CS 880: Advanced Complexity Theory
Final Project: Hardness Amplification within NP

Priyananda Shenoy
May 2008

Abstract

We survey the current status of the problem of *Hardness amplification within NP*, reviewing known results and evaluating various approaches for improving on them. We also prove that under some stronger assumptions, we can amplify the hardness to $(\frac{1}{2} - 1/2^{\Omega(n)})$.

1 Introduction

Average-case complexity has been studied extensively in recent times, due to it's practical applications. One of the core problems in this field is, given a function which is "mildly hard" in the average-case, can we construct a "very hard" function? This is the problem of *Hardness Amplification*. This paper looks at a restricted version of the problem, where the resultant hard function needs to be in NP. Intuitively, we can "amplify" the hardness of a function by invoking multiple copies of the function and combining them using another function.

In this paper, we deal with functions which are $1/poly(n)$-hard (precise definition of *hardness* will be given in section 1.3). We want the resultant hardness to be as close to $\frac{1}{2}$ as possible. We will look at the results of O'Donnell [ROD02], Healy et. al. [HVV04] and Lu et. al. [LTW07]. The main focus of this paper will be on the Lu-Tsai-Wu result, and how we can improve upon that result.

1.1 Organization of the paper

Section 1.2 introduces the basic terminology and definitions used in the rest of the paper. Section 2 goes into detail about previous results, specifically the XOR Lemma and the result due to O'Donnell. Section 3 details the result due to Healy-Vadhan-Viola. Section 4 deals with the Lu-Tsai-Wu result. Section 5 describes how a stronger assumption about the "mildly hard" function can be used to improve upon the result. Section 6 contains an appendix for some background information about Combinatorial Rectangles.

1.2 Preliminaries

We will begin by defining precisely the concept of *hardness* of functions:

Definition 1. *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be δ-hard for circuits of size s, if no circuit of size s can correctly compute f on a δ fraction of the inputs $\{0,1\}^n$.*

The goal of *Hardness Amplification within NP* is to answer this question:

Question 1. *If there exists an $f \in NP$ which is δ-hard against circuits of size s, then is there a function $f' \in NP$ which is δ'-hard against circuits of size s', where δ' is close to $\frac{1}{2}$ and $s' = poly(s)$?*

Since the NP vs. P/poly question is unresolved, we don't even know if there is a function in NP which is $1/2^n$-hard. But under the reasonable assumption that NP is slightly hard on average, we can obtain a function in NP which is very hard on average.

Definition 2. *A function $f \in NP$ is said to be* mildly hard on average *if it is $1/poly(n)$-hard against circuits of polynomial size.*

The general strategy for hardness amplification is the so called *direct product construction*, where multiple independent copies of the function are aggregated using a combining function, the result of which is significantly harder than the original function. The question of hardness amplification within NP can be rephrased as:

Question 2. *If $f : \{0,1\}^n \to \{0,1\} \in NP$ is δ-hard for circuits of size s, is there a function $g : (\{0,1\}^n)^k \to \{0,1\}$ such that $g(f, \cdots, f) \in NP$ is δ'-hard against circuits of size s', where δ' is close to $\frac{1}{2}$ and $s' = poly(s)$?*

We will define few more terms which will be useful later on.

Definition 3. *The* Bias *of a function $f : \{0,1\}^n \to \{0,1\}$ is defined as*

$$bias(f) = |\Pr_x[f(x) = 0] - \Pr_x[f(x) = 1]|.$$

Definition 4. *The* Expected Bias *of a function $f : \{0,1\}^n \to \{0,1\}$ at ρ is*

$$ExpBias_\rho(f) = E_{R_\rho}[bias(f|_R)],$$

where $f|_R$ denotes f under a random restriction R, and the expectation is over all random restrictions with parameter ρ.

Definition 5. *The* Expected Collision Probability *of a probabilistic function $f : \{0,1\}^n \to \{0,1\}$ is defined as*

$$ExpCP(f) = E_x[2 \Pr_{y,y'}[h_y(x) = h_{y'}(x)] - 1].$$

Definition 6. *The* Noise Stability *of f at δ is*

$$NoiseStab_\delta(f) = 2\Pr[f(x) = f(N_\delta(x))] - 1,$$

where $N_\delta(x)$ is the string obtained by flipping each bit of x with probability δ.

Definition 7. *A generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ is said to be* explicitly computable *if given seed $\sigma \in \{0,1\}^l$, we can compute $X_i \in \{0,1\}^n$ for all $i \leq k$ in time $poly(l, \log k)$, where $G(\sigma) = X_1 X_2 \cdots X_k$.*

Definition 8. *A probabilistic, read-once, oblivious* branching program *of size s and block-size n, is a finite state machine with s states, over the alphabet $\{0,1\}^n$, with a fixed start state and an arbitrary number of final states, Each edge is labeled with a symbol from the alphabet. For every state a and symbol α, the edges leaving a and labeled α are assigned a probability distribution. Then computation proceeds as follows. The input is read sequentially, one block of n-bits at a time. If the machine is in state a and it reads symbol α, it chooses an edge leaving a with symbol α according to its probability. The width of a branching program is the maximum, over i, of the number of states that are reachable after reading i symbols. Let BP(s,n) denote the class of functions computed by such programs.*

Definition 9. *A generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ is said to be ε-pseudo random against branching programs of size s and block size n if for every branching program B,*

$$|\Pr[B(G(U_l)) = 1] - \Pr[B(G(U_{nk}) = 1]| \leq \varepsilon.$$

2 Prior work

2.1 XOR Lemma

One simple candidate for the combining function is the XOR function. We can define the combining function g as

$$g(f, \cdots, f) \overset{def}{=} f(x_1) \oplus \cdots \oplus f(x_k),$$

where $k = poly(n)$. The XOR Lemma states that the hardness of this function approaches $\frac{1}{2}$ exponentially with k.

Lemma 1 (Yao's XOR Lemma)**.** *If f is δ-hard for size s, then g is $\left(\frac{1}{2} - 1/2^{\Omega(\delta k)} - 1/s'\right)$ for size $s' = poly(s)$.*

Unfortunately it's unknown if NP is closed under XOR, which means that we cannot guarantee that the resultant function is in NP.

2.2 O'Donnel's Result

Since XOR is unsuitable for your purposes, we need to look for other combining functions. O'Donnell [ROD02] characterized the properties that the combining function needs to have as follows:

- To ensure that the resultant function remains in NP, the combining function needs to be *monotone*.
- It must be (nearly) balanced.
- The *expected bias* of the combining function must be much smaller than δ.

O'Donnell related the hardness achieved by the combining function to a combinatorial property of the function known as it's *expected bias* . Using the functions "Recursive Majority of 3" and "Tribes", he was able to get the following result [ROD02]:

Theorem 1 (O'Donnell)**.** *If there is a balanced function in NP which is $1/poly(n)$-hard for circuits of polynomial size, then there is a function in NP which is $(\frac{1}{2} - 1/n^{1/2-\alpha})$-hard, where α is arbitrarily small.*

The amplification obtained through this result is not as good as the XOR Lemma, since this result is independent of the circuit size against which the original function was hard. O'Donnell proved this technique cannot be used to amplify more than $(\frac{1}{2} - 1/n)$.

2.2.1 Details of the O'Donnell construction

O'Donnell in [ROD02] related the hardness of the resultant function to a property of the combining function, called it's *Expected Bias.* Specifically,

Theorem 2 (O'Donnell)**.** *For every $r > 0$, the function $g(f, \cdots, f)$ is $(1 - ExpBias_{(2-r)\delta}(g) - \varepsilon)$ -hard for circuits of size $\Omega(\frac{\varepsilon^2}{k \cdot \log(1/\delta)} \cdot s)$.*

Expected Bias in turn could be related to the *Noise stability* (or noise sensitivity) of the combining function as

$$NoiseStab_\delta(h) \leq ExpBias_{2\delta}(h) \leq \sqrt{NoiseStab_\delta(h)}.$$

The idea therefore is to use a function with low noise stability, i.e. high sensitivity to noise as a combining function. It turns out that the Recursive Majority function has this desired property. Using Recursive Majority as the combining function, we can amplify from $1/poly(n)$-hardness to $(\frac{1}{2} - 1/n^\alpha)$-hardness, where $\alpha \approx 0.13$. To further amplify hardness from some suitably large value, the Tribes function can be used, which also has low noise stability. Using this function as the combining function, we can amplify the hardness up to $\frac{1}{2} - 1/n^{\frac{1}{2}-\beta}$, where $\beta > 0$ is some small constant. Taking these two functions together, we can amplify a mildly hard function to get a function which is quite close to $\frac{1}{2} - 1/\sqrt{n}$-hard.

The bottleneck for this approach comes from the fact that for monotone Boolean functions, the Expected Bias is bounded on the lower side. In this sense, the Tribes function is close to optimal. O'Donnell showed that using just these techniques, the best amplification one can hope to achieve is $(\frac{1}{2} - 1/n)$.

3 The Healy-Vadhan-Viola Result

To amplify beyond the $(\frac{1}{2} - 1/n)$ barrier, Healy et. al. [HVV04] used *derandomization* and *nondeterminism.* They noted that the actual barrier was $(\frac{1}{2} - 1/n')$ where $n' = n \cdot k$ is the input length of the new function f'. This meant that taking k to be super-polynomial would break the barrier faced by the O'Donnell construction. Simply taking k to be super-polynomial would not help, because the resultant function needs to be in NP. We can get over these problems using the following techniques:

- *Derandomization*: The first idea is to break the linear dependence of n' on k. That way k can grow rapidly without the input size to the new function f' blowing up. Instead of obtaining the inputs independently, we can use a pseudo random generator to generate the inputs for each of the k copies of f. The seed length of the PRG determines the input length of the function. Healy et. al. use Nisan's PRG for space bounded computation, which has a seed length of $O(n^2 + \log^2 k)$.
- *Nondeterminism*: To keep the resultant function in NP even when k is super polynomial, we need to use nondeterminism. Instead of evaluating all k copies, we will nondeterministically choose and evaluate a small number of copies (say $O(\log k)$).

Combining these techniques, they obtain the following result:

Theorem 3 (Healy, Vadhan, Viola)**.** *If there exists a balanced $f \in NP$ which is $1/poly(n)$-hard for circuits of size $s = s(n)$, then there exists $f' \in NP$ which is $(\frac{1}{2} - 1/s')$-hard for circuits of size $s' = s'(n) = s(\sqrt{n})^{\Omega(1)}$. Specifically,*

1. *If $s(n) = n^{\omega(1)}$, then we can achieve hardness $(\frac{1}{2} - 1/n^{\omega(1)})$.*
2. *If $s(n) = 2^{n^{\Omega(1)}}$, then we can achieve hardness $(\frac{1}{2} - 1/2^{n^{\Omega(1)}})$.*
3. *If $s(n) = 2^{\Omega(n)}$, then we can achieve hardness $(\frac{1}{2} - 1/2^{\Omega(\sqrt{n})})$.*

3.1 Details of the Healy-Vadhan-Viola construction

The basis of the Healy-Vadhan-Viola construction is the same as O'Donnell's: The combining function C is a combination of Recursive Majority function and the Tribes function. The first step is to break the linear dependence of the input size of f' to k, we is done by rerandomization. Instead of generating $n \cdot k$ bits randomly for k copies of f, we use a much smaller seed and a suitable pseudo random generator to "fool" our combining function. If $C : \{0,1\}^k \to 0, 1$ is our combining function, then the resultant function we get is $f' : (\{0,1\}^n)^k \to 0, 1$ defined as $f' = C \circ f^{\otimes k}$. We need a generator $\Gamma : \{0,1\}^l \to (\{0,1\}^n)^k$ with the following properties:

- **Γ is indistinguishability-preserving**: We need the computational hardness of $(C \circ f^{\otimes k}) \circ \Gamma$ to be at least $\frac{1}{2} - ExpBias[(C \circ g^{\otimes k}) \circ \Gamma]$, where g is some δ-random function. This implies that for any random seed $\sigma \in \{0,1\}^l$, and for any δ-random g,

$$\sigma \cdot f(\Gamma(\sigma)|_1) \cdots f(\Gamma(\sigma)|_k) \text{ and } \sigma \cdot g(\Gamma(\sigma)|_1) \cdots g(\Gamma(\sigma)|_k)$$

  are indistinguishable.

- **Γ fools the bias of C**: For any δ-random function g, $ExpBias[(C \circ g^{\otimes k}) \circ \Gamma]$ should be close to $ExpBias[C \circ g^{\otimes k}]$, which gives us

$$ExpBias[(C \circ g^{\otimes k}) \circ \Gamma] \leq \sqrt{NoiseStab(C) + \varepsilon},$$

  where ε is some small constant.

Healy et. al. use two separate generators to fulfill the two properties, and obtain the final generator by taking the XOR of the two generators. For preserving indistinguishability, they use Nisan and Widgerson's combinatorial generator, which has the following property:

Lemma 2. *For every $n > 1$ and every k, there is an explicitly constructable generator $IP_k : \{0,1\}^l \to (\{0,1\}^n)^k$ with seed length $l = O(n^2)$ that is indistinguishability preserving for size k^2.*

To obtain a PRG to fool the bias of C, the key observation is that C can be modeled as a branching program with block-size n and size s. Since branching programs correspond to space bounded computation, Nisan's pseudo random generators against space bounded computations can be used, which has the following property:

Lemma 3. *For every $n > 1$ and every $k \leq 2^n$, there is an explicitly constructable generator $N_k : \{0,1\}^l \rightarrow (\{0,1\}^n)^k$ which is $(1/2^n)$-pseudo random against branching programs of size 2^n, block-size n with seed length $l = O(n \log k)$ which fools the bias of C.*

The combined function $\Gamma_k(x, y) = IP_k(x) \oplus N_k(y)$ inherits the required properties from both the generators. The combined generator has an input length $O(n^2)$. Thus derandomization allows us to decouple the input length of the new function from k.

To achieve hardness exponentially close to $\frac{1}{2}$, k needs to be super polynomial. So we cannot possibly evaluate all copies of f in polynomial time, but since we have nondeterminism at our disposal, we can guess a suitably smaller number of functions to evaluate. As the Tribes function is a DNF with clause size $O(logk)$, we can nondeterministically choose one of the clauses and evaluate only that clause. This can be done in polynomial time as long as $k \leq 2^{poly(n)}$.

3.2 Cause of the gap

(This section is based mostly on Section 2.6 of [VIO06]).

Even though this construction achieves hardness exponentially close to $\frac{1}{2}$, there is still a gap between this and the result obtained by the XOR Lemma. The source of this gap turns out to be the quadratic seed length for the generator used in section 3.1. Recall that the generator was a XOR of an indistinguishability-preserving generator and Nisan's pseudo-random generator for branching programs. Impagliazzo and Widgerson showed than an indistinguishability-preserving generator with linear seed length exists. On the other hand, it is still open whether a linear length PRG exists for branching programs.

One thing to note is that while having a linear length PRG for branching programs is sufficient for getting the required result, it is not necessary. The BP corresponding to the combining function has certain characteristics: it's width is much smaller corresponding to their size. We only require a PRG to fool such restricted BPs.

4 The Lu-Tsai-Wu Result

Lu et.al [LTW07] take a different approach to improve upon the Healy-Vadhan-Viola result. They use the same construction as the previous one, but treat the combining function not as a branching program, but as an entity known as as *Combinatorial Rectangle*. The improvement comes from the fact that for these entities, a pseudo random generator exists with seed length $O(n^{3/2})$. The result they obtain is:

Theorem 4. *If there is a balanced function $f \in NP$ which is δ-hard for circuits of size $s(n) = 2^{\Omega(n)}$, then there exists a function $f' \in NP$ which is $(\frac{1}{2} - 1/s'(n))$-hard for circuits of size $s'(n) = 2^{\Omega(n^{2/3})}$.*

4.1 Details of the Lu-Tsai-Wu construction

Lu et. al. use the same construction developed by Healy et al. They expand the combining function as

$$A = C \circ f^{\otimes k} = OR_d \circ (AND_b \circ RecMaj_r^{\otimes b} \circ f^{\otimes b3^r})^{\otimes d},$$

where $k = db3^r = 2^{O(n)}, d = 2^{O(n)}, b = poly(n), r = O(\log(1/\delta)) = O(\log n)$. A is a probabilistic function, and let A_y indicate the invocation of A with some random string y. The key observation to make is that when the function outputs 0 i.e. for any $A_y^{-1}(0)$, we can view it as a rectangle $R \in \mathcal{R}(2^{b3^r n}, d)$. Instead of working with the expected bias of the combining function, they proved that the hardness obtained to be related to another property of the combining function, it's expected collision probability. What Lu et. al. showed was that to fool the expected collision probability of C, it is enough to have a good PRG for rectangles in $\mathcal{R}(2^{b3^r n}, d)$. Lu presents the construction of a suitable PRG in [CLU98], but this cannot be used directly since each dimension is very large. To resolve this, they use d copies of Nisan's PRG against BPs to fool d functions in each dimension. The seeds for these PRGs come from using Lu's PRG for rectangles.

Let $\varepsilon = 1/2^{\Omega(n)}$, then the generators used by this construction are

- The generator $G_N : \{0,1\}^q \to \{0,1\}^{b3^r n}$ is Nisan's (ε/d)-generator for $BP(n^c, n)$ for some constant c, where $q = O(n \log n)$.
- The generator $G_L : \{0,1\}^l \to \{0,1\}^{dq}$ if Lu's ε-PRG for $\mathcal{R}(2^q, d)$, with seed length $l = O(n^{3/2}$.

The combined generator they use is

$$G(u) = (G_N^{\otimes d} \circ G_L)(u) = G_N^{\otimes d}(G_L(u)).$$

The total seed length to this combined generator is $O(n^{3/2}) + O(n \log n) = O(n^{3/2})$.

4.2 Cause of the gap

As with the Healy-Vadhan-Viola result, the amplification obtained through this technique falls short of the one obtained by the XOR Lemma. Again, the bottleneck is the seed length of Lu's generator for rectangles. It is not enough however to improve just the generator for rectangles, because the other generator used, Nisan's PRG for branching programs, has a seed length of $O(n \log n)$. With just a $O(n)$ generator for rectangles, the best we could achieve would be $(\frac{1}{2} - 1/2^{\Omega(n/\log n)})$ hardness.

5 Amplification under a stronger assumption

Until now we have not considered anything about the function f, other than it was balanced and mildly hard on average. Our bottleneck in both the Healy-Vadhan-Viola construction and the Lu-Tsai-Wu construction was in the seed length of the psuedo random generator used to supply the inputs to the multiple invocations of f. One way to get around this problem is to approximate f by another function which takes less bits, but retains most of it's hardness. We introduce the following notation to help make this nothing precise.

Definition 10. *Let $f : \{0,1\}^n \to \{0,1\}$ be δ-hard for circuits of size $s = s(n)$. Then a function $g : \{0,1\}^m \to \{0,1\}$ which is δ'-hard for sets of size s is termed as a (m, δ')-* hardness preserving approximation *of f if g is ε close to f where $\varepsilon > 0$ is some constant.*

The idea is, instead of evaluating f by generating n random bits, we evaluate an approximation of f which takes fewer input bits. If the hardness of this approximate function is not too different from the original function, then we can reduce the seed length of our generator appropriately, and leave the rest of the construction the same. Plugging in the values from Lu-Tsai-Wu we get:

Theorem 5. *Let $f : \{0,1\}^n \to \{0,1\} \in NP$ be $1/poly(n)$-hard for circuits of size $s = s(n)$. If there exists a $(n^{2/3}, 1/poly(n))$-hardness preserving approximation of f in NP, then there exists an f' which is $(\frac{1}{2} - 1/2^{\Omega(n)})$-hard for circuits of size $2^{\Omega(n)}$.*

Note that this in itself is not very interesting, since what we're just stating is that if there exists a function g on $m = n^{2/3}$ variables which is hard for circuits of size $2^{\Omega(n)} = 2^{\Omega(m^{3/2})}$, then there exists an f' which is $(\frac{1}{2} - 1/2^{\Omega(m^{3/2})})$-hard, which is directly obtainable from the Healy-Vadhan-Viola result. The real utility in this approach is that we can get a sense of how close to n m needs to get in order to become mildly hard on average. Intuitively as m gets closer to n, g becomes more accurate in approximating f and it's hardness increases. Since f is a balanced Boolean function, there is some m strictly smaller than n, for which g is $1/poly(n)$-hard. Turning this around, we can now obtain bounds on seed length of the PRGs used in the construction.

In the following section, we will try to develop the following: We will try to establish a relationship between the parameters ε (the closeness of g to f), and δ' (the hardness of g). Harmonic analysis then allows us to relate ε with the Influence of f. Since we know that f is a balanced Boolean function, with a distinct bound on Influence, we can show that for m sufficiently close to n, $\delta' = 1/poly(n)$ i.e. there always exists a $(m, 1/poly(n))$- hardness preserving approximation of f.

In the following discussion, it is helpful to think of g as a function on n variables where only m out of the n bits are used. Let g be an ε-approximation of f, i.e. g matches with f on only $(1-\varepsilon)$ fraction of the input.

Theorem 6. *If f is δ-hard for circuits of size s, and g is an ε-approximation of f with $\varepsilon < \delta$, then g is δ'-hard for circuits of size s where $\delta' > 0$.*

Proof. We will prove this by contradiction. Assume g is easy for circuits of size s, so there is some circuit C which computes g correctly on all inputs. Since g differs from f in only ε fraction of the input, we can use the same circuit to compute f correctly on $(1-\varepsilon)$ fraction of the input, which is a contradiction since no circuit can compute greater than $(1-\delta)$ fraction. This implies that g is not easy for circuits of size s. $\square$

We will try to relate the hardness of f with the hardness of g. Let f and g be δ_f and δ_g hard respectively for circuits of size s. Let C_g be the circuit which computes g with the least error. Suppose we try to use the same circuit to output the value of f. Then the probability that this circuit outputting the correct value is

$$\begin{aligned}\Pr_x[C_g(x)=f(x)] &= \Pr_x[C_g(x)=g(x) \text{ and } g(x)=f(x)] + \Pr_x[C_g(x)\neq g(x) \text{ and } g(x)\neq f(x)]\\ &= (1-\delta_g)(1-\varepsilon)+\delta_g\varepsilon\\ &= 1-\delta_g-\varepsilon+2\delta_g\varepsilon.\end{aligned}$$

Since f is δ_f-hard, the success probability cannot be more than $(1-\delta)$, so we get

$$\begin{aligned}1-\delta_g-\varepsilon+2\delta_g\varepsilon &\leq 1-\delta_f\\ \delta_g+\varepsilon-2\delta_g\varepsilon &\geq \delta_f\\ \delta_g(1-2\varepsilon) &\geq \delta_f-\varepsilon\\ \delta_g &\geq \frac{\delta_f-\varepsilon}{1-2\varepsilon}.\end{aligned}$$

Friedgut's theorem states that we can ε-approximate f by a m junta where $m = 2^{O(\frac{I(f)}{\varepsilon})}$. We get

$$\begin{aligned}m &\leq 2^{c\frac{I(f)}{\varepsilon}}\\ \log m &\leq c'\frac{I(f)}{\varepsilon}\\ \varepsilon &\leq c'\frac{I(f)}{\log m}\end{aligned}$$

Since f is a balanced Boolean function, it's influence is at least $O(\log n)$ by the Kahn-Kalai-Linial theorem. So we get

$$\varepsilon \leq c''\frac{\log n}{\log m}.$$

If we take $m = n^\alpha$ where $\alpha \leq 1$ is some constant, then we get $\log m = \alpha \log n$, giving us a constant upper bound on the error rate.

So for some value of m, the approximation error we get is bounded by a constant, which implies from the above expression for δ_g that $\delta_g = 1/poly(n)$ if $d_f = 1/poly(n)$. So we get g to be a $(m, 1/poly(n))$-hardness preseverving approximation of f.

Note to Professor van Melkebeek

I'm not sure how to procede because of these constants. What I want to show is that there is some $\alpha < 1$ such that plugging $m = n^\alpha$ gives us an constant error rate which is less than δ, by which we can unconditionally say that g is $1/poly(n)$ hard.

6 Appendices

6.1 Combinatorial Rectangles

(This section is mainly based on [CLU98])
The Lu-Tsai-Wu result result relies on a tool called *Combinatorial Rectangle*.

Definition 11. *For positive integers m and d, a* combinatorial rectangle of type (m,d) *is a subset of $[m]^d$ of the form $R_1 \times R_2 \times \cdots \times R_d$ where $R_i \subseteq [m]$ for all $i \in [d]$. The family of such rectangles is denoted as $\mathcal{R}(m, d)$.*

We are interested in a measure of a rectangle, called the *volume*. The general definition for volume is:

Definition 12. *Let V be a finite set with uniform distribution. Then the volume of a set $A \subseteq V$ is defined as*

$$vol(A) = \Pr_{x \in V}[x \in A] = \frac{|A|}{|V|}.$$

Specifically for combinatorial rectangles, the volume is defined as follows:

Definition 13. *The* Volume *of a rectangle $R \in \mathcal{R}(m, d)$ is given by*

$$\prod_{i \in [d]} \frac{|R_i|}{m}.$$

Estimating the volume of a rectangle plays an important role in certain theoretical results, e.g. in derandomizing RL. Often, we don't want to sample the entire space V (which is of size m^d in the case of combinatorial rectangles), but a smaller subspace of it, while obtaining a reasonable approximation for the volume. For this, we can use a pseudo random generator.

Definition 14. *Let $\mathcal{A}$ be a family of subsets of V. Then a deterministic function $g : \{0, 1\}^l \to V$ is an* ε-generator using l bits *for $\mathcal{A}$, if for all $A \in \mathcal{A}$,*

$$|\Pr_{y \in \{0,1\}^l}[g(y) \in A] - vol(A)| \leq \varepsilon.$$

The goal is to find an explicitly constructable generator with small seed length. Lu [CLU98] obtained the following result, which is the best known one:

Theorem 7 (Lu). *For any $m, d \in N$ and any $\varepsilon \in (0, 1)$, there exists an explicitly constructable ε-generator for $\mathcal{R}(m, d)$ with $l = O(\log m + \log d + \log^{3/2}(\frac{1}{\varepsilon}))$.*

When m, d and $\frac{1}{\varepsilon}$ are all $n^{\Theta(1)}$, the seed length becomes $O(\log^{3/2} n)$.

References

[ROD02] R. O'Donnell. Hardness Amplification within NP. In *Proceedings of* 34th *STOC*, May 2002.

[HVV04] A. Healy, S. Vadhan, E. Viola. Using Nondeterminism to Amplify Hardness. In *Proceedings of* 36th *STOC*, June 2004.

[LTW07] C. Lu, S. Tsai, H. Wu. Improved hardness amplification in NP. In *Theoretical Computer Science*, Feb 2007.

[BTR08] A. Bogdanov, L. Trevisan. Average-Case Complexity. Manuscript.
http://arxiv.org/abs/cs.CC/0606037

[VIO06] E. Viola. The Complexity of Hardness Amplification and Derandomization. Ph.D Thesis, The Division of Engineering and Applied Sciences, Harvard University.

[SVI08] R. Shaltiel, E. Viola. Hardness Amplification Proofs Require Majority. To appear in *Proceedings of* 40th *STOC*, 2008.

[CLU98] C. Lu. Improved Pseudorandom Generators for combinatorial rectangles. In *Combinatorica* 22(3) 2002.

[FRI98] E. Friedgut. Boolean Functions with Low Average Sensitivity Depend on Few coordinates. In *Combinatorica* 18(1) 1998.