An Introduction to Probabilistically Checkable Proofs and the PCP Theorem

Jose Falcon, Mitesh Jain

April 25, 2011

1 Introduction

Define the languages of NP as the languages with efficient proof systems.

Definition 1. *A language L is in* NP *if there is a polynomial-time deterministic Turing machine $\mathcal{V}$ that, given an input x, verifies proofs, denoted π, of $x \in L$. The following properties hold:*

$$x \in L \implies \exists \pi : \mathcal{V}^{\pi}(x) = 1 \tag{1}$$

$$x \notin L \implies \forall \pi : \mathcal{V}^{\pi}(x) = 0. \tag{2}$$

$\mathcal{V}^{\pi}(x)$ has access to an input string x and a certificate string π. Note that we use "proof" and "certificate" interchangeably.

It is instructive to pose NP-problems as a game between a solver and a verifier. A solver must produce a certificate π of the existence of $x \in L$, whereas a verifier correctly and efficiently determines the validity of π. Consider the problem SAT: a certificate is merely an assignment to the variables of the input formula. The verifier can substitute the assignment into the original formula and quickly discern satisfiability. Notice, however, the verifier scans every bit of π.

An astute reader may ask if it is necessary to read every bit of a certificate. Certainly we must if we rely on substitution as a means of verification. However, if we change the structure of our certificate, can we provide a *new* proof system which is verifiable by reading a constant number of bits of π, if only probabilistically? Surprisingly, we can.

1.1 PCP Verifiers

Let us generalize our notion of a verifier. First, outfit the verifier with an address tape so that it may randomly access individual bits of π. An address is written to the tape by throwing $r(n)$ random coins, where each coin corresponds to a bit on the tape. Next, limit the number of queries to the proof to $q(n)$. Relative to the classical verifier, this restricts the power of the PCP verifier. We allow the verifier to err, and consider the following questions:

- what is the tradeoff between the query complexity and the error incurred by the verifier; and
- how small can the probabilistically checkable proof be relative to the classical proof?

1.1.1 Formalization

A probabilistic verifier admits the following properties: a correct proof of $x \in L$ is always accepted, i.e.,

$$\Pr[\text{accepting a correct proof}] = 1,$$

and, if $x \notin L$, then every claimed certificate of $x \in L$ is rejected with high probability.

Definition 2. *Let L be a language and $q, r : \mathbb{N} \to \mathbb{N}$. L has an $(r(n), q(n))$-*PCP *verifier if there is a polynomial-time probabilistic algorithm $\mathcal{V}$ such that it is:*

- ***Efficient:*** *given an input $x \in \{0,1\}^n$ and random access to a certificate $\pi \in \{0,1\}^*$, $\mathcal{V}$ uses at most $r(n)$ random coins and makes at most $q(n)$ queries to π. $\mathcal{V}$ accepts or rejects when it outputs "1" or "0" respectively.*
- ***Complete:*** *if $x \in L$ then there exists a certificate $\pi \in \{0,1\}^*$ such that $\Pr[\mathcal{V}^\pi(x) = 1] = 1$.*
- ***Sound:*** *if $x \notin L$ then for every certificate $\pi \in \{0,1\}^*$, $\Pr[\mathcal{V}^\pi(x) = 0] > \frac{1}{2}$.*

A language L is in PCP$(r(n), q(n))$, if L has a $(c \cdot r(n), d \cdot q(n))$-PCP verifier, for constants $c, d > 0$.

Notice this formulation dictates the maximum size of a proof that can be verified. Given an address tape of length $r(n)$ we may access $2^{r(n)}$ bits. Querying the proof $q(n)$ times is equivalent to accessing $q(n)$ proofs of size $2^{r(n)}$. Thus, proofs may be at most $q(n) \cdot 2^{r(n)}$ bits.

1.2 The PCP Theorem

The following theorem is known as *the* PCP theorem and was shown by Arora et al. in 1992 [1].

Theorem 3. *NP = PCP$(\log n, 1)$.*

A proof of this result is outside the scope of this project. Theorem 3 shows, in short, that every NP-language has a PCP verifier that verifies certificates of at most *poly*(n) bits by reading a constant number of bits. The following section outlines a proof of a weaker PCP theorem that provides some intuition of the proof of 3.

2 A Proof of a Weak PCP Theorem

The primary contribution of this project is a pedagogical proof of the following PCP theorem:

Theorem 4. *NP $\subseteq$ PCP$(poly(n), 1)$.*

Theorem 4 is weaker than the PCP theorem stated in the previous section in the sense that the proofs it validates may be much larger. The former verifies proofs of exponential size, whereas the latter verifies proofs of polynomial size. It is interesting, still, that exponentially sized proofs can be verified by a constant number of queries.

The remainder of this section outlines a proof of a $(poly(n), 1)$-PCP verifier for CIRCUIT-SAT. Because CIRCUIT-SAT is known to be NP-complete, any NP-language has a PCP verifier by first reducing to CIRCUIT-SAT. For simplicity, we assume the problem is translated into a set of equivalent boolean quadratic constraints. The pith of the verifier is to encode solutions to these constraints as functions which can be quickly tested and decoded. Our main tools include, boolean linear functions, Walsh-Hadamard codes and boolean quadratic equations. Finally, we prove the verifier is efficient, complete and sound.

2.1 Linear Functions and Bit Vectors

Throughout our study of the Walsh-Hadamard code (section 2.2) and theorem 4, we frequently depend on linear functions from $\{0,1\}^n$ to $\{0,1\}$.

Definition 5. *A function $f : \{0,1\}^n \to \{0,1\}$ is linear if, for every $x, y \in \{0,1\}^n$, $f(x+y) = f(x) + f(y)$, and $f(\alpha x) = \alpha f(x)$.*

It is also convenient to reason about bit strings as vectors. Consider the dot product of $u, x \in \{0,1\}^n$ as vectors.

Definition 6. *The dot product of any $u, x \in \{0,1\}^n$ is defined as*

$$\ell_u(x) = \sum_{i=1}^{n} u_i x_i \pmod 2$$

Corollary 7. *$\ell_u(x)$ is a linear function.*

The next lemma shows that if two bit strings u, v are different, then for half the choices of x, $\ell_u(x) \neq \ell_v(x)$. It appears frequently throughout the design of the PCP verifier (section 2.3) in the form, $\Pr_x[\ell_u(x) \neq \ell_v(x)] = \frac{1}{2}$.

Lemma 8. *Let $u, v \in \{0,1\}^n$. If $u \neq v$, then for half the choices of x, $\ell_u(x) \neq \ell_v(x)$.*

Proof. Let $u = u_1 u_2 \ldots u_n \in \{0,1\}^n$ and $v = v_1 v_2 \ldots v_n \in \{0,1\}^n$ such that $u \neq v$. Then u and v differ in at least one bit. Without loss of generality, let k be the least index such that $u_k \neq v_k$. We show that $\ell_u(x) \neq \ell_v(x)$ for half the choices of $x \in \{0,1\}^n$ by a simple counting argument. Let $x = x_1 x_2 \ldots x_n$ and consider the following.

$$\sum_{i=1, i\neq k}^{n} u_i x_i \tag{3}$$

$$\sum_{i=1, i\neq k}^{n} v_i x_i \tag{4}$$

By definition,

$$\ell_u(x) = (3) + u_k x_k \pmod 2 \tag{5}$$

$$\ell_v(x) = (4) + v_k x_k \pmod 2 \tag{6}$$

Suppose (3) = (4); we are forced to set $x_k = 1$ to ensure $\ell_u(x) \neq \ell_v(x)$. Otherwise (3) $\neq$ (4) and setting $x_k = 0$ ensures the inequality. Since there are 2^n possible choices of x, but a single bit is fixed for every choice, there are 2^{n-1} possible choices of x where $\ell_u(x) \neq \ell_v(x)$. $\square$

A useful fact for conceptualizing Walsh-Hadamard codewords (section 2.2) is that the set $\mathcal{L}_n = \{\ell_u \,|\, u \in \{0,1\}^n\}$ is equal to the set of *all* linear functions from $\{0,1\}^n$ to $\{0,1\}$.

Lemma 9. *A function $f : \{0,1\}^n \to \{0,1\}$ is linear if and only if there exists some $u \in \{0,1\}^n$ such that $f(x) = \ell_u(x)$.*

Proof. Starting in the "if" direction, suppose $f(x) = \ell_u(x)$. It follows from corollary 7 that f is linear. In the "only if" direction, suppose $f : \{0,1\}^n \to \{0,1\}$ is linear. We must show there exists some u such that $f(x) = \ell_u(x)$. Consider the following bit vectors,

$$e_1 = 100\ldots 0, e_2 = 010\ldots 0, \ldots, e_n = 000\ldots 1,$$

where $e_i \in \{0,1\}^n$. Any bit vector $x = x_1 x_2 \ldots x_n$ can be decomposed as the summation of the following unit vectors,

$$x = x_1 e_1 + x_2 e_2 + \ldots + x_n e_n$$

Here, x_i acts as a scalar. Since $f(x)$ is linear,

$$\begin{aligned} f(x) &= f(x_1 e_1) + f(x_2 e_2) + \ldots + f(x_n e_n) \\ &= x_1 f(e_1) + x_2 f(e_2) + \ldots + x_n f(e_n) \\ &= \sum_{i=1}^{n} x_i u_i, \quad \text{where } u_i = f(e_i) \\ &= \ell_u(x). \end{aligned}$$

□

Corollary 10. *The set $\mathcal{L}_n = \{\ell_u \,|\, u \in \{0,1\}^n\}$ is the set of all linear functions from $\{0,1\}^n$ to $\{0,1\}$.*

2.2 The Walsh-Hadamard Code

The Walsh-Hadamard code is an encoding of binary strings of length n as binary strings of length 2^n. The Walsh-Hadamard encoding function $\mathcal{WH} : \{0,1\}^n \to \{0,1\}^{2^n}$ is defined as:

Definition 11. $\mathcal{WH}(u) = \ell_u$.

This definition may seem counterintuitive; $\mathcal{WH}$ maps binary strings to binary strings, but clearly ℓ_u is a function. It is useful, then, to think of $\mathcal{WH}(u)$ as a binary string encoding the dot product of u with every $i \in \{0,1\}^n$. Observe that the i^{th} bit of $\mathcal{WH}(u)$, written $\mathcal{WH}(u)_i$, is equal to the dot product of u with i. From corollary 10, we conclude $\mathcal{WH}(u)$ is equivalently the evaluation of every linear function at u.

Definition 12. *If $f \in \{0,1\}^{2^n}$ is equal to $\mathcal{WH}(u)$ for some u, then f is a Walsh-Hadamard codeword.*

Corollary 13. *The set of Walsh-Hadamard codewords $S_{\mathcal{WH}}$ is equivalent to the set of all linear functions.*

2.2.1 Linearity Testing

Linear functions are ideal proof encodings because they can be efficiently tested for membership in $S_{\mathcal{WH}}$. Suppose we are given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$; if $f(x+y) = f(x) + f(y)$, for every $x, y \in \{0,1\}^n$, conclude f is linear. Clearly this test is inefficient, as it requires $O(2^{2^n})$ queries to f. With high confidence, can we determine linearity by querying f a constant number of times? We begin by defining the "closeness" of two functions.

Definition 14. *Let $\rho \in [0,1]$. $f, g : \{0,1\}^n \to \{0,1\}$ are ρ-close if*

$$\Pr_{x \in_R {}^1 \{0,1\}^n} [f(x) = g(x)] \geq \rho.$$

f is ρ-close to a linear function if there is a linear function g such that f and g are ρ-close.

Theorem 15. *Let $f : \{0,1\}^n \to \{0,1\}$ such that for some $\rho > \frac{1}{2}$,*

$$\Pr_{x,y \in_R \{0,1\}^n} [f(x+y) = f(x) + f(y)] \geq \rho.$$

Then f is ρ-close to a linear function.

Now we define an efficient linearity test $\mathcal{T}(f)$: choose $x, y \in \{0,1\}^n$ independently at random and output $f(x+y) = f(x) + f(y)$. Similar to the PCP verifier, we must argue that linear functions always pass $\mathcal{T}(f)$,

[1] $x \in_R \{0,1\}^n$ is x chosen uniformly at random from $\{0,1\}^n$.

and that functions not ρ-close to a linear function are rejected with high probability. Formally, theorem 15 states, for $\rho > \frac{1}{2}$,

$$\Pr[\mathcal{T}(f) = 1] = 1 \quad \text{when } f \text{ is linear} \tag{7}$$

$$\Pr[\mathcal{T}(f) = 0] \geq \frac{1}{2} \quad \text{when } f \text{ is not } \rho\text{-close to linear} \tag{8}$$

2.2.2 Local Decoding

Let $f : \{0,1\}^n \to \{0,1\}$ be a function that is 1- δ-close to a Walsh-Hadamard codeword (read "linear function") g. We argue that g is uniquely defined by lemma 8.

Lemma 16. *Suppose $\delta < \frac{1}{4}$ and the function $f : \{0,1\}^n \to \{0,1\}$ is $(1-\delta)$-close to some linear function g. Then g is unique.*

Proof. Assume by way of contradiction, that g is not unique, i.e., there exists some linear function g' such that $g \neq g'$ and f is $(1-\delta)$-close to both g and g'.

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = g(x)] \geq 1 - \delta > 3/4 \quad \text{(definition)} \tag{9}$$

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = g'(x)] \geq 1 - \delta > 3/4 \quad \text{(definition)} \tag{10}$$

$$\begin{aligned}(9) \text{ and } (10) \implies &\Pr_{x \in_R \{0,1\}^n}[(f(x) = g(x)) \wedge (f(x) = g'(x))] \\ = &\Pr_{x \in_R \{0,1\}^n}[g(x) = g'(x)] > 9/16 > 1/2.\end{aligned}$$

But from lemma 8,

$$\Pr_{x \in_R \{0,1\}^n}[g(x) = g'(x)] = 1/2$$

Hence contradiction. □

Because f is only ρ-close to g, it is possible that $f(x) \neq g(x)$. However, we can easily recover $g(x)$, with high probability. Notice that since g is a Walsh-Hadamard codeword, $g(x+r) = g(x) + g(r)$, for any $r \in \{0,1\}^n$. Then, $g(x+r) - g(r) = g(x)$. Therefore, $f(x+r) - f(r)$ is highly likely to be equal to $g(x)$, thereby allowing efficient decoding.

Definition 17. *Let $\mathcal{D}(f)_x$ be a procedure such that given an input x and oracle access to a function f, decodes $f(x)$. We define $\mathcal{D}(f)_x$ as follows:*

1. *choose $r \in \{0,1\}^n$ at random.*

2. *query $f(x+r)$, $f(r)$ and output $f(x+r) - f(r)$.*

Lemma 18. *If $f : \{0,1\}^n \to \{0,1\}$ is $(1-\delta)$-close to a Walsh-Hadamard codeword g, then, for all $x \in \{0,1\}^n$,*

$$\Pr[\mathcal{D}(f)_x = g(x) = \ell_u(x)] \geq 1 - 2\delta$$

Proof. Let $u \in \{0,1\}^n$ be fixed. Because f is $(1-\delta)$-close to g, we have

$$\Pr_{v \in_R \{0,1\}^n}[f(v) \neq g(v)] \leq \delta.$$

Similarly, $\Pr[f(u+v) \neq g(u+v)] \leq \delta$. By the union-bound,

$$\Pr_{v \in \{0,1\}^n}[f(u) \neq g(u) \vee f(u+v) \neq g(u+v)] \leq 2\delta.$$

Thus, with probability at least $1 - 2\delta$, we have

$$f(u+v) - f(u) = g(u+v) - g(u) = g(v).$$

□

2.3 A PCP Verifier for CIRCUIT-SAT

In this section we will describe the NP-complete language CIRCUIT-SAT, and formulate an equivalent problem of finding satisfying assignments for a set of polynomials.

2.3.1 CIRCUIT-SAT is NP-complete

Definition 19. *The language CIRCUIT-SAT consists of all circuits, represented as strings, that produce a single bit of output and which have a satisfying assignment. An n-input circuit C is in CIRCUIT-SAT iff there exists $u \in \{0,1\}^n$ such that $C(u) = 1$.*

CIRCUIT-SAT is clearly in NP because the satisfying assignment can serve as the certificate, and it can be verified in polynomial-time in the size of the circuit.

Lemma 20. *CIRCUIT-SAT $\leq_p$ 3SAT*

Proof. Let C be a circuit. Without loss of generality, assume that the circuit C has only AND (fan-in = 2) and NOT (fan-in = 1) gates. Now, we map circuit C to a $3CNF$ formula ϕ as follows.

- For all gate nodes v_i of C, create a corresponding variable z_i in ϕ.
- If the node v_i is the output of an AND gate with input from nodes v_j and v_k, then add clauses equivalent to $\neg(z_i \oplus (z_j \wedge z_k))$ to ϕ, i.e.,

  $$(\neg z_i \vee \neg z_j \vee z_k) \wedge (\neg z_i \vee z_j \vee \neg z_k) \wedge (\neg z_i \vee z_j \vee z_k) \wedge (z_i \vee \neg z_j \vee \neg z_k).$$

- If v_i is an output of a NOT gate with input v_j, then add $(z_i \vee z_j) \wedge (\neg z_i \vee \neg z_k)$ to ϕ.
- If v_i is an output gate, then add (z_i) to ϕ.

The formula ϕ is satisfiable if and only if the circuit C is satisfiable, by construction. The reduction is also polynomial-time in the input size of the circuit. $\square$

CIRCUIT-SAT is equivalently expressible as set of boolean constraints. The boolean constraint for any gate i with input j, k is,

$$P_i(z) = \begin{cases} z_i - z_j z_k & \textit{if } i \textit{ is an AND gate} \\ z_i - (1 - z_j) & \textit{if } i \textit{ is a NOT gate} \\ 1 - z_j & \textit{if } i \textit{ is an OUTPUT gate} \\ 0 & \textit{if } i \textit{ is an INPUT gate} \end{cases}$$

Let us phrase CIRCUIT-SAT as the following question: given a boolean circuit C with n gates and $k \leq n$ input gates, is there an assignment $w \in \{0,1\}^n$ such that $C(w) = 1$? In terms of boolean constraints, is there an assignment z to all the gates such that $\forall i, P_i(z) = 0$?

2.3.2 Building a Proof

Suppose the set of constraint equations for CIRCUIT-SAT were polynomials of degree one. In that case, we could easily use the Walsh-Hadamard encoding f as a satisifying assignment, i.e., a proof π. Then to verify π, define the following procedure.

1. Use linearity testing to verify f is δ-close to a linear function. By lemma 15 we accept with probability $1 - \delta$.

2. Since f is δ-close to a linear function, use local decoding to verify that the assignment satisfies the constraints.

With this procedure, along with lemma 8, we can design a PCP verifier satisfying completeness, soundness and efficiency.

However, the constraint equations are not polynomials of degree one, but polynomials of degree two. Decompose a constraint $P_i(z) = Q_i(z)+L_i(z)+c_i$ into three parts,

- $Q_i(z)$: polynomials of degree two,
- $L_i(z)$: polynomials of degree one, and
- c_i : constants.

This suggests that we need to extend the Walsh-Hadamard encoding to include evaluation of all quadratic polynomials at some point $x \in \{0,1\}^n$.

Definition 21. *A quadratic evaluation $quad_x : \{0,1\}^{n\times n} \to \{0,1\}$ is defined as*

$$quad_x(C) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_ix_j = x^TCx.$$

The following properties hold for $quad_x$:

1. $quad_x$ can be represented as a bit string of length 2^{n^2};

2. $quad_x$ is the evaluation of all quadratic polynomials at x, i.e., it is a truth table for all quadratic polynomials, and therefore, a single query retrieves the value of any polynomial of degree 2; and

3. $quad_z$ is linear, namely, for any $n \times n$ matrix $C_1, C_2, quad_x(C_1+C_2) = quad_x(C_1) + quad_x(C_2)$. This is trivially derived from the linearity of matrix operation $(A_1 + A_2)x = A_1x + A_2x$.

Notice the quadratic encoding is similar to the Walsh-Hadmard encoding.

The prover can give the verifier a proof of the assignment z, consisting of a Walsh-Hadamard encoding f and a quadratic encoding g for z. f encodes the satisfying assignment for the linear part of the boolean constraints, while g encodes the corresponding quadratic portion.

2.3.3 Test Procedures for the PCP Verifier

Given two strings f and g, where f is a Walsh-Hadamard encoding and g is a quadratic encoding of z, we define the following test procedure.

1. Test f and g for linearity (lemma 15).
2. Test f and g for consistentency, i.e., the encode the same source word.
3. If f and g are δ-close to linear, then perform self-correction, and recover the unique linear function closest to it.
4. Test that the assignment satisfies the constraint equation using lemma 8.

Intutively, the consistency requirement comes from the observation that although we have two codewords f and g, the source word may *not* be the same. Therefore, we must test the consistency of the codewords, i.e., to verify that for any u_1 and u_2, $f(u_1 u_2^T) = f(u1)f(u2)$.

Definition 22 (Consistency Test)**.** *Given $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^{n\times n} \to \{0,1\}$, the test procedure Quad-Consistency^{f,g} is defined as:*

1. choose z_1 and $z_2 \in_R \{0,1\}^n$;

2. accept if $g(z_1 z_2^T) = f(z_1)f(z_2)$.

$z_1 z_2^T$ is a multiplicatin of an $n \times 1$ vector with a $1 \times n$ vector, giving an $n \times n$ matrix whose ij^{th} element is $(z_1)_i(z_2)_j$

Lemma 23 (Completeness)**.** *If $f : \{0,1\}^n \to \{0,1\}$ is a linear function, say ℓ_z, and $g : \{0,1\}^{n\times n} \to \{0,1\}$ is a linear function, say $quad_z$, for some $z \in \{0,1\}^n$, then Pr[Quad-Consistency^{f,g} accepts] = 1.*

Proof. Since $f = \ell_z$, then for any $y \in \{0,1\}^n$, $f(y) = y^T z$ (matrix representation of the dot product). Similarly, since $g = quad_z$, by the definition of $quad_z$, for any $z_1, z_2 \in_R \{0,1\}^n$,

$$\begin{aligned} g(z_1 z_2^T) &= \sum_{i,j} z_i z_j [z_1 z_2^T]_{i,j} \\ &= \sum z_i z_j [z_1]_i [z_2]_j \\ &= \left(\sum_i z_i [z_1]_i\right)\left(\sum_i z_j [z_2]_j\right) \\ &= f(z_1)f(z_2) \end{aligned}$$

□

The completeness criterion suggests that if f is a Walsh-Hadamard encoding of z, and g is a quadratic encoding of the same z, then Quad-Consistency will always accept. A proof of soundness is given below.

Lemma 24 (Soundness)**.** *If f and g are linear functions and*

$$\Pr[\textit{Quad-Consistency}^{f,g} \textit{ accepts}] > \frac{3}{4},$$

then there exists $z \in \{0,1\}^n$, such that $f = \ell_z$ and $g = quad_z$.

Proof. We will use linearity of f and g and lemma 8 to prove the soundness of the Quad-consistency test procedure.

1. $f = \ell_z$ for some $z \in \{0,1\}^n$ $\quad$ {f is linear}
   therefore for any $z_1, z_2 \in \{0,1\}^n$,

$$\begin{aligned} f(z_1)f(z_2) &= z_1^T z z^T z_2 \\ &= z_1^T C z_2 \quad \textit{where } C = zz^T \textit{ is a } n \times n \textit{ matrix} \end{aligned}$$

2. Since g is linear, there exists a matrix $B = b_{ij}$ such that

$$\begin{aligned} g(z_1 z_2^T) &= \sum_{i,j} b_{ij} \left[z_1 z_2^T\right]_{ij} \\ &= \sum_{i,j} b_{ij} \left(z_1)_i (z_2^T\right)_j \\ &= \sum_{i,j} \left(z_1)_i b_{ij} (z_2^T\right)_j \\ &= z_1^T B z_2 \end{aligned}$$

   So we should compare the matrix B and $C = zz^T$.
   If B = C then it must be the case that $g = quad_z$. Other-wise we will show that the Test *rejects* with probability at-least 1/4 which is 1 - Pr[accept] condition in the definition.
   If $B \neq C$ then for a random vector $z \in \{0,1\}^n$, we have $Bz_2 \neq Cz_2$ with probability at least 1/2 that is,
   $Pr[Bz_2 \neq Cz_2 | B \neq C] \geq 1/2$. [2]

[2]Proof Sketch: Bz_2 and Cz_2 are $n \times 1$ vectors (multiplication of $n \times n$ matrix with $n \times 1$). For any two column vectors to be different, they must differ in at least one row. Let R_i be a row vector. Then, $\Pr[R_1 \neq R_2] = \frac{\sum_1^n \binom{n}{i}}{2^n}$. Using binomial expansion $(1+x)^n + (1-x)^n$ for $x = 1$, and the above expression is equal to $1 - \frac{1}{2^n} \geq 1/2 \; for \; n > 0$

3. For any two bit vectors $y_1, y_2 \in \{0,1\}^n$, and $y_1 \neq y_2$

$$\Pr[z_1^T y_1 \neq z_1^T y_2] = 1/2 \quad \textit{(Lemma 8)}.$$

Choose $y_1 = Bz_1$ and $y_2 = Cz_1$. Thus we have

$$\Pr[z_1^T B z_2 \neq z_1^T C z_2 | Bz_2 \neq Cz_2] = 1/2$$

From Conditional Probability $\Pr[E] = \sum \Pr[E|A_i]Pr[A_i]$

$$\Pr[z_1^T B z_2 \neq z_1^T C z_2 | B \neq C] \geq 1/4$$

That is the Quad-Consistency test accepts with probability $> 3/4$.

□

Recall that we cannot guarantee that the prover provides linear functions f and g. With a linearity test, we are assured that f and g are δ-close to linear function. Thus, we can recover the encoded word by defining a Quad-Correction procedure similar to local decoding for Walsh-Hadamard codewords.

We show that it is possible to do self-correction (quad-correction) using the consistency test above, with required soundness and completeness properties for the procedure. Then, we query the oracle (Walsh-Hadamard and Quadratic encoding of the assignment) to get the evaluation of any linear and quadratic polynomial in our set of constraints at any given input.

Definition 25 (Quad-Correction). *Given $f : \{0,1\}^n \rightarrow \{0,1\}$ and $g : \{0,1\}^{n\times n} \rightarrow \{0,1\}$, $Quad - Correction^{f,g}$ is defined as:*

1. *choose $z_1, z_2 \in_R \{0,1\}^n$ and $M \in_R \{0,1\}^{n\times n}$;*
2. *accept if $g(z_1 z_2^T + M) - g(M) = f(z_1)f(z_2)$.*

Lemma 26 (Completeness Property). *If $f = \ell_z$ and $g = quad_z$ for some $z \in \{0,1\}^n$, then*

$$Pr[Quad - correction^{f,g} \ accepts = 1]$$

Proof. For any $z_1, z_2 \in \{0,1\}^n$ and $M \in \{0,1\}^{n\times n}$

$$\begin{aligned} g(z_1 z_2^T + M) - g(M) &= g(z_1 z_2^T) + g(M) - g(M) \quad \text{(linearity of } quad_z\text{)} \\ &= g(z_1 z_2^T) \\ &= f(z_1)f(z_2) \quad \text{(given } f = \ell_z \text{ and Quad-Consistency)} \end{aligned}$$

Therefore, $\Pr[g(z_1 z_2^T + M) - g(M) = f(z_1)f(z_2)] = 1$. □

Lemma 27. *If f is δ-close to ℓ_z for some $z \in \{0,1\}^n$, g is δ-close to some linear function and* $\Pr[$*Quad-Correction*^{f,g} *accepts*$] > \frac{3}{4} + 4\delta$*, then g is δ-close to $quad_z$.*

Proof. Since f is δ-close to ℓ_z and $z_1, z_2 \in_R \{0,1\}^n$,

$$\Pr[f(z_1) \neq \ell_z(z_1)] \leq \delta \tag{11}$$
$$\Pr[f(z_2) \neq \ell_z(z_2)] \leq \delta. \tag{12}$$

Let f' be the linear function that is δ-close to g. Since $z_1 z_2^T + M$ and $M \in_R \{0,1\}^{n \times n}$ (uniformly random elements, but not independent)

$$\Pr[g(z_1 z_2^T + M) \neq f'(z_1 z_2^T + M)] \leq \delta \tag{13}$$
$$\Pr[g(M) \neq f'(M)] \leq \delta \tag{14}$$

From (11), (12), (13) and (14),

$$\begin{aligned}\Pr[&f(z_1) \neq \ell_z(z_1) \cup \\ &f(z_2) \neq \ell_z(z_2) \cup \\ &g(z_1 z_2^T + M) \neq f'(z_1 z_2^T + M) \cup \\ &g(M) \neq f'(M)] \leq 4\delta.\end{aligned}$$

Since,

$$\begin{aligned}&\Pr[g(z_1 z_2^T + M) - g(M) = f(z_1) f(z_2)] > 3/4 + 4\delta \\ &\implies \\ &\Pr[\ell_z(z_1)\ell_z(z_2) = g(z_1 z_2^T + M) - g(M)] \geq \frac{3}{4},\end{aligned}$$

it follows from the soundness of Quad-Consistency that $g = quad_z$. $\square$

2.3.4 PCP Verifier for CIRCUIT-SAT

Based on the test procedures developed so far, we now design a PCP verifier for CIRCUIT-SAT that accepts a correct proof with probability 1, and rejects with high probabilty, if the proof is not δ-close to a satisfying assignment. In both the scenarios, the verifier makes only a constant number of queries to the proof.

Definition 28 (PCP verifier for CIRCUIT-SAT)**.** *The* PCP*-verifier expects as proof the functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^{n \times n} \to \{0,1\}$. Given a circuit C and oracle access to f and g, the verifier performs the following steps:*

1. *Linearity testing for f:*
   (a) *choose $z_1, z_2 \in_R \{0,1\}^n$, and*
   (b) *output $f(z_1 + z_2) = f(z_1) + f(z_2)$.*

   Query Complexity = 3, Randomness = 2n

2. *Linearity testing for g:*
   (a) *choose $M_1, M_2 \in_R \{0,1\}^{n\times n}$, and*
   (b) *output $g(M_1 + M_2) = g(M_1) + g(M_2)$.*

   Query Complexity = 3, Randomness = $2n^2$

3. *Consistency testing of f and g given both are δ-close to linear functions:*
   (a) *choose $z_1, z_2 \in_R \{0,1\}^n$ and $M \in_R \{0,1\}^{n\times n}$, and*
   (b) *output $f(z_1)\dot{f}(z_2) = g(z_1 z_2^T + M) - g(M)$.*

   Query Complexity = 4, Randomness = $2n + n^2$

4. *Satisfiability of C:*
   (a) *choose $\alpha_1, \alpha_2, \ldots, \alpha_n \in_R \{0,1\}, r \in_R \{0,1\}^n$ and $M \in_R \{0,1\}^{n\times n}$;*
   (b) *decompose the function $P(z) = \sum_{i=1}^n \alpha_i P_i(z)^3$ as the sum of quadratic part $Q(z) = z^T Bz$, a linear part $L(z) = y^T z$ and a constant c; and*
   (c) *check that $[g(M+B) - g(M)] + [f(y+r) - f(r)] + c = 0$.*

   Query Complexity = 4, Randomness = $2n + n^2$

2.3.5 Analysis

Let us analyze the PCP verifier $\mathcal{V}$ described in the preceding section. Recall from definition 2, we must show efficiency, completeness and soundness.

Lemma 29 (Efficiency). *$\mathcal{V}$ is efficient, that is, $r(n) = poly(n)$ and $q(n) = O(1)$.*

[3]Without loss of generality, assume that all constraints are linearly independent. To reduce the number of queries needed to test the satisfiability of the circuit, use lemma 8.

Proof. The number of random bits required for each step of the verifier is provided up. Summing them together yields

$$r(n) = 2n + 2n^2 + (2n + n^2) + (2n + 2n^2) = O(n^2).$$

Likewise, for the query complexity,

$$q(n) = 3 + 3 + 4 + 4 = 14 = O(1).$$

Furthermore, the length of a proof is $2^n + 2^{n^2}$. □

Lemma 30 (Completeness)**.** *If z is a satisfying assignment for a circuit C, then $\Pr[\mathcal{V}^\pi = 1] = 1$, where π is a proof of z.*

Proof. By construction. □

Lemma 31 (Soundness)**.** *There exists a $\delta_0 > 0$, such that for all $\delta < \delta_0$, $\mathcal{V}$ accepts with probability at least $1 - \delta$, then the following conditions hold:*

1. *there exists a satisfying assignment z for C;*

2. *f is δ-close to ℓ_z; and*

3. *g is δ-close to $quad_z$.*

Proof by contradiction. Suppose the claim is false for some $\delta_0 = \frac{1}{20}$. Then there exists a $\delta < \delta_0$ such that $\Pr[\mathcal{V} = 1] > 1 - \delta$, but there does not exist a satisfying assignment z such that f is δ-close to ℓ_z and g is δ-close to $quad_z$. Then at least one of the following should be true:

1. f is not δ-close from linear. $\mathcal{V}$ will reject with probability $\geq \delta$ (by soundness of linearity testing).

2. g is not δ-close from linear. $\mathcal{V}$ will reject with probability $\geq \delta$ (by soundness of linearity testing).

3. f is δ-close to a linear function ℓ_z and g is δ-close to some linear function g', but $g' \neq quad_z$. $\mathcal{V}$ rejects with probability $\geq \frac{1}{4} - 4\delta$ (by soundness of Quad-Correction).

4. f is δ-close to some ℓ_z and g is δ-close to $quad_z$ for some z, but z is not a satisfying assignment for C, that is, $\exists i : P_i(z) \neq 0$. Notice:

$$\textit{Lemma 8} \implies \Pr[\textstyle\sum \alpha_i P_i(z) \neq 0] = 1/2 \tag{15}$$

$$\textit{Lemma 18} \implies \Pr[f(y+u) - f(u) \neq \ell_z(y)] \leq 2\delta \tag{16}$$

$$\textit{Lemma 18} \implies \Pr[g(M+B) - g(M) \neq quad_z(B)] \leq 2\delta \tag{17}$$

Therefore, (15), (16) and (17) imply

$$\Pr[(g(M+B)-g(M))+(f(y+r)-f(r))+c=\sum \alpha_i P_i(z) \neq 0] \geq 1/2-4\delta,$$

that is, the satisfiability test rejects with probability at least $1/2-4\delta$. For sufficiently small δ, $1/4-4\delta$ and $1/2-4\delta$ are no smaller than δ. Hence, in each of the four cases above, the PCP verifier rejects with probability at least δ. This is a contradiction.

$\square$

Thus, there is a verifier which accepts an exponentially sized proof for CIRCUIT-SAT, and satisfies the three properties of the PCP verifier outlined in definition 2. In particular, we have shown that CIRCUIT-SAT $\in$ $\text{PCP}(O(n^2), 14)$.

2.4 Conclusion

In this paper, we proved a weaker notion of the PCP theorem. We hope that the intution developed in the process encourages the reader to explore the complete proof for the theorem. For an informal history and the connection of the interactive proofs with PCP theorem, we refer the reader to [3]. Irit Dinur gives a dramatically simple construction of the probabilitically checkable proofs. Interested reader can refer to these excellent introduction and complete proof of the PCP theorem [2] [4] [5].

We conclude by stating minor observations. First, adding randomness to the verifier does not increase the power of the verifier, but does increase efficiency. We can improve confidence by repeating tests throughout verification. By endowing the proofs with redundancy, they have the resilience to convince the verifier even when a fraction of the bits are flipped. Low degree polynomials (over a field) are known to have efficient error-correcting properties in addition to their nice algebraic structure. This motiviates the use of Walsh-Hadamard encoding, as well as the quadratic encoding developed in 2.3.2. Lastly, the seemingly counterintuitive properties of PCP highlight the fact that the "format" in which proofs are expected is a powerful tool to aid verifiers. However, the format present in the preceeding sections is exceptionally large. The PCP theorem proves that it is possible to maintain constant query complexity, but reduce the length of the proof to a size polynomial in the length of the input. Recent research has improved the size of PCP certificates to $O(n \log n)^{O(1)}$, although, the reduction comes at the cost of query complexity. This is an active research topic.

References

[1] *Probabilistic checking of proofs: A new characterization of NP* Arora, S. and Safra, S. Journal of the ACM (JACM), Volume 45, 1998.

[2] Radhakrishnan, J. and Sudan, M. *On Dinur's proof of the PCP theorem* BULLETIN-AMERICAN MATHEMATICAL SOCIETY Volume 44, 2007.

[3] Dana Moshkovitz. *A History of the PCP Theorem* http://people.csail.mit.edu/dmoshkov/courses/pcp/pcp-history.pdf Course 6.895, Fall 2010.

[4] Arora, S. and Barak, B. *Computational Complexity: A Modern Approach* Chapter 11 and 22, Cambridge University Press, 2009.

[5] *PCPs, codes and inapproximability - Autumn 2007* Prahladh Harsha http://www.tcs.tifr.res.in/ prahladh/teaching/07autumn/.