'}$. Observe that the DNF ϕ' is satisfied with probability $1 - 2^{-\Omega(k)}$, for it is not satisfied with probability $\prod_{i=1}^{m'}(1 - \Pr[A_i]) \leq (1 - k/(2\mathrm{e}m'))^{m'} \leq 2^{-\Omega(k)}$. (Again by a convexity argument, $\prod_i (1 - \Pr[A_i])$ is maximized when $\Pr[A_i]$s are equal.)

Let $\phi'_r(x)$ denote the rth approximation to ϕ'. Also, (without loss of generality) let k be even so that $\phi'_k \leq \phi' \leq \phi$. Note that while ϕ'_{k-1} is a an upper bound on ϕ', it is *not* an upper bound on ϕ. We shall use ϕ'_k and identically 1 function respectively as lower and upper bounds for applying Proposition 2.12 to ϕ.

From argument above, we know that $\mathbb{E}[1 - \phi] \leq \mathbb{E}[1 - \phi'] \leq 2^{-\Omega(k)}$. To bound $\mathbb{E}[\phi - \phi'_k]$, we note that

$$\mathbb{E}\left[\phi - \phi'_k\right] \;=\; \mathbb{E}\left[\phi - \phi'\right] + \mathbb{E}\left[\phi' - \phi'_k\right] \;\leq\; \mathbb{E}\left[1 - \phi'\right] + \mathbb{E}\left[\phi'_{k-1} - \phi'_k\right] \;\leq\; 2^{-\Omega(k)}$$

where in the last inequality we used that $\mathbb{E}[\phi'_{k-1} - \phi'_k]$ as in the previous case, since $\sum_{i=1}^{m'} \Pr[A_i] < k/(2\mathrm{e}) + 1$. The bound on the $\|\phi'_k\|_1^{\neq \emptyset}$ is as before. Applying Proposition 2.12, we then get that ϵ-biased sets $O(2^{-\Omega(k)} + \epsilon m'^k)$-fool ϕ. ■

3.2 Fooling General DNF Formulas

In this section, we show that small-bias distributions fool general DNFs. While the seed length will not be as good as in the previous section, the result will be more general. Also, this section will involve use of more analytic tools. Our proof shall be along the lines of Razborov's simplified proof [52] of Bazzi's theorem [12]. The following two theorems will be the main theorems of this section.

Theorem 3.2 *Let ϕ be a width w-DNF formula with m terms. Then, ϕ is δ-fooled by an ϵ-biased distribution where $\epsilon = w^{-O(w\log(m/\delta))}$.*

Theorem 3.3 *Let ϕ be a DNF formula with m terms. Then, ϕ is δ-fooled by an ϵ-biased distribution where $\epsilon = (\log(m/\delta))^{O(-\log^2(m/\delta))}$.*

Plugging in the pseudorandom generator construction from Proposition 2.8 in Theorem 3.2, we get a pseudorandom generator which δ-fools width-w DNFs with m terms over n variables and has a seed of length $O(\log n + w\log w\log(m/\delta))$. Doing the same for Theorem 3.3, we get a pseudorandom generator which δ-fools DNFs with m terms over n variables and has a seed of length $O(\log n + \log^2(m/\delta)\log\log(m/\delta))$.

Theorem 3.3 follows by a reduction to Theorem 3.2, by deleting the terms with large width, as we describe later. For most of this section, we will be concerned with DNFs of a bounded width. To prove Theorem 3.2, we will be interested in finding sandwiching functions f_l and f_u to apply Proposition 2.12.

Using an argument similar to [12], we reduce this to the problem of finding a function g such that $\|\phi - g\|_2$ and $\|g\|_1$ are small, and $\phi(x) = 0 \implies g(x) = 0$. We then show how to remove the last condition and then find an appropriate g using a Fourier concentration result of Mansour [40]. More formally, we prove the following three lemmas.

Lemma 3.4 *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a DNF with m terms and $g : \{0,1\}^n \to \mathbb{R}$ be such that: $\|g\|_1 \le l$, $||\phi - g||_2 \le \epsilon_1$ and $g(x) = 0$ whenever $\phi(x) = 0$. Then, we can get $f_\ell, f_u : \{0,1\}^n \to \mathbb{R}$ such that*

- $\forall\ x,\ f_\ell(x) \le \phi(x) \le f_u(x)$
- $\mathbb{E}_{x\in U_n}[f_u(x) - \phi(x)] \le m\epsilon_1^2$ *and* $\mathbb{E}_{x\in U_n}[\phi(x) - f_\ell(x)] \le m\epsilon_1^2$.
- $\|f_\ell\|_1,\ \|f_u\|_1 \le (m+1)(l+1)^2 + 1$

Lemma 3.5 *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a width-w DNF with m terms. Suppose for every width-w DNF ϕ_1, there is a function $g_1 : \{0,1\}^n \to \mathbb{R}$ such that: $\|g_1\|_1 \le l_1$ and $||\phi_1 - g_1||_2 \le \epsilon_2$. Then, we can get $g : \{0,1\}^n \to \mathbb{R}$ such that $\|g\|_1 \le m(l_1+1)$, $||\phi - g||_2 \le m\epsilon_2$ and $g(x) = 0$ whenever $\phi(x) = 0$.*

Lemma 3.6 *Let $\phi : \{0,1\}^n \to \{0,1\}$ be a width w DNF and $\epsilon_2 > 0$. Then there is a function $g_1 : \{0,1\}^n \to \mathbb{R}$ such that $||\phi - g_1||_2 \le \epsilon_2$ and $\|g_1\|_1 = w^{O(w\log(1/\epsilon_2))}$*

Before, we prove these lemmas, we show how it implies Theorem 3.2.

Proof: [of Theorem 3.2] Set $\epsilon_2 = \sqrt{\delta/2m^3}$ and $\epsilon_1 = \sqrt{\delta/2m}$. By applying Lemma 3.6, for every width-w DNF ϕ_1, we can get a function $g_1 : \{0,1\}^n \to \mathbb{R}$ such that

- $||\phi_1 - g_1||_2 \le \epsilon_2 = \sqrt{\delta/2m^3}$
- $||g_1||_1 = w^{O(w\log(1/\epsilon_2))} = w^{O(w\log(m/\delta))}$

Now, we apply Lemma 3.5 with $l_1 = w^{O(w \log(m/\delta))}$ and $\epsilon_2 = \sqrt{\delta/2m^3}$. Then, for the given DNF ϕ, we get a function g such that $||g||_1 = w^{O(w \log(m/\delta))}$ and $||g - \phi||_2 \leq m\epsilon_2 = \epsilon_1 = \sqrt{\delta/2m}$. Finally, we apply Lemma 3.4 with g and ϵ_1 as defined and $l = w^{O(w \log(m/\delta))}$ to get f_ℓ and f_u such that ϕ is sandwiched by f_ℓ and f_u, $||f_\ell||_1, ||f_u||_1 \leq w^{O(w \log(m/\delta))}$ and

$$\mathop{\mathbb{E}}_{x \in U_n}[f_u(x) - \phi(x)] \leq \frac{\delta}{2} \text{ and } \mathop{\mathbb{E}}_{x \in U_n}[\phi(x) - f_\ell(x)] \leq \frac{\delta}{2}$$

By applying Proposition 2.12, we get that an $\epsilon = w^{-O(w \log(m/\delta))}$ (for an appropriately large constant inside $O(\cdot)$) biased set fools ϕ by $\delta/2 + \epsilon l \leq \delta$. ■

We now get back to proofs of Lemma 3.4, Lemma 3.5 and Lemma 3.6. We start with proof of Lemma 3.4.

Proof: [of Lemma 3.4] Let $\phi = \bigvee_{i=1}^m A_i$ where A_i are the terms. We define f_ℓ and f_u as follows:

- $f_\ell = 1 - (1 - g)^2$
- $f_u = 1 - (1 - \sum_{i=1}^m A_i)(1 - g)^2$

We note that this is the same construction of functions as in Lemma 3.3 in [12]. In particular, the following two things are already proven there.

- $\forall\ x,\ f_\ell(x) \leq \phi \leq f_u(x)$
- $\mathbb{E}_{x \in U_n}[f_u(x) - \phi(x)] \leq m||\phi - g||_2^2$ and $\mathbb{E}_{x \in U_n}[\phi(x) - f_\ell(x)] \leq m||\phi - g||_2^2$

Using this, we have the proof of the first two items in the lemma. Only the third item *i.e.,* bound on $\|f_\ell\|_1$ and $\|f_u\|_1$ remains to be proven. To get this, we use Observation 2.4 and Observation 2.5 along with the hypothesis $\|g\|_1 \leq l$. Using this, we get that $\|f_\ell\|_1 \leq 1 + (1 + l)^2$ and $\|f_u\|_1 \leq 1 + (m + 1)(l + 1)^2$ which proves the lemma. ■

We now turn to the proof of Lemma 3.5. The proof follows the proof by Razborov [52] with some changes.

Proof: [of Lemma 3.5] We first observe as in [52] (attributed to Avi Wigderson) that if $\phi = \bigvee_{i=1}^m A_i$ where A_i are the individual terms, then ϕ can be rewritten as $\sum_{i=1}^m A_i(1 - \bigvee_{j=1}^{i-1} A_j)$. Let us write $\bigvee_{j=1}^{i-1} A_j = \phi_i$ ($\phi_i = 0$ if $i = 1$). Then, we can say that $\phi = \sum_{i=1}^m A_i(1 - \phi_i)$. Note that each of the ϕ_i is a width w-DNF. Hence, we can apply our hypothesis to get functions $g_1, \ldots, g_m : \{0,1\}^n \to \mathbb{R}$ such that for all i, $\|g_i\|_1 \leq l_1$ and $\|g_i - \phi_i\|_2 \leq \epsilon_2$. Let us now consider the function $g : \{0,1\}^n \to \mathbb{R}$ defined as

$$g = \sum_{i=1}^m A_i(1 - g_i)$$

We observe that if $\phi(x) = 0$ for some x, then $\forall\ i$, $A_i(x) = 0$ which implies that $g(x) = 0$. Applying Observation 2.4 and using that A_i's are terms and hence $||A_i||_1 = 1$, we also get that $\|g\|_1 \leq m(l_1+1)$. So, the only thing that remains to be proven is that $\|\phi - g\|_2 \leq m\epsilon_2$. Though this is done in [52], we do it here for the sake of completeness.

$$\begin{aligned}
\|g-\phi\|_2^2 &= \mathop{\mathbb{E}}_{x\in U_n}\left[\left(\sum_{i=1}^{m} A_i(\phi_i - g_i)(x)\right)^2\right] \\
&\leq m \mathop{\mathbb{E}}_{x\in U_n}\left[\sum_{i=1}^{m}(A_i(\phi_i - g_i)(x))^2\right] \qquad \text{(By Jensen's inequality)} \\
&= m\sum_{i=1}^{m} \mathop{\mathbb{E}}_{x\in U_n}\left[(A_i(\phi_i - g_i)(x))^2\right] \\
&\leq m\sum_{i=1}^{m} \mathop{\mathbb{E}}_{x\in U_n}\left[(\phi_i - g_i)(x)^2\right] \qquad \text{(Using } A_i \text{ is bounded by 1)} \\
&= m\sum_{i=1}^{m} ||\phi_i - g_i||_2^2 \;\leq\; m^2\epsilon_2^2 \qquad \text{(Using } ||\phi_i - g_i||_2 \leq \epsilon_2\text{)}
\end{aligned}$$

This proves that $||\phi - g||_2 \leq m\epsilon_2$ which finishes the proof. ■

We now come to the proof of Lemma 3.6. The proof is dependent upon the following well-known concentration result by Mansour [40].

Theorem 3.7 *[40] Let $\phi : \{0,1\}^n \to \{0,1\}$ be a width w-DNF with m terms and $\epsilon_2 > 0$. Let $\sum_{S\subset[n]} \hat{\phi}(S)\chi_S$ be the Fourier expansion of ϕ. Then there is a subset $\Gamma \subset 2^{[n]}$ of size $w^{O(w\log(1/\epsilon_2))}$ such that g defined as $g_1 = \sum_{S\in\Gamma} \hat{\phi}(S)\chi_S$ is such that $||\phi - g_1||_2 \leq \epsilon_2$.*

Proof: [of Lemma 3.6] For the given ϕ and ϵ_2, let g_1 be the function given by Theorem 3.7. Clearly, it satisfies $||\phi - g_1||_2 \leq \epsilon_2$. To bound $\|g_1\|_1$, note that $\|g_1\|_1 = \sum_{S\in\Gamma} |\hat{\phi}(S)|$ where $|\Gamma| = w^{O(w\log(1/\epsilon_2))}$. Note that $\sum_{S\in\Gamma} |\hat{\phi}(S)|^2 = \alpha$ for some $\alpha \in [0,1]$ (by Parseval-Plancherel identity and the fact that ϕ lies in $[0,1]$). Now, we have

$$\left(\sum_{S\in\Gamma} |\hat{\phi}(S)|\right)^2 \leq |\Gamma| \left(\sum_{S\in\Gamma} |\hat{\phi}(S)|^2\right) \leq |\Gamma| \qquad \text{(By Jensen's inequality)}$$

Hence, this gives us $\sum_{S\in\Gamma} |\hat{\phi}(S)| \leq \sqrt{|\Gamma|} = w^{O(w\log(1/\epsilon_2))}$ which proves the lemma. ■

Theorem 3.3 now follows by reducing the case of arbitrary DNFs to that of bounded width, by deleting the terms with width greater than $\log(m/2\delta)$ and arguing that the change in the distinguishing probability is small.

Proof: [of Theorem 3.3] Let ϕ_w be the DNF obtained by removing all the terms from ϕ which have more than w literals, for a value of w to be specified later. Note that $\forall\ x$, $\phi_w(x) \leq \phi(x)$. Also, note that

$$\mathop{\mathbb{E}}_{x\in U_n}[\phi(x) - \phi_w(x)] \leq \Pr_{x\in U_n}[\exists \text{ term present in } \phi \text{ but not in } \phi_w \text{ which is satisfied}] \leq m2^{-w}$$

The last inequality uses that all the terms present in ϕ but not ϕ_w have more than w literals and hence are satisfied with probability at most 2^{-w} under the uniform distribution. Also, let D be any ϵ-biased distribution. We can again say that

$$\mathop{\mathbb{E}}_{x\in D}[\phi(x)-\phi_w(x)] \leq \Pr_{x\in D}[\exists \text{ term present in } \phi \text{ but not in } \phi_w \text{ which is satisfied}] \leq m(2^{-w}+\epsilon)$$

The last inequality uses that under a ϵ-biased distribution, a term of width-w is satisfied with probability at most $2^{-w}+\epsilon$. This is because a term has ℓ_1 norm 1 and hence is ϵ fooled by a ϵ-biased distribution. Using the above two inequalities as well as $\phi_w \leq \phi$, we can say

$$\mathop{\mathbb{E}}_{x\in D}\phi(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi(x) \geq \mathop{\mathbb{E}}_{x\in D}\phi_w(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi(x) \geq \mathop{\mathbb{E}}_{x\in D}\phi_w(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi_w(x) - m2^{-w}$$

$$\mathop{\mathbb{E}}_{x\in D}\phi(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi(x) \leq \mathop{\mathbb{E}}_{x\in D}\phi(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi_w(x) \leq \mathop{\mathbb{E}}_{x\in D}\phi_w(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi_w(x) + m(\epsilon+2^{-w})$$

which together imply that

$$|\mathop{\mathbb{E}}_{x\in D}\phi(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi(x)| \leq |\mathop{\mathbb{E}}_{x\in D}\phi_w(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi_w(x)| + m(\epsilon+2^{-w})$$

Let us put $w = \log(2m/\delta)$. Then, Theorem 3.2 says that $|\mathbb{E}_{x\in D}\phi_w(x) - \mathbb{E}_{x\in U_n}\phi_w(x)|$ is $\delta/4$ fooled by an ϵ biased distribution where $\epsilon = w^{-O(w\log(m/\delta))} = (\log(m/\delta))^{-O(\log^2(m/\delta))}$. Then,

$$|\mathop{\mathbb{E}}_{x\in D}\phi(x) - \mathop{\mathbb{E}}_{x\in U_n}\phi(x)| \leq \frac{\delta}{4} + m(\epsilon+2^{-w}) \leq \frac{\delta}{4} + \frac{\delta}{2} + m(\log(m/\delta))^{-O(\log^2(m/\delta))} \leq \delta$$

■

Chapter 4

Limitations of Small-Bias Distributions

In this section we provide various lower bounds on fooling DNFs by ϵ-biased distributions. Recall that in Section 3.1, we showed that a bias less than $m^{-O(\log(1/\delta))}$ is sufficient to δ-fool a read-once DNF with m terms. We first give a simple example which shows that this bound is optimal when δ is a small constant.

For smaller values of δ, we give a somewhat more technical construction, which shows that the bias needs to be less than $m^{-\Omega(\log(1/\delta)/\log\log(1/\delta))}$ to δ-fool a read-once DNF with m terms. Note that this would also imply the optimality for constant δ but we choose to retain the previous example due to its simplicity.

For the case of general DNFs, we give an instance showing that ϵ must be necessarily less than $m^{-\Omega(\log(1/\delta))}$. This does match our bound for the case of read-once DNFs, but is somewhat far from the upper bound we provide in Section 3.2 (which uses $\epsilon = (\log(m/\delta))^{-O(\log^2(m/\delta))}$).

4.1 Lower Bounds for Read-Once DNF for Constant Error Probability

Our analysis gives that for $\delta = \Theta(1)$ and $m = n^{\Theta(1)}$, an ϵ-biased distribution with $\epsilon = n^{-\Theta(1)}$ suffices to δ-fool a read-once DNF with m terms. The following theorem shows this tradeoff is optimal.

Theorem 4.1 *There is read-once DNF $\phi : \{0,1\}^n \to \{0,1\}$ with $\Theta(n/\log n)$ terms and an ϵ-biased distribution D over $\{0,1\}^n$ where $\epsilon = n^{-\Theta(1)}$ such that*

$$|\Pr_{x\in U_n}[\phi(x) = 1] - \Pr_{x\in D}[\phi(x) = 1]| = \Omega(1)$$

Proof: Let t be an integer such that $t \equiv 2(mod\ 4)$ and for $x \in \{0,1\}^t$, define the inner product

$$IP(x) = \left(\sum_{i=1}^{t/2} x_i x_{t/2+i}\right) \quad (mod\ 2)$$

Define distribution D over $\{0,1\}^{t+1}$ as follows. It is a uniform distribution on $x \circ IP(x)$ for $x \in \{0,1\}^t$. The following fact is easy to verify.

Fact 4.2 *For all subsets $S \subset [t]$, $\chi_S : \{0,1\}^t \to \{-1,1\}$,*

$$\left| \mathop{\mathbb{E}}_{x \in U_t} \left[\chi_S(x)(-1)^{IP(x)}\right] \right| \;=\; 2^{-t/2}$$

Claim 4.3 *D is $2^{-\Omega(t)}$ biased distribution over $\{0,1\}^{t+1}$.*

Proof: Consider any character $\chi_S : \{0,1\}^{t+1} \to \{0,1\}$. In case, $(t+1) \notin S$, then clearly $\mathbb{E}_{x \in D}[\chi_S(x)] = 0$. If $(t+1) \in S$, then let $S' = S \backslash \{t+1\}$

$$\left| \mathop{\mathbb{E}}_{x \in D}[\chi_S(x)] \right| \;=\; \left| \mathop{\mathbb{E}}_{x' \in U_t}[\chi_{S'}(x')(-1)^{IP(x')}] \right| \;=\; 2^{-\Omega(t)}$$

This implies that D is $2^{-\Omega(t)}$ biased distribution over $\{0,1\}^{t+1}$. ■

Let $n = (t+1)2^t$ for $t \equiv 2(mod\ 4)$. Split $\{0,1\}^n$ into 2^t chunks. Let the variables in the i^{th} chunk be $y_{i,1}, \ldots, y_{i,t+1}$. Let $D_1, \ldots, D_{2^t}$ be 2^t independent copies of D such that D_i is over $y_{i,1}, \ldots, y_{i,t+1}$. Let D' defined over $\{0,1\}^n$ be the product distribution of $D_1, \ldots, D_{2^t}$. Clearly, D' is a $2^{-\Omega(t)}$ biased distribution. Now, consider the read-once DNF ϕ defined as

$$\phi = \bigvee_{i=1}^{2^t} \left(\bigwedge_{j=1}^{t+1} y_{i,j}\right)$$

Under the uniform distribution, each term is satisfied with probability $1/2^{t+1}$ while note that under D', each term is satisfied with probability $1/2^t$. This is because once the first t variables in a term are 1, the $t+1^{th}$ variable is 1 in D as $t \equiv 2(mod\ 4)$. As the terms are over disjoint sets of variables, hence we can say that

$$\left| \Pr_{y \in D}[\phi(y) = 0] - \Pr_{y \in U}[\phi(y) = 0] \right| = \left| \left(1 - \frac{1}{2^t}\right)^{2^t} - \left(1 - \frac{1}{2^{t+1}}\right)^{2^t} \right| = \Omega(1)$$

This proves the theorem. ■

4.2 Lower Bounds for Read-Once DNF for Sub-constant Error Probability

The obvious scaling of the previous example would give $\epsilon = 2^{-\Omega(\log m + \log\log(1/\delta))}$. Here we give a construction of a specific ϵ-biased distribution which shows that to δ-fool the "tribes" DNF (described below), one must have $\epsilon = m^{-\Omega(\log(1/\delta)/\log\log(1/\delta))}$. We first state the more general form of the theorem claiming the existence of such a DNF and a distribution and as a subsequent corollary, we get the bias in terms of the distinguishing probability.

Theorem 4.4 *For every sufficiently large integer n of the form $n = m \log m$ for m which is power of 2 and for every integer $d \geq 1$, there is an $(m/2)^{-d}$-biased distribution D over $\{0,1\}^n$ and a read-once DNF ϕ with m terms such that ϕ distinguishes D from uniform by at least* $1/(2d+3)!$.

Proof: We first describe the DNF. The DNF is defined by splitting the n variables into m chunks of size $\log m$. Let the variables in the i^{th} chunk be $x_{i,1}, \ldots, x_{i,\log m}$. The DNF is

$$\phi(x) = \bigvee_{i=1}^{m} C_i \quad \text{where } C_i \equiv \bigwedge_{j=1}^{\log m} x_{i,j}$$

The following two claims, describe the required distribution D.

Claim 4.5 *There is a distribution $Y = Y_1 \circ \ldots \circ Y_m$ over $\{0,1\}^m$ with the following properties*

- *for every $1 \leq i \leq m$, $\Pr[Y_i = 1] = 1/m$.*
- *$Y_1, \ldots, Y_m$ are d-wise independent;*
- *For every $y \in Supp(Y)$, $y_1 + \ldots + y_m \leq d$.*

We can now describe the distribution D in terms of the random variables $Y_1, \ldots, Y_m$. Given values $y_1, \ldots, y_m$, we choose $x_{i,1}, \ldots, x_{i,\log m}$ to be all 1, if $y_i = 1$ and uniformly from $\{0,1\}^{\log m} \setminus 1^{\log m}$ if $y_i = 0$. In particular, this ensures that $\bigwedge_{j=1}^{\log m} x_{i,j} = y_i$ and hence C_i is satisfied if and only if $y_i = 1$. We claim that the distribution has a small bias.

Claim 4.6 *The distribution D defined above has bias at most $(m/2)^{-d}$.*

Before proving these two claims, lets see why they suffice to construct the counterexample. First, observe that by Claim 4.6, term C_i being satisfied is equivalent to $y_i = 1$. By inclusion-exclusion principle, the probability that $x \in_r D$ satisfies ϕ is

$$\begin{aligned}\Pr_{x\in D}[\phi \text{ is satisfied}] &= \sum_{S\in[m],|S|>0} (-1)^{|S|-1}\Pr[\forall i \in S,\ C_i \text{ is satisfied}] \\ &= \sum_{S\in[m],|S|>0} (-1)^{|S|-1}\Pr[\forall i \in S, y_i = 1] \\ &= \sum_{S\in[m],d\geq|S|>0} (-1)^{|S|}\Pr[\forall i \in S,\ y_i = 1] \qquad (\text{Using } \sum_i^m y_i \leq d) \\ &= \sum_{t=1}^{d}(-1)^{t-1}\binom{m}{t}\frac{1}{m^t}\end{aligned}$$

The last equality uses that y_i's are d-wise independent and $\Pr[y_i = 1] = 1/m$. To estimate the above probability for the uniform distribution, we can obtain upper and lower bounds on it by truncating the inclusion-exclusion respectively at $d+1$ and $d+2$ when d is even (the upper and lower bounds are switched when d is odd). Thus ϕ distinguishes D from uniform with probability at least

$$\begin{aligned}\binom{m}{d+1}\frac{1}{m^{d+1}} - \binom{m}{d+2}\frac{1}{m^{d+2}} &= \frac{m!}{m^{d+1}(d+1)!(m-d-2)!}\left(\frac{1}{m-d-1} - \frac{1}{m(d+2)}\right) \\ &\geq \frac{m!}{m^{d+1}(d+1)!(m-d-2)!}\frac{1}{2m} \\ &\geq \frac{1}{2(d+1)!}\prod_{i=1}^{d+1}\left(1-\frac{i}{m}\right) \\ &= \frac{1}{2(2d+2)!}\prod_{i=1}^{d+1}\left((d+1+i)\left(1-\frac{i}{m}\right)\right) \geq \frac{1}{(2d+3)!}\end{aligned}$$

The last inequality uses that $(d+1+i)(1-i/m) \geq 1$. Hence, we need to prove Claims 4.5 and 4.6. We start with Claim 4.5.

Proof: [of Claim 4.5] Let $p_0, \ldots, p_d \geq 0$ such that $\sum p_i = 1$ (We will non-constructively describe p_i's later). The distribution Y is chosen as following. Pick i, $0 \leq i \leq d$ with probability p_i. Choose a uniformly random subset $S \subset [m]$ of size d and set $y_i = 1$ if $i \in S$ and $y_i = 0$ if $i \notin S$. By construction, trivially the third property is satisfied. We need to set $p_0, \ldots, p_d$ such that the first and the second properties are satisfied. Note that to ensure that Y_i's are d-wise independent, it suffices to show that for every $0 \leq i \leq d$ and $1 \leq j_1 < \ldots < j_i \leq m$, we have $\mathbb{E}[y_{j_1} \cdot \ldots \cdot y_{j_i}] = \mathbb{E}[y_{j_1}] \cdot \ldots \cdot \mathbb{E}[y_{j_i}] = 1/m^i$ (because each variable y_k takes only two possible values.) By symmetry of the construction, it suffices to ensure these properties when $\{j_1, \ldots, j_i\} = \{1, \ldots, i\}$ for every $0 \leq i \leq d$. Thus we only need to select $p_0, \ldots, p_d$ such that for every $0 \leq i \leq d$,

$$\mathbb{E}[y_1 \cdot \ldots \cdot y_i] = \sum_{t=i}^{d}\frac{\binom{m-i}{t-i}}{\binom{m}{t}}p_t = 1/m^i.$$

This is a triangular system of $d+1$ linear equations which has a unique solution $p_0, \ldots, p_d$. However, we must make sure that the values of the solution $p_0, \ldots, p_d$ are nonnegative. We use descent on i to show $p_i \geq 0$. We have $p_d = \binom{m}{d}/m^d \geq 0$. For $i < d$, we have:

$$\begin{aligned} p_i &= \binom{m}{i}\left[\frac{1}{m^i} - \sum_{t=i+1}^{d} \frac{\binom{m-i}{t-i}}{\binom{m}{t}} p_t\right] \\ &\geq \binom{m}{i}\left[\frac{1}{m^i} - \sum_{t=i+1}^{d} \frac{\binom{m-i-1}{t-i-1}}{\binom{m}{t}} m p_t\right] \\ &= m\binom{m}{i}\left[\frac{1}{m^{i+1}} - \sum_{t=i+1}^{d} \frac{\binom{m-i-1}{t-i-1}}{\binom{m}{t}} p_t\right] = 0 \end{aligned}$$

■

We will also later give a constructive proof of the above claim. However, we choose to retain this argument as the technique used to justify existence of the distribution is more general.

Proof: [of Claim 4.6] To compute the bias of the distribution D, consider any character χ_S where $S \subset [m \log m]$ is non-empty. For any $i \in [m]$, let us define $S_i = S \cap \{(i-1)\log m + 1, \ldots, i \log m\}$. Note that

$$\mathop{\mathbb{E}}_{x \in D}\left[\chi_S(x)\right] = \mathop{\mathbb{E}}_{x \in D}\left[\prod_{i : S_i \neq \phi} \chi_{S_i}(x)\right]$$

Our proof will only depend on the number of non-empty sets S_i. Without loss of generality, we can assume that the non-empty sets are $S_1, \ldots, S_t$ for some $t > 0$. We denote the set of variables $x_{i,1}, \ldots, x_{i,\log m}$ by x_i. To compute the bias, we then need to calculate

$$\mathop{\mathbb{E}}_{x \in D}\left[\prod_{i=1}^{t} \chi_{S_i}(x_i)\right] = \mathop{\mathbb{E}}_{Y}\left[\prod_{i=1}^{t} \mathop{\mathbb{E}}_{x_i}\left[\chi_{S_i}(x_i) | y_i\right]\right]$$

as the variables $x_1, \ldots x_m$ are independent given Y. We now note that

$$\mathop{\mathbb{E}}_{x_i}\left[\chi_{S_i}(x_i) | y_i = 1\right] = (-1)^{|S_i|} \qquad \text{and} \qquad \mathop{\mathbb{E}}_{x_i}\left[\chi_{S_i}(x_i) | y_i = 0\right] = -\frac{(-1)^{|S_i|}}{m-1}$$

If $t \leq d$, then $y_1, \ldots, y_t$ are independent and the bias simply becomes 0 as below.

$$\begin{aligned} \mathop{\mathbb{E}}_{Y}\left[\prod_{i=1}^{t} \mathop{\mathbb{E}}_{x_i}\left[\chi_{S_i}(x_i) | y_i\right]\right] &= \prod_{i=1}^{t} \mathop{\mathbb{E}}_{x_i, y_i}\left[\chi_{S_i}(x_i)\right] \\ &= \prod_{i=1}^{t}\left(\frac{1}{m} \cdot (-1)^{|S_i|} - \left(1 - \frac{1}{m}\right) \cdot \frac{(-1)^{|S_i|}}{m-1}\right) = 0 \end{aligned}$$

If $t > d$, we can bound the bias as

$$\begin{aligned}
\mathop{\mathbb{E}}_{Y}\left[\prod_{i=1}^{t}\mathop{\mathbb{E}}_{x_i}[\chi_{S_i}(x_i)|y_i]\right] &\leq \mathop{\mathbb{E}}_{Y}\left[\prod_{i=1}^{t}\left|\mathop{\mathbb{E}}_{x_i}[\chi_{S_i}(x_i)|y_i]\right|\right] \\
&\leq \mathop{\mathbb{E}}_{Y}\left[\prod_{i=1}^{d}\left|\mathop{\mathbb{E}}_{x_i}[\chi_{S_i}(x_i)|y_i]\right|\right] \\
&= \prod_{i=1}^{d}\left(\frac{1}{m}+\left(1-\frac{1}{m}\right)\cdot\frac{1}{m-1}\right) \;=\; \left(\frac{2}{m}\right)^{d}
\end{aligned}$$

which proves the claim. ■

■

By plugging $d = \log(1/\delta)/\log\log(1/\delta)$ in the above theorem, we get the following corollary.

Corollary 4.7 *For m which is a power of 2 and $\delta > 0$, there is a read-once DNF ϕ over $n = m \log m$ variables and a distribution D over $\{0,1\}^n$ which has bias $m^{-O(\log(1/\delta)/\log\log(1/\delta))}$ and ϕ distinguishes D from uniform by δ.*

We give below an alternate proof of Claim 4.5 which gives an explicit construction for the d-wise independent distribution mentioned in the claim.

Claim 4.8 *There is a distribution $Y = Y_1 \circ \ldots \circ Y_m$ over $\{0,1\}^m$ with the following properties*

- *for every $1 \leq i \leq m$, $\Pr[Y_i = 1] = 1/m$.*
- *$Y_1, \ldots, Y_m$ are d-wise independent;*
- *For every $y \in Supp(Y)$, $y_1 + \ldots + y_m \leq d$.*

Proof: Since m is taken to be a power of 2, there exists a field $\mathbb{F}$ with $|\mathbb{F}| = m$, the elements of which we identify with the numbers $0, \ldots, m-1$. Choose d independent random elements $a_0, \ldots, a_{d-1} \in \mathbb{F}$ and define the (random) degree-d polynomial

$$P(z) \;:=\; z^d + a_{d-1}z^d + \ldots + a_0.$$

We define the random variables $Y_1, \ldots, Y_m$ as

$$Y_i \;:=\; \begin{cases} 1 & \text{if } P(i-1) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Since P is a degree d-polynomial, for any $y_1, \ldots, y_m \in Supp(Y)$, at most d of $y_1, \ldots, y_m$ are 1 and hence $y_1 + \ldots + y_m \leq d$. Also, since P is equally likely to take any of the m values at the point $i-1$ (as a_0 is uniform in $\mathbb{F}$), $\Pr[Y_i = 1] = \Pr[P(i-1) = 0] = 1/m$.

Note that for any d distinct points $i_1, \ldots, i_d$ and the polynomial P as above, the vector $(P(i_1), \ldots, P(i_d))$ can be computed as

$$(P(i_1), \ldots, P(i_d)) \;=\; (a_0, \ldots, a_{d-1}) \cdot A + (i_1^d, \ldots, i_d^d)$$

where $A \in \mathbb{F}^{d\times d}$ is a matrix with the j^{th} column as $(1, i_j, \ldots, i_j^{d-1})^T$. Since all the columns of A are linearly independent, and $(a_0, \ldots, a_{d-1})$ is a random element of $\mathbb{F}^d$, $(P(i_1), \ldots, P(i_d))$ is also uniformly distributed in $\mathbb{F}^d$. This gives that the values of P at any d points are independent and hence $Y_1, \ldots, Y_m$ form the required d-wise independent distribution. ■

4.3 Lower Bounds for General DNF

Below we show that to δ-fool general DNFs with m terms, one requires a $m^{-\Omega(\log 1/\delta)}$ biased set. Before, we state the theorem, we state the following technical lemma.

Lemma 4.9 *For $x \in \{0,1\}^n$, let $MOD_3(x) = \sum_{i=1}^n x_i \pmod 3$. Consider the distribution D over $\{0,1\}^n$ which is the uniform distribution on the set D_0 defined as*

$$D_0 = \{x | MOD_3(x) \neq 0\}$$

Then D is $2^{-\Omega(n)}$ biased distribution.

Proof: Consider any linear function $\chi : \{0,1\}^n \to \{-1,1\}$. Lemma 2.9 in [54] says that

$$|\Pr_{x:MOD_3(x)=0}[\chi(x)=1] - \Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]| = 2^{-\Omega(n)}$$

Also, $|x : \chi(x)=1| = \left(\Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]\right)|D_0| + \left(\Pr_{x:MOD_3(x)=0}[\chi(x)=1]\right)(2^n - |D_0|)$

$$\Longrightarrow$$

$$|x : \chi(x)=1| \geq \Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]|D_0| + \left(\Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1] - 2^{-\Omega(n)}\right)(2^n - |D_0|)$$

$$\Longrightarrow \frac{|x : \chi(x)=1|}{2^n} + \frac{2^{-\Omega(n)}(2^n - |D_0|)}{2^n} \geq \left(\Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]\right)$$

$$\Longrightarrow \frac{1}{2} + 2^{-\Omega(n)} \geq \left(\Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]\right)$$

Similarly, we can prove that

$$\frac{1}{2} - 2^{-\Omega(n)} \leq \left(\Pr_{x:MOD_3(x)\neq 0}[\chi(x)=1]\right)$$

This implies that $|\mathbb{E}_{x\in D}[\chi(x)]| = 2^{-\Omega(n)}$ which implies that D is a $2^{-\Omega(n)}$ biased set. ■

We now prove the existence of small biased sets which are distinguished by DNFs. The bound on the bias in terms of number of terms and distinguishing probability is in the subsequent corollary.

Theorem 4.10 *For any $t \geq 3, \ell$, there exists a DNF ϕ over ℓt variables with $O(t2^\ell)$ terms and an $\epsilon = 2^{-\Omega(\ell t)}$-biased distribution D such that ϕ distinguishes D from uniform with probability $2^{-O(t)}$.*

Proof: The distribution D will be the uniform distribution over $D_0 \subset \{0,1\}^{\ell t}$ which is defined as

$$D_0 = \left\{x \in \{0,1\}^{\ell t} | MOD_3(x) \neq 0\right\}$$

By Lemma 4.9, the bias of D_0 is $2^{-\Omega(\ell t)}$. To define the DNF ϕ, we partition the variables into t blocks, each block having ℓ variables. The j^{th} variable in the i^{th} block is denoted by x_{ij}. The DNF ϕ is defined as $\phi_1 \vee \ldots \vee \phi_t$ where ϕ_i is a DNF over the i^{th} block of variables which is 1 if and only if the sum of the variables in the i^{th} block is non-zero modulo 3. Note that ϕ is only a function of variables in the i^{th} block. Thus, we can always write ϕ_i using 2^ℓ terms. Hence, ϕ can be written using $t2^\ell$ terms. We first observe that

$$\Pr_{x \in D}[\phi(x) = 1] = 1$$

This is because if the sum of the variables in all the blocks is non-zero mod 3, then there must be at least one block i in which the sum is non-zero mod 3 which ensures that $\phi_i = 1$ implying $\phi = 1$. Now, note that under the uniform distribution, each $\phi_i = 0$ with probability at least $1/3 - 2^{-\ell} \geq 1/4$. This is because $\phi_i = 1$ iff $\sum_{j=1}^{\ell} x_{ij} \neq 0 (mod\ 3)$. As all ϕ_i's are over disjoint sets of variables, this implies

$$\Pr_{x \in U}[\phi(x) = 1] = 1 - \Pr_{x \in U}[\phi(x) = 0] = 1 - (\wedge_{i=1}^{t} \Pr_{x \in U}[\phi_i(x) = 0]) \leq 1 - \frac{1}{4^t}$$

This implies that ϕ distinguishes D from uniform by $1/4^t = 2^{-O(t)}$. ■

Corollary 4.11 *For arbitrarily large m and arbitrary small δ such that $2^{-m/2} < \delta$, there exists a DNF ϕ over $O(\log m \log(1/\delta))$ variables and a distribution D such that ϕ has m terms, D has bias $m^{-\Omega(\log(1/\delta))}$ and ϕ distinguishes D from uniform with probability δ.*

Proof: From the above theorem, we can say that for every t, ℓ there is a DNF ϕ and a distribution D such that ϕ has $t2^\ell$ terms, D is $2^{-\Omega(t\ell)}$ biased and ϕ can distinguish D from uniform by $2^{-O(t)}$. By setting $t = \Theta(\log(1/\delta))$, we can get the distinguishing probability to be equal to δ. Similarly, we set $\ell = \log m - \log\log(1/\delta) - \Theta(1)$, we can get the number of terms to be m. Then the bias of the distribution D guaranteed by the theorem is $2^{-\Omega(t\ell)} = 2^{-\Omega((\log m - \log\log(1/\delta) - \Theta(1))\log(1/\delta)} = m^{-\Omega(\log(1/\delta))}$ as long as $\delta > 2^{-m/2}$. ■

Chapter 5

Related Work and Possible Further Work

There has been several recent results showing that k-wise independent distributions fool various classes of functions:

- Bazzi [12] shows that $O(\log^2(m/\delta))$-wise independent distributions δ-fool DNF formulas with m terms.
- Braverman [15] shows that $O(\log O(d^2)(m/\delta))$-wise independent distributions δ-fool depth-d circuits of size m.
- Diakonikolas et. al. [19] show that $O(\delta^{-2}\log(1/\delta))$-wise independent distributions δ-fool linear threshold functions (half-spaces).
- Diakonikolas, Kane, and Nelson [20] show that poly$(1/\delta)$-wise independent distributions δ-fool degree-2 threshold functions.

By a result of Alon, Goldreich, and Mansour [7], every ϵ-biased distribution over n bits is ϵn^k-close to a k-wise independent distribution. Furthermore, an ϵ-biased distribution can be generated using $O(\log(n/\epsilon))$ bits. Therefore, when it is possible to δ-fool by k-wise independent distributions using $O(k \log n)$ random bits, it is possible to 2δ-fool by ϵ-biased distributions using $O(k \log n + \log(1/\delta))$ random bits. This shows that ϵ-biased distributions can fool at least as good as k-wise independent distributions using the same seed-length. Therefore, a natural question is whether ϵ-biased distributions give pseudorandom generators with better seed-lengths for the above mentioned classes of functions.

Klivans, Lee, and Wan extend our proof that logarithmic seed small-bias distributions fool read-once DNFs to read-k DNFs for constant k. Read-k DNFs are those where every variable appears in at most k terms.

One interesting problem that has been left open in our work is whether ϵ-biased sets with seed-length $O(\log m)$ can δ-fool general DNF formulas with m terms when δ is a const

Part II

Analysis of Goldreich's Candidate One-Way Function

Chapter 6

Preliminaries

6.1 Goldreich's Function

Goldreich [24] constructs a function $f = f_{P,G} : \{0,1\}^n \to \{0,1\}^m$ parameterized by a d-ary predicate $P : \{0,1\}^d \to \{0,1\}$ and $G \in [n]^{m\times d}$, where $[n] = \{1,\ldots,n\}$. The function is defined by

$$(f(x))_i = P(x_{G_{i,1}},\ldots,x_{G_{i,d}}) \quad \text{for } i \in [m].$$

G may be thought of as the adjacency list of a bipartite graph, with n nodes on the left-hand side and m nodes on the right-hand side. Each node on the left represents a bit of input to f, and each node on the right represents a bit of output of f. We name the vertices on the left by $1,\ldots,n$, and we name the vertices on the right by $1,\ldots,m$. We use $L = [n]$ to denote the set of all vertices on the left, and call these vertices *input nodes*. We use $R = [m]$ to denote the set of all vertices on the right, and call these vertices *output nodes*. Each vertex $i \in R$ is connected to vertices $G_{i,1},\ldots,G_{i,d} \in L$, so an output node is connected to the input bits it depends on.

Goldreich suggests using a random predicate P, and a graph G with expansion properties and $m = n$.

6.2 Backtracking Algorithms

Definition 6.1 (Partial Assignment) *A* partial truth 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* $f : \{0,1\}^n \to \{0,1\}^m$, *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 of the variables of f to vary.*

For a partial truth assignment ρ, an index $i \in [n] \setminus \mathrm{Vars}(\rho)$, and a value $a \in \{0,1\}$, we define the partial truth assignment $\rho[i \leftarrow a]$ by

$$\rho[i \leftarrow a](j) = \begin{cases} a, & j = i; \\ \rho(j), & j \neq i. \end{cases}$$

Definition 6.2 (Backtracking Algorithm) *Assume $f = f_{P,G}$ is an instance of Goldreich's function and let $b \in \{0,1\}^m$. A* backtracking algorithm *for finding a solution $x \in \{0,1\}^n$ to the equation $f(x) = b$, is defined by a scheduler $\mathbf{S}$. $\mathbf{S}$ is a mapping which is given as input the string b and the current partial assignment ρ, and returns a pair (i, a) where $i \in [n] \setminus \mathrm{Vars}(\rho)$ is the index of an input variable to assign, and a is the first value to try assigning to x_i. We denote by $\mathrm{BT}_{\mathbf{S}}$ the backtracking algorithm with scheduler $\mathbf{S}$. When given $b \in \{0,1\}^m$ and the current partial assignment ρ, $\mathrm{BT}_{\mathbf{S}}(b, \rho)$ behaves as follows.*

- *If there exists an output node $i \in R$ such that $G_{i,k}$ is fixed by ρ for every $k \in [d]$, and $f(\rho)_i \neq b_i$, then $\mathrm{BT}_{\mathbf{S}}(b, \rho) = \mathrm{FAIL}$.*
- *Otherwise, if ρ is a complete assignment, $\mathrm{BT}_{\mathbf{S}}(b, \rho) = \rho$.*
- *Otherwise, let $(i, a) = \mathbf{S}(b, \rho)$ and compute $x = \mathrm{BT}_{\mathbf{S}}(b, \rho[i \leftarrow a])$. If x is a truth assignment, then $\mathrm{BT}_{\mathbf{S}}(b, \rho) = x$.*
- *Otherwise, $\mathrm{BT}_{\mathbf{S}}(b, \rho) = \mathrm{BT}_{\mathbf{S}}(b, \rho[i \leftarrow 1 - a])$.*

To solve $f(x) = b$, we call $\mathrm{BT}_{\mathbf{S}}(b, *^n)$, so that ρ is initialized to the empty assignment.

Definition 6.3 (Backtracking Tree) *Given a scheduler $\mathbf{S}$, a value $b \in \{0,1\}^m$, and a partial assignment ρ, the* backtracking tree $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, \rho))$ *is a binary tree which records all the decisions made by $\mathbf{S}$ when evaluating $\mathrm{BT}_{\mathbf{S}}(b, \rho)$. Every node of the tree is labeled by a partial assignment.*

The root of the tree is labelled by ρ. If $\mathrm{BT}_{\mathbf{S}}(b, \rho)$ returns immediately, i.e. without recursion, either of FAIL or ρ, then the root has no children. Otherwise, the left child of the root is the subtree $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, \rho[i \leftarrow a]))$, where $(i, a) = \mathbf{S}(b, \rho)$. If $\mathrm{BT}_{\mathbf{S}}(b, \rho[i \leftarrow a]) = \mathrm{FAIL}$, then the root has no right child, and otherwise the root's right child is $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, \rho[i \leftarrow 1 - a]))$. We consider the number of nodes in a subtree of the backtracking tree to be the time *required to explore that subtree.*

Definition 6.4 (Path String) *Given a string $\sigma \in \{\mathrm{left}, \mathrm{right}\}^\ell$, we can think of σ as a path of length ℓ from the root of a binary tree downward. We call a string σ viewed in this way a* path string. *We also let $\mathrm{BT}^{\sigma}_{\mathbf{S}}(b)$ denote the partial assignment which is found by starting at the root of $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, *^n))$ and following ℓ edges down, at the i-th step taking either the left or right child depending on σ_i.*

For example, here is a backtracking tree $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, *^n))$, with the partial assignment $\mathrm{BT}^{\mathrm{left,right,left}}_{\mathbf{S}}(b)$ boxed.

$$\begin{tikzcd}
 & & & *^n \arrow[dll] & \\
 & {[x_5 = 0]} \arrow[dl, dashed] \arrow[drr] & & & \\
 {} & & & {[x_5 = 0, x_7 = 1]} \arrow[dll] \arrow[dr, dashed] & \\
 & \boxed{[x_5 = 0, x_7 = 1, x_8 = 0]} \arrow[dl, dashed] \arrow[dr, dashed] & & & {} \\
 {} & & {} & &
\end{tikzcd}$$

Since the procedure **S** is not restricted in any way, it is free to guide the algorithm immediately toward a correct solution (perhaps by doing an exhaustive search of potential preimages). Thus, to hope for non-linear lower bounds on inversion time, some sort of restriction must be placed on the scheduler. We study backtracking algorithms that are *myopic* and backtracking algorithms that are *drunken*.

6.2.1 Myopic Backtracking Algorithms

The first restricted class of algorithms we consider is that of *myopic* backtracking algorithms, whose decisions must be based on a limited view of the problem instance.

The adjective "myopic," in the context of search algorithms for the satisfiability of CNF formulas, was first introduced by Achlioptas and Sorkin [1]. Later, Alekhnovich et al. [4] defined a more general class of myopic backtracking algorithms, which we will now discuss in the context of finding a solution to the equation $f(x) = b$: The myopic backtracking algorithm has full access to the function f, but limited access to b. In fact, the algorithm does not have access to any of b when it starts. But at every node of the backtracking tree, the myopic scheduler is allowed to read up to K bits of b. Then the algorithm must decide which assignment to make next. The scheduler has access to the bits it read at any ancestor node in the backtracking tree, but not to the bits from other branches: so the algorithm is allowed to remember information when making recursive calls, but must forget whenever it backtracks.

We have found that our approach for proving lower bounds on myopic backtracking algorithms, as well as the approach of [4] on which we build, works for a notion of myopicness which is in some ways more general than that considered in [4]. It is more general in the following two specific ways:

- The algorithm is allowed to amortize its dependency on b. Rather than restricting the scheduler's decisions to depend on K bits of b at each step, we only require that the algorithm's behavior in the first s steps depends on some t bits of b, where s is a parameter and $t = Ks$.

- The algorithm is allowed a more powerful dependence on the bits of b. As an example, assume that the algorithm begins by setting $x_1 = g(b_1, \ldots, b_4)$, where

  $$g = (\bar{b_1} \wedge \bar{b_2} \wedge \bar{b_3}) \vee (\bar{b_1} \wedge b_2 \wedge \bar{b_4}) \vee (b_1 \wedge \bar{b_3} \wedge \bar{b_4}) \vee (\bar{b_2} \wedge b_3 \wedge b_4).$$

  It is possible to see that the value of the function g depends on $b_1, \ldots, b_4$ in such a way that g cannot be calculated using any decision tree of depth ≤ 3; therefore any algorithm that begins by setting $x_1 = g(b_1, \ldots, b_4)$ needs to begin by reading at least 4 bits of b in the worst case. However, since g and

  $$\bar{g} = (b_1 \wedge b_3 \wedge b_2) \vee (b_1 \wedge \bar{b_3} \wedge b_4) \vee (\bar{b_1} \wedge b_2 \wedge b_4) \vee (\bar{b_2} \wedge b_3 \wedge \bar{b_4})$$

  are both 3-DNF formulas, there always exist 3 bits out of b_1, $\ldots$, b_4 who can act as a "certificate" to the value of g: i.e. by knowing the value of only these 3 bits, we can be sure about the value of g. (For example, when $b_1 = 1, b_2 = 0, b_3 = 1, b_4 = 1$, we are sure that $g = 0$ based solely on the values of b_1, b_2, and b_4, irrespective of the value of b_3.) It is also possible to see that for each assignment of values to $b_1, \ldots, b_4$, the 3-bit certificate to the value of g in our example is *unique*. Since g satisfies all these properties, our more general notion of *myopicness* considers setting $x_1 = g(b_1, \ldots, b_4)$ to be possible by "looking at" only 3 bits. (The indices of the 3 bits that act as "certificate" correspond to the set T in Definition 6.5 below. The existence and uniqueness of "certificates" correspond respectively to Properties 1 and 2 in Definition 6.5.)

Definition 6.5 (Myopic Backtracking Algorithm) *A backtracking algorithm for solving the equation $f(x) = b$ is said to be (s,t)-*myopic *if for every $b \in \{0,1\}^n$ and every length-s path string $\sigma \in \{\text{left}, \text{right}\}^s$ there exists $T \subseteq R$ with $|T| \leq t$ such that the decisions made by the algorithm along the path σ depend only on the bits b_T, and T also depends only on b_T. (Here b_T denotes the bits b_i that are indexed by $i \in T$.) More precisely, there exists a function T_σ which assigns to every $b \in \{0,1\}^m$ a set $T_\sigma(b) \subseteq R$, satisfying the following two properties for every $b, b' \in \{0,1\}^m$ such that $b'_{T_\sigma(b)} = b_{T_\sigma(b)}$.*

1. *If $\tau \sqsubseteq \sigma$ is any prefix of σ, then the partial assignments $\mathrm{BT}^\tau_{\mathbf{S}}(b)$ and $\mathrm{BT}^\tau_{\mathbf{S}}(b')$ are equal.*
2. *$T_\sigma(b') = T_\sigma(b)$.*

Notice that the first property is equivalent to saying that for every strict prefix $\tau \sqsubset \sigma$, we have $\mathbf{S}(b', \mathrm{BT}^\tau_{\mathbf{S}}(b')) = \mathbf{S}(b, \mathrm{BT}^\tau_{\mathbf{S}}(b))$.

Note that a myopic scheduler $\mathbf{S}$ has full access to P and G at all times: only the access to b is restricted.

6.2.2 Drunken Backtracking Algorithms

The second class of restricted backtracking algorithms that we consider are *drunken* backtracking algorithms. In these algorithms, the scheduler can use any complicated procedure to choose the variable to assign next, but the value first assigned to that variable should be random. A lower bound for finding satisfying assignments to CNF formulas using drunken backtracking algorithms were first proved in [4]. We will describe such a lower bound for inverting Goldreich's function. Similar bounds were given by [41] and independently by [30]. We choose to describe the lower bound since it fits well in the framework that we have developed for myopic backtracking algorithms.

In order to define drunken backtracking algorithms, we first need to define randomized backtracking algorithms.

Definition 6.6 (Randomized Backtracking Algorithm) *A* randomized scheduler $\mathbf{S}$ *is a random variable whose possible values are deterministic schedulers as defined in Definition 6.2.*

A randomized scheduler $\mathbf{S}$ *leads to a* randomized backtracking algorithm $\mathrm{BT}_{\mathbf{S}}$.

Besides myopic backtracking algorithms, we consider backtracking algorithms where every time the scheduler assigns a new variable, the value assigned to that variable is chosen uniformly at random. The following definition captures this property.

Definition 6.7 (Drunken Backtracking Algorithm) *A randomized backtracking algorithm* $\mathrm{BT}_{\mathbf{S}}$ *is said to be* drunken *if for every $b \in \{0,1\}^m$ and partial assignment ρ, the following conditions are satisfied by the random pair $(i,a) = \mathbf{S}(b,\rho)$. First,* $\Pr[a = 0] = \Pr[a = 1] = \frac{1}{2}$*. Second, a is independent of i, and is also independent of $\mathbf{S}(b,\rho')$ for every $\rho' \neq \rho$.*

Notice that our definition of randomized schedulers allows the bit assignments returned by the scheduler on two inputs (b,ρ) and (b',ρ') to be correlated, but Definition 6.7 stipulates, in the case of drunken schedulers, an independence condition on the bit assignments returned by the scheduler.

6.3 The Predicate Used

Goldreich's function is paramaterized by a predicate $P : \{0,1\}^d \to \{0,1\}$. Goldreich [24] suggests choosing P uniformly at random. In this paper, we consider random predicates.

We also consider predicates of the form $P_{h,Q}(x_1,\ldots,x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1},\ldots,x_d)$ for any predicate $Q : \{0,1\}^h \to \{0,1\}$. Predicates of this type, specifically $P(x_1,\ldots,x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$, have been shown in [42] to yield outputs, through Goldreich's function, that have the pseudorandom property of small-bias [45]. So it is natural to consider predicates of the form $P_{h,Q}$ as the building block for Goldreich's candidate one-way function.

Here we discuss two useful properties predicates can have, and show that they are satisfied by the predicates we consider.

Definition 6.8 (robust predicate) $P : \{0,1\}^d \to \{0,1\}$ *is h-*robust *iff every restriction ρ such that $P|_\rho$ is constant satisfies $d - |\rho| \leq h$ [3, Definition 2.2]. For example, the predicate that sums all its inputs modulo 2 is 0-robust.*

Definition 6.9 (balanced predicate) *For predicate $P : \{0,1\}^d \to \{0,1\}$, real number $\epsilon_{\text{bal}} \in [0, 1/2)$, and integer $h \in [0, d-1]$, we say predicate P is* $(h, \epsilon_{\text{bal}})$-balanced *if, after fixing all variables but $h+1$ of them,*

$$|\Pr[P(x_1, \ldots, x_d) = 0| \textit{ fixed variables}] - \tfrac{1}{2}| \leq \epsilon_{\text{bal}}.$$

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

Remark 6.10 *A predicate is $(h,0)$-balanced iff all its Fourier weight is concentrated on the last $h+1$ levels. That is, every nonzero fourier coefficient corresponds to an XOR of at least $d-h$ variables.*

Lemma 6.11 *A random predicate on d variables is $(\Theta(\log \frac{d}{\epsilon_{\text{bal}}}), \epsilon_{\text{bal}})$-balanced with probability $1 - \exp[-\text{poly}(d/\epsilon_{\text{bal}})]$. The predicate $P_{h,Q}$ is $(h,0)$-balanced.*

Proof: Let ρ be any partial assignment which fixes all but $h+1$ variables. There are 2^{h+1} inputs consistent with ρ: call them $x_1^\rho, \ldots, x_{2^{h+1}}^\rho$. Define the event $E_\rho = \{|\#\{i : P(x_i^\rho) = 1\} - 2^h| > 2^{h+1}\epsilon_{\text{bal}}\}$: E_ρ is the event that P is not balanced under the partial assignment ρ.

By a Chernoff bound, $\Pr[E_\rho] \leq 2e^{-\epsilon_{\text{bal}}^2 2^{h+2}}$. P is balanced if all events E_ρ hold. There are $2^{d-h-1}\binom{d}{h+1}$ partial assignments ρ to consider. Taking a union bound,

$$\begin{aligned}
\Pr[P \text{ is not } (h, \epsilon_{\text{bal}})\text{-balanced}] &\leq 2^{d-h-1}\binom{d}{h+1} 2e^{-\epsilon_{\text{bal}}^2 2^{h+2}} \\
&\leq 2^{d-h} d^{h+1} e^{-\epsilon_{\text{bal}}^2 2^{h+2}} \\
&= \exp[(h+1)\ln d + (d-h)\ln 2 - \epsilon_{\text{bal}}^2 2^{h+2}].
\end{aligned}$$

Finally, take $h = \Theta(\log \frac{d}{\epsilon_{\text{bal}}})$.

To see that the predicate $P_{h,Q}(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1}, \ldots, x_d)$ is $(h,0)$-balanced, notice that any subset of $h+1$ variables includes at least one of the variables $x_1, \ldots, x_{d-h}$. ■

Corollary 6.12 *A random predicate on d variables is $\Theta(\log d)$-robust with probability $1 - \exp[-\text{poly}(d)]$. Any predicate of the form $P_{h,Q}$ is h-robust.*

6.4 Expansion Properties of the Graph Used

Let $G \in [n]^{m \times d}$ be the bipartite graph from Section 6.1. The following definitions and lemmas are generalized from [4].

Definition 6.13 (Boundary and Neighborhood) [4, Definition 2.1].
Let $I \subseteq R$. Its neighborhood *$\Gamma(I) \subseteq L$ is the set of all nodes adjacent to nodes in I. For $i \in I$, the* boundary of *i* in *I, denoted $\partial_I i$, is the set of nodes with one edge to i but no other edges to I. The* boundary *of I, denoted ∂I, is the set of all nodes $j \in L$ such that there is exactly one edge from j to I. Equivalently, $\partial I = \bigcup_{i \in I} \partial_I i$.*

Definition 6.14 (Expansion) [4, Definition 2.1].
G is an (r, c)-expander *if for all $I \subseteq R$ such that $|I| \le r$, we have $|\Gamma(I)| \ge c|I|$. G is an* (r, c)-boundary expander *if for all $I \subseteq R$ such that $|I| \le r$, we have $|\partial I| \ge c|I|$.*

Lemma 6.15 Analogous to [4, Lemma 2.1].
Every (r, c)-expander is an $(r, 2c - d)$-boundary expander.

Proof: It suffices to show that whenever $|\Gamma(I)| \ge c|I|$, we have $|\partial I| \ge (2c - d)|I|$. Indeed, every node in $\Gamma(I) \setminus \partial I$ is connected to at least two nodes in I. Since each node in I has degree d, then $|\Gamma(I)| + |\Gamma(I) \setminus \partial I| \le d|I|$. It follows that $|\partial I| \ge 2|\Gamma(I)| - d|I|$. ■

Lemma 6.16 *A random bipartite graph $G \in [n]^{n\times d}$ with n left nodes and n right nodes, and of right-degree d*

- *has with probability $\ge 1 - 2^{-n_{\text{bad}}}$ at most n_{bad} left nodes of degree $> d_{\text{left}}$, provided $d_{\text{left}} \ge 2de$ and $n_{\text{bad}} \ge 2ne2^{-d}$;*
- *is with probability $\ge 1 - 1/n$ an (r, c')-expander for any $d = o(n/\lg n)$, $c' = d - \Omega(d)$, provided $r \le r_{\max}(n, d, c')$ where $r_{\max} = \Omega(n/d)$.*

Proof: The probability that there are $> n_{\text{bad}}$ left vertices of degree $> d_{\text{left}}$ is at most the probability that there exists a set S of n_{bad} left vertices such that at least $n_{\text{bad}}d_{\text{left}}$ of the edges of the graph have an endpoint in S. For each set S of n_{bad} left vertices, this happens with probability at most

$$\binom{nd}{n_{\text{bad}}d_{\text{left}}}(\frac{n_{\text{bad}}}{n})^{n_{\text{bad}}d_{\text{left}}} \le (\frac{nde}{n_{\text{bad}}d_{\text{left}}})^{n_{\text{bad}}d_{\text{left}}}(\frac{n_{\text{bad}}}{n})^{n_{\text{bad}}d_{\text{left}}} \le 2^{-dn_{\text{bad}}},$$

where the last inequality is true provided $d_{\text{left}} \ge 2de$. Now by a union bound, the probability that such an S exists is at most

$$\binom{n}{n_{\text{bad}}}2^{-dn_{\text{bad}}} \le (\frac{ne}{n_{\text{bad}}})^{n_{\text{bad}}}2^{-dn_{\text{bad}}} \le 2^{-n_{\text{bad}}}$$

provided $n_{\text{bad}} \ge 2ne2^{-d}$. This proves the first part of the lemma. To prove the second part of the lemma, we note that the probability that the expansion of a specific set of i right nodes be $< c'$ is at most

$$\binom{n}{\lfloor c'i\rfloor}(\frac{\lfloor c'i\rfloor}{n})^{di} \le (\frac{ne}{\lfloor c'i\rfloor})^{\lfloor c'i\rfloor}(\frac{\lfloor c'i\rfloor}{n})^{di}.$$

Thus the probability that the graph is not an (r, c')-expander is at most

$$\sum_{i=1}^{r}\binom{n}{i}(\frac{ne}{\lfloor c'i\rfloor})^{\lfloor c'i\rfloor}(\frac{\lfloor c'i\rfloor}{n})^{di} \le \sum_{i=1}^{r}(\frac{ne}{i})^{i}(\frac{ne}{c'i})^{c'i}(\frac{c'i}{n})^{di} = \sum_{i=1}^{r}((\frac{i}{n})^{d-c'-1}c'^{d-c'}e^{1+c'})^{i}.$$

Let $a_i = (\frac{i}{n})^{d-c'-1} c'^{d-c'} e^{1+c'}$. By the above, to prove the second part of the lemma, it is enough to show that $a_i^i \leq n^{-2}$ for every $1 \leq i \leq r$. Define

$$r_{\max} = \frac{n}{c'}(4c' e^{1+c'})^{\frac{-1}{d-c'-1}} = \Omega(n/d).$$

For $i \leq r_{\max}$, we have $a_i \leq 1/4$. Thus, if $i \geq \lg n$, we have $a_i^i \leq n^{-2}$. And if $i < \lg n$, it can be seen that $a_i = o(n^{-2})$ for d bigger than a large enough constant but at the same time $d = o(n/\lg n)$. ■

Lemma 6.17 *Assume G is an (r,c)-boundary expander of right-degree d, with at most n_{bad} left-nodes of degree bigger than d_{left}. Assume $\hat{c} = \lceil c - h - 1 \rceil$ is positive. Let W be a subset of the input nodes L. Then there exists $U \subseteq W$ such that*

$$|U| \geq \frac{\hat{c}(|W| - n_{\text{bad}})}{\hat{c} + d_{\text{left}}(d-1)}$$

and for every $A \subseteq R$ with $|A| \leq r$, there is some $i \in A$ such that $|\partial_A i \setminus U| > h$.

Proof: Construct U using the following algorithm:

- $U \leftarrow \varnothing$.
- For every $i \in L$, set $n_i \leftarrow \begin{cases} \hat{c} & \text{if } i \in W \text{ and } i \text{ has degree at most } d_{\text{left}}; \\ 0 & \text{otherwise.} \end{cases}$
- The following invariant holds every time the following **while** loop checks its loop condition: every output node connected to an input node j has at most $\hat{c} - n_j$ adjacent input nodes in U.
- **while** $\exists i,\ n_i > 0$,
  - $U \leftarrow U \cup \{i\}$.
  - $n_i \leftarrow 0$.
  - For every $j \in L$ distinct from i, if i and j have a common neighbor, then $n_j \leftarrow \max\{0, n_j - 1\}$.

In the beginning, $\sum_i n_i \geq \hat{c}(|W| - n_{\text{bad}})$, and in the end, $\sum_i n_i = 0$. At each step, $\sum_i n_i$ decreases by at most $\hat{c} + d_{\text{left}}(d-1)$. Therefore the number of steps we took is

$$|U| \geq \frac{\hat{c}(|W| - n_{\text{bad}})}{\hat{c} + d_{\text{left}}(d-1)}.$$

By the loop invariant, every output node has at most $\hat{c}$ adjacent input nodes in U. Let $A \subseteq R$ have size $|A| \leq r$. Then by the expansion of G, there is some $i \in A$ such that $|\partial_A i| \geq c$. It follows that $|\partial_A i \setminus U| \geq c - \hat{c} > h$. ■

6.4.1 Closure Operation

We will make use of the notion of the *closure* of a set of input nodes. Our definition is based on the one given in [4, Section 3.2], which is in turn based on [5].

Definition 6.18 (Closure) *Fix a bipartite graph G with vertex parts L and R, and numbers $r, c > 0$ such that G is an (r, c)-boundary expander. Fix a subset of input nodes $J \subseteq L$. We say a subset of output nodes $I \subseteq R$ is* closed *with respect to J if the subgraph of G obtained by deleting nodes in $J \cup \Gamma(I)$ and nodes in I is an $(r/2, c/2)$-boundary expander. The* closure *of J, denoted $\mathrm{Cl}(J)$, is the collection of all sets which are closed with respect to J and have size at most $r/2$.*

Note that an element of the closure of a set of left-nodes is a set of right-nodes: so, for example, it does not make sense to say for $C \in \mathrm{Cl}(I)$ that $C \supseteq I$.

Lemma 6.19 Analogous to [4, Lemma 3.5].
If G is an (r, c)-boundary expander and $|J| \leq cr/4$, then $\mathrm{Cl}(J)$ is non-empty and contains some C such that $|C| < 2c^{-1}|J|$.

Proof: Call $I \subseteq R$ *nonexpanding* if $|\partial I \setminus J| < c|I|/2$. For a nonexpanding I we have $|\partial I| < c|I|/2 + |J|$. If, furthermore, $|I| \leq r$, by the boundary-expansion of G we have $|\partial I| \geq c|I|$, so $|I| < 2c^{-1}|J| \leq r/2$. Therefore a nonexpanding $I \subseteq R$ has either $> r$ vertices or $\leq r/2$ vertices.

If the empty set is closed, we are done. Otherwise, let I_0 be a largest nonexpanding set with $\leq r/2$ vertices. We claim that I_0 is closed. Indeed, let S be any subset of $R \setminus I_0$ with $\leq r/2$ vertices. It suffices to show that $|\partial S \setminus (J \cup \Gamma(I_0))| \geq c|S|/2$. Suppose otherwise: then S is nonempty, and also $|\partial(I_0 \cup S) \setminus J| \leq |\partial I_0 \setminus J| + |\partial S \setminus (J \cup \Gamma(I_0))| < c|I_0 \cup S|/2$. $I_0 \cup S$ is then a nonexpanding set with $\leq r$ vertices, and therefore $\leq r/2$ vertices. This contradicts our assumption that I_0 was a largest nonexpanding set with $\leq r/2$ vertices.

I_0 is closed, and we showed at the start of the proof that $|I_0| < 2c^{-1}|J|$. ■

Lemma 6.20 *For any set $J \subseteq L$, $\mathrm{Cl}(J) = \mathrm{Cl}(J \cup \bigcap_{I \in \mathrm{Cl}(J)} \Gamma(I))$.*

Proof: Any set $I \subseteq R$ which is closed with respect to $J \cup \bigcap_{I' \in \mathrm{Cl}(J)} \Gamma(I')$ is also closed with respect to the smaller set J, so $\mathrm{Cl}(J) \supseteq \mathrm{Cl}(J \cup \bigcap_{I' \in \mathrm{Cl}(J)} \Gamma(I'))$. Conversely, let $I \in \mathrm{Cl}(J)$. We must show that I is closed with respect to $J \cup \bigcap_{I' \in \mathrm{Cl}(J)} \Gamma(I')$: that is, that the subgraph $\hat{G}$ of G obtained by deleting nodes in I and in $J' = J \cup \left(\bigcap_{I' \in \mathrm{Cl}(J)} \Gamma(I')\right) \cup \Gamma(I)$ is an $(r/2, c/2)$-boundary expander. J' is equal to $J \cup \Gamma(I)$, and $I \in \mathrm{Cl}(J)$, so $\hat{G}$ is indeed an $(r/2, c/2)$-boundary expander. ■

We now relate Goldreich's function to our notions of expansion, robust predicates, and closure. The following definition of local consistency is equivalent with [4, Definition 3.4].

Definition 6.21 (Locally Consistent) *Let f be Goldreich's function for graph G and predicate P. Let $b \in \{0,1\}^m$ and let ρ be a partial truth assignment. For a set of output nodes $I \subseteq R$, we say ρ is* consistent *with I if ρ can be extended to some $x' \in \{0,1\}^n$ such that $f(x')_I = b_I$. We say ρ is* locally consistent *if for all $I \subseteq R$ such that $|I| \leq r/2$, ρ is consistent with I.*

Lemma 6.22 Analogous to [4, Lemma 3.6].
Let P be a $c/2$-robust predicate. Let $b \in \{0,1\}^m$. Let ρ be a partial truth assignment, and let $C \in \mathrm{Cl}(\mathrm{Vars}(\rho))$. Then ρ is locally consistent iff ρ is consistent with C.

Proof: By Definition 6.18 we have $|C| \leq r/2$. Thus if ρ is locally consistent then it is consistent with C.

Conversely, assume ρ is consistent with C. If ρ is not locally consistent, let I be a smallest set such that ρ is not consistent with I. Let $I' = I \setminus C$. Since ρ is consistent with C, we have $I' \neq \emptyset$.

Let $L^- = \mathrm{Vars}(\rho) \cup \Gamma(C)$. Since C is closed with respect to $\mathrm{Vars}(\rho)$, $|\partial I' \setminus L^-| \geq c|I'|/2$. In particular, there must be some $i \in I'$ with $|\partial_{I'} i \setminus L^-| \geq c/2$.

Since I is a smallest set with which ρ is not consistent, ρ is consistent with $I \setminus \{i\}$: so extend ρ to a partial assignment x' which satisfies $(f(x'))_{I\setminus\{i\}} = b_{I\setminus\{i\}}$. Since P is a $c/2$-robust predicate, we can modify input bits in the set $|\partial_{I'} i \setminus L^-|$ and leave all other input bits the same to produce an input x'' such that $(f(x''))_i = b_i$. Since x'' is equal to x' on every input bit in $\Gamma(I \setminus \{i\})$, we have $(f(x''))_I = b_I$, which contradicts the assumption that ρ was not consistent with I. ■

Lemma 6.23 *If P is a $c/2$-robust predicate, the partial assignment ρ is locally consistent, and the closure $\mathrm{Cl}(T)$ is nonempty for some set of input nodes $T \supseteq \mathrm{Vars}(\rho)$, then ρ can be extended to a locally consistent partial assignment ρ' such that $\mathrm{Vars}(\rho') = T$.*

Proof: Pick any $C \in \mathrm{Cl}(T)$. Since ρ is locally consistent, ρ is consistent with C. That is, ρ can be extended to a complete assignment $x' \in \{0,1\}^n$ such that $f(x')_C = b_C$. Let ρ' be the restriction of x' to T: then by Lemma 6.22, ρ' is locally consistent. ■

6.5 Experimental Study of Inverting Goldreich's Function

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 [22, 21], 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.

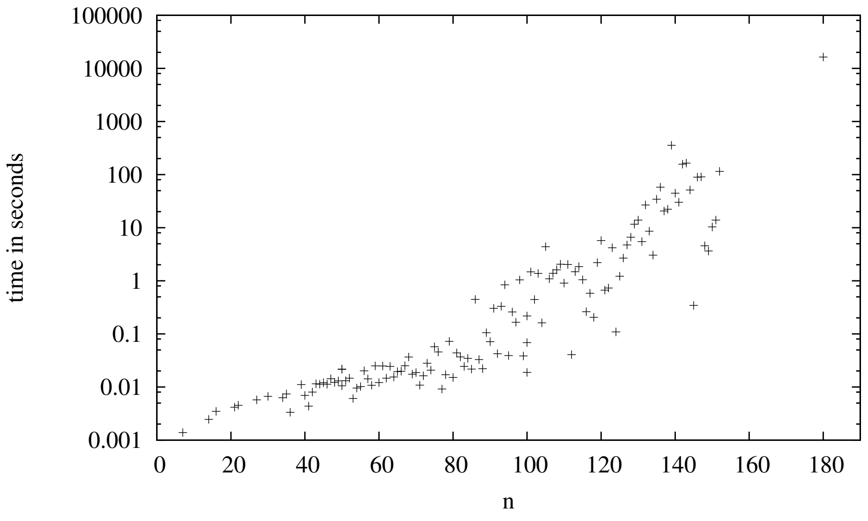

Figure 6.1: Number of seconds taken by MiniSat to invert Goldreich's function for different values of n. We use the degree-five predicate $P_5(x) = x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5)$ and a random bipartite graph of right-degree five.

We ran MiniSat on a Lenovo T61 laptop with 2GB of RAM and a 2.00GHz Intel T7300 Core Duo CPU. Fig. 6.1 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.

Chapter 7

Lower Bounds on Myopic and Drunken Backtracking Algorithms

Here we state and prove Theorem 7.1, our main result about the complexity of inverting Goldreich's function with myopic and drunken backtracking algorithms. We note that for drunken algorithms, similar results were given by [41] and independently by [30].

Our result has a large number of parameters and assumptions which we list after the statement of the theorem, in Setups 7.2, 7.3 and 7.4 below. We then list some example settings of the parameters in Section 7.0.1. We will first give an overview of the proof of Theorem 7.1 in Section 7.1. Following that, we give the actual proof, completing the proof in Section 7.7.

Theorem 7.1

- *Assume* $\mathrm{BT}_{\mathbf{S}}$ *is an* (s,t)*-myopic backtracking algorithm, where* $s \leq cr/4$ *and* $t \leq r/2$. *Import all the parameters and assumptions from the myopic setup (Setups 7.2 and 7.3 below). Then the probability that* $\mathrm{BT}_{\mathbf{S}}$ *on input* b *finds some* $x' \in f^{-1}(b)$ *in time* $\leq 2^{(c/2-h)r/4-d}$ *is at most*

$$M^{1/2}2^{1-u/2}\left(\frac{1+2\epsilon_{\mathrm{bal}}}{1-2\epsilon_{\mathrm{bal}}}\right)^{r/2}.$$

- *Assume* $\mathrm{BT}_{\mathbf{S}}$ *is a drunken backtracking algorithm, and import all the parameters and assumptions from the drunken setup (Setups 7.2 and 7.4 below). Then the probability that* $\mathrm{BT}_{\mathbf{S}}$ *on input* b *finds some* $x' \in f^{-1}(b)$ *in time* $\leq 2^{(c/2-h)r/4-d}$ *is at most*

$$M^{1/2}2^{1-\lfloor\frac{cr}{4}\rfloor/2}.$$

Setup 7.2 (common setup) *These parameters and assumptions are common to the myopic and drunken analyses.*

- *Let* $f = f_{P,G} : \{0,1\}^n \to \{0,1\}^m$ *be Goldreich's function for graph* $G \in [n]^{m\times d}$ *and predicate* $P : \{0,1\}^d \to \{0,1\}$, *where* n, m, *and* d *are positive integers.*
- *Let* $M = \mathbf{E}_{x\in\{0,1\}^n}[|f^{-1}(f(x))|]$. *We say that* f *is* M-to-one on average *and discuss this property more in Section 8.*

Figure 7.1: The probability that a random graph $G \in [n]^{n\times d}$ is not good is given. If the graph is good, the probability that a $(s, n/\Theta(d))$-myopic or drunken bactracking algorithm can invert $f_{P,G}$, for any suitable choice of P, in time better than $2^{\Theta(n)}$ is given. For the myopic algorithm, we should have $s \geq 2^{-\Theta(d)}n$.

Backtracking Algorithm:	$(s, n/\Theta(d))$-myopic	drunken
Running time:	$2^{\Theta(n)}$	$2^{\Theta(n)}$
Probability algorithm can invert:	$2^{-s/\Theta(d)}$	$2^{-\Theta(n)}$
Probability random graph not good:	$2^{-s/\Theta(d^2)} + n^{O(1)}2^{-s/\Theta(d)}$	$2^{-n/\Theta(d)}$

- *Assume G is an (r, c)-boundary expander of right-degree d (Definition 6.14) where r is a positive integer and $c > 0$.*
- *Assume P is h-robust (Definition 6.8), where $h \in (0, c/2)$.*

Setup 7.3 (setup for myopic algorithms) *Import everything from Setup 7.2, and add:*

- *Assume P is an $(h, \epsilon_{\text{bal}})$-balanced predicate (Definition 6.9), for some $\epsilon_{\text{bal}} \in [0, 1/2)$.*
- *Assume G has at most n_{bad} left vertices in G with degree $> d_{\text{left}}$, for some positive integers n_{bad} and d_{left}.*
- *Let $u = \hat{c}(s - n_{\text{bad}})/(\hat{c} + d_{\text{left}}(d - 1))$ where $\hat{c} = \lceil c - h - 1\rceil$.*
- *Choose $x \in \{0, 1\}^n$ uniformly at random and set $b = f(x)$.*

Setup 7.4 (setup for drunken algorithms) *Import everything from Setup 7.2, and add:*

- *Choose $x \in \{0, 1\}^n$ uniformly at random and set $b = f(x)$.*

7.0.1 Applications of Theorem 7.1

Random graphs: Theorem 7.1, applied in the setting where G is a random graph, yields concrete lower bounds on the expected running time of myopic and drunken backtracking algorithms to invert Goldreich's function: Indeed, let $G \in [n]^{n\times d}$ be a random bipartite graph of right-degree d with n left nodes and n right nodes. Throughout we assume that $d = O(\log n)$, so $d = \Theta(1)$ or $d = \Theta(\log n)$ or anything in between. (For a quick look at the final lower bounds that we can achieve, see Figure 7.1).

By Lemma 6.16, with probability at least $1-1/n$, the graph G is an (r, c')-expander with $c' = d/2 + \Theta(d)$ and $r = \Theta(n/d)$. Hence by Lemma 6.15, G is an (r, c)-boundary expander for $c = \Theta(d)$.

In order to be able to get a good final lower bound on the running time, we assume the predicate P satisfies the following two properties:

1. P is h-robust for $h \leq c/2(1-\Omega(1))$. For lower bounds on myopic algorithms, we also need the stronger condition that P be $(h, 2^{-\Omega(d)})$-balanced.

2. P is such that Equation 8.1 of Theorem 8.1 holds.

Two types of predicates satisfy the above two properties:

1. Predicates $P = P_{h,Q}(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1}, \ldots, x_d)$: By Lemma 6.11, not only does P satisfy Property 1 above, but in fact P is $(h, 0)$-balanced. Also, by Theorem 8.1 it satisfies Property 2 above.

2. Random predicates P on d variables: By Lemma 6.11, with high probability, P is $(O(\lg d), 2^{-\Theta(d)})$-balanced, and hence Property 1 above is satisfied (since $O(\log d) \leq c/2(1-\Omega(1)) = \Theta(d)$). Also, by Theorem 8.1, with high probability, P satisfies Property 2 above. Furthermore, by Remark 8.2 we can with high probability verify this in time $2^{O(d)}$ to be sure that predicate P satisfies Property 2 above. Verifying whether P satisfies Property 1 can also clearly be done in time $2^{O(d)}$. Hence we can keep repeating the generation of the random predicate P until we are sure P satisfies both Properties 1 and 2.

Now Theorem 7.1 can be applied to the function $f_{P,G}$ as follows:

1. Assume that the backtracking algorithm is (s,t)-myopic where $t \leq r/2$.

   Let $n_{\text{bad}} = s/2$ and $d_{\text{left}} = \lceil 2de \rceil$. Then by Lemma 6.16 with probability at least $1 - 2^{-\Theta(s)}$ there are at most n_{bad} nodes of degree $> d_{\text{left}}$, provided $s \geq 4ne2^{-d}$. In Setup 7.3, we have
   $$u = \frac{\lceil c-h-1 \rceil (s - n_{\text{bad}})}{\lceil c-h-1 \rceil + d_{\text{left}}(d-1)} = \frac{s}{\Theta(d)}.$$
   Now by Theorem 8.1, with probability $1 - n^{O(1)}q$ over the randomness of G, the function $f_{P,G}$ satisfies many-to-oneness $M \leq 2^{2^{-\Theta(d)}n}/q$. (As we will see, we will choose $q = 2^{-s/\Theta(d)}$, so the probability that a random G is not good is at most $n^{O(1)}2^{-s/\Theta(d)}$.) Now by Theorem 7.1 the probability that the algorithm does not take time $\geq 2^{(c/2-h)r/4-d} = 2^{\Theta(n)}$ is at most
   $$M^{1/2}2^{1-u/2}\left(\frac{1+2\epsilon_{\text{bal}}}{1-2\epsilon_{\text{bal}}}\right)^r \leq \left(2^{2^{-\Theta(d)}n}q^{-1/2}\right)2^{-s/\Theta(d)}2^{2^{-\Theta(d)}n},$$
   which is at most probability $2^{-s/\Theta(d)}$ when $s = \Omega(2^{-\Theta(d)}n)$ and $q = 2^{-s/\Theta(d)}$.

2. Assume that the backtracking algorithm is drunken.

   With probability $1 - qn^{O(1)}$ over the randomness of G, we know that the average many-to-oneness of $f_{P,G}$ is $M \leq 2^{2^{-\Theta(d)}n}/q$. (As we will see, we will choose $q = 2^{-\Theta(n)}$.) Now by Theorem 7.1, the probability that the algorithm does not take time $\geq 2^{(c/2-h)r/4-d} = 2^{\Theta(n)}$ is at most
   $$M^{1/2}2^{1-\lfloor \frac{cr}{4} \rfloor/2} = \left(2^{2^{-\Theta(d)}n}q^{-1/2}\right)2^{-\Theta(n)},$$
   which is probability $2^{-\Theta(n)}$ for a suitable choice of $q = 2^{-\Theta(n)}$.

Reducing the effect of random non-expanding graphs: As was just mentioned, our upper bound on the probability that a random graph G is not an expander (Lemma 6.16) is only inverse polynomial (more specifically $1/n$). This means that the function $f_{P,G}$ is secure against myopic and drunken backtracking algorithms only with inverse polynomial probability. In order to improve (i.e. decrease) this probability to something which is inverse superpolynomial, we suggest two ways:

1. Use the function $f^{\otimes k} : \{0,1\}^{n\times k} \to \{0,1\}^{n\times k}$ defined as the concatenation of k Goldreich functions:
$$f^{\otimes k}(x^1, \dots, x^k) = f_{P,G_1}(x^1) \dots f_{P,G_k}(x^k),$$
where $G_1, \dots, G_k \in [n]^{n\times d}$ are chosen independently and uniformly at random. $f^{\otimes k}$ is as secure as each of the individual functions $f_{P,G_1}, \dots, f_{P,G_k}$. Thus, the probability that $f^{\otimes k}$ is not secure is $\le n^{-k}$. To get exponentially small probability, let for example $k = n$.

2. In Section 7.8, we show how we can relax the expansion condition and still get lower bounds on the running time of myopic and drunken backtracking algorithms. Specifically, Theorem 7.22 could give for r_{bad}-imperfect expanders G (see Definition 7.20) the same lower bounds as we got above from Theorem 7.1 for perfect expanders G, if in the myopic case $r_{\text{bad}} \le u/4d = s/\Theta(d^2)$ and in the drunken case $r_{\text{bad}} \le cr/(8d) = n/\Theta(d)$. Thus by Lemma 7.21, we have the following: (See Figure 7.1)
   - With probability $1 - 2^{-s/\Theta(d^2)} - n^{O(1)}2^{-s/\Theta(d)}$ over the randomness of graph G, an $(s, n/\Theta(d))$-myopic algorithm cannot invert $f_{P,G}$ with probability better than $2^{-s/\Theta(d)}$ in time better than $2^{\Theta(n)}$, provided $s \ge 2^{-\Theta(d)}n$.
   - With probability $1 - 2^{-n/\Theta(d)}$ over the randomness of graph G, a drunken algorithm cannot invert $f_{P,G}$ with probability better than $2^{-\Theta(n)}$ in time better than $2^{\Theta(n)}$.

7.1 Proof Overview

Our proof is similar to the proof of Alekhnovich et al. [4] that myopic backtracking algorithms take an exponentially long time to solve systems of linear equations.

Let $\mathrm{BT}_{\mathbf{S}}$ be a myopic or drunken backtracking algorithm. We pick a random $x \in \{0,1\}^n$, and run $\mathrm{BT}_{\mathbf{S}}$ on input $b = f(x)$. Our goal is to show that $\mathrm{BT}_{\mathbf{S}}$ will run for a long time before returning any $x' \in f^{-1}(b)$, but we begin with an easier goal: to show that $\mathrm{BT}_{\mathbf{S}}$ will either run for a long time or return a value that is not *exactly* equal to x. In the case that f is an injective function, these goals are one and the same. In the general case, Lemma 7.18 allows us to reduce the first goal, Theorem 7.1, to the second goal, Lemma 7.17.

Our strategy to prove Lemma 7.17 is to show first that with high probability $\mathrm{BT}_{\mathbf{S}}$ will choose an incorrect value for a variable, and second, that it will take a long time to notice its mistake. We are only able to prove the second part when the mistake the algorithm made is locally consistent (Definition 6.21).

At every point during the execution of $\mathrm{BT}_{\mathbf{S}}$, its partial truth assignment ρ can be in one of three states:

1. ρ is consistent with x.
2. ρ is not consistent with x, but is locally consistent.
3. ρ is not locally consistent.

We show (in Lemma 7.13 when $\mathrm{BT}_{\mathbf{S}}$ is myopic and in Lemma 7.14 when $\mathrm{BT}_{\mathbf{S}}$ is drunken) that with high probability, the algorithm reaches state 2 at some point when all but at most $cr/4$ bits of ρ are either unspecified ($\rho_i = *$) or are "locally forced", meaning if ρ_i were flipped then ρ would not be locally consistent (Definition 7.5). At this point, we can apply the proof in Section 7.4: the algorithm will either return a value $x' \neq x$ which is consistent with ρ, or, in the case that ρ can't be extended to any $x' \in f^{-1}(b)$, $\mathrm{BT}_{\mathbf{S}}$ will take exponentially long to backtrack and undo its mistake

In order to make our task simpler, we start in Section 7.2 by modifying $\mathrm{BT}_{\mathbf{S}}$ so that it becomes a *clever* algorithm that never enters state 3 before it has made more than $cr/4$ locally unforced assignments. Then our task is reduced to showing that after the clever version of $\mathrm{BT}_{\mathbf{S}}$ has made $cr/4$ locally unforced assignments, it is much more likely to be in state 2 than state 1. We show this in Section 7.3.

7.2 Clever Backtracking Algorithms

As in the proof of Alekhnovitch et al, [4] we find it convenient to assume the algorithm $\mathrm{BT}_{\mathbf{S}}$'s first $cr/4$ variable substitutions are locally consistent. We use their notion of a *clever myopic* algorithm, which we call a *clever backtracking algorithm.*

Definition 7.5 (Locally Forced Assignment) *Let $b \in \{0,1\}^m$, let ρ be a partial truth assignment, and let $i \in [n] \setminus \mathrm{Vars}(\rho)$. We say an assignment $i \leftarrow a$ is* locally forced *by b and ρ if $\rho[i \leftarrow a]$ is locally consistent but $\rho[i \leftarrow 1-a]$ is not. (Definition 6.21 defines local consistency.)*

Definition 7.6 (Clever Backtracking Algorithm) *We call a scheduler* $\mathbf{S}$ clever *if it avoids making assignments which are not locally consistent. More precisely, it is never the case that $i \leftarrow a$ is locally forced by b and ρ but $\mathbf{S}(b,\rho) = (i, 1-a)$. If the scheduler $\mathbf{S}$ is clever, we also call the associated backtracking algorithm $\mathrm{BT}_{\mathbf{S}}$ clever.*

We show that we may assume without loss of generality that a backtracking algorithm is clever. We begin by defining the clever version of a backtracking algorithm, and then show that the clever version is at least as good as the original, and does not lose the property of being myopic.

Definition 7.7 (The Clever Version of a Backtracking Algorithm) *Let* $\mathbf{S}$ *be a scheduler. The* clever version of $\mathbf{S}$*, denoted* $\mathcal{C}(\mathbf{S})$*, is defined as follows. If* $\mathbf{S}(b, \rho) = (i, a)$*, then*

$$\mathcal{C}(\mathbf{S})(b, \rho) = \begin{cases} (i, 1-a) & \textit{if } i \leftarrow 1-a \textit{ is locally forced by } b \textit{ and } \rho \\ (i, a) & \textit{otherwise.} \end{cases}$$

The clever version of a backtracking algorithm $\mathrm{BT}_{\mathbf{S}}$ *is* $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$.

As the name indicates, the clever version of a backtracking algorithm is also clever.

Lemma 7.8 $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}(b, \rho)$ *produces the same output as* $\mathrm{BT}_{\mathbf{S}}(b, \rho)$*, and does not take any more steps.*

Proof: Observe that whenever $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ behaves differently from $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$, it is because $\rho[i \leftarrow a]$ is locally inconsistent, where $(i, a) = \mathbf{S}(b, \rho)$. Any locally inconsistent assignment is also globally inconsistent, so $\mathrm{BT}_{\mathbf{S}}$ must backtrack after trying $\rho[i \leftarrow a]$. Therefore $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ does not waste any time or change its output by trying $\rho[i \leftarrow 1-a]$ before $\rho[i \leftarrow a]$. ■

The following notion of a "clever unwinding" is only used in the proof of Lemma 7.10.

Lemma and Definition 7.9 (Clever Unwinding) *Let* $\mathbf{S}$ *be a scheduler. Let* $b \in \{0,1\}^m$ *and let* $\sigma \in \{\mathrm{left}, \mathrm{right}\}^\ell$ *be a path string. Then there exists a unique path string* $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$ *such that* $\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b) = \mathrm{BT}^{\sigma^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b)$*: that is to say, if* $\mathrm{BT}_{\mathbf{S}}$ *follows the path* $\sigma^{\mathcal{C}}_{\mathbf{S}}$*, then it arrives at the same partial assignment that* $\mathcal{C}(\mathbf{S})$ *reaches after following the path* σ*. (The notation* $\mathrm{BT}^{\sigma}_{\mathbf{S}}$ *is introduced in Definition 6.3.) We call* $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$ *the* clever unwinding *of* σ *under* b.

The clever unwinding of any prefix of σ *is a prefix of the clever unwinding of* σ.

Now, let $b' \in \{0,1\}^m$ *be such that for every prefix* $v \sqsubseteq \sigma^{\mathcal{C}}_{\mathbf{S}}(b)$, $\mathrm{BT}^{v}_{\mathbf{S}}(b') = \mathrm{BT}^{v}_{\mathbf{S}}(b)$*. Let* $C \in \mathrm{Cl}(\mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b)))$ *and assume further that* $b'_C = b_C$*. Then if* P *is a* $c/2$*-robust predicate,* $\sigma^{\mathcal{C}}_{\mathbf{S}}(b') = \sigma^{\mathcal{C}}_{\mathbf{S}}(b)$.

Proof: The general idea is that the backtracking tree of the clever algorithm $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ is the same as the backtracking tree of the original algorithm $\mathrm{BT}_{\mathbf{S}}$, except some left- and right- children have been switched. This gives a natural correspondence between paths in $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, *^n))$ and paths in $\mathrm{Tree}(\mathrm{BT}_{\mathcal{C}(\mathbf{S})}(b, *^n))$.

More precisely, pick $i \in \{1, \ldots, \ell\}$. We specify explicitly the value of $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)_i$, the ith character of the path string $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$. Consider the node in the clever backtracking tree $\mathrm{Tree}(\mathrm{BT}_{\mathcal{C}(\mathbf{S})}(b, *^n))$ reached from the root of the tree by following the branches $\sigma_1, \ldots, \sigma_{i-1}$. We set $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)_i = \sigma_i$ except when the clever scheduler $\mathcal{C}(\mathbf{S})$ makes a "clever" choice different from the original scheduler $\mathbf{S}$ at this node, in which case we assign to $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)_i$ the opposite of σ_i.

Now, let $b' \in \{0,1\}^m$ and $C \in \mathrm{Cl}(\mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b)))$ as in the statement of the lemma. To show $\sigma^{\mathcal{C}}_{\mathbf{S}}(b) = \sigma^{\mathcal{C}}_{\mathbf{S}}(b')$, it suffices to show that along the path σ, the clever scheduler $\mathcal{C}(\mathbf{S})$ makes a different choice from the original scheduler $\mathbf{S}$ at exactly the same places on input b as on input b'.

Let $v \sqsubset \sigma^{\mathcal{C}}_{\mathbf{S}}(b)$ be a strict prefix, let $\rho_v = \mathrm{BT}^{v}_{\mathbf{S}}(b) = \mathrm{BT}^{v}_{\mathbf{S}}(b')$, and let $(j, a) = \mathbf{S}(b, \rho_v) = \mathbf{S}(b', \rho_v)$. Since $\mathrm{Vars}(\rho_v) \cup \{j\} \subseteq \mathrm{Vars}(\mathrm{BT}^{\sigma^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b)) = \mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b))$, we know that $C \in \mathrm{Cl}(\mathrm{Vars}(\rho_v) \cup \{j\})$. Because $b_C = b'_C$, by Lemma 6.22 and the definition of local forcing (Definition 7.5), $j \leftarrow 1 - a$ is locally forced by b iff it is locally forced by b', so $\mathcal{C}(\mathbf{S})$ makes a different choice from $\mathbf{S}$ at (b, ρ_v) iff it does for (b', ρ_v). ■

Lemma 7.10 *If* $\mathbf{S}$ *is an* (s,t)*-myopic scheduler,* P *is a* $c/2$*-robust predicate, and* $s \leq cr/4$*, then* $\mathcal{C}(\mathbf{S})$ *is* $(s, t + 2c^{-1}s)$*-myopic.*

Proof: Let $b \in \{0,1\}^m$ and let $\sigma \in \{\text{left}, \text{right}\}^s$ be a path string. Our task is to find a set $T \subseteq L$ of size $\leq t + 2c^{-1}s$ such that for every prefix $\tau \sqsubseteq \sigma$, the partial assignment $\mathrm{BT}^{\tau}_{\mathcal{C}(\mathbf{S})}(b)$ depends only on $b_{T(b)}$, and T itself also depends only on $b_{T(b)}$.

Apply the myopic property of $\mathbf{S}$ with b and the clever unwinding path string $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$, to get a set $T_0 \subseteq L$ such that $|T_0| \leq t$ and whenever $b'_{T_0} = b_{T_0}$, $\mathrm{BT}_{\mathbf{S}}$ makes the same decisions along the path $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$ on inputs b and b':

$$\forall v \sqsubseteq \sigma^{\mathcal{C}}_{\mathbf{S}}(b),\ \mathrm{BT}^{v}_{\mathbf{S}}(b') = \mathrm{BT}^{v}_{\mathbf{S}}(b). \tag{7.1}$$

Next, let $C \in \mathrm{Cl}(\mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b)))$ be such that $|C| < 2c^{-1}s$ (Lemma 6.19). Finally, set $T = T_0 \cup C$.

Notice that $|T| \leq t + 2c^{-1}s$. To prove $\mathcal{C}(\mathbf{S})$ is $(s, t + 2c^{-1}s)$-myopic, we consider $b' \in \{0,1\}^m$ such that $b'_T = b_T$. By Lemma 7.9 we have $\sigma^{\mathcal{C}}_{\mathbf{S}}(b') = \sigma^{\mathcal{C}}_{\mathbf{S}}(b)$. Therefore by the myopic property of $\mathbf{S}$, the set T_0 for b' is the same set T_0 as we get for b. Moreover, for any $\tau \sqsubseteq \sigma$, since $\tau^{\mathcal{C}}_{\mathbf{S}}(b)$ is a prefix of $\sigma^{\mathcal{C}}_{\mathbf{S}}(b)$, by (7.1) we have $\mathrm{BT}^{\tau^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b) = \mathrm{BT}^{\tau^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b')$. Therefore, by the definition of clever unwinding, we have

$$\mathrm{BT}^{\tau}_{\mathcal{C}(\mathbf{S})}(b) = \mathrm{BT}^{\tau^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b) = \mathrm{BT}^{\tau^{\mathcal{C}}_{\mathbf{S}}(b)}_{\mathbf{S}}(b') = \mathrm{BT}^{\tau^{\mathcal{C}}_{\mathbf{S}}(b')}_{\mathbf{S}}(b') = \mathrm{BT}^{\tau}_{\mathcal{C}(\mathbf{S})}(b').$$

In particular, $\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b) = \mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b')$; hence the same $C \in \mathrm{Cl}(\mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b')))$ is chosen for b' as is chosen for b (provided that the choice of C only depends on $\mathrm{Vars}(\mathrm{BT}^{\sigma}_{\mathcal{C}(\mathbf{S})}(b))$). Given that T_0 and C are the same for b' and b, we have shown that $T = T_0 \cup C$ is also the same for b' and b. The proof is complete. ■

The following is a technical lemma that will be used in the proof of the lemma after. It concerns the distinction between assignments a bactracking algorithm makes that are locally forced and those which are not locally forced. Loosely speaking, the lemma says that closure of the set of locally unforced variables is large enough that its neighborhood includes all the variables which were locally forced. Combined with Lemma 6.20, this allows us to ignore locally forced variables when taking closures.

Lemma 7.11 *Let* $\mathbf{S}$ *be any scheduler and* $\sigma \in \{\text{left}, \text{right}\}^*$ *be any path string. Assume* P *is a* $c/2$*-robust predicate. Let* $\rho = \mathrm{BT}^{\sigma}_{\mathbf{S}}(b)$ *(Definition 6.3). Let* $T \subseteq \mathrm{Vars}(\rho)$ *be any set of variables whose assignments were locally forced along the path* σ*: that is, for all* $i \in T$*, if* ρ' *was the partial assignment just before variable* i *was assigned, then* $i \leftarrow \rho(i)$ *is locally forced by* b *and* ρ'*.*

Then for any $C \in \mathrm{Cl}(\mathrm{Vars}(\rho) \setminus T)$ *we have* $T \subseteq \Gamma(C)$*.*

Proof: Let ρ^s denote the partial assignment reached after taking s steps along the path σ, and let $T^s = T \cap \mathrm{Vars}(\rho^s)$. We will prove by induction on s that

$$\forall C \in \mathrm{Cl}(\mathrm{Vars}(\rho^s) \setminus T^s),\ T^s \subseteq \Gamma(C). \tag{$*$}$$

The base case is trivial: $T^0 = \varnothing$.

Now, assume $(*)$ holds for $s-1$. Let $(i, a) = \mathbf{S}(b, \rho^{s-1})$, the index and value of the next assignment made by $\mathrm{BT}_\mathbf{S}$. Let $C \in \mathrm{Cl}(\mathrm{Vars}(\rho^s) \setminus T^s)$. Since Cl is monotone, $C \in \mathrm{Cl}(\mathrm{Vars}(\rho^{s-1}) \setminus T^{s-1})$, so $T^{s-1} \subseteq \Gamma(C)$ by $(*)$. If $T^s = T^{s-1}$, then we are done. Otherwise $T^s = T^{s-1} \cup \{i\}$, and $i \leftarrow a$ is locally forced by b and ρ^{s-1}. Define the partial assignment $\rho^* \in \{0,1\}^n$ be the same as ρ^{s-1} except that $\rho^*_i = *$ for every $i \in T$. Notice that $\mathrm{Vars}(\rho^*) = \mathrm{Vars}(\rho^{s-1}) \setminus T^s$. We know that $\rho^*[i \leftarrow a]$ is locally consistent, but $\rho^*[i \leftarrow 1-a]$ is not. Lemma 6.22 says that local consistency is entirely determined by variables in $\Gamma(C)$, because $C \in \mathrm{Cl}(\mathrm{Vars}(\rho^*) \cup \{i\})$. Therefore $i \in \Gamma(C)$, so $T^s = T^{s-1} \cup \{i\} \subseteq \Gamma(C)$. ■

Finally, we come to the reason for replacing our algorithm with a clever algorithm. The following lemma will be used in the next section.

Lemma 7.12 *Assume P is a $c/2$-robust predicate. Any clever backtracking algorithm* $\mathrm{BT}_\mathbf{S}$ *will make at least $\lfloor cr/4 \rfloor$ locally unforced assignments on its leftmost branch. Let ρ be the partial assgnment reached after making $\lfloor cr/4 \rfloor$ locally unforced assignments. Then ρ is locally consistent, and* $\mathrm{Cl}(\mathrm{Vars}(\rho))$ *is nonempty.*

Proof: Suppose $\mathrm{BT}_\mathbf{S}$ runs for at least s steps on its leftmost branch, resulting in partial assignment ρ, and that $s' \leq s$ of the assignments were not locally forced. It suffices to prove the following two facts:

1. If $s' < \lfloor cr/4 \rfloor$ and ρ is locally consistent then the next assignment $\mathrm{BT}_\mathbf{S}$ makes will be locally consistent.

2. If $s' \leq \lfloor cr/4 \rfloor$, then $\mathrm{Cl}(\mathrm{Vars}(\rho))$ is nonempty (and therefore $s < n$).

We will prove the second fact first. Let $T \subseteq \mathrm{Vars}(\rho)$ be the set of variables that were locally forced. Then by Lemma 7.11,

$$T \subseteq \bigcap_{C \in \mathrm{Cl}(\mathrm{Vars}(\rho) \setminus T)} \Gamma(C).$$

By Lemma 6.20 and since Cl is monotone, $\mathrm{Cl}(\mathrm{Vars}(\rho)) = \mathrm{Cl}(\mathrm{Vars}(\rho) \setminus T)$. Now, $|\mathrm{Vars}(\rho) \setminus T| = s' \leq cr/4$, so Lemma 6.19 completes our proof of the second fact.

Having proved the second fact, we now prove the first. Let $(i, a) = \mathbf{S}(b, \rho)$ be the next assignment chosen by the scheduler $\mathbf{S}$. Assume for a contradiction that $\rho[i \leftarrow a]$ is not locally consistent. Since $\mathbf{S}$ is clever, then, $\rho[i \leftarrow 1-a]$ is not locally consistent either. By Lemma 6.23 (using the fact just proved), ρ is not locally consistent, which contradicts our assumption. ■

7.3 The Probability of a Correct Guess is Small

In this section, we assume $\mathbf{S}$ is a clever scheduler which is either myopic or drunken. Sample $x \in \{0,1\}^n$ uniformly at random and let $b = f_{P,G}(x)$. Run $\mathrm{BT}_{\mathbf{S}}$ for s steps, and call the resulting partial assignment ρ. Since $\mathbf{S}$ is clever, ρ is locally consistent. In what follows, we show that with high probability ρ will disagree with x.

7.3.1 Main Myopic Lemma

Lemma 7.13 (Main Myopic Lemma) *Let $\mathrm{BT}_{\mathbf{S}}$ be any clever (s,t)-myopic backtracking algorithm for $s \le cr/4$ and $t \le r$, and import the parameters and assumptions from Setups 7.2 and 7.3. Run $\mathrm{BT}_{\mathbf{S}}$ on input b. Let ρ be the partial assignment consisting of the first s choices made by $\mathrm{BT}_{\mathbf{S}}$; in other words, $\rho = \mathrm{BT}_{\mathbf{S}}^{\mathrm{left}^s}(b)$. (By Lemma 7.12, ρ is well-defined.) Then the probability that ρ can be extended to x is at most*

$$p = 2^{-u}\left(\frac{1+2\epsilon_{\mathrm{bal}}}{1-2\epsilon_{\mathrm{bal}}}\right)^t.$$

(u is defined in Setup 7.3.)

Proof: (Within this proof we will always have $\sigma = \mathrm{left}^s$, so we shorten the notation T_σ to T.)

For any $\hat{b} \in \{0,1\}^m$, define the event

$$E_{\hat{b}} = \{x \in \{0,1\}^n : f(x)_{T(\hat{b})} = \hat{b}_{T(\hat{b})}\}.$$

We begin by showing that the events $\{E_{\hat{b}} : \hat{b} \in \{0,1\}^n\}$ form a partition of the sample space $\{x : x \in \{0,1\}^n\}$. These events cover the whole sample space because $x \in E_{f(x)}$ for every $x \in \{0,1\}^n$. Now assume, for $\hat{b}, \hat{b}' \in \{0,1\}^m$, the events $E_{\hat{b}}$ and $E_{\hat{b}'}$ share a sample point x. Then $f(x)_{T(\hat{b})} = \hat{b}_{T(\hat{b})}$, hence, by the second property of myopic backtracking algorithms in Definition 6.5, we have $T(f(x)) = T(\hat{b})$. Similarly, we have $T(f(x)) = T(\hat{b}')$. Thus, $T(\hat{b}) = T(\hat{b}')$, and $\hat{b}_{T(\hat{b})} = f(x)_{T(\hat{b})} = \hat{b}'_{T(\hat{b}')}$. This means that any two intersecting events $E_{\hat{b}}$ and $E_{\hat{b}'}$ are actually equal.

Since the events $E_{\hat{b}}$ form a partition, we can prove the Lemma by showing that for every $\hat{b}$, the probability that ρ can be extended to x, conditioned on event $E_{\hat{b}}$, is at most p. Therefore from now on we fix $\hat{b}$. By the first property of myopic backtracking algorithms in Definition 6.5, conditioning on event $E_{\hat{b}}$ fixes ρ. By applying Lemma 6.17 with $W = \mathrm{Vars}(\rho)$, there exists a set of input nodes $U \subseteq \mathrm{Vars}(\rho)$ of size $\le u$ such that every subset of $T(b)$ has boundary expansion $> h$ outside U. We know that

$$\Pr[\rho \text{ can be extended to } x | E_{\hat{b}}] \le \Pr[x_U = \rho_U | E_{\hat{b}}],$$

so it suffices to show

$$\Pr[x_U = y | E_{\hat{b}}] \le p, \tag{7.2}$$

for $y = \rho_U$.

Here is an overview of the proof of (7.2) for any $y \in \{0,1\}^{|U|}$. We first show that x_U has little influence on the distribution of $b_{T(\hat{b})}$. Then by Bayes' rule, we conclude that the bits $b_{T(\hat{b})}$ do not contain much information about x_U.

Order the nodes in $T(\hat{b})$ as $v_1, v_2, \ldots, v_{|T(\hat{b})|}$ such that for every $1 \le i \le |T(\hat{b})|$, we have $|\Gamma(T_i) \setminus (\Gamma(T_{i-1}) \cup U)| \ge h+1$ for $T_i = \{v_1, \ldots, v_i\}$. This ordering is possible because every subset of $T(\hat{b})$ has boundary $> h$ outside U. For any $y \in \{0,1\}^{|U|}$, we have

$$\begin{aligned}\Pr[E_{\hat{b}} | x_U = y] &= \prod_{i=1}^{|T(\hat{b})|} \Pr[b_{v_i} = \hat{b}_{v_i} | b_{T_{i-1}} = \hat{b}_{T_{i-1}}, x_U = y] \\ &\in [(\tfrac{1}{2} - \epsilon_{\text{bal}})^{|T(\hat{b})|}, (\tfrac{1}{2} + \epsilon_{\text{bal}})^{|T(\hat{b})|}],\end{aligned}$$

since P is $(h, \epsilon_{\text{bal}})$-balanced and for every $1 \le i \le |T(\hat{b})|$, v_i has at least $h+1$ neighbors outside $\Gamma(T_{i-1})$ and U. Using Bayes' rule, for any $y, y' \in \{0,1\}^{|U|}$,

$$\begin{aligned}\frac{\Pr[x_U = y' | E_{\hat{b}}]}{\Pr[x_U = y | E_{\hat{b}}]} &= \frac{\Pr[E_{\hat{b}} | x_U = y'] \Pr[x_U = y']}{\Pr[E_{\hat{b}} | x_U = y] \Pr[x_U = y]} \\ &= \frac{\Pr[E_{\hat{b}} | x_U = y']}{\Pr[E_{\hat{b}} | x_U = y]} \\ &\ge \left(\frac{1-2\epsilon}{1+2\epsilon}\right)^{|T(\hat{b})|}.\end{aligned}$$

Fixing y and summing the above inequality over all $y' \in \{0,1\}^{|U|}$, we get

$$\frac{1}{\Pr[x_U = y | E_{\hat{b}}]} \ge 2^{|U|} \left(\frac{1-2\epsilon}{1+2\epsilon}\right)^{|T(\hat{b})|}.$$

Equation (7.2) follows immediately. ■

7.3.2 Main Drunken Lemma

Next, we will present the main lemma for drunken backtracking algorithms.

Lemma 7.14 (Main Drunken Lemma) *Let* $\mathbf{S}$ *be any drunken scheduler, and consider the backtracking algorithm* $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ *which uses the clever version of* $\mathbf{S}$*. Let* $s' \le cr/4$*, and import all the parameters and assumptions from Setups 7.2 and 7.4. Run* $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ *on input* b*. Let* $\rho^{s'}$ *be the partial assignment consisting of the choices made by* $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ *on its left-most branch up to the point where it has made* s' *non-forced assignments. (By Lemma 7.12,* $\rho^{s'}$ *is well-defined.) Then the probability that* ρ *can be extended to* x *is exactly* $2^{-s'}$.

Proof: We shall prove by induction on s' that the probability that $\rho^{s'}$ can be extended to x is $2^{-s'}$, even after conditioning on the value of $x \in \{0,1\}^n$. The statement for the base case $s' = 0$ is clear: $*^n$ can always be extended to x.

For $s' \geq 1$, let $\rho^{s'}_{\text{bef}}$ be the partial assignment just before the s'th non-forced assignment is made. Since $\rho^{s'}_{\text{bef}}$ is obtained from $\rho^{s'-1}$ by adding some forced assignments, we know that $\rho^{s'}_{\text{bef}}$ can be extended to x if and only if $\rho^{s'-1}$ can be.

Furthermore, to get from $\rho^{s'}_{\text{bef}}$ to $\rho^{s'}$, the drunken scheduler $\mathbf{S}$ makes a choice (i, a) where $\Pr[a = 0] = \Pr[a = 1] = 1/2$ given i and all the events that have happened so far in the algorithm. Since this step is not locally forced, the clever version of $\mathbf{S}$ allows that the backtracking algorithm $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ first try the choice of a for the ith variable, which is inconsistent with x_i with probability 1/2. Therefore,

$$\begin{aligned}\Pr[\rho^{s'} \text{ can be extended to } x] =& \Pr[\rho^{s'}_{\text{bef}} \text{ can be extended to } x]\cdot \\ & \Pr[a = x_i | \rho^{s'}_{\text{bef}} \text{ can be extended to } x] \\ =& \Pr[\rho^{s'-1} \text{ can be extended to } x] \cdot 1/2 \\ =& 2^{-s'+1} \cdot 1/2 = 2^{-s'}.\end{aligned}$$

■

Let us compare the lemma above with the similar result [30, Lemma 6]. In our argument above, we only need to show that at least one of the drunken choices made by the algorithm is unlucky, whereas [30] uses a concentration result (Chernoff's bound) to show that approximately half of the drunken choices is unlucky. On the other hand, the argument of [30] deals with the many-to-oneness of Goldreich's function directly, whereas we deal with this issue of many-to-oneness separately in Section 7.6.

7.4 Refutation of Locally Consistent but Globally Inconsistent Partial Assignments

Lemma 7.15 *Consider running a backtracking algorithm* $\mathrm{BT}_{\mathbf{S}}$ *for solving* $f_{P,G}(x) = b$. *Import the parameters and assumptions from Setup 7.2. Assume that* $\mathrm{BT}_{\mathbf{S}}$ *reaches a partial assignment* ρ, *where* $\mathrm{Cl}(\mathrm{Vars}(\rho))$ *is nonempty and* ρ *is locally consistent, but* ρ *cannot be extended to any solution of* $f_{P,G}(x) = b$. *Then* $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b, \rho))$, *the backtracking subtree rooted at* ρ, *has size at least* $2^{(c/2-h)r/4-d}$.

In order to prove the above, we will make use of the well-known connection between the size of backtracking trees and the size of "tree-like resolution proofs."

We will make use of the following theorem from [13]. 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.

Theorem 7.16 [13, Theorem 3.3]
The size of any tree-like resolution refutation of a CNF 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.

Proof: [Lemma 7.15] (Our proof follows the proof of [4, Lemma 9], which in turn uses the Ben-Sasson-Wigderson measure of [13].)

We define a CNF formula $\Psi(x)$ with width d 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 $P(x_{G_{i,1}}, x_{G_{i,2}}, \ldots, x_{G_{i,d}}) = b_i$. For any $S \subseteq L$ and $y \in \{0,1\}^S$, we denote by $\Psi[x_S = y]$ the CNF formula of width $\leq d$, whose variables are $x_{L \setminus S}$, and which is obtained from Ψ by replacing the variables x_S with y.

The bactracking subtree $\mathrm{Tree}(\mathrm{BT}_{\mathbf{S}}(b,\rho))$ can be converted to a tree-like resolution proof (of the same size as the subtree) that the formula $\Psi[x_{\mathrm{Vars}(\rho)} = \rho_{\mathrm{Vars}(\rho)}]$ is unsatisfiable (See for example [32, Proposition 1]). Thus, by Theorem 7.16, it suffices to show:

Every resolution refutation of $\Psi[x_{\mathrm{Vars}(\rho)} = \rho_{\mathrm{Vars}(\rho)}]$ has width at least $w \overset{\mathrm{def}}{=} (c/2 - h)r/4$. (*)

By assumption, $\mathrm{Cl}(\mathrm{Vars}(\rho))$ contains some set I of size at most $r/2$ which is closed with respect to $\mathrm{Vars}(\rho)$. Since ρ is locally consistent, ρ can be extended to some $x' \in \{0,1\}^n$ such that $f(x')_I = b_I$. Let $J = \mathrm{Vars}(\rho) \cup \Gamma(I)$. Instead of proving (*), we will prove the stronger statement that every resolution refutation of $\Psi[x_J = x'_J]$ has width at least w.

For any clause C on the variables $x_{L \setminus J}$ and a set $I' \subseteq R \setminus I$, we say I' *implies* C if for every x such that $(f(x)_{I'} = b_{I'} \wedge x_J = x'_J)$, the clause C is satisfied by x. We define the *measure* of C to be

$$\mu(C) = \min\{|I'| : I' \subseteq R \setminus I, \text{ and } I' \text{ implies } C\}.$$

Assume $\mu(C) \leq r/2$, and let I' be a smallest subset of $R \setminus I$ which implies C. No vertex $i \in I'$ contains more than h neighbors in $L \setminus J$ that do not appear in C or $\Gamma(I' \setminus \{i\})$, since otherwise, by the h-robustness of the predicate P, $I' \setminus \{i\}$ would also imply C. Since C is closed with respect to $\mathrm{Vars}(\rho)$ and $|I'| \leq r/2$, we know $|\partial I' \setminus J| \geq c|I'|/2$, so C consists of at least $(c/2 - h)\mu(C)$ variables. Thus we have proved:

$$\text{For any clause } C,\ \mu(C) \text{ is either } \leq \frac{\mathrm{width}(C)}{c/2 - h} \text{ or } > r/2. \qquad (\#)$$

We have:

1. $\mu(C) = 1$ for any clause C in the CNF formula $\Psi[x_J = x'_J]$.

2. $\mu(C) > r/2$ for the empty clause $C = \text{False}$, because of (#) and because $\mu(\text{False}) > 0$.

3. μ is subadditive: If C_2 is the resolution of C_0 and C_1, then $\mu(C_2) \leq \mu(C_0) + \mu(C_1)$, because whenever I'_0 implies C_0 and I'_1 implies C_1, it follows that $I'_0 \cup I'_1$ implies C_2.

Putting 1, 2 and 3 together, we find that every resolution refutation of $\Psi[x_J = x'_J]$ contains a clause C whose measure is in the range $(r/4, r/2]$. By (#), the width of C is at least $w = (c/2 - h)r/4$, which completes the proof. ■

7.5 Inverting Goldreich's Function Exactly

We are now ready to prove that myopic and drunken backtracking algorithms cannot efficiently determine the exact value of x given $f(x)$. Here we stop for a moment to state and prove this partial result before we go on to complete the proof of Theorem 7.1.

Lemma 7.17

- *Assume* $\mathrm{BT}_{\mathbf{S}}$ *is an* (s,t)*-myopic backtracking algorithm, where* $s \le cr/4$ *and* $t \le r/2$*. Import all the parameters and assumptions from Setups 7.2 and 7.3. Then the probability that* $\mathrm{BT}_{\mathbf{S}}$ *on input* b *runs in time* $\le 2^{(c/2-h)r/4-d}$ *and returns the exact solution* x *is at most*
$$2^{-u}\left(\frac{1+2\epsilon_{\text{bal}}}{1-2\epsilon_{\text{bal}}}\right)^r.$$
- *Assume* $\mathrm{BT}_{\mathbf{S}}$ *is a drunken backtracking algorithm, and import all the parameters and assumptions from Setups 7.2 and 7.4. Then the probability that* $\mathrm{BT}_{\mathbf{S}}$ *on input* b *rens in time* $\le 2^{(c/2-h)r/4-d}$ *and returns the exact solution* x *is at most*
$$2^{-\lfloor \frac{cr}{4} \rfloor}.$$

Proof: By Lemma 7.8, it suffices to complete the proof for the clever version of the algorithm $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$.

- *Myopic Case.*

  By Lemma 7.10, $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ is (s,r)-myopic. Let ρ be the partial assignment consisting of the first s choices made by $\mathrm{BT}_{\mathbf{S}}$. Then by Lemma 7.13, applied to $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$, the probability that ρ can be extended to x is at most $2^{-u}\left(\frac{1+2\epsilon_{\text{bal}}}{1-2\epsilon_{\text{bal}}}\right)^r$.

- *Drunken Case.*

  Let ρ be the partial assignment consisting of the first $\lfloor cr/4 \rfloor$ choices made by $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$. Then by Lemma 7.14, the probability that ρ can be extended to x is at most $2^{-\lfloor cr/4 \rfloor}$.

Henceforth, assume ρ cannot be extended to x. Our goal is to prove that in this case, $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ will either return some $x' \ne x$, or that its running time will be at least $2^{(c/2-h)r/4-d}$

Since $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ will explore the entire backtracking subtree rooted at ρ before backtracking, it cannot return x before finishing that exploration. We consider two cases: either ρ can be extended to some $x' \in f^{-1}(f(x))$, or it can be extended to no such x'

In the first case, $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ must return some $x' \in f^{-1}(f(x))$ before it finishes exploring the subtree, so $x' \ne x$.

In the second case, $\mathrm{BT}_{\mathcal{C}(\mathbf{S})}$ will not return anything before exploring every node of the backtracking subtree rooted at ρ. Lemma 7.12 guarantees all the preconditions for Lemma 7.15, so the number of nodes in that subtree is at least $2^{(c/2-h)r/4-d}$. ■

7.6 Accounting for the Size of Pre-Images

So far, we have shown that given $f(x)$, certain backtracking algorithms cannot efficiently guess x. We now show that such algorithms also cannot efficiently find any $x' \in f^{-1}(f(x))$, assuming the pre-images $f^{-1}(f(x))$ are small enough on average.

Lemma 7.18 *Let* $\mathrm{BT}_{\mathbf{S}}$ *be a backtracking algorithm, choose* $x \in \{0,1\}^n$ *uniformly at random and let* $b = f(x)$*. Let* E *be the event that when solving* $f(x') = b$*,* $\mathrm{BT}_{\mathbf{S}}$ *runs in time* $\leq t$ *and returns the exact solution* x*. Let* F *be the event that* $\mathrm{BT}_{\mathbf{S}}$ *runs in time* $\leq t$ *and returns any* $x' \in f^{-1}(b)$*.*

Then $\Pr[F] \leq 2\sqrt{M \Pr[E]}$*, where* $M = \mathbf{E}_{x \in \{0,1\}^n}[|f^{-1}(f(x))|]$*.*

Proof: Consider the event

$$H = \{x \in \{0,1\}^n : |f^{-1}(f(x))| \leq M'\},$$

where M' will be chosen shortly. We have $\Pr[F] \leq \Pr[F, H] + \Pr[H^c]$. We can upper-bound $\Pr[F, H]$ in terms of $\Pr[E]$:

$$\begin{aligned}\Pr[E] &\geq \Pr[E|F, H] \cdot \Pr[F, H] \\ &\geq \frac{1}{M'} \cdot \Pr[F, H],\end{aligned}$$

and we get $\Pr[F, H] \leq M' \Pr[E]$. To upper-bound $\Pr[H^c]$, we use Markov's inequality on the size of the preimage of $f(x)$, and we get $\Pr[H^c] \leq M/M'$. We complete the proof by taking $M' = \sqrt{M/\Pr[E]}$, so that

$$\Pr[F] \leq \Pr[F, H] + \Pr[H^c] \leq M' \Pr[E] + M/M' = 2\sqrt{M \Pr[E]}.$$

■

7.7 Proof of Theorem 7.1

Proof: We prove the myopic and drunken cases at the same time. Consider the events E and F from the statement of Lemma 7.18. Theorem 7.1 can be restated in terms of the event F as follows:

- In the myopic case, $\Pr[F] \leq M^{1/2} 2^{1-u/2} \left(\frac{1+2\epsilon_{\mathrm{bal}}}{1-2\epsilon_{\mathrm{bal}}}\right)^{r/2}$.
- In the drunken case, $\Pr[F] \leq M^{1/2} 2^{1-\lfloor \frac{cr}{4} \rfloor/2}$.

By Lemma 7.18, it suffices to bound the probability of E as follows:

- In the myopic case, $\Pr[E] \leq 2^{-u} \left(\frac{1+2\epsilon_{\mathrm{bal}}}{1-2\epsilon_{\mathrm{bal}}}\right)^{r}$.
- In the drunken case, $\Pr[E] \leq 2^{-\lfloor \frac{cr}{4} \rfloor}$.

These two bounds are proved in Lemma 7.17. ■

7.8 Coping With Imperfect Expansion

One way to choose the graph $G \in [n]^{n\times d}$ is uniformly at random. Such a graph will fail the expansion condition of Theorem 7.1 with probability $\geq 1/n^d$. On the other hand, the probability that the graph will fail any of the other conditions of the theorem is exponentially small, for a suitable choice of predicate P. In this section, we show that the expansion condition of Theorem 7.1 can be weakened, so that a random graph fails with exponentially small probability.

Definition 7.19 (Removing Nodes) *Let $G \in [n]^{m\times d}$ be a bipartite graph with nodes $L = [n]$ on the left and $R = [m]$ on the right. Let $I \subseteq R$. Then we define $G \setminus I \in [n]^{(m-|I|)\times d}$ to be the graph G without the nodes in I or the edges connected to them.*

Definition 7.20 (Imperfect Expansion) *A graph $G \in [n]^{n\times d}$ is an r_{bad}-*imperfect (r, c)-boundary expander *if there exists a subset $I_{\text{bad}} \subseteq R$ of size $|I_{\text{bad}}| \leq r_{\text{bad}}$ such that $G \setminus I_{\text{bad}}$ is an (r, c)-boundary expander (Definition 6.14). We call I_{bad} an* extraneous set *of G.*

Lemma 7.21 *A random bipartite graph $G \in [n]^{n\times d}$ with n left nodes and n right nodes, and of right-degree d, is with probability $1 - (1/4)^{r_{\text{bad}}}$ an r_{bad}-imperfect (r, c)-boundary expander for any $c = d - \Omega(d)$, provided $r + r_{\text{bad}} \leq r_{\max}(n, d, c)$, where $r_{\max} = \Omega(n/d)$.*

Proof: Let $c' = (c + d)/2$. Let I_{bad} be a largest set of right-nodes $I \subseteq R$ of size at most $r + r_{\text{bad}}$ such that $|\Gamma(I)| \leq c'|I|$. Then $G \setminus I_{\text{bad}}$ is an $(r + r_{\text{bad}} - |I_{\text{bad}}|, c')$-expander, and hence by Lemma 6.15, an $(r + r_{\text{bad}} - |I_{\text{bad}}|, c)$-boundary expander. All that remains is to show $|I_{\text{bad}}| \leq r_{\text{bad}}$.

Note that $c' = d - \Omega(d)$, so from the proof of Lemma 6.16, the probability that there exists a set of size $i \in [r_{\text{bad}} + 1, r + r_{\text{bad}}]$ with expansion $\leq c'$ is at most $\sigma_{i\in[r_{\text{bad}}+1,r+r_{\text{bad}}]} a_i^i$, where a_i is defined in the same proof and is shown to be at most 1/4. Thus the sum is at most $(1/4)^{r_{\text{bad}}}$ ■

Now here we state the theorem corresponding to Theorem 7.1 adjusted to imperfect expanders. For applications of this theorem, we refer the reader to Section 7.0.1.

Theorem 7.22

- *Assume* $\text{BT}_{\mathbf{S}}$ *is an (s, t)-myopic backtracking algorithm, where $s \leq cr/4$ and $t \leq r/2$. Import all the parameters and assumptions from Setup 7.23. Then the probability that* $\text{BT}_{\mathbf{S}}$ *on input b finds some $x' \in f^{-1}(b)$ in time $\leq 2^{(c/2-h)r/4-d} - dr_{\text{bad}}$ is at most*

$$M^{1/2} 2^{1+dr_{\text{bad}}-u/2} \left(\frac{1 + 2\epsilon_{\text{bal}}}{1 - 2\epsilon_{\text{bal}}}\right)^{r/2}.$$

- *Assume* $\text{BT}_{\mathbf{S}}$ *is a drunken backtracking algorithm, and import all the parameters and assumptions from Setup 7.24. Then the probability that* $\text{BT}_{\mathbf{S}}$ *on input b finds some $x' \in f^{-1}(b)$ in time $\leq 2^{(c/2-h)r/4-d} - dr_{\text{bad}}$ is at most*

$$M^{1/2} 2^{1+dr_{\text{bad}}-\lfloor \frac{cr}{4} \rfloor/2}.$$

Setup 7.23 (Imperfect-expansion setup for myopic algorithms) *The same as Setup 7.3, except that we do not assume G to be an (r, c)-boundary expander, but instead assume G is an r_{bad}-imperfect (r, c)-boundary expander, where r_{bad} is a positive integer.*

Setup 7.24 (Imperfect-expansion setup for drunken algorithms) *The same as Setup 7.4, except that we do not assume G to be an (r, c)-boundary expander, but instead assume G is an r_{bad}-imperfect (r, c)-boundary expander, where r_{bad} is a positive integer.*

Lemma 7.25 *Let $f = f_{G,P} : \{0,1\}^n \to \{0,1\}^m$ be an instance of Goldreich's function for graph G and predicate P. Let $I \subseteq R$ be a set of right-nodes in G and define $\hat{f} = f_{(G\setminus I),P} : \{0,1\}^n \to \{0,1\}^{m-|I|}$.*

*Let $\text{BT}_{\mathbf{S}}$ be (a) an (s,t)-myopic backtracking algorithm, or (b) a drunken backtracking algorithm, for inverting f. Then there exists (a) an (s,t)-myopic backtracking algorithm or (b) a drunken backtracking algorithm, called $\text{BT}_{\hat{\mathbf{S}}}$, for inverting $\hat{f}$, which has the following relation to $\text{BT}_{\mathbf{S}}$. Sample $x \in \{0,1\}^n$ uniformly at random and let $b = f(x)$ and $\hat{b} = \hat{f}(x)$. If p is the probability that $\text{BT}_{\mathbf{S}}$ on input b runs in time $\leq$ **maxtime** and returns the exact solution x, then the probability that $\text{BT}_{\hat{\mathbf{S}}}$ on input $\hat{b}$ runs in time $\leq$ **maxtime** $+ |\Gamma(I)|$ and returns the exact solution x is at least $p2^{-|\Gamma(I)|}$, where $\Gamma(I)$ is the set of nodes connected to I.*

A first attempt to prove Lemma 7.25 is to have $\text{BT}_{\hat{\mathbf{S}}}$ convert the output $\hat{b} \in \{0,1\}^{m-|\Gamma(I)|}$ into a complete output $b \in \{0,1\}^m$ by guessssing the output values b_I randomly. This guess would be correct with probability $2^{-|I|}$, and $\text{BT}_{\hat{\mathbf{S}}}$ could then try to emulate the original algorithm $\text{BT}_{\mathbf{S}}$ on the complete input b, by making the same decision that $\text{BT}_{\mathbf{S}}$ would make at each node of the backtracking tree. The trouble with this approach is that when $\text{BT}_{\mathbf{S}}$ reaches a node whose partial assignment is inconsistent with a bit in b_I, it will backtrack. $\text{BT}_{\hat{\mathbf{S}}}$ is only allowed to backtrack from nodes which are inconsistent with bits in $\hat{b}$, and so $\text{BT}_{\hat{\mathbf{S}}}$ may be forced to explore backtracking subtrees that $\text{BT}_{\mathbf{S}}$ is allowed to skip. To fix this problem, we guess the input bits $x_{\Gamma(I)}$ instead of guessing the output bits b_I.

Proof: $\text{BT}_{\hat{\mathbf{S}}}$ will begin by guessing the bits $x_{\Gamma(I)}$ uniformly at random. Call the resulting partial assignment ρ_{guess}. We are not interested in the behavior of $\text{BT}_{\hat{\mathbf{S}}}$ after it undoes any of these initial assignments, so it is enough to describe the backtracking subtree below the node ρ_{guess}, which is denoted $\text{Tree}(\text{BT}_{\hat{\mathbf{S}}}(\hat{b}, \rho_{\text{guess}}))$ in Definition 6.3. Extend $\hat{b}$ to $\bar{b} \in \{0,1\}^m$ using the rule

$$\bar{b}_i = \begin{cases} \hat{b}_i & \text{if } i \notin I \\ f(\rho_{\text{guess}})_i & \text{if } i \in I \end{cases}$$

Note that if ρ_{guess} is a correct guess, i.e. $\rho_{\text{guess}} = x_{\Gamma(I)}$, then $\bar{b} = f(x) = b$. To make $\text{Tree}(\text{BT}_{\hat{\mathbf{S}}}(\hat{b}, \rho_{\text{guess}}))$, start with $\text{Tree}(\text{BT}_{\mathbf{S}}(\bar{b}, *^n))$ of the original algorithm and make the following changes:

- First, delete any nodes whose partial assignments disagree with ρ_{guess}.

- Next, add ρ_{guess} to the partial assignment at every node: if the node originally had partial assignment ρ, then the new version of the node will have partial assignment $\rho \cup \rho_{\text{guess}}$, defined as follows:

$$(\rho \cup \rho_{\text{guess}})_j = \begin{cases} *, & j \notin \text{Vars}(\rho) \cup \Gamma(I) \\ \rho_j, & j \in \text{Vars}(\rho) \\ (\rho_{\text{guess}})_j, & j \in \Gamma(I). \end{cases}$$

- Finally, sometimes a node will have the same label as its child, because the scheduler $\mathbf{S}$ decided to assign a variable which is in ρ_{guess}. Replace each such node with its child. (There will only be one child, because we deleted the child which was inconsistent with ρ_{guess}).

The effect of our three changes is to ignore all branches where $\text{BT}_{\mathbf{S}}$ makes an assignment inconsistent with the assignments ρ_{guess} which $\text{BT}_{\hat{\mathbf{S}}}$ already made. In particular, if $\text{BT}_{\mathbf{S}}$ returns x as solution, then the leaf labeled x is not removed from the tree. Thus we have

$$\begin{aligned}
&\Pr_{x,\rho_{\text{guess}}}[\text{BT}_{\hat{\mathbf{S}}}(\hat{b}, *^n) \text{ returns } x \text{ in time } \mathbf{maxtime} + |\Gamma(I)|] \\
&\geq \Pr_{x,\rho_{\text{guess}}}[\rho_{\text{guess}} = x_{\Gamma(I)} \text{ and } \text{BT}_{\hat{\mathbf{S}}}(\hat{b}, \rho_{\text{guess}}) \text{ returns } x \text{ in time } \mathbf{maxtime}] \\
&\geq \Pr_{x,\rho_{\text{guess}}}[\rho_{\text{guess}} = x_{\Gamma(I)} \text{ and } \text{BT}_{\mathbf{S}}(\bar{b}, *^n) \text{ returns } x \text{ in time } \mathbf{maxtime}] \\
&= \Pr_{x,\rho_{\text{guess}}}[\rho_{\text{guess}} = x_{\Gamma(I)} \text{ and } \text{BT}_{\mathbf{S}}(b = f(x), *^n) \text{ returns } x \text{ in time } \mathbf{maxtime}] \\
&= \Pr_{x,\rho_{\text{guess}}}[\rho_{\text{guess}} = x_{\Gamma(I)}] \cdot \Pr_x[\text{BT}_{\mathbf{S}}(b = f(x), *^n) \text{ returns } x \text{ in time } \mathbf{maxtime}] \\
&\geq 2^{-|\Gamma(I)|}p.
\end{aligned}$$

It remains only to show that $\text{BT}_{\hat{\mathbf{S}}}$ is (a) an (s,t)-myopic backtracking algorithm (Definition 6.5) or (b) a drunken backtracking algorithm (Definition 6.7).

- For the drunken case (b), the first $|\Gamma(I)|$ assignments are random values by design. Every other assignment $\text{BT}_{\hat{\mathbf{S}}}$ makes copies a decision made by $\text{BT}_{\mathbf{S}}$, so $\text{BT}_{\hat{\mathbf{S}}}$ is at least as drunk as $\text{BT}_{\mathbf{S}}$.

- For the myopic case (a), $\text{BT}_{\hat{\mathbf{S}}}$ is actually not myopic, since a myopic algorithm is by definition deterministic. However, as we will see, $\text{BT}_{\hat{\mathbf{S}}}$ is a distribution over (s,t)-myopic algorithms. For every $\rho_{\text{guess}} \in \{0,1\}^{|\Gamma(I)|}$, define the deterministic backtracking algorithm $\text{BT}_{\hat{\mathbf{S}}}\,|\rho_{\text{guess}}$ as $\text{BT}_{\hat{\mathbf{S}}}$ conditioned that the value first guessed for $x_{\Gamma(I)}$ is ρ_{guess}. Consider any path string $\hat{\sigma} \in \{\text{left}, \text{right}\}^s$. Assuming $s > |\Gamma(I)|$, the backtracking tree node $\text{BT}_{\hat{\mathbf{S}}}\,|\rho^{\hat{\sigma}}_{\text{guess}}(\hat{b})$ corresponds to some equivalent node $\text{BT}^{\sigma}_{\mathbf{S}}(\bar{b})$ in the original backtracking tree, for some path string $\sigma \in \{\text{left}, \text{right}\}^{s'}$ where $s' \leq s$. Since $\text{BT}_{\mathbf{S}}$ is (s,t)-myopic, there exists a set $T \subseteq R$ with $|T| \leq t$ such that the decisions made by $\text{BT}_{\mathbf{S}}$ along the path σ depend only on $\bar{b}_T$, and the choice of T also depends only on $\bar{b}_T$. Then for every fixed ρ_{guess}, the decisions made by $\text{BT}_{\hat{\mathbf{S}}}\,|\rho_{\text{guess}}$ depend only on the bits $\hat{b}_{T\setminus I}$, and the set T only depends on $\hat{b}_{T\setminus I}$. (Also if $s \leq |\Gamma(I)|$, one can choose the set T to be the empty set.) Therefore $\mathrm{BT}_{\hat{\mathbf{S}}}\,|\rho_{\text{guess}}$ is (s,t)-myopic for every ρ_{guess}. Choose the ρ_{guess} that maximizes the probability of returning x in time **maxtime**. Then $\mathrm{BT}_{\hat{\mathbf{S}}}\,|\rho_{\text{guess}}$ returns x in time **maxtime** with probability at least $p2^{-|\Gamma(I)|}$.

■

We are now ready to prove the main theorem of this section.

Proof: [Theorem 7.22] Let I_{bad} be an extraneous set of G and let $\hat{f} = f_{(G\setminus I_{\text{bad}}),P}$. Let p be the probability that $\mathrm{BT}_{\mathbf{S}}$ on input b runs in time $\leq 2^{(c/2-h)r/4-d} - dr_{\text{bad}}$ and returns the exact solution x.

By Lemma 7.25, there is an algorithm $\mathrm{BT}_{\hat{\mathbf{S}}}$ which with probability at least $p2^{-dr_{\text{bad}}}$, on input $\hat{f}(x)$, runs in time $\leq 2^{(c/2-h)r/4-d}$ and returns the exact solution x.

Since $\hat{f}$ uses an (r,c)-boundary expander, we can apply Lemma 7.17 and conclude that in the myopic case, $p2^{-dr_{\text{bad}}} \leq 2^{-u}\left(\frac{1+2\epsilon_{\text{bal}}}{1-2\epsilon_{\text{bal}}}\right)^{r}$, and in the drunken case, $p2^{-dr_{\text{bad}}} \leq 2^{-\lfloor\frac{cr}{4}\rfloor}$. Lemma 7.18 completes the proof. ■

Chapter 8

The Size of Pre-images of Goldreich's Function

In this chapter we prove that Goldreich's function has pre-images sufficiently small for Theorem 7.1 to work.

Theorem 8.1 *There exist positive constants c_1 and c_2 such that, for sufficiently large d, if we choose $P : \{0,1\}^d \to \{0,1\}$ uniformly at random, then with probability $1 - 2^{-2^{c_1 d}}$ over the choice of P,*

$$\mathop{\mathbf{E}}_{\substack{G \sim \mathrm{Unif}([n]^{n \times d}) \\ x \sim \mathrm{Unif}(\{0,1\}^n)}} [\# y : f_{P,G}(x) = f_{P,G}(y)] \leq n^{O(1)} 2^{2^{-c_2 d} n}. \tag{8.1}$$

Also, for any constant $c_3 \in (0,1)$, there exists a positive constant c_2 such that the predicate $P_{h,Q}(x_1, \ldots, x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1}, \ldots, x_d)$ satisfies Equation (8.1), for $h \leq c_3 d$ and any h-ary predicate Q.

Furthermore, by Markov's Inequality, for any P which satisfies Equation (8.1) and any $q > 0$, with probability $\geq 1 - n^{O(1)} q$ over the choice of graph $G \in [n]^{n \times d}$,

$$\mathop{\mathbf{E}}_{x \sim \mathrm{Unif}(\{0,1\}^n)} [\# y : f_{P,G}(x) = f_{P,G}(y)] \leq 2^{2^{-c_2 d} n} / q.$$

This theorem was shown for $h = 2$ and $Q(x, y) = x \wedge y$ in [16]. Itsykson [30] pointed out that the same proof works for general predicates Q for $h + 1 < d/4$. The proof for random predicates is original to this work.

Remark 8.2 *Looking at the proof of Theorem 8.1 for random predicates, the following can be observed: Not only does a random predicate P satisfy Equation 8.1 of Theorem 8.1 with probability $\geq 1 - 2^{-2^{c_1 d}}$, but also a random predicate P with probability $\geq 1 - 2^{-2^{c_1 d}}$ can be* verified *to satisfy Equation 8.1 of Theorem 8.1 in time $2^{O(d)}$. More precisely, for the suitable choice of $\delta > 0$:*

- *A random predicate is δ-good (see Definition 8.10) with probability $\geq 1 - 2^{-2^{c_1 d}}$ by Lemma 8.11.*

- *A δ-good predicate satisfies Equation 8.1 of Theorem 8.1 by Lemma 8.13 and the proof of Theorem 8.1.*
- *It is possible to verify in time $2^{O(d)}$ whether a given predicate P is δ-good or not.*

Definition 8.3
Let $\Delta^2 = \{p : \{0,1\}^2 \to \mathbf{R}^{\geq 0} | \sum p = 1\}$ be the set of probability distributions over $\{0,1\}^2$. For $\alpha \in \Delta^2$,

- *We define α^d to be the distribution over $(\{0,1\}^d)^2$ where $(x,y) \sim \alpha^d$ means each $(x_i, y_i) \sim \alpha$ independently. For example, if α is the uniform distribution, α^d is also the uniform distribution, and if α assigns a probability of one to the string* 01*, then α^d assigns a probability of* 1 *to the pair $(0 \cdots 0, 1 \cdots 1)$.*
- *$\mathcal{H}(\alpha)$ denotes the base-*2 *entropy of the distribution: $\mathcal{H}(\alpha) = -\sum_{i,j \in \{0,1\}} \alpha_{i,j} \lg \alpha_{i,j}$.*

Similarly, let $\Delta^2(d) = \{\beta : \{0,1\}^2 \to \mathbf{N} | \sum \beta = d\}$.

- *For a pair $(x,y) \in (\{0,1\}^d)^2$, and for $(a,b) \in \{0,1\}^2$, let $\#ab(x,y)$ be the number of indices i such that $x_i = a$ and $y_i = b$. For example, if $x = 0110$ and $y = 1000$, then $\#00(x,y) = 1$ and $\#10(x,y) = 2$.*
- *Define* **NA** *(for Number of Appearances) by*
  $\mathbf{NA}(x,y) = (\#00(x,y), \#01(x,y), \#10(x,y), \#11(x,y)) \in \Delta^2(d)$.
- *For $\beta \in \Delta^2(d)$, $\mathbf{NA}^{-1}(\beta)$ is a set of pairs $(x,y) \in (\{0,1\}^d)^2$. We shall sometimes use $\mathbf{NA}^{-1}(\beta)$ to denote the uniform probability distribution over that set.*

Definition 8.4 *For a predicate $P : \{0,1\}^d \to \{0,1\}$ and $\alpha \in \Delta^2$, the* probability of equality *of P over α is*

$$\mathrm{PE}(P, \alpha) = \Pr_{(x,y) \sim \alpha^d}[P(x) = P(y)].$$

Lemma 8.5 *There exist positive constants c_1 and c_2 such that, for sufficiently large d, the following holds. Choose $P : \{0,1\}^d \to \{0,1\}$ uniformly at random. Then with probability $1 - 2^{-2^{c_1 d}}$ over the choice of P,*

$$\forall \alpha \in \Delta^2,\ \mathcal{H}(\alpha) + \lg \mathrm{PE}(P, \alpha) \leq 1 + 2^{-c_2 d}. \tag{8.2}$$

Also, for any constant $c_3 \in (0,1)$, there exists a constant $c_2 > 0$ such that $P_{h,Q}$ defined in Theorem 8.1 satisfies Equation (8.2), for $h \leq c_3 d$ and any h-ary predicate Q.

We defer the proof of Lemma 8.5 until after the proof of Theorem 8.1.

Proof: [Theorem 8.1] Let P be any predicate satisfying Equation (8.2) in Lemma 8.5. Then

$$\begin{aligned}
&\mathop{\mathbf{E}}_{G\sim\mathrm{Unif}([n]^{n\times d})}[\#(x,y) : f_{P,G}(x) = f_{P,G}(y)] \\
&= \sum_{x,y\in\{0,1\}^n} \Pr_{G\sim\mathrm{Unif}([n]^{n\times d})}[f_{P,G}(x) = f_{P,G}(y)] \\
&= \sum_{x,y\in\{0,1\}^n} \prod_{i=1}^{n} \Pr_{G_{i,1},\ldots,G_{i,d}\sim\mathrm{Unif}([n])}[P(x_{G_{i,1}},\ldots,x_{G_{i,d}}) = P(y_{G_{i,1}},\ldots,y_{G_{i,d}})] \\
&= \sum_{x,y\in\{0,1\}^n} \prod_{i=1}^{n} \mathrm{PE}(P, \mathbf{NA}(x,y)/n) \\
&= \sum_{\beta\in\Delta^2(n)} \binom{n}{\beta_{00},\beta_{01},\beta_{10},\beta_{11}} \mathrm{PE}(P,\beta/n)^n
\end{aligned}$$

(using Stirling's approximation)

$$\begin{aligned}
&\le \sum_{\beta\in\Delta^2(n)} O(2^{n\mathcal{H}(\beta/n)})\,\mathrm{PE}(P,\beta/n)^n \\
&\le |\Delta^2(n)| \max_{\alpha\in\Delta^2} O(2^{n\mathcal{H}(\alpha)})\,\mathrm{PE}(P,\alpha)^n \\
&= O(n^3) \max_{\alpha\in\Delta^2} [2^{\mathcal{H}(\alpha)}\,\mathrm{PE}(P,\alpha)]^n.
\end{aligned}$$

Lemma 8.5 completes the proof. ■

8.0.1 A Technical Lemma

Lemma 8.6 $\forall\tau \in (0,1],\ \exists\epsilon > 0,\ \forall p \in [0,1]\ \forall d \ge 1,\ \mathcal{H}(p) + \lg(1 + (1-p)^{\tau d}) \le 1 + 2^{-\epsilon d}$.

Proof: Assume $\tau \in (0,1]$ is given. First we show that we can choose positive D and ϵ' such that

$$\forall p \in [0,1],\ \forall d > D,\ \mathcal{H}(p) + \lg(1 + (1-p)^{\tau d}) \le 1 + 2^{-\epsilon' d}. \tag{8.3}$$

We prove this by considering four possible cases for the value of p, namely, $p \in (\epsilon_1, 1]$, $p \in (\epsilon_2/d, \epsilon_1]$, $p \in (2^{-\epsilon_3 d}, \epsilon_2/d]$, $p \in [0, 2^{-\epsilon_3 d}]$, where ϵ_1, ϵ_2, ϵ_3 are positive constants to be chosen. We will choose the numbers D and $\epsilon', \epsilon_1, \epsilon_2, \epsilon_3$ as we go along, but according to the following dependency graph:

$$\begin{tikzcd}
\epsilon_2 \arrow[r] & \epsilon_1 \arrow[d] \arrow[dr] & \\
& \epsilon' & D \\
& \epsilon_3 \arrow[u] \arrow[ur] &
\end{tikzcd}$$

- Case 1: $p > \epsilon_1$. Then

$$\mathcal{H}(p) + \lg(1 + (1-p)^{\tau d}) \le 1 + (1-\epsilon_1)^{\tau d} \lg e \le 1 + 2^{-\epsilon' d},$$

  for $\epsilon_1 < 1$, $\epsilon' \le -\frac{1}{2}\tau \lg(1-\epsilon_1)$, $d > D \ge -2\lg\lg e/(\tau \lg(1-\epsilon_1))$.

For the remaining three cases, p is small. Using the Taylor expansion of lg around 2, we get

$$\lg(1 + (1-p)^{\tau d}) \le 1 + \frac{(1-p)^{\tau d} - 1}{2\ln 2} \le 1 + \frac{e^{-\tau p d} - 1}{2 \ln 2}.$$

- Case 2: $p \in (\epsilon_2/d, \epsilon_1]$. Then

$$\mathcal{H}(p) + \lg(1 + (1-p)^{\tau d}) \le \mathcal{H}(\epsilon_1) + 1 + \frac{e^{-\tau\epsilon_2} - 1}{2\ln 2} \le 1,$$

  if we choose ϵ_1 small enough that $\epsilon_1 \le 1/2$ and $\mathcal{H}(\epsilon_1) \le (1 - e^{-\tau\epsilon_2})/(2\ln 2)$.

For the remaining two cases we fix $\epsilon_2 = (2\tau)^{-1}$. Now, $\tau p d \le \frac{1}{2}$, and we have the approximation

$$\mathcal{H}(p) + 1 + \frac{e^{-\tau p d} - 1}{2\ln 2} \le (p\lg(1/p) + 2p) + 1 - \frac{\tau p d}{4 \ln 2} = 1 + p\left(\lg(1/p) - \frac{\tau}{4\ln 2}d + 2)\right).$$

- Case 3: $p \in (2^{-\epsilon_3 d}, \epsilon_2/d]$.

  For $\epsilon_3 < \frac{\tau}{4\ln 2}$ and $d > D$ for sufficiently large D (depending on ϵ_3): $\lg(1/p) - \frac{\tau}{4\ln 2}d + 2 < 0$.

- Case 4: $p \le 2^{-\epsilon_3 d}$.

  For $\epsilon' \le \frac{1}{2}\epsilon_3$ and $d > D$ for sufficiently large D (depending on ϵ_3): $p\lg(1/p) \le \epsilon_3 d 2^{-\epsilon_3 d} \le 2^{-\epsilon' d}$.

We have proved (8.3). It remains to prove the lemma for $d \in [1, D]$. Let $f(p,d) = \mathcal{H}(p) + \lg(1 + (1-p)^{\tau d})$. Since f is a continuous function on the compact set $[0,1] \times [1, D]$, it achieves a finite maximum $M = f(p_*, d_*)$ on this set. It is easy to see that $M \in (1,2)$. Let $\epsilon = \min\{\epsilon', -D^{-1}\lg(M-1)\}$. Then for $d \in [1, D]$, $f(p,d) \le M \le 1 + 2^{-\epsilon d}$, and for $d \in (D, \infty)$, $f(p,d) \le 1 + 2^{-\epsilon' d} \le 1 + 2^{-\epsilon d}$. ■

8.1 Proof of Lemma 8.5 for the predicate $P_{h,Q}$

For predicates of the form $P_{h,Q}(x_1,\ldots,x_d) = x_1 \oplus \cdots \oplus x_{d-h} \oplus Q(x_{d-h+1},\ldots,x_d)$, we have

$$\begin{aligned}
\mathrm{PE}(P_{h,Q},\alpha) =& \frac{1+\mathbf{E}[(-1)^{P(x)+P(y)}]}{2}\\
=& \frac{1+\left(\prod_{i=1}^{d-h}\mathbf{E}[(-1)^{x_i+y_i}]\right)\mathbf{E}[(-1)^{Q(x_{d-h+1},\ldots,x_d)+Q(y_{d-h+1},\ldots,y_d)}]}{2}\\
\le& \frac{1+\left|\prod_{i=1}^{d-h}\mathbf{E}[(-1)^{x_i+y_i}]\right|}{2}\\
=& \frac{1+|\alpha_{00}+\alpha_{11}-\alpha_{10}-\alpha_{01}|^{d-h}}{2}.
\end{aligned}$$

Take $p = \min\{\alpha_{00}+\alpha_{11}, \alpha_{01}+\alpha_{10}\}$. Given p, the maximum values of $\mathcal{H}(\alpha)$ is acheived when $\alpha_{00} = \alpha_{11}$ and $\alpha_{01} = \alpha_{10}$; thus $\mathcal{H}(\alpha) \le 1 + \mathcal{H}(p)$. By the above, $\mathrm{PE}(P_{h,Q},\alpha) \le (1+(1-2p)^{d-h})/2 \le (1+(1-p)^{d-h})/2$. Therefore it suffices to show $\mathcal{H}(p)+\lg(1+(1-p)^{d-h}) \le 1+2^{-\epsilon d}$. Lemma 8.6 completes the proof with $\tau = (d-h)/d$.

8.1.1 Proof of Lemma 8.5 for Random Predicates

It remains to prove Lemma 8.5 for predicates chosen uniformly at random.

Definition 8.7 *For $\beta \in \Delta^2(d)$, the probability of equality of P over β is*

$$\mathrm{PE}(P,\beta) = \Pr_{(x,y)\sim \mathbf{NA}^{-1}(\beta)}[P(x) = P(y)].$$

$\mathrm{PE}(P,\beta)$ is similar to $\mathrm{PE}(P,\alpha)$, but there are finitely many possible values for $\beta \in \Delta^2(d)$. as opposed to Δ^2 where there are infinitely many possible values for $\alpha \in \Delta^2$. The fact that β has finite range helps: Once you show that given a β, the number $\mathrm{PE}(P,\beta)$ is upper-bounded with high probability over the randomness of P, you can take a union bound to get with high probability over the randomness of P an upper bound on $\mathrm{PE}(P,\beta)$ for *all* β (see Lemma 8.11 below).

Lemma 8.8

$$\mathrm{PE}(P,\alpha) = \mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\mathrm{PE}(P,\beta)].$$

Proof: We use the partition $(\{0,1\}^d)^2 = \bigcup_{\beta\in\Delta^2(d)} \mathbf{NA}^{-1}(\beta)$.

$$\begin{aligned}
&\mathrm{PE}(P,\alpha)\\
=& \Pr_{(x,y)\sim\alpha^d}[P(x)=P(y)]\\
=& \sum_{\beta\in\Delta^2(d)} \Pr_{(x,y)\sim\alpha^d}[P(x)=P(y) | (x,y)\in\mathbf{NA}^{-1}(\beta)] \Pr_{(x,y)\sim\alpha^d}[(x,y)\in\mathbf{NA}^{-1}(\beta)]\\
=& \mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}\left[\Pr_{(x,y)\sim\alpha^d}[P(x)=P(y) | (x,y)\in\mathbf{NA}^{-1}(\beta)]\right].
\end{aligned}$$

For any fixed $\beta \in \Delta^2(d)$, α^d restricted to $\mathbf{NA}^{-1}(\beta)$ is simply the uniform distribution on $\mathbf{NA}^{-1}(\beta)$. Therefore the quantity inside the expectation is equal to $\mathrm{PE}(P, \beta)$. ■

Definition 8.9 (One-Bit Entropy, α_{a*}, α_{*a}, $\mathcal{H}^*(\alpha)$) *Let $\alpha \in \Delta^2$.*

- *For $a \in \{0,1\}$, $\alpha_{a*} \stackrel{\text{def}}{=} \alpha(a,0) + \alpha(a,1)$ and $\alpha_{*a} \stackrel{\text{def}}{=} \alpha(0,a) + \alpha(1,a)$.*
- *We define the* one-bit entropy of α *to be $\mathcal{H}^*(\alpha) \stackrel{\text{def}}{=} \max\{\mathcal{H}(\alpha_{0*}, \alpha_{1*}), \mathcal{H}(\alpha_{*0}, \alpha_{*1})\}$.*

Definition 8.10 (Good Predicate) *Let $E(d)$ (for **E**qual) be the set of $\beta \in \Delta^2(d)$ such that $\beta(0,1) = \beta(1,0) = 0$. Notice that if $(x,y) \in \mathbf{NA}^{-1}(\beta)$, then $x = y$ iff $\beta \in E(d)$.*

For any $\delta > 0$, we call P to be a δ-good predicate if for every $\beta \in \Delta^2(d) \setminus E(d)$, we have

$$\mathrm{PE}(P, \beta) \le \tfrac{1}{2} + 2^{-d(\mathcal{H}^*(\beta/d) - \delta)/2}.$$

Lemma 8.11 *Fix any $\delta > 0$ and $\beta \in \Delta^2(d) \setminus E(d)$. Choose $P : \{0,1\}^d \to \{0,1\}$ uniformly at random. Then*

$$\Pr[\mathrm{PE}(P, \beta) > \tfrac{1}{2} + 2^{-d(\mathcal{H}^*(\beta/d) - \delta)/2}] \le \exp(-\tfrac{1}{2} 2^{\delta d} / poly(d)).$$

Taking a union bound,

$$\Pr[P \textit{ is not a } \delta\textit{-good predicate}] \le \exp(-2^{\delta d - O(\log d)}).$$

Proof: Without loss of generality, assume $\mathcal{H}^*(\beta/d) = \mathcal{H}(\beta_{0*}/d, \beta_{1*}/d)$. Let $S \subseteq \{0,1\}^d$ be the support of the marginal distribution on x when $(x,y) \sim \mathbf{NA}^{-1}(\beta)$: that is, S is the set of strings with β_{0*} zeroes and β_{1*} ones.

Pick any $x \in S$. If $P, P' : \{0,1\}^d \to \{0,1\}$ are predicates which differ only at x, then $\mathrm{PE}(P', \beta) - \mathrm{PE}(P, \beta) \le c_x$, where

$$c_x = \Pr_{(x',y') \sim \mathbf{NA}^{-1}(\beta)}[x' = x \vee y' = x] \le \frac{2}{|S|} = \frac{2}{\binom{d}{\beta_{0*}}}.$$

Fix P arbitrarily on all $x \notin S$, but choose the value of P independently at random for all $x \in S$: then $\mathbf{E}[\mathrm{PE}(P, \beta)] = \frac{1}{2}$, since $\beta \notin E(d)$. By McDiarmid's inequality, for any ϵ,

$$\begin{aligned}
\Pr[\mathrm{PE}(P, \beta) > \tfrac{1}{2} + \epsilon] &\le \exp\left(-\frac{2\epsilon^2}{\sum_{x \in S} c_x^2}\right) \\
&\le \exp\left(-\tfrac{1}{2}\epsilon^2 \binom{d}{\beta_{0*}}\right) \\
&\le \exp\left(-\tfrac{1}{2}\epsilon^2 2^{d\mathcal{H}(\beta_{0*}/d, \beta_{1*}/d)} / \mathrm{poly}(d)\right).
\end{aligned}$$

To complete the proof, take $\epsilon = 2^{-d(\mathcal{H}(\beta_{0*}/d, \beta_{1*}/d) - \delta)/2}$. ■

Lemma 8.12 *For every $p_0 \in (0, \frac{1}{2}]$, there exist $c_2, \delta, \mu > 0$, such that $c_2 < \frac{\epsilon}{2}$ where ϵ is taken from Lemma 8.6 (with $\tau = 1$), and for all sufficiently large $d \in \mathbf{N}$,*

$$\forall p \in [p_0, 1-p_0], \Pr_{k\sim\mathrm{Binom}(p,d)}[\mathcal{H}(k/d) < c_2 + \delta + \mu] + 2^{-(c_2+\mu)d/2} \le \tfrac{1}{6}2^{-c_2 d}.$$

Proof: Let D_{KL} denote the Kullback-Leibler divergence

$$D_{KL}(q\|p) = q \lg \frac{q}{p} + (1-q)\lg\frac{1-q}{1-p}.$$

Fix $\lambda > 0$ to be small enough that $\lambda < 1/2$, $\mathcal{H}(\lambda) < 3\epsilon$, and for any $p \in [p_0, 1-p_0]$,

$$D_{KL}(\lambda\|p) > \mathcal{H}(\lambda).$$

Now, choose any $p \in [p_0, 1-p_0]$.

$$\begin{aligned}
\Pr_{k\sim\mathrm{Binom}(p,d)}[\mathcal{H}(k/d) < \mathcal{H}(\lambda)] &= \Pr[k/d < \lambda] + \Pr[k/d > 1-\lambda] \\
\textit{(Without loss of generality, assume } p \le \tfrac{1}{2}.\textit{)} \quad &\le 2\Pr[k/d < \lambda] \\
\textit{(Apply Chernoff's bound.)} \quad &\le 2\exp(-D_{KL}(\lambda\|p)d) \\
&< 2\exp(-\mathcal{H}(\lambda)d).
\end{aligned}$$

Complete the proof by taking $c_2 = \mathcal{H}(\lambda)/6 < \frac{\epsilon}{2}$, $\delta = \mathcal{H}(\lambda)/3$ and $\mu = \mathcal{H}(\lambda)/2$, and taking d to be sufficiently large that $2\exp(-\mathcal{H}(\lambda)d) < \frac{1}{6}2^{-c_2 d} - 2\cdot 2^{-(c_2+\mu)d/2}$. ■

Lemma 8.13 *Let $p_0 < \frac{1}{2}$ be the unique number satisfying $\mathcal{H}(p_0) = \frac{1}{2}$. Choose c_2, δ, and μ according to Lemma 8.12. Let P be any δ-good predicate. Then for every $\alpha \in \Delta^2$, we have $\mathcal{H}(\alpha) + \lg \mathrm{PE}(P,\alpha) \le 1 + 2^{-c_2 d}$.*

Proof: Let $\alpha \in \Delta^2$. Let $E(d)$ be defined as in Definition 8.10. Let $\gamma = \Pr_{\beta\sim\mathrm{Mult}(\alpha,d)}[\beta \in E(d)]$. Since P is a δ-good predicate, we have

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\mathrm{PE}(P,\beta)] \\
\le&(1-\gamma)\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\min\{1, \tfrac{1}{2} + 2^{-d(\mathcal{H}^*(\beta/d)-\delta)/2}\} | \beta \notin E(d)] + \gamma \\
=&(1-\gamma)\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\min\{\tfrac{1}{2}, 2^{-d(\mathcal{H}^*(\beta/d)-\delta)/2}\} | \beta \notin E(d)] + \tfrac{1+\gamma}{2}. \\
\le&\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\min\{\tfrac{1}{2}, 2^{-d(\mathcal{H}^*(\beta/d)-\delta)/2}\}] + \tfrac{1+\gamma}{2}.
\end{aligned}$$

Therefore, by the concavity of the logarithm function and taking a linear approximation at $\frac{1+\gamma}{2}$, we have

$$\begin{aligned}
\lg \mathrm{PE}(P,\alpha) &\le \lg\tfrac{1+\gamma}{2} + \tfrac{2}{(1+\gamma)\ln 2}(PE(P,\alpha) - \tfrac{1+\gamma}{2}) \\
(\textit{Apply Lemma } 8.8.) \quad &= \lg\tfrac{1+\gamma}{2} + \tfrac{2}{(1+\gamma)\ln 2}(\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)} PE(P,\beta) - \tfrac{1+\gamma}{2}) \\
&\le \lg\tfrac{1+\gamma}{2} + 3\mathop{\mathbf{E}}_{\beta\sim\mathrm{Mult}(\alpha,d)}[\min\{\tfrac{1}{2}, 2^{-d(\mathcal{H}^*(\beta/d)-\delta)/2}\}].
\end{aligned}$$

Since $\lg \mathrm{PE}(P,\alpha) \leq 0$, the claim of the lemma follows when $\mathcal{H}(\alpha) \leq 1$; so henceforth we assume $\mathcal{H}(\alpha) \geq 1$, and hence $\mathcal{H}^*(\alpha) \geq 1/2$. We may further assume without loss of generality that $\mathcal{H}(\alpha_{0*}, \alpha_{1*}) \geq 1/2$, or in other words, $\alpha_{0*} \in [p_0, 1-p_0]$. Applying Lemma 8.12 (recall that c_2, δ and μ were chosen according to that lemma), we have

$$\Pr_{\beta \sim \mathrm{Mult}(\alpha,d)}[\mathcal{H}(\beta_{0*}/d, \beta_{1*}/d) < c_2 + \delta + \mu] + 2^{-(c_2+\mu)d/2} \leq \tfrac{1}{6}2^{-c_2 d}.$$

Thus, by Markov's inequality and since $\mathcal{H}^*(\beta/d) \geq \mathcal{H}(\beta_{0*}/d, \beta_{1*}/d)$, we have

$$\mathop{\mathbf{E}}_{\beta \sim \mathrm{Mult}(\alpha,d)}[\min\{\tfrac{1}{2}, 2^{-d(\mathcal{H}^*(\beta/d)-\delta)/2}\}] \leq \tfrac{1}{6}2^{-c_2 d}.$$

Therefore, we have

$$\mathcal{H}(\alpha) + \lg \mathrm{PE}(P,\alpha) \leq \mathcal{H}(\alpha) + (\lg \tfrac{1+\gamma}{2} + \tfrac{1}{2}2^{-c_2 d}),$$

and our goal reduces to showing $\mathcal{H}(\alpha) + \lg(1+\gamma) \leq 2 + \frac{1}{2}2^{-c_2 d}$. Let ϵ be chosen to satisfy Lemma 8.6 with $\tau = 1$. Note that c_2 in Lemma 8.12 is $< \epsilon/2$, therefore we can make sure d is large enough that $\frac{1}{2}2^{-c_2 d} \geq 2^{-\epsilon d}$. We have $\gamma = (\alpha_{00} + \alpha_{11})^d$, $\mathcal{H}(\alpha) \leq 1 + \mathcal{H}(\alpha_{00} + \alpha_{11})$, so Lemma 8.6 completes our proof with $p = \alpha_{00} + \alpha_{11}$ and $\tau = 1$. ■

Now we are ready to complete the proof of Lemma 8.5

Proof: [of Lemma 8.5 for Random Predicates] Choose δ and c_2 according to Lemma 8.13. By Lemma 8.11, a predicate P chosen uniformly at random is δ-good with probability $\geq 1 - \exp(-2^{\delta d - O(\log d)}) \geq 1 - 2^{-2^{c_1 d}}$ for $c_1 < \delta$ and large enough d. By Lemma 8.13, for every $\alpha \in \Delta^2$ and such a predicate P, we have $\mathcal{H}(\alpha) + \lg \mathrm{PE}(P,\alpha) \leq 1 + 2^{-c_2 d}$. ■

Chapter 9

Related and Possible Future Work

Cryan and Miltersen [17] first raised the question of whether cryptographic primitives (their work focused on pseudorandom generators) can be computed in NC_0, where every output bit depends on a constant number of input bits. Mossel, Shpilka and Trevisan [42] 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. Applebaum, Ishai and Kushilevtiz [9, 10] show that, under standard assumptions, one can construct one-way functions and pseudorandom generators that can be computed in NC_0. Goldreich's function is in NC_0 when $d = O(1)$.

There has been a number of other works about the security of Goldreich's candidate one-way function:

- Bogdanov and Qiao [14] show that when the number of output bits of the function is a large enough constant factor larger than the number of input bits, if the used predicate has correlation with either one or two of its variables, then Goldreich's function can be inverted. (Note that this does not break Goldreich's original proposal where the number of input and output bits are equal.)

- As was mentioned, Miller [41] and Itsykson [30] show lower bounds on the running time of drunken backtracking algorithms for inverting Goldreich's function.

- Itsykson and Sokolov [31] gave a lower bound on the running time of myopic backtracking algorithms which can read more than a constant number of output nodes at each round. Their lower bound is for an explicit function, as opposed to Goldreich's function for random graphs.

Applebaum, Barak, and Wigderson [8] use an assumption about Goldreich's function together with other assumptions to create public key cryptography. The assumption is that a variant of Goldreich's function with more output bits than input bits is a pseudorandom generator. The assumption is stronger than Goldreich's conjecture that the function is one-way.

The main limitation of the present work are the somewhat artificial setups of both myopic algorithms and drunk algorithms. It would also be interesting to show that no "variation of

Gaussian elimination" can invert Goldreich's function when non-linear predicates are used. The first step is to formalize such a statement.

Bibliography

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

[2] Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constand-depth circuits. *Advances in Computing Research - Randomness and Computation*, 5:199–223, 1989. Preliminary version in *Proc. of FOCS'85*.

[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] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *J. Autom. Reasoning*, 35:51–72, 2005.

[5] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. In *FOCS*, pages 190–199, 2001.

[6] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[7] Noga Alon, Oded Goldreich, and Yishay Mansour. Almost k-wise independence versus k-wise independence. *Information Processing Letters*, 88(3):107–110, 2003.

[8] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Symposium on Theory of Computing*, pages 171–180, 2010.

[9] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC0. *SIAM J. on Computing*, 36(4):845–888, 2006.

[10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in NC0. In *APPROX-RANDOM*, pages 260–271, 2006.

[11] Louay Bazzi. *Minimum Distance of Error Correcting Codes versus Encoding Complexity, Symmetry, and Pseudorandomness.* PhD thesis, MIT, 2003.

[12] Louay M. J. Bazzi. Polylogarithmic independence can fool dnf formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009.

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

[14] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. In *APPROX-RANDOM*, pages 392–405, 2009.

[15] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *Journal of ACM*, 57(5), 2010.

[16] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC, San Francisco, CA, USA. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 521–538. Springer, 2009.

[17] Mary Cryan and Peter B. Miltersen. On pseudorandom generators in NC0. In *Proceedings of MFCS'01*, 2001.

[18] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[19] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. Bounded independence fools halfspaces. In *Foundations of Computer Science*, pages 171–180, 2009.

[20] Ilias Diakonikolas, Daniel Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *focs*, 2010.

[21] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT, St. Andrews, UK, Proceedings*, pages 61–75, 2005.

[22] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT. Santa Margherita Ligure, Italy, Selected Revised Papers*, pages 502–518, 2003.

[23] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Approximations of general independent distributions. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 10–16, 1992.

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

[25] Oded Goldreich. *Foundations of Cryptography - Basic Applications.* Cambridge University Press, 2004.

[26] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[27] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[28] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[29] Russell Impagliazzo and Avi Wigderson. = *BPP* if requires exponential circuits: Derandomizing the xor lemma. In *STOC*, pages 220–229, 1997.

[30] Dmitry Itsykson. Lower bound on average-case complexity of inversion of goldreich's function by drunken backtracking algorithms. In *Computer Science - Theory and Applications, 5th International Computer Science Symposium in Russia, CSR*, pages 204–215, 2010.

[31] Dmitry Itsykson and Dmitry Sokolov. The complexity of inversion of explicit goldreich's function by dpll algorithms. *Preprint*, 2010.

[32] Kazuo Iwama and Shuichi Miyazaki. Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle. In *Proceedings of ISAAC*, volume 1741 of *Lecture Notes in Computer Science*, pages 133–142, 1999.

[33] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC*, pages 355–364, 2003.

[34] Adam Klivans, Homin Lee, and Andrew Wan. Mansour's conjecture is true for random dnf formulas. In *Proceedings of the 23rd Conference on Learning Theory (COLT)*, 2010.

[35] Leslie Lamport. Constructing digital signatures from a one-way function. *Technical Report SRI-CSL-98, SRI International Computer Science Laboratory*, 1979.

[36] Leonid Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003.

[37] Michael Luby and Boban Velickovic. On deterministic approximation of DNF. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 430–438, 1991.

[38] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Israel Symposium on Theory of Computing and Systems*, pages 18–24, 1993.

[39] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[40] Yishay Mansour. An $o(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50(3):543–550, 1995.

[41] Rachel Miller. Goldreich's one-way function candidate and drunken backtracking algorithms. Master's thesis, University of Virginia, 2009. Distinguished Majors Thesis.

[42] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ϵ-biased generators in NC^0. *Random Structures and Algorithms*, 29(1):56–81, 2006.

[43] Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[44] Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[45] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput*, 22:838–856, 1993.

[46] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[47] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[48] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[49] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 12(4):63–70, 1991.

[50] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[51] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[52] Alexander Razborov. A Simple Proof of Bazzi's Theorem. *ACM Trans. Comput. Theory*, 1(1):1–5, 2009.

[53] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[54] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(1):137–168, 2008.