Advanced Computational Complexity

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 600

1. PCP Theorem and Hardness of Approximation
   - Approximate Solutions to NP-Hard Optimization Problems
   - Two Views of the PCP Theorem
   - Equivalence of the Two Views
   - Hardness of Approximation for Vertex Cover and Independent Set
   - $\mathrm{NP} \subseteq \mathrm{PCP}(\mathrm{poly}(n), 1)$: PCP from the Walsh-Hadamard Code

Subsection 1

Approximate Solutions to NP-Hard Optimization Problems

Approximating NP-Hard Optimization Problems

- One of the main motivations for the theory of NP-completeness was to understand the computational complexity of computing optimum solutions to combinatorial problems such as TSP or INDSET.
- Since $\mathsf{P} \neq \mathsf{NP}$ implies that thousands of NP-hard optimization problems do not have efficient algorithms, attention then focused on whether or not they have efficient approximation algorithms.
- In many practical settings, obtaining an approximate solution may be almost as good as solving it exactly and could be a lot easier.
- Therefore , finding the best possible approximation algorithms for NP-hard optimization problems is important.

Limits of Approximation and PCP Theorem

- We would like to understand whether or not we could approximate interesting NP-hard problems within an arbitrary precision.
- Many researchers suspected that there are inherent limits to approximation.
- Proving such limits was the main motivation behind the discovery of the PCP Theorem.

MAX3SAT

- MAX3SAT is the following problem:
  - The input is a 3CNF Boolean formula φ.
  - We seek an assignment that maximizes the number of satisfied clauses.
- The corresponding decision problem, 3SAT, is NP-complete.
- So MAX3SAT is NP-hard.

Approximation of MAX3SAT

- We define an approximation algorithm for MAX3SAT.

Definition (Approximation of MAX3SAT)

- Consider a 3CNF formula φ.
- The **value** of φ, denoted by $\mathsf{val}(\varphi)$, is the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

  In particular, φ is satisfiable iff $\mathsf{val}(\varphi) = 1$.
- For every $\rho \leq 1$, an algorithm A is a **ρ-approximation algorithm** for MAX3SAT if, for every 3CNF formula φ with m clauses, $A(\varphi)$ outputs an assignment satisfying at least

$$\rho \cdot \mathsf{val}(\varphi) m$$

  of φ's clauses.

1/2-Approximation for MAX3SAT

- We describe a polynomial-time algorithm that computes a $\frac{1}{2}$-approximation for MAX3SAT.
- The algorithm assigns values to the variables in a greedy fashion.
  - The i-th variable is assigned the value that results in satisfying at least $\frac{1}{2}$ the clauses in which it appears.
  - Any clause that gets satisfied is removed and not considered in assigning values to the remaining variables.
- The final assignment satisfies at least $\frac{1}{2}$ of all clauses.
- This is at least half of the maximum that the optimum assignment could satisfy.

7/8-Approximation for MAX3SAT (Cont'd)

- Using semidefinite programming one can also design a polynomial-time $\left(\frac{7}{8} - \epsilon\right)$-approximation algorithm, for every $\epsilon > 0$.
- Obtaining such a ratio is trivial if we restrict ourselves to 3CNF formulae with three distinct variables in each clause.
- A random assignment has probability $\frac{7}{8}$ to satisfy each clause.
- Linearity of expectations implies that a random assignment is expected to satisfy a $\frac{7}{8}$ fraction of the clauses.
- Thus, we get a simple probabilistic $\frac{7}{8}$-approximation algorithm.

MinVertexCover

- We already encountered the decision problem VertexCover.
- The optimization version is MinVertexCover.
  - The input is a graph.
  - We wish to determine the size of the minimum vertex cover.
- For $\rho \leq 1$, a ρ-approximation algorithm for MinVertexCover:
  - Receives input a graph G;
  - Outputs a vertex cover whose size is at most $\frac{1}{\rho}$ times the size of the minimum vertex cover.

1/2-Approximation for MinVertexCover

- We present a $\frac{1}{2}$-approximation algorithm for MinVertexCover.
- Start with $S \leftarrow \emptyset$.
- Pick any edge in the graph e_1, and add both its endpoints to S.
- Delete these two vertices and all edges adjacent to them.
- Iterate this process:
  - Picking edges $e_2, e_3, \ldots$;
  - Adding their endpoints to S until the graph becomes empty.
- At the end, S is such that every graph edge has an endpoint in S.
- Thus, S is a vertex cover.

The 1/2-Approximation Property

- The sequence of edges
$$e_1, e_2, \ldots$$
used to buildup S are pairwise disjoint.
- That is, they form a matching.
- The cardinality of S is twice the number of edges in this matching.
- The minimum vertex cover must by definition include at least one endpoint of each matching edge.
- Thus the cardinality of S is at most twice the cardinality of the minimum vertex cover.

Subsection 2

Two Views of the PCP Theorem

PCP Theorem and Locally Testable Proofs

- The PCP Theorem can be viewed in two alternative ways:
  - One view involves new, extremely robust proof systems.
  - The other approximating combinatorial optimization problems.
- First, the PCP Theorem provides a new kind of proof systems.

  Example: Suppose the goal is to convince that a Boolean formula is satisfiable.
- One way is to present the usual certificate, i.e., a satisfying assignment, which one can then check by substituting back into the formula.
- The drawback is that this requires reading the entire certificate.

PCP Theorem and Locally Testable Proofs (Cont'd)

- The PCP Theorem presents an interesting alternative.
- The certificate may be rewritten so that it can be verified by probabilistically selecting a constant number of locations (as low as 3 bits) to examine in it.
- Furthermore, this probabilistic verification has the following properties.
  1. A correct certificate will never fail to convince.
     That is, no choice of the random coins will cause rejection.
  2. If the formula is unsatisfiable, then rejection is guaranteed for every claimed certificate with high probability.

Randomized Mathematical Proofs

- Boolean satisfiability is NP-complete.
- Thus, every other NP language can be deterministically and efficiently reduced to it.
- This implies that the PCP Theorem applies to every NP language.
- There is one counterintuitive consequence.
- Let $\mathcal{A}$ be any one of the usual axiomatic systems of mathematics for which proofs can be verified by a deterministic TM in time that is polynomial in the length of the proof.
- Consider the language in NP

$$L = \{\langle \varphi, 1^n \rangle : \varphi \text{ has a proof in } \mathcal{A} \text{ of length} \leq n\}.$$

Randomized Mathematical Proofs (Cont'd)

- The PCP Theorem asserts that L has probabilistically checkable certificates.
- Such a certificate can be viewed as an alternative notion of "proof" for mathematical statements that is just as valid as the usual notion.
- In standard mathematical proofs, every line of the proof has to be checked to verify its validity.
- This new notion guarantees that proofs are probabilistically checkable by examining only a constant number of bits in them.

From NP to PCP

- A language L is in NP if there is a poly-time Turing machine V ("verifier") that, given input x, checks certificates (or membership proofs) to the effect that $x \in L$.
- Let V^π denote "a verifier with access to certificate π".
- Then

$$\begin{array}{rcl} x \in L & \Longrightarrow & \exists\pi(V^\pi(x) = 1) \\ x \notin L & \Longrightarrow & \forall\pi(V^\pi(x) = 0). \end{array}$$

- The class PCP is a generalization adopting the following changes.
- First, the verifier is probabilistic.
- Second, the verifier has random access to the proof string Π.

From NP to PCP (Cont'd)

- The second condition means that each bit of the proof string can be independently queried by the verifier via a special address tape.
- That is, if the verifier desires the i-th bit in the proof string, it:
  - Writes i on the address tape;
  - Receives the bit $\pi[i]$.
- In PCP the queries are a precious resource to be used sparingly.
- The address size is logarithmic in the proof size.
- So this model in principle allows a polynomial-time verifier to check membership proofs of exponential size.

Adaptive vs. Nonadaptive Verifiers

- Verifiers can be adaptive or nonadaptive.
- A **nonadaptive verifier** selects its queries based only on:
  - Its input;
  - Its random tape.
- That is, no query depends upon the responses to any of the prior queries.
- An **adaptive verifier** can rely upon bits it has already queried in π to select its next queries.
- In the definition of PCP, we restrict to nonadaptive verifiers.
- The choice is made because most PCP Theorems can be proved using nonadaptive verifiers.

PCP-Verifier

Definition (PCP Verifier)

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

- **Efficiency**: On input a string $x \in \{0,1\}^n$ and, given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$, called the proof:
  - V uses at most $r(n)$ random coins;
  - V makes at most $q(n)$ nonadaptive queries to locations of π.

  Then, it outputs 1 (for "accept") or 0 (for "reject").

  We let $V^\pi(x)$ denote the random variable representing V's output on input x and with random access to π.

PCP-Verifier (Cont'd)

Definition (PCP Verifier Cont'd)

- **Completeness**: If $x \in L$, then there exists a proof $\pi \in \{0,1\}^*$, such that
$$\Pr[V^\pi(x) = 1] = 1.$$
We call this π the **correct proof** for x.
- **Soundness**: If $x \notin L$, then for every $\pi \in \{0,1\}^*$,
$$\Pr[V^\pi(x) = 1] \leq \frac{1}{2}.$$

We say that a language L is in $\mathrm{PCP}(r(n), q(n))$ if there are some constants $c, d > 0$, such that L has a $(c \cdot r(n), d \cdot q(n))$-PCP verifier.

Remarks on the PCP Theorem

- The content of the PCP Theorem is that every NP language has a highly efficient PCP verifier.

Theorem (The PCP Theorem)

$\mathsf{NP} = \mathsf{PCP}(\log n, 1)$.

- We begin with some remarks.

1. The Soundness condition stipulates that if $x \notin L$, then the verifier has to reject every proof with probability at least $\frac{1}{2}$.

   This is the most difficult part of the proof.
2. The restriction that proofs are of length $\leq q2^r$ is inconsequential.

   Such a verifier can look on at most this number of locations with nonzero probability over all 2^r choices for its random string.

Remarks on the PCP Theorem (Cont'd)

3. We have

$$\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{O(r(n))} q(n)).$$

A nondeterministic machine could:

- Guess the proof in $2^{O(r(n))} q(n)$ time;
- Verify it deterministically by running the verifier for all $2^{O(r(n))}$ possible choices of its random coin tosses.

If the verifier accepts for all these possible coin tosses, then the nondeterministic machine accepts.

As a special case of this, we get

$$\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NTIME}(2^{O(\log n)}) = \mathrm{NP}.$$

This is the trivial direction of the PCP Theorem.

Remarks on the PCP Theorem (Cont'd)

4. The statement of the PCP Theorem allows verifiers for different NP languages to use a different number of query bits (so long as this number is constant).

   But every NP language is polynomial-time reducible to SAT.

   So all these numbers can be upper bounded by a universal constant.

   Namely, the number of query bits required by a verifier for SAT.

5. The constant $\frac{1}{2}$ in the soundness requirement is arbitrary.

   A PCP verifier with soundness $\frac{1}{2}$ that uses r coins and makes q queries can be converted into a PCP verifier using cr coins and cq queries with soundness 2^{-c} by just repeating its execution c times.

   Consequently, changing $\frac{1}{2}$ to any other positive constant smaller than 1 will not change the class of languages defined.

The PCP System for Graph Non-Isomorphism

- The language GNI of pairs of non-isomorphic graphs is in $\mathrm{PCP}(\mathrm{poly}(n), 1)$.
- The input for GNI is $\langle G_0, G_1 \rangle$, where G_0, G_1 have both n nodes.
- The verifier expects the proof π to contain, for each labeled graph H with n nodes, a bit $\pi[H] \in \{0, 1\}$, such that:
  - $\pi[H] = 0$, if $H \cong G_0$;
  - $\pi[H] = 1$, if $H \cong G_1$;
  - $\pi[H]$ arbitrary, if neither case holds.
- So π is an (exponentially long) array of bits indexed by the (adjacency matrix representations of) all possible n-vertex graphs.

The PCP System for Graph Non-Isomorphism (Cont'd)

- The verifier picks:
    - $b \in \{0,1\}$ at random;
    - A random permutation.

  She applies the permutation to the vertices of G_b to obtain an isomorphic graph H.

  She queries the corresponding bit of π.

  She accepts iff the bit queried is b.
- If $G_0 \ncong G_1$, then clearly a proof π that makes the verifier accept with probability 1 can be constructed.
- If $G_1 \cong G_2$, then the probability that any π makes the verifier accept is at most $\frac{1}{2}$.

The Scaled-Up PCPTheorem

- We use the notation

$$\mathrm{PCP}(\mathrm{poly}(n), 1) = \bigcup_{c \geq 1} \mathrm{PCP}(n^c, 1).$$

Theorem (Scaled-up PCP Theorem)

$\mathrm{PCP}(\mathrm{poly}(n), 1) = \mathrm{NEXP}$.

- This theorem can be thought of as a "scaled-up" version of the PCP Theorem.
- The proof uses similar techniques to the original proof of the PCP Theorem and the IP = PSPACE theorem.

PCP and Hardness of Approximation

- The second view of the PCP Theorem is that it shows that for many NP optimization problems, computing approximate solutions is no easier than computing exact solutions.
- For concreteness, we focus, first, on MAX3SAT.
  - Until 1992, we did not know whether or not MAX3SAT has a polynomial-time ρ-approximation algorithm for every $\rho < 1$.
  - The PCP Theorem means that the answer is NO (unless P = NP).

Theorem (PCP Theorem: Hardness of Approximation View)

There exists $\rho < 1$, such that for every $L \in \mathrm{NP}$, there is a polynomial-time function f mapping strings to (representations of) 3CNF formulas, such that

$$x \in L \implies \mathrm{val}(f(x)) = 1,$$
$$x \notin L \implies \mathrm{val}(f(x)) < \rho$$

More on Hardness of Approximation

- The PCP Theorem immediately implies the following

Corollary

There exists some constant $\rho < 1$, such that, if there is a polynomial-time ρ-approximation algorithm for $\textsc{Max3Sat}$, then $\mathsf{P} = \mathsf{NP}$.

- Suppose that:
  - $L \in \mathsf{NP}$;
  - A is a ρ-approximation algorithm for $\textsc{Max3Sat}$.

  The two conditions in the PCP Theorem imply that $x \in L$ iff $A(f(x))$ yields an assignment satisfying at least a ρ fraction of $f(x)$'s clauses.

  Therefore, for every $L \in \mathsf{NP}$, we get a way to convert the ρ-approximation algorithm A for $\textsc{Max3Sat}$ into an algorithm deciding L.

Subsection 3

Equivalence of the Two Views

Introducing Constraint Satisfaction Problems

- We show the equivalence of:
  - The "proof view" of the PCP Theorem;
  - The "hardness of approximation view" of the PCP Theorem.
- A constraint satisfaction problem (CSP) is a generalization of 3SAT.
- It allows clauses of arbitrary form (instead of just OR of literals).
- It also allows dependence upon more than 3 variables.

Constraint Satisfaction Problems

Definition (Constraint Satisfaction Problems (CSP))

Let q be a natural number.
A $q\mathrm{CSP}$ instance φ is a collection of functions

$$\varphi_1, \ldots, \varphi_m : \{0,1\}^n \to \{0,1\},$$

called **constraints**, such that each function φ_i depends on at most q of its input locations.
This means that, for every $i \in [m]$, there exist $j_1, \ldots, j_q \in [n]$ and $f : \{0,1\}^q \to \{0,1\}$, such that

$$\varphi_i(\boldsymbol{u}) = f(u_{j_1}, \ldots, u_{j_q}), \quad \text{for every } \boldsymbol{u} \in \{0,1\}^n.$$

Constraint Satisfaction Problems (Cont'd)

Definition (Constraint Satisfaction Problems (CSP) Cont'd)

An **assignment** $\boldsymbol{u} \in \{0,1\}^n$ **satisfies** constraint φ_i if

$$\varphi_i(\boldsymbol{u}) = 1.$$

The fraction of constraints satisfied by $\boldsymbol{u}$ is

$$\frac{\sum_{i=1}^m \varphi_i(\boldsymbol{u})}{m}.$$

We let $\mathrm{val}(\varphi)$ denote the max over all $\boldsymbol{u} \in \{0,1\}^n$.
We say that φ is **satisfiable** if

$$\mathrm{val}(\varphi) = 1.$$

We call q the **arity** of φ.

Size and Representation

- 3SAT is the subcase of qCSP where:
  - $q = 3$;
  - The constraints are OR's of the involved literals.
- We define the **size** of a qCSP-instance φ to be the number m of constraints it has.
- Because variables not used by any constraints are redundant, we always assume $n \leq qm$.
- Note that a qCSP instance over n variables with m constraints can be described using $O(mq \log n 2^q)$ bits.
- In all cases of interest q will be a constant independent of n, m.

Greedy Approximation Algorithm

- Consider the problem $\text{Max}q\text{CSP}$ of maximizing the number of satisfied constraints in a given $q\text{CSP}$ instance.
- Recall the simple greedy approximation algorithm for 3SAT.
- It can be generalized for $\text{Max}q\text{CSP}$.
- Let φ be any $q\text{CSP}$ instance with m constraints.
- This algorithm will output an assignment satisfying

$$\frac{\mathsf{val}(\varphi)}{2^q}m$$

constraints.

Gap CSP

- We prove the equivalence of the two formulations of the PCP Theorem.
- We show that they are both equivalent to the NP-hardness of a certain gap version of $q\mathrm{CSP}$.

Definition (Gap CSP)

Let $q \in \mathbb{N}$ and assume $\rho \leq 1$.
Define ρ-$\mathrm{G}\textsc{ap}q\mathrm{CSP}$ to be the problem of determining, for a given $q\mathrm{CSP}$-instance φ, which of the following holds:

(1) $\mathrm{val}(\varphi) = 1$;

In this case we say φ is a **YES instance** of ρ-$\mathrm{G}\textsc{ap}q\mathrm{CSP}$;

(2) $\mathrm{val}(\varphi) < \rho$;

In this case we say φ is a **NO instance** of ρ-$\mathrm{G}\textsc{ap}q\mathrm{CSP}$.

Gap CSP (Cont'd)

Definition (Gap CSP Cont'd)

We say that ρ-$\text{GAP}q\text{CSP}$ is NP-**hard** if, for every language L in NP, there is a polynomial time function f mapping strings to (representations of) $q\text{CSP}$ instances satisfying:

- **Completeness**: $x \in L \Rightarrow \text{val}(f(x)) = 1$;
- **Soundness**: $x \notin L \Rightarrow \text{val}(f(x)) < \rho$.

Theorem (Gap CSP)

There exist constants $q \in \mathbb{N}$, $\rho \in (0, 1)$, such that ρ-$\text{GAP}q\text{CSP}$ is NP-hard.

The PCP Theorem Implies the Gap CSP Theorem

- Assume that $\mathsf{NP} \subseteq \mathsf{PCP}(\log n, 1)$.

  We will show that $\frac{1}{2}$-$\mathrm{GAP}q\mathrm{CSP}$ is NP-hard, for some constant q.

  It is enough to reduce a single NP-complete language, such as $3\mathrm{SAT}$, to $\frac{1}{2}$-$\mathrm{GAP}q\mathrm{CSP}$, for some constant q.

  By hypothesis, $3\mathrm{SAT}$ has a PCP system in which the verifier V:
  - Makes a constant number of queries, which we denote by q;
  - Uses $c \log n$ random coins, for some constant c.

  Suppose we are given input x and $r \in \{0,1\}^{c \log n}$.

  Define $V_{x,r}$ to be the function that, on input a proof π, outputs 1 if the verifier will accept the proof π on input x and coins r.

The PCP Theorem Implies the Gap CSP Theorem (Cont'd)

- Define $V_{x,r}$ to be the function that, on input a proof π, outputs 1 if the verifier will accept the proof π on input x and coins r.

  Note that $V_{x,r}$ depends on at most q locations.

  Thus, for every $x \in \{0,1\}^n$, the collection

$$\varphi = \{V_{x,r}\}_{r \in \{0,1\}^{c \log n}}$$

  is a polynomial-sized $q\mathrm{CSP}$ instance.

  Furthermore, since V runs in polynomial-time, the transformation of x to φ can also be carried out in polynomial-time.

  By the completeness and soundness of the PCP system:
  - If $x \in 3\mathrm{SAT}$, then φ will satisfy $\mathrm{val}(\varphi) = 1$.
  - If $x \notin 3\mathrm{SAT}$, then φ will satisfy $\mathrm{val}(\varphi) \leq \frac{1}{2}$.

The Gap CSP Theorem Implies the PCP Theorem

- Suppose that ρ-$\mathrm{G}\scriptstyle\mathrm{AP}$$q\mathrm{CSP}$ is NP-hard for some constants $q, \rho < 1$.

  We translate this into a PCP system with q queries, ρ soundness, and logarithmic randomness, for any language L:

  Suppose the input is x.
  The verifier runs the reduction $f(x)$ to obtain a $q\mathrm{CSP}$ instance

$$\varphi = \{\varphi_i\}_{i=1}^m.$$

  It expects the proof π to be an assignment to the variables of φ.
  It verifies the proof by choosing a random $i \in [m]$ and checking that φ_i is satisfied (by making q queries).

  - If $x \in L$, then the verifier accept with probability 1;
  - If $x \notin L$, it accepts with probability at most ρ.

  The soundness can be boosted to $\frac{1}{2}$ at the expense of a constant factor in the randomness and number of queries.

The Second Version of the PCP is Equivalent to the CSP

- 3CNF formulas are a special case of 3CSP instances.
  So the PCP Theorem implies the CSP Theorem.
- We now show the converse.
  Let $\rho > 0$ and $q \in \mathbb{N}$ be such that $(1-\epsilon)$-$\textsc{Gap}q\mathrm{CSP}$ is NP-hard.
  Let φ be a $q\mathrm{CSP}$ instance over n variables with m constraints.
  Each constraint φ_i of φ can be expressed as an AND of $\leq 2^q$ clauses, with each clause the OR of $\leq q$ variables or their negations.
  Let φ' be the collection of at most $m2^q$ clauses corresponding to all the constraints of φ.
  - Suppose φ is a YES instance of $(1-\epsilon)$-$\textsc{Gap}q\mathrm{CSP}$ (i.e., satisfiable).
    Then there exists an assignment satisfying all the clauses of φ'.
  - Suppose φ is a NO instance of $(1-\epsilon)$-$\textsc{Gap}q\mathrm{CSP}$.
    Then every assignment violates at least an ϵ-fraction of the φ_i's.
    Hence, it violates at least an $\frac{\epsilon}{2^q}$ fraction of the constraints of φ'.

Second Version of PCP and CSP (Cont'd)

- We can use the Cook-Levin technique to transform any clause C on q variables $u_1, \ldots, u_q$ to a set $C_1, \ldots, C_q$ of clauses over the variables $u_1, \ldots, u_q$ and additional auxiliary variables $y_1, \ldots, y_q$, such that:
  (1) Each clause C_i is the OR of at most three variables or their negations;
  (2) If $u_1, \ldots, u_q$ satisfy C, then there is an assignment to $y_1, \ldots, y_q$, such that $u_1, \ldots, u_q$, $y_1, \ldots, y_q$ simultaneously satisfy $C_1, \ldots, C_q$;
  (3) If $u_1, \ldots, u_q$ does not satisfy C then for every assignment to $y_1, \ldots, y_q$, there is some clause C_i that is not satisfied by $u_1, \ldots, u_q$, $y_1, \ldots, y_q$.

  Let φ'' denote the collection of at most $qm2^q$ clauses over the $n + qm2^q$ variables obtained in this way from φ'.

Second Version of PCP and CSP (Cont'd)

- Note that φ'' is a 3SAT formula.
  Our reduction will map φ to φ''.
  - Completeness Suppose φ is satisfiable.
    Then so is φ'.
    Hence, the same holds for φ''.
  - Soundness Suppose that every assignment violates at least an ϵ fraction of the constraints of φ.
    Then every assignment violates at least an $\frac{\epsilon}{2^q}$ fraction of the constraints of φ'.
    So every assignment violates at least an $\frac{\epsilon}{q2^q}$ fraction of the constraints of φ''.

Summary: Two views of the PCP Theorem

Proof view		Hardness of Approximation View
PCP verifier (V)	$\longleftrightarrow$	CSP instance (φ)
PCP proof (π)	$\longleftrightarrow$	Assignment to variables ($\boldsymbol{u}$)
Length of proof	$\longleftrightarrow$	Number of variables (n)
Number of queries (q)	$\longleftrightarrow$	Arity of constraints (q)
Number of random bits (r)	$\longleftrightarrow$	Logarithm of number of constraints ($\log m$)
Soundness parameter (typically $\frac{1}{2}$)	$\longleftrightarrow$	Maximum of val(φ) for a NO instance
NP $\subseteq$ PCP($\log n, 1$)	$\longleftrightarrow$	ρ-GAPqCSP is NP-hard, MAX3SAT is NP-hard to ρ-approximate.

Subsection 4

Hardness of Approximation for Vertex Cover and Independent Set

Other Hard to Approximate Problems

- The PCP Theorem implies hardness of approximation results for many more problems than just 3SAT and qCSP.
- We show a hardness of approximation result for:
  - The maximum independent set MAXINDSET problem;
  - The minimum vertex cover MINVERTEXCOVER problem.
- The inapproximability result for MAXINDSET is stronger than the result for MINVERTEXCOVER, since it rules out ρ-approximation, for every $\rho < 1$.

Theorem

- There is some $\gamma < 1$, such that computing a γ-approximation to MINVERTEXCOVER is NP-hard.
- For every $\rho < 1$, computing a ρ-approximation to MAXINDSET is NP-hard.

Exact Solutions vs. Approximate Solutions

- A vertex cover is a set of vertices touching all edges of the graph.
- So its complement is an independent set.
- Thus, the two problems are equivalent with respect to exact solution.
- The largest independent set is simply the complement of the smallest vertex cover.
- However, this does not imply that they are equivalent with respect to approximation.

Exact Solutions vs. Approximate Solutions (Cont'd)

- Let the size of the minimum vertex cover be VC.
- Let the size of the largest independent set be IS.
- We see that $\text{VC} = n - \text{IS}$.
- A ρ-approximation for IndSet would produce an independent set of size $\rho \cdot \text{IS}$.
- Using this to compute an approximation to MinVertexCover, gives a vertex cover of size $n - \rho \cdot \text{IS}$.
- This yields an approximation ratio of

$$\frac{n - \text{IS}}{n - \rho \cdot \text{IS}}$$

  for MinVertexCover.
- This could be arbitrarily small if IS is close to n.

From 3CNF Formulas to Independent Sets

- The preceding theorem shows that, unless $\mathsf{P} = \mathsf{NP}$, the approximability of the two problems is inherently different.
  - $\textsc{MinVertexCover}$ has a polynomial time $\frac{1}{2}$-approximation algorithm;
  - $\textsc{IndSet}$ does not have a ρ-approximation algorithm, for every $\rho < 1$.
- We first show, using the PCP Theorem, that there is some constant $\rho < 1$, such that both problems cannot be ρ-approximated in polynomial time (unless $\mathsf{P} = \mathsf{NP}$).
- We then show how to "amplify" the approximation gap and make ρ as small as desired in case of $\textsc{IndSet}$.

From 3CNF Formulas to Independent Sets (Cont'd)

Lemma

There exist a polynomial time computable transformation f from 3CNF formulas to graphs, such that, for every 3CNF formula φ, $f(\varphi)$ is an n-vertex graph whose largest independent set has size

$$\text{val}(\varphi)\frac{n}{7}.$$

- We apply the "normal" NP-completeness reduction for IndSet on this 3CNF formula and observe that it satisfies the desired property.

Non-Approximability of IndSet

Corollary

If P $\neq$ NP, then there are some constants $\rho < 1$ and $\rho' < 1$, such that:

- The problem IndSet cannot be ρ-approximated in polynomial time;
- The problem MinVertexCover cannot be ρ'-approximated.

- Let L be an NP language.

  The PCP Theorem implies that the decision problem for L can be reduced to approximating Max3Sat.

  Specifically, the reduction produces a 3CNF formula φ that is either satisfiable or satisfies $\text{val}(\varphi) < \rho$, where $\rho < 1$ is some constant.

  We apply the reduction of the lemma on this 3CNF formula.

  We conclude that a ρ-approximation to IndSet would allow us to do a ρ-approximation to Max3Sat on φ.

  Thus, ρ-approximation to IndSet is NP-hard.

Non-Approximability of MinVertexCover

- We now obtain the result for MinVertexCover.

  We observe that the minimum vertex cover in the graph resulting from the reduction of the previous paragraph has size

  $$n - \text{val}(\varphi)\frac{n}{7}.$$

  Suppose MinVertexCover had a ρ'-approximation for $\rho' = \frac{6}{7-\rho}$.

  Then, in case $\text{val}(\varphi) = 1$, we can find a vertex cover of size

  $$\frac{1}{\rho'}\left(n - \frac{n}{7}\right).$$

  This size is at most $n - \rho\frac{n}{7}$.

  Thus, such an approximation would allow us to distinguish the cases $\text{val}(\varphi) = 1$ and $\text{val}(\varphi) < \rho$.

  The latter task is NP-hard.

Proof of Universal Hardness of IndSet

- Finally, we need to amplify this approximation gap for IndSet.

  This is possible thanks to a "self-improvement" property.

  For self-improvement here, one uses a graph product.

  For any n-vertex graph G, define G^k to be a graph on $\binom{n}{k}$ vertices whose vertices correspond to all subsets of vertices of G of size k.

  Two subsets S_1, S_2 are adjacent if $S_1 \cup S_2$ is an independent set in G.

  The largest independent subset of G^k corresponds to all k-size subsets of the largest independent set in G.

Proof of Universal Hardness of IndSet (Cont'd)

- Let IS be the size of the largest independent set in G.

  Thus, he largest independent subset of G^k has size $\binom{\text{IS}}{k}$.

  Take the graph produced by the reduction of the preceding corollary.

  Take its k-wise product.

  Then the ratio of the size of the largest independent set in the two cases is

$$\frac{\binom{\text{IS}}{k}}{\binom{\rho\cdot\text{IS}}{k}}.$$

  This is approximately a factor ρ^k.

  Choosing k large enough, ρ^k can be made smaller than any desired constant.

  The running time is n^k, a polynomial for any fixed k.

Levin Reductions

- We defined L' to be NP-hard if every $L \in \text{NP}$ reduces to L'.
- The reduction was a polynomial-time function f, such that

$$x \in L \quad \text{iff} \quad f(x) \in L'.$$

- In all cases, we proved that $x \in L$ implies $f(x) \in L'$ by showing a way to map a certificate for the fact that $x \in L$ to a certificate for the fact that $f(x) \in L'$.
- The definition of a Karp reduction does not require that this mapping between certificates be efficient.
- However, this was often the case.
- Similarly, we proved that $f(x) \in L'$ implies $x \in L$ by showing a way to map a certificate for the fact that $x' \in L'$ to a certificate for the fact that $x \in L$.
- Again the proofs typically yield an efficient way to compute this mapping.
- We call reductions with these properties **Levin reductions**.

PCP Reductions and Levin Reductions

- The PCP reductions of this chapter also satisfy this Levin reduction property.
- We mapped a CNF formula φ into a graph G such that φ is satisfiable iff G has a “large” independent set.
- Additionally, we efficiently mapped:
  - A satisfying assignment for φ into a large independent set in G;
  - A not-too-small independent set in G into a satisfying assignment for φ.

Subsection 5

NP $\subseteq$ PCP(poly(n), 1): PCP from the Walsh-Hadamard Code

Exponential-Sized PCP System for NP

- We now prove a weaker version of the PCP Theorem.
- We show that every NP statement has an exponentially long proof that can be locally tested by only looking at a constant number of bits.
- In addition to a taste of how one proves PCP theorems, the techniques are the same used in the proof of the full-fledged PCP.

Theorem (Exponential-Sized PCP System for NP)

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

- We design an appropriate verifier for an NP-complete language.
- The verifier expects the proof to contain an encoded version of the usual certificate.
- It checks such an encoded certificate by simple probabilistic tests.

The Walsh-Hadamard Code

- We use the **Walsh-Hadamard code**, which is a way to encode bit strings of length n by linear functions in n variables over GF(2).
- For $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, we define

$$\boldsymbol{x} \odot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i \pmod 2.$$

- The Walsh-Hadamard encoding function

$$\mathrm{WH} : \{0,1\}^* \to \{0,1\}^*$$

maps a string $\boldsymbol{u} \in \{0,1\}^n$ to the truth table of the function

$$\boldsymbol{x} \mapsto \boldsymbol{u} \odot \boldsymbol{x}.$$

The Walsh-Hadamard Code (Cont'd)

- An n-bit string $\boldsymbol{u} \in \{0,1\}^n$ is encoded using $|\text{WH}(\boldsymbol{u})| = 2^n$ bits.
- A function $f \in \{0,1\}^{2^n}$ is a **Walsh-Hadamard codeword** if it is equal to $\text{WH}(\boldsymbol{u})$ for some $\boldsymbol{u}$
- Such a string

$$f \in \{0,1\}^{2^n}$$

can also be viewed as a function from $\{0,1\}^n$ to $\{0,1\}$.

The Random Subsum Principle

Random Subsum Principle

If $\boldsymbol{u} \neq \boldsymbol{v}$ then, for $\frac{1}{2}$ the choices of $\boldsymbol{x}$,

$$\boldsymbol{u} \odot \boldsymbol{x} \neq \boldsymbol{v} \odot \boldsymbol{x}.$$

- The random subsum principle implies that the Walsh-Hadamard code is an error correcting code with minimum distance $\frac{1}{2}$.
- That is, for every $\boldsymbol{u} \neq \boldsymbol{v} \in \{0,1\}^n$, the encodings WH($\boldsymbol{u}$) and WH($\boldsymbol{v}$) differ in at least half the bits.

Testing for Codewords

- Given access to a function $f : \{0,1\}^n \to \{0,1\}$, we would like to test whether or not f is actually a codeword of Walsh-Hadamard.
- The Walsh-Hadamard codewords are precisely the set of all linear functions from $\{0,1\}^n$ to $\{0,1\}$.
- So we can test f by checking that, for all 2^{2n} pairs $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$,

$$f(\boldsymbol{x}+\boldsymbol{y}) = f(\boldsymbol{x}) + f(\boldsymbol{y}).$$

- This test works by definition, but it involves reading all 2^n values of f.

Local Testing for Codewords

- To test f by reading only a constant number of its values, we:
  - Choose $\boldsymbol{x}, \boldsymbol{y}$ at random;
  - Verify the equation.
- Even such a local test accepts a linear function with probability 1.
- On the other hand, there is no longer a guarantee that every function that is not linear is rejected with high probability!
- E.g., suppose f is very close to being a linear function.

  I.e., f is obtained by modifying a linear function on a very small fraction of its inputs.

  Then such a local test will encounter the nonlinear part with very low probability.

  Thus, it will not be able to distinguish f from a linear function.

Revising the Goal

- We revise the goal to design a test that:
  - Accepts every linear function;
  - Rejects with high probability every function that is far from linear.

Definition

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

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

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

Linearity Testing

Theorem (Linearity Testing)

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

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

Then f is ρ-close to a linear function.

- For every $\rho \in (0, \frac{1}{2})$, we can obtain a linearity test that rejects with probability at least $\frac{1}{2}$ every function that is not $(1-\delta)$-close to a linear function, by testing the linearity condition repeatedly $O\left(\frac{1}{\delta}\right)$ times with independent randomness.
- We call such a test a $(1-\delta)$-**linearity test**.

Local Decoding of Walsh-Hadamard Code

- Suppose that, for $\delta < \frac{1}{4}$, the function $f : \{0,1\}^n \to \{0,1\}$ is $(1-\delta)$-close to some linear function $\widetilde{f}$.
- Every two linear functions differ on half of their inputs.
- So the function $\widetilde{f}$ is uniquely determined by f.
- Suppose we are given $\boldsymbol{x} \in \{0,1\}^n$ and random access to f.
- Can we obtain $\widetilde{f}(\boldsymbol{x})$ using only a constant number of queries?
- Naively, since most $\boldsymbol{x}$'s satisfy $f(\boldsymbol{x}) = \widetilde{f}(\boldsymbol{x})$, we should be able to learn $\widetilde{f}(\boldsymbol{x})$ with good probability by making only the single query $\boldsymbol{x}$ to f.
- However, $\boldsymbol{x}$ could be one of the places where f and $\widetilde{f}$ differ.
- Fortunately, there is still a simple way to learn $\widetilde{f}(\boldsymbol{x})$ while making only two queries to f.

The Local Decoding Procedure

- Suppose that, for $\delta < \frac{1}{4}$, the function $f : \{0,1\}^n \to \{0,1\}$ is $(1-\delta)$-close to some linear function $\widetilde{f}$.
- We learn $\widetilde{f}(\boldsymbol{x})$ by making only two queries to f:
  1. Choose $\boldsymbol{x}' \in_R \{0,1\}^n$.
  2. Set $\boldsymbol{x}'' = \boldsymbol{x} + \boldsymbol{x}'$.
  3. Let $\boldsymbol{y}' = f(\boldsymbol{x}')$ and $\boldsymbol{y}'' = f(\boldsymbol{x}'')$.
  4. Output $\boldsymbol{y}' + \boldsymbol{y}''$.

Correctness of the Procedure

- Both $\boldsymbol{x}'$ and $\boldsymbol{x}''$ are individually uniformly distributed (even though they are dependent).

  Thus, by the union bound, with probability at least $1 - 2\delta$, we have

  $$\boldsymbol{y}' = \widetilde{f}(\boldsymbol{x}') \quad \text{and} \quad \boldsymbol{y}'' = \widetilde{f}(\boldsymbol{x}'').$$

  Yet by the linearity of $\widetilde{f}$,

  $$\widetilde{f}(\boldsymbol{x}) = \widetilde{f}(\boldsymbol{x}' + \boldsymbol{x}'') = \widetilde{f}(\boldsymbol{x}') + \widetilde{f}(\boldsymbol{x}'').$$

  Hence, with at least $1 - 2\delta$ probability,

  $$\widetilde{f}(\boldsymbol{x}) = \boldsymbol{y}' + \boldsymbol{y}''.$$

- This technique allows recovering any bit of the correct codeword ($\widetilde{f}$) from a corrupted version (f) by making a constant number of queries.
- So it is called **local decoding** of the Walsh-Hadamard code.
- It is also known as **self correction** of the Walsh-Hadamard code.

Satisfiable Quadratic Equations over GF(2)

- To show NP $\subseteq$ PCP(poly(n), 1), we construct a (poly(n), 1)-verifier proof system for a particular NP-complete language L.
- The result follows since every NP language is reducible to L.
- The NP-complete language L is QuadEq, the language of systems of quadratic equations over GF(2) $= \{0, 1\}$ that are satisfiable.

  Example: The following is an instance of QuadEq over the variables $u_1, \ldots, u_5$:

$$\begin{aligned} u_1u_2 + u_3u_4 + u_1u_5 &= 1 \\ u_2u_3 + u_1u_4 &= 0 \\ u_1u_4 + u_3u_5 + u_3u_4 &= 1 \end{aligned}$$

  This instance is satisfiable, since the all-1 assignment satisfies all the equations.

NP-Completeness of QuadEq

- To check QuadEq is NP-complete one reduces the NP-complete language CktSat of satisfiable Boolean circuits to QuadEq.
- The idea is to:
  - Have a variable represent the value of each wire in the circuit (including the input wires);
  - Express AND and OR using the equivalent quadratic polynomial. E.g.,

    $$x \vee y = 1 \quad \text{iff} \quad (1-x)(1-y) = 0,$$

    and so on.

Matrix Formulation of QuadEq

- In GF(2), we have $u_i = (u_i)^2$.
- So we can assume that the equations do not contain terms of the form u_i (i.e., all terms are of degree exactly two).
- Hence, m quadratic equations over the variables $u_1, \ldots, u_n$ can be described by:
  - An $m \times n^2$ matrix A over GF(2);
  - An m-dimensional vector $\boldsymbol{b}$ over GF(2).
- The problem QuadEq can be phrased as the task, given such $A, \boldsymbol{b}$, of finding an n^2-dimensional vector U satisfying:
  (1) $AU = \boldsymbol{b}$;
  (2) U is the tensor product $\boldsymbol{u} \otimes \boldsymbol{u}$ of some n-dimensional vector $\boldsymbol{u}$.
- If $\boldsymbol{x}, \boldsymbol{y}$ are two n-dimensional vectors, then their **tensor product** $\boldsymbol{x} \otimes \boldsymbol{y}$ is the n^2-dimensional vector (or $n \times n$ matrix) whose (i, j)-th entry is $x_i y_j$ (identifying $[n^2]$ with $[n] \times [n]$ in some canonical way).

The PCP Verifier

- We now describe the PCP system for QUADEQ.
- Let $A, \boldsymbol{b}$ be an instance of QUADEQ.
- Suppose that $A, \boldsymbol{b}$ is satisfiable by an assignment $\boldsymbol{u} \in \{0,1\}^n$.
- The verifier V gets access to a proof $\pi \in \{0,1\}^{2^n + 2^{n^2}}$.
- π is interpreted as a pair of functions

$$f : \{0,1\}^n \to \{0,1\} \quad \text{and} \quad g : \{0,1\}^{n^2} \to \{0,1\}.$$

- In the correct PCP proof π for $A, \boldsymbol{b}$:
  - The function f will be the Walsh-Hadamard encoding for $\boldsymbol{u}$;
  - The function g will be the Walsh-Hadamard encoding for $\boldsymbol{u} \otimes \boldsymbol{u}$.
- Verifier V will be designed to accepts proofs of this form with probability one.
- The analysis repeatedly uses the Random Subsum Principle.

Step 1: Check that f, g are Linear Functions

- The verifier performs a 0.999-linearity test on both f, g.
- If either of f, g is not 0.999-close to a linear function, then V rejects with high probability.
- We, thus, assume there exist two linear functions

$$\widetilde{f} : \{0,1\}^n \to \{0,1\} \quad \text{and} \quad \widetilde{g} : \{0,1\}^{n^2} \to \{0,1\},$$

  such that:
  - $\widetilde{f}$ is 0.999-close to f;
  - $\widetilde{g}$ is 0.999-close to g.
- For Steps 2 and 3, the verifier can query $\widetilde{f}$, $\widetilde{g}$ at any desired point.
  - Local decoding allows the verifier to recover any desired value of $\widetilde{f}, \widetilde{g}$ with good probability.
  - Steps 2 and 3 will only use a small (< 20) number of queries to $\widetilde{f}, \widetilde{g}$.
  - Thus, with high probability (say > 0.9) local decoding will succeed on all these queries.

Notation

- To simplify notation, in the rest of the procedure we use f, g for $\widetilde{f}, \widetilde{g}$, respectively.
- In particular, we assume both f and g are linear.
- Thus, they must encode some strings

$$\boldsymbol{u} \in \{0,1\}^n \quad \text{and} \quad \boldsymbol{w} \in \{0,1\}^{n^2}.$$

- In other words, f, g are the functions given by

$$f(\boldsymbol{r}) = \boldsymbol{u} \odot \boldsymbol{r} \quad \text{and} \quad g(\boldsymbol{z}) = \boldsymbol{w} \odot \boldsymbol{z}.$$

Step 2: Verify that g encodes $\boldsymbol{u} \otimes \boldsymbol{u}$

- We verify that g encodes $\boldsymbol{u} \otimes \boldsymbol{u}$, where $\boldsymbol{u} \in \{0,1\}^n$ is the string encoded by f.
- The verifier performs the following test ten times using independent random bits.
  - Choose $\boldsymbol{r}, \boldsymbol{r}'$ independently at random from $\{0,1\}^n$.
  - If $f(\boldsymbol{r})f(\boldsymbol{r}') \neq g(\boldsymbol{r} \otimes \boldsymbol{r}')$ then halt and reject.
- In a correct proof, we have $\boldsymbol{w} = \boldsymbol{u} \otimes \boldsymbol{u}$.
- So

$$\begin{array}{rcl} f(\boldsymbol{r})f(\boldsymbol{r}') & = & \left(\sum_{i\in[n]} u_i r_i\right)\left(\sum_{j\in[n]} u_j r'_j\right) \\ & = & \sum_{i,j\in[n]} u_i u_j r_i r'_j \\ & = & (\boldsymbol{u} \otimes \boldsymbol{u})(\boldsymbol{r} \otimes \boldsymbol{r}'). \end{array}$$

- In the correct proof this is equal to $g(\boldsymbol{r} \otimes \boldsymbol{r}')$.
- Thus, Step 2 never rejects a correct proof.

The Case of an Incorrect Proof, $\mathbf{w} \neq \mathbf{u} \otimes \mathbf{u}$

- Suppose, now, that $\mathbf{w} \neq \mathbf{u} \otimes \mathbf{u}$.
- We claim that in each of the ten trials V will halt and reject with probability at least $\frac{1}{4}$.
- Thus, the probability of rejecting in at least one trial is

$$\geq 1 - \left(\frac{3}{4}\right)^{10} > 0.9.$$

- Let W be an $n \times n$ matrix with the same entries as $\mathbf{w}$.
- Let U be the $n \times n$ matrix such that $U_{i,j} = u_i u_j$.
- Think of $\mathbf{r}$ as a row vector and $\mathbf{r}'$ as a column vector.
- In this notation,

$$\begin{aligned} g(\mathbf{r} \otimes \mathbf{r}') &= \mathbf{w} \odot (\mathbf{r} \otimes \mathbf{r}') \\ &= \textstyle\sum_{i,j \in [n]} w_{i,j} r_i r'_j \\ &= \mathbf{r} W \mathbf{r}'. \end{aligned}$$

The Case of an Incorrect Proof (Cont'd)

- Moreover,

$$\begin{aligned} f(\boldsymbol{r})f(\boldsymbol{r}') &= (\boldsymbol{u} \odot \boldsymbol{r})(\boldsymbol{u} \odot \boldsymbol{r}') \\ &= (\sum_{i=1}^n u_i r_i)(\sum_{j=1}^n u_j r_j') \\ &= \sum_{i,j \in [n]} u_i u_j r_i r_j' \\ &= \boldsymbol{r} U \boldsymbol{r}'. \end{aligned}$$

- V rejects if $\boldsymbol{r} W \boldsymbol{r}' \neq \boldsymbol{r} U \boldsymbol{r}'$.
- The Random Subsum Principle implies that, if $W \neq U$, then at least $\frac{1}{2}$ of all $\boldsymbol{r}$ satisfy $\boldsymbol{r} W \neq \boldsymbol{r} U$.
- Applying the Random Subsum Principle for each such $\boldsymbol{r}$, we conclude that at least $\frac{1}{2}$ of all $\boldsymbol{r}'$ satisfy

$$\boldsymbol{r} W \boldsymbol{r}' \neq \boldsymbol{r} U \boldsymbol{r}'.$$

- Hence, the trial rejects for at least $\frac{1}{4}$ of all pairs $\boldsymbol{r}, \boldsymbol{r}'$.

Step 3: Verify that g Encodes a Satisfying Assignment

- We use all that has been verified about f, g in the preceding steps.
- It is easy to check that any particular equation, say the kth equation of the input, is satisfied by $\boldsymbol{u}$.
- That is,

$$\sum_{i,j} A_{k,(i,j)} u_i u_j = b_k.$$

- If $\boldsymbol{z}$ is the n^2 dimensional vector

$$(A_{k,(i,j)}), \quad i,j = 1, \ldots, n,$$

  the left-hand side is $g(\boldsymbol{z})$.
- The verifier knows $A_{k,(i,j)}$ and b_k.
- So it simply queries g at $\boldsymbol{z}$ and checks that $g(\boldsymbol{z}) = b_k$.

Step 3: A Hurdle and a Solution

- The drawback is that, in order to check that $\boldsymbol{u}$ satisfies the entire system, the verifier needs to make a query to g, for each $k = 1, 2, \ldots, m$.
- However, the number of queries is required to be independent of m.
- So the verifier relies again on the Random Subsum Principle.
- The verifier takes a random subset of the equations and computes their sum mod 2.
- This sum is a new quadratic equation.
- The Random Subsum Principle implies that, if $\boldsymbol{u}$ does not satisfy even one equation in the original system, then, with probability at least $\frac{1}{2}$, it will not satisfy this new equation.
- The verifier checks that $\boldsymbol{u}$ satisfies this new equation.

Concluding the Proof of NP $\subseteq$ PCP(poly(n), 1)

- Overall, we get a verifier V such that:
  (1) If $A, \boldsymbol{b}$ is satisfiable, then V accepts the correct proof with probability 1;
  (2) If $A, \boldsymbol{b}$ is not satisfiable then V accepts every proof with probability at most 0.8.
- The probability of accepting a proof for a false statement can be reduced to $\frac{1}{2}$ by simple repetition.