PCP's, Hardness of Approximation and the Unique Games Conjecture

Haris Angelidakis

MPLA

May 3 & 10, 2012

Outline

Outline

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

The PCP Idea

The main idea behind PCP's is to check a proof faster than usual. How is this done?

- We first rewrite the proof in a certain format, the **PCP format**.
- We then check randomly a constant number of its bits:
  - A correct proof always convinces us.
  - A false proof will convince us with probability $\leq 1/2$.

Detail: The rewriting is completely mechanical and does not greatly increase its size. **But**, it requires proofs to be written in a formal axiomatic system (such as ZF Set Theory).

Towards a new definition of *NP*

Note: From now on, we shall refer to languages $L \subseteq \{0,1\}^*$.

Definition (*NP* "yes"-certificate definition)

A language L is in *NP* if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a deterministic polynomial-time TM M (called the **verifier** of L) such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a **certificate** for x (with respect to the language L and machine M).

Informally, *NP* is the complexity class of problems for which it is easy to check that a solution is correct.

Towards a new definition of *NP*

Note: From now on, we shall refer to languages $L \subseteq \{0,1\}^*$.

Definition (*NP* "yes"-certificate definition)

A language L is in *NP* if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a deterministic polynomial-time TM M (called the **verifier** of L) such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a **certificate** for x (with respect to the language L and machine M).

Informally, *NP* is the complexity class of problems for which it is easy to check that a solution is correct.

Towards a new definition of *NP*

Note: From now on, we shall refer to languages $L \subseteq \{0,1\}^*$.

Definition (*NP* "yes"-certificate definition)

A language L is in *NP* if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a deterministic polynomial-time TM M (called the **verifier** of L) such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a **certificate** for x (with respect to the language L and machine M).

Informally, *NP* is the complexity class of problems for which it is easy to check that a solution is correct.

Towards a new definition of *NP*

Some comments

- What is a mathematical proof? Anything that can be verified by a rigorous procedure, i.e., an algorithm.
- A theorem = a problem.
- A proof = a solution.

Towards a new definition of *NP*

Some comments

- What is a mathematical proof? Anything that can be verified by a rigorous procedure, i.e., an algorithm.
- A theorem = a problem.
- A proof = a solution.

Towards a new definition of *NP*

Some comments

- What is a mathematical proof? Anything that can be verified by a rigorous procedure, i.e., an algorithm.
- A theorem = a problem.
- A proof = a solution.

Towards a new definition of *NP*

Some comments

- What is a mathematical proof? Anything that can be verified by a rigorous procedure, i.e., an algorithm.
- A theorem = a problem.
- A proof = a solution.

Towards a new definition of *NP*

Definition (*NP* alternative definition)

An alternative way to define *NP* is as the class of all languages $L \subseteq \{0,1\}^*$ that have **efficient** proof systems: proof systems in which there is a polynomial-time algorithm that verifies correctness of the statement $x \in L$ with assistance of a proof.

- One problem with the usual proof systems (i.e. the "yes"-certificates for *NP*) is that these proofs are very sensitive to error. A false theorem can be "proven" by a proof that consists of only one erroneous step. Similarly, a 3-SAT formula ϕ can be unsatisfiable, yet have an assignment that satisfies all clauses but one. In these cases, the verifier must check every single proof step / clause in order to make sure that the proof is correct.

Towards a new definition of *NP*

Definition (*NP* alternative definition)

An alternative way to define *NP* is as the class of all languages $L \subseteq \{0,1\}^*$ that have **efficient** proof systems: proof systems in which there is a polynomial-time algorithm that verifies correctness of the statement $x \in L$ with assistance of a proof.

- One problem with the usual proof systems (i.e. the "yes"-certificates for *NP*) is that these proofs are very sensitive to error. A false theorem can be "proven" by a proof that consists of only one erroneous step. Similarly, a 3-SAT formula ϕ can be unsatisfiable, yet have an assignment that satisfies all clauses but one. In these cases, the verifier must check every single proof step / clause in order to make sure that the proof is correct.

Towards a new definition of *NP*

Definition (*NP* alternative definition)

An alternative way to define *NP* is as the class of all languages $L \subseteq \{0,1\}^*$ that have **efficient** proof systems: proof systems in which there is a polynomial-time algorithm that verifies correctness of the statement $x \in L$ with assistance of a proof.

- One problem with the usual proof systems (i.e. the "yes"-certificates for *NP*) is that these proofs are very sensitive to error. A false theorem can be "proven" by a proof that consists of only one erroneous step. Similarly, a 3-SAT formula ϕ can be unsatisfiable, yet have an assignment that satisfies all clauses but one. In these cases, the verifier must check every single proof step / clause in order to make sure that the proof is correct.

Towards a new definition of *NP*

- In contrast, the **PCP theorem** gives each set in *NP* an alternative proof system, in which proofs are robust.
- In this system a proof for a false statement is guaranteed to have many errors.
- As a result, a verifier can randomly read only a few bits from the proof and decide, with *high probability* of success, whether the proof is valid or not.

Towards a new definition of *NP*

- In contrast, the **PCP theorem** gives each set in *NP* an alternative proof system, in which proofs are robust.
- In this system a proof for a false statement is guaranteed to have many errors.
- As a result, a verifier can randomly read only a few bits from the proof and decide, with *high probability* of success, whether the proof is valid or not.

Towards a new definition of *NP*

- In contrast, the **PCP theorem** gives each set in *NP* an alternative proof system, in which proofs are robust.
- In this system a proof for a false statement is guaranteed to have many errors.
- As a result, a verifier can randomly read only a few bits from the proof and decide, with *high probability* of success, whether the proof is valid or not.

Towards a new definition of *NP*

Definition (*NP* revisited - The *NP* verifier)

$L \in NP$ iff there exists a poly-time TM V (the verifier) such that:

$$x \in L \Rightarrow \exists \pi \text{ such that } V^{\pi}(x) = 1,$$
$$x \notin L \Rightarrow \forall \pi,\ V^{\pi}(x) = 0.$$

(π is a proof)

Towards a new definition of *NP*

Definition (The PCP verifier)

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

- **Efficiency**: On input $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (the *proof*), V uses at most $r(n)$ random coins and 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.
- **Completeness**: $x \in L \Rightarrow \exists \pi \in \{0,1\}^*$ such that $Pr[V^\pi(x) = 1] = 1$. (We call this string π the correct proof for x.)
- **Soundness**: $x \notin L \Rightarrow \forall \pi \in \{0,1\}^*, Pr[V^\pi(x) = 1] \leq 1/2$.

We say that a language L is in $PCP[r(n), q(n)]$ if there are some constants $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP verifier.

Towards a new definition of *NP*

Definition (The PCP verifier)

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

- **Efficiency**: On input $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (the *proof*), V uses at most $r(n)$ random coins and 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.
- **Completeness**: $x \in L \Rightarrow \exists \pi \in \{0,1\}^*$ such that $Pr[V^\pi(x) = 1] = 1$. (We call this string π the correct proof for x.)
- **Soundness**: $x \notin L \Rightarrow \forall \pi \in \{0,1\}^*, Pr[V^\pi(x) = 1] \leq 1/2$.

We say that a language L is in $PCP[r(n), q(n)]$ if there are some constants $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP verifier.

Towards a new definition of *NP*

Definition (The PCP verifier)

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

- **Efficiency**: On input $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (the *proof*), V uses at most $r(n)$ random coins and 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.
- **Completeness**: $x \in L \Rightarrow \exists \pi \in \{0,1\}^* \text{ such that } Pr[V^\pi(x) = 1] = 1$. (We call this string π the correct proof for x.)
- **Soundness**: $x \notin L \Rightarrow \forall \pi \in \{0,1\}^*, Pr[V^\pi(x) = 1] \leq 1/2$.

We say that a language L is in $PCP[r(n), q(n)]$ if there are some constants $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP verifier.

Towards a new definition of *NP*

Definition (The PCP verifier)

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

- **Efficiency**: On input $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (the *proof*), V uses at most $r(n)$ random coins and 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.
- **Completeness**: $x \in L \Rightarrow \exists \pi \in \{0,1\}^* \text{ such that } Pr[V^\pi(x) = 1] = 1$. (We call this string π the correct proof for x.)
- **Soundness**: $x \notin L \Rightarrow \forall \pi \in \{0,1\}^*, Pr[V^\pi(x) = 1] \leq 1/2$.

We say that a language L is in $PCP[r(n), q(n)]$ if there are some constants $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP verifier.

Towards a new definition of *NP*

Definition (The PCP verifier)

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

- **Efficiency**: On input $x \in \{0,1\}^n$ and given random access to a string $\pi \in \{0,1\}^*$ of length at most $q(n)2^{r(n)}$ (the *proof*), V uses at most $r(n)$ random coins and 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.
- **Completeness**: $x \in L \Rightarrow \exists \pi \in \{0,1\}^* \text{ such that } Pr[V^\pi(x) = 1] = 1$. (We call this string π the correct proof for x.)
- **Soundness**: $x \notin L \Rightarrow \forall \pi \in \{0,1\}^*, Pr[V^\pi(x) = 1] \leq 1/2$.

We say that a language L is in $PCP[r(n), q(n)]$ if there are some constants $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP verifier.

The PCP Theorem

Theorem (PCP Theorem - Arora, Lund, Motwani, Sudan, Szegedy, Safra)

$NP = PCP[O(\log n), O(1)]$.

Gap-introducing reductions and *NP*-completeness

Dinur's proof of the PCP theorem is based on finding **gap-introducing** reductions, i.e. reductions of the following form:

(we want to reduce $L \in NP$, to a 3CNF formula ϕ_x with m clauses and with the following properties)

$$x \in L \Rightarrow \phi_x \text{ is satisfiable}$$
$$x \notin L \Rightarrow \text{ no assignment satisfies more than } (1-\epsilon_1)m \text{ clauses of } \phi_x.$$

Theorem

If there is a gap-introducing reduction for some problem L in NP, then $L \in PCP[O(\log n), O(1)]$. *In particular, if L is NP-complete then the PCP theorem holds.*

Outline

How can we get inapproximability results

- In general, standard *NP*-hardness proofs are not powerful enough to give inapproximability results.
- In order to get such a result, we will need stronger reductions, the gap-introducing reductions we have already mentioned.

How can we get inapproximability results

- In general, standard *NP*-hardness proofs are not powerful enough to give inapproximability results.
- In order to get such a result, we will need stronger reductions, the gap-introducing reductions we have already mentioned.

Approximability of Max-3SAT

Theorem

The PCP theorem implies that there is an $\epsilon_1 > 0$ such that there is no polynomial $(1 - \epsilon_1)$-approximation algorithm for Max-3SAT, unless $P = NP$.

Proof.

On board... □

Note: This directly implies that there is no PTAS for Max-3SAT.

Approximability of Max-3SAT

Theorem

The PCP theorem implies that there is an $\epsilon_1 > 0$ such that there is no polynomial $(1 - \epsilon_1)$-approximation algorithm for Max-3SAT, unless $P = NP$.

Proof.

On board... □

Note: This directly implies that there is no PTAS for Max-3SAT.

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (1 / 3)

Theorem (Håstad)

For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i, j, k in the proof w and a bit b, and it accepts iff $w_i \oplus w_j \oplus w_k = b$.

Consequences:

- Reduction of an NP-complete problem, say SAT, to Max-E3LIN-2:
  - Take the $PCP_{1-\epsilon,\frac{1}{2}+\epsilon}[O(\log n), 3]$ for SAT.
  - If a SAT formula ϕ is satisfiable, then there is a proof w such that at least $1 - \epsilon$ fraction of the system of 3ELIN-2 equations $w_i \oplus w_j \oplus w_k = b$ are satisfied by the corresponding bits.
  - If ϕ is not satisfiable, then for every proof w at most $\frac{1}{2} + \epsilon$ fraction of equations can be satisfied.
  - Thus, we cannot approximate Max-E3LIN-2 within a factor better than 2 unless $P = NP$. (can you find an algorithm that achieves this factor?)

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (2 / 3)

- Continuing the reduction: Max-E3LIN-2 to Max-E3SAT:
  - Every equation $w_i \oplus w_j \oplus w_k = b$ can be expressed as the conjunction of 4 clauses, so that the equation is satisfied iff all 4 clauses can be satisfied. (how?)
  - Thus, from a system I of m equations we can construct a formula ϕ_I of $4m$ clauses.
  - If we can satisfy $\geq m(1-\epsilon)$ equations, then the same assignment can satisfy $\geq 4m(1-\epsilon)$ clauses.
  - Conversely, if it is impossible to satisfy more than $m(\frac{1}{2}+\epsilon)$ equations, then it is impossible to satisfy more than $4m - m(\frac{1}{2}+\epsilon) = \frac{7}{2}m - m\epsilon$ (at least one clause for each unsatisfied equation)
  - Thus, Max-E3SAT, and so Max-3SAT, cannot be approximated within a factor better than $7/8$, unless $P = NP$.
  - Actually, the trivial randomized algorithm achieves the $7/8$ factor for Max-E3SAT. Observe that it can be easily derandomized with the method of conditional expectation.

Optimized PCP constructions (3 / 3)

- Finally, with a similar reduction from Max-E3LIN-2 to Max-CUT, we can prove that Max-Cut has an approximability bound of $16/17 \approxeq 0.94117$. (the currently best result is the $\approxeq 0.878$ SDP algorithm due to Goemans and Williamson)

Improvement on Håstad's Theorem

Theorem (Guruswami, Lewin, Sudan, Trevisan 98)

$NP = PCP_{1,\frac{1}{2}+\epsilon}[O(\log n), 3],\ \forall \epsilon > 0$

Proof of Optimality of the above result

Theorem (Karloff, Zwick 97)

$P = PCP_{1,\frac{1}{2}}[O(\log n), 3]$

Improvement on Håstad's Theorem

Theorem (Guruswami, Lewin, Sudan, Trevisan 98)

$NP = PCP_{1,\frac{1}{2}+\epsilon}[O(\log n), 3],\ \forall \epsilon > 0$

Proof of Optimality of the above result

Theorem (Karloff, Zwick 97)

$P = PCP_{1,\frac{1}{2}}[O(\log n), 3]$

Outline

A Threshold of $\ln n$ for Approximating Set Cover

We will now try to sketch the proof of the following theorem:

Theorem

If there is some $\epsilon > 0$ such that a polynomial time algorithm can approximate set cover within $(1 - \epsilon) \ln n$, then $NP \subset TIME(n^{O(\log \log n)})$.

Note: The above result is due to Feige (1996). Under the weaker assumption that $P \neq NP$, Raz and Safra (1997) established a lower bound of $c \cdot \ln n$ where c is a constant, and Alon, Moshkovitz and Safra (2006) improved the result with a higher constant.

A Threshold of $\ln n$ for Approximating Set Cover

We will now try to sketch the proof of the following theorem:

Theorem

If there is some $\epsilon > 0$ such that a polynomial time algorithm can approximate set cover within $(1 - \epsilon) \ln n$, then $NP \subset TIME(n^{O(\log \log n)})$.

Note: The above result is due to Feige (1996). Under the weaker assumption that $P \neq NP$, Raz and Safra (1997) established a lower bound of $c \cdot \ln n$ where c is a constant, and Alon, Moshkovitz and Safra (2006) improved the result with a higher constant.

Main idea of the proof (1 / 2)

- The proof is based on a reduction from a multi-prover proof system.
- We start with the Max-3SAT-B problem, for which we know the following:

Theorem

It is MAX-SNP hard to approximate MAX-3SAT-B. For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-B formulas and 3CNF-B formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

- We will actually use Max-E3SAT-5, in which we have a CNF formula with n variables and $5n/3$ clauses, in which every clause contains **exactly** 3 literals and every variable appears in **exactly** 5 clauses.

Theorem

For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-5 formulas and 3CNF-5 formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

Main idea of the proof (1 / 2)

- The proof is based on a reduction from a multi-prover proof system.
- We start with the Max-3SAT-B problem, for which we know the following:

Theorem

It is MAX-SNP hard to approximate MAX-3SAT-B. For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-B formulas and 3CNF-B formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

- We will actually use Max-E3SAT-5, in which we have a CNF formula with n variables and $5n/3$ clauses, in which every clause contains **exactly** 3 literals and every variable appears in **exactly** 5 clauses.

Theorem

For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-5 formulas and 3CNF-5 formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

Main idea of the proof (1 / 2)

- The proof is based on a reduction from a multi-prover proof system.
- We start with the Max-3SAT-B problem, for which we know the following:

Theorem

It is MAX-SNP hard to approximate MAX-3SAT-B. For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-B formulas and 3CNF-B formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

- We will actually use Max-E3SAT-5, in which we have a CNF formula with n variables and $5n/3$ clauses, in which every clause contains **exactly** 3 literals and every variable appears in **exactly** 5 clauses.

Theorem

For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-5 formulas and 3CNF-5 formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

Main idea of the proof (1 / 2)

- The proof is based on a reduction from a multi-prover proof system.
- We start with the Max-3SAT-B problem, for which we know the following:

Theorem

It is MAX-SNP hard to approximate MAX-3SAT-B. For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-B formulas and 3CNF-B formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

- We will actually use Max-E3SAT-5, in which we have a CNF formula with n variables and $5n/3$ clauses, in which every clause contains **exactly** 3 literals and every variable appears in **exactly** 5 clauses.

Theorem

For some $\epsilon > 0$, it is NP-hard to distinguish between satisfiable 3CNF-5 formulas and 3CNF-5 formulas in which at most an $(1 - \epsilon)$-fraction of the clauses can be simultaneously satisfied.

Main idea of the proof (2 / 2)

- By using a k-prover proof system for Max-3SAT-5, we reduce, by using some "partition systems" the problem to an instance of the set cover with universe size m such that:
  - If ϕ is satisfiable, then the instance created can be covered by kQ subsets for some Q.
  - If only a $(1-\epsilon)$ fraction of the clauses of ϕ can be simultaneously satisfied, then the created instance requires $(1-2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \to 0$ as $k \to \infty$.
- Thus, we cannot approximate set cover within a factor of $(1-o(1)) \ln n$, unless $NP \subset TIME(n^{O(\log \log n)})$.

Main idea of the proof (2 / 2)

- By using a k-prover proof system for Max-3SAT-5, we reduce, by using some "partition systems" the problem to an instance of the set cover with universe size m such that:
  - If ϕ is satisfiable, then the instance created can be covered by kQ subsets for some Q.
  - If only a $(1-\epsilon)$ fraction of the clauses of ϕ can be simultaneously satisfied, then the created instance requires $(1-2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \to 0$ as $k \to \infty$.
- Thus, we cannot approximate set cover within a factor of $(1-o(1)) \ln n$, unless $NP \subset TIME(n^{O(\log \log n)})$.

Main idea of the proof (2 / 2)

- By using a k-prover proof system for Max-3SAT-5, we reduce, by using some "partition systems" the problem to an instance of the set cover with universe size m such that:
  - If ϕ is satisfiable, then the instance created can be covered by kQ subsets for some Q.
  - If only a $(1-\epsilon)$ fraction of the clauses of ϕ can be simultaneously satisfied, then the created instance requires $(1-2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \to 0$ as $k \to \infty$.
- Thus, we cannot approximate set cover within a factor of $(1-o(1)) \ln n$, unless $NP \subset TIME(n^{O(\log \log n)})$.

Main idea of the proof (2 / 2)

- By using a k-prover proof system for Max-3SAT-5, we reduce, by using some "partition systems" the problem to an instance of the set cover with universe size m such that:
  - If ϕ is satisfiable, then the instance created can be covered by kQ subsets for some Q.
  - If only a $(1-\epsilon)$ fraction of the clauses of ϕ can be simultaneously satisfied, then the created instance requires $(1-2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \to 0$ as $k \to \infty$.
- Thus, we cannot approximate set cover within a factor of $(1-o(1)) \ln n$, unless $NP \subset TIME(n^{O(\log \log n)})$.

Multiprover Interactive Protocols

Definition (k-Prover Proof Systems)

Let $P_1, P_2, ..., P_k$ be infinitely powerful machines and V be a probabilistic polynomial-time machine, all of which share the same read-only input tape. The verifier V shares tapes with each P_i, but different provers P_i and P_j have no communication between them. Formally, each P_i is a function from the input and the conversation it has seen so far to a message.

$P_1, ..., P_k$ and V form a multiprover interactive protocol for a language L if:

1. If $x \in L$ then $Pr[P_1, ..., P_k \text{ and } V \text{ on } x \text{ accept}] \geq \frac{2}{3}$.
2. If $x \notin L$ then **for all** provers $P'_1, ..., P'_k$, $Pr[P'_1, ..., P'_k \text{ and } V \text{ on } x \text{ accept}] \leq \frac{1}{3}$.

Note: A round of a multiprover interactive protocol consists of messages from the verifier to some or all of the provers followed by messages from these provers to the verifier. In general, interactive protocols can have a polynomial number of rounds.

Multiprover Interactive Protocols

Definition (k-Prover Proof Systems)

Let $P_1, P_2, ..., P_k$ be infinitely powerful machines and V be a probabilistic polynomial-time machine, all of which share the same read-only input tape. The verifier V shares tapes with each P_i, but different provers P_i and P_j have no communication between them. Formally, each P_i is a function from the input and the conversation it has seen so far to a message.

$P_1, ..., P_k$ and V form a multiprover interactive protocol for a language L if:

1. If $x \in L$ then $Pr[P_1, ..., P_k \text{ and } V \text{ on } x \text{ accept}] \geq \frac{2}{3}$.
2. If $x \notin L$ then **for all** provers $P'_1, ..., P'_k$, $Pr[P'_1, ..., P'_k \text{ and } V \text{ on } x \text{ accept}] \leq \frac{1}{3}$.

Note: A round of a multiprover interactive protocol consists of messages from the verifier to some or all of the provers followed by messages from these provers to the verifier. In general, interactive protocols can have a polynomial number of rounds.

Multiprover Interactive Protocols

Definition (k-Prover Proof Systems)

Let $P_1, P_2, ..., P_k$ be infinitely powerful machines and V be a probabilistic polynomial-time machine, all of which share the same read-only input tape. The verifier V shares tapes with each P_i, but different provers P_i and P_j have no communication between them. Formally, each P_i is a function from the input and the conversation it has seen so far to a message. **$P_1, ..., P_k$ and V form a multiprover interactive protocol for a language L** if:

1. If $x \in L$ then $Pr[P_1, ..., P_k \text{ and } V \text{ on } x \text{ accept}] \geq \frac{2}{3}$.
2. If $x \notin L$ then **for all** provers $P'_1, ..., P'_k$, $Pr[P'_1, ..., P'_k \text{ and } V \text{ on } x \text{ accept}] \leq \frac{1}{3}$.

Note: A round of a multiprover interactive protocol consists of messages from the verifier to some or all of the provers followed by messages from these provers to the verifier. In general, interactive protocols can have a polynomial number of rounds.

k-Prover Proof Systems as PCP's

- PCP witness (not in general encoded in binary, but in an alphabet with cardinality depending on input size) partitioned into k segments.
- Each segment is controlled by a prover.
- The verifier reads one character from each segment. (can be viewed as a query to each prover)
- The above description corresponds to one round multiprover proof systems.

k-Prover Proof Systems as PCP's

- PCP witness (not in general encoded in binary, but in an alphabet with cardinality depending on input size) partitioned into k segments.
- Each segment is controlled by a prover.
- The verifier reads one character from each segment. (can be viewed as a query to each prover)
- The above description corresponds to one round multiprover proof systems.

k-Prover Proof Systems as PCP's

- PCP witness (not in general encoded in binary, but in an alphabet with cardinality depending on input size) partitioned into k segments.
- Each segment is controlled by a prover.
- The verifier reads one character from each segment. (can be viewed as a query to each prover)
- The above description corresponds to one round multiprover proof systems.

k-Prover Proof Systems as PCP's

- PCP witness (not in general encoded in binary, but in an alphabet with cardinality depending on input size) partitioned into k segments.
- Each segment is controlled by a prover.
- The verifier reads one character from each segment. (can be viewed as a query to each prover)
- The above description corresponds to one round multiprover proof systems.

Reducing the probability error (1 / 4)

- It can be shown that if we have a constant error of probability, we can easily reduce it by running the protocol several times serially **only** in the case of one-prover model.
- However, if we have 2 or more provers, it isn't obvious whether a parallel repetition works or not.

Reducing the probability error (1 / 4)

- It can be shown that if we have a constant error of probability, we can easily reduce it by running the protocol several times serially **only** in the case of one-prover model.
- However, if we have 2 or more provers, it isn't obvious whether a parallel repetition works or not.

Reducing the probability error (2 / 4)

Counter-example:

(2-prover 1-round)

- V: Pick two bits a and b uniformly and independently at random.
- $V \rightarrow P_1 : a$
- $V \rightarrow P_2 : b$
- $P_1 \rightarrow V : c$
- $P_2 \rightarrow V : d$
- Accept if $(a \vee c) \neq (b \vee d)$.

It is easy to show that the best strategy for two provers causes the verifier to accept with probability $\frac{1}{2}$.

Reducing the probability error (2 / 4)

Counter-example:

(2-prover 1-round)

- V: Pick two bits a and b uniformly and independently at random.
- $V \rightarrow P_1 : a$
- $V \rightarrow P_2 : b$
- $P_1 \rightarrow V : c$
- $P_2 \rightarrow V : d$
- Accept if $(a \vee c) \neq (b \vee d)$.

It is easy to show that the best strategy for two provers causes the verifier to accept with probability $\frac{1}{2}$.

Reducing the probability error (3 / 4)

Now, the parallel version of the same protocol:

- V: Pick bits a_1, a_2 and b_1, b_2 uniformly and independently at random.
- $V \rightarrow P_1 : a_1, a_2$
- $V \rightarrow P_2 : b_1, b_2$
- $P_1 \rightarrow V : c_1, c_2$
- $P_2 \rightarrow V : d_1, d_2$
- Accept if $(a_1 \vee c_1) \neq (b \vee d)$ and $(a_2 \vee c_2) \neq (b_2 \vee d_2)$.

If the parallel runs of the protocol behaved independently we would expect that the optimum strategy for the provers causes the verifier to accept with probability $(1/2)^2 = \frac{1}{4}$. However the following strategy for the provers causes the verifier to accept with probability $\frac{3}{8}$.

- P_1: If $a_1 = a_2 = 0$ respond $c_1 = c_2 = 0$ otherwise respond $c_1 = c_2 = 1$.
- P_2: If $b_1 = b_2 = 0$ respond $d_1 = d_2 = 0$ otherwise respond $d_1 = d_2 = 1$.

Reducing the probability error (3 / 4)

Now, the parallel version of the same protocol:

- V: Pick bits a_1, a_2 and b_1, b_2 uniformly and independently at random.
- $V \rightarrow P_1 : a_1, a_2$
- $V \rightarrow P_2 : b_1, b_2$
- $P_1 \rightarrow V : c_1, c_2$
- $P_2 \rightarrow V : d_1, d_2$
- Accept if $(a_1 \vee c_1) \neq (b \vee d)$ and $(a_2 \vee c_2) \neq (b_2 \vee d_2)$.

If the parallel runs of the protocol behaved independently we would expect that the optimum strategy for the provers causes the verifier to accept with probability $(1/2)^2 = \frac{1}{4}$. However the following strategy for the provers causes the verifier to accept with probability $\frac{3}{8}$.

- P_1: If $a_1 = a_2 = 0$ respond $c_1 = c_2 = 0$ otherwise respond $c_1 = c_2 = 1$.
- P_2: If $b_1 = b_2 = 0$ respond $d_1 = d_2 = 0$ otherwise respond $d_1 = d_2 = 1$.

Reducing the probability error (3 / 4)

Now, the parallel version of the same protocol:

- V: Pick bits a_1, a_2 and b_1, b_2 uniformly and independently at random.
- $V \rightarrow P_1 : a_1, a_2$
- $V \rightarrow P_2 : b_1, b_2$
- $P_1 \rightarrow V : c_1, c_2$
- $P_2 \rightarrow V : d_1, d_2$
- Accept if $(a_1 \vee c_1) \neq (b \vee d)$ and $(a_2 \vee c_2) \neq (b_2 \vee d_2)$.

If the parallel runs of the protocol behaved independently we would expect that the optimum strategy for the provers causes the verifier to accept with probability $(1/2)^2 = \frac{1}{4}$. However the following strategy for the provers causes the verifier to accept with probability $\frac{3}{8}$.

- P_1: If $a_1 = a_2 = 0$ respond $c_1 = c_2 = 0$ otherwise respond $c_1 = c_2 = 1$.
- P_2: If $b_1 = b_2 = 0$ respond $d_1 = d_2 = 0$ otherwise respond $d_1 = d_2 = 1$.

Reducing the probability error (4 / 4)

However, it is true that parallel repetition reduces the error at an exponential rate:

Theorem (Raz Parallel Repetition Theorem)

If a **2-prover 1-round** *proof system is repeated λ times* **independently in parallel**, *then the error is $2^{-c\lambda}$, where $c > 0$ is a constant that depends only on the error of the original proof system (assuming this error was less than one) and on the length of the answers of the provers in the original proof system.*

Reducing the probability error (4 / 4)

However, it is true that parallel repetition reduces the error at an exponential rate:

Theorem (Raz Parallel Repetition Theorem)

If a **2-prover 1-round** *proof system is repeated λ times* **independently in parallel**, *then the error is $2^{-c\lambda}$, where $c > 0$ is a constant that depends only on the error of the original proof system (assuming this error was less than one) and on the length of the answers of the provers in the original proof system.*

A k-prover proof system for Max-3SAT-5 (1 / 3)

1. We use a binary code that contains k code words, each of length λ and weight (number of 1's) $\lambda/2$, and Hamming distance at least $\lambda/3$. We choose $\lambda = \Theta(\log \log n)$ and k an arbitrarily large constant. (such a code can be easily found, but it is out of the scope of this presentation)
2. The verifier V selects λ clauses, say $C_1, ..., C_\lambda$ u.i.r.
3. From each C_j V selects a single variable u.i.r. These $x_1, ..., x_\lambda$ are called the *distinguished* variables.
4. Prover P_i receives C_j for those coordinates j in its code word that have the bit 1 and x_j for all the others.
5. Each P_i replies with a string of 2λ bits. This string is interpreted as an assignment to all the variables that the prover received. ($\lambda/2$ distinguished variables plus three variables in each of the $\lambda/2$ clauses)

A k-prover proof system for Max-3SAT-5 (1 / 3)

1. We use a binary code that contains k code words, each of length λ and weight (number of 1's) $\lambda/2$, and Hamming distance at least $\lambda/3$. We choose $\lambda = \Theta(\log \log n)$ and k an arbitrarily large constant. (such a code can be easily found, but it is out of the scope of this presentation)
2. The verifier V selects λ clauses, say $C_1, ..., C_\lambda$ u.i.r.
3. From each C_j V selects a single variable u.i.r. These $x_1, ..., x_\lambda$ are called the *distinguished* variables.
4. Prover P_i receives C_j for those coordinates j in its code word that have the bit 1 and x_j for all the others.
5. Each P_i replies with a string of 2λ bits. This string is interpreted as an assignment to all the variables that the prover received. ($\lambda/2$ distinguished variables plus three variables in each of the $\lambda/2$ clauses)

A k-prover proof system for Max-3SAT-5 (1 / 3)

1. We use a binary code that contains k code words, each of length λ and weight (number of 1's) $\lambda/2$, and Hamming distance at least $\lambda/3$. We choose $\lambda = \Theta(\log \log n)$ and k an arbitrarily large constant. (such a code can be easily found, but it is out of the scope of this presentation)
2. The verifier V selects λ clauses, say $C_1, ..., C_\lambda$ u.i.r.
3. From each C_j V selects a single variable u.i.r. These $x_1, ..., x_\lambda$ are called the *distinguished* variables.
4. Prover P_i receives C_j for those coordinates j in its code word that have the bit 1 and x_j for all the others.
5. Each P_i replies with a string of 2λ bits. This string is interpreted as an assignment to all the variables that the prover received. ($\lambda/2$ distinguished variables plus three variables in each of the $\lambda/2$ clauses)

A k-prover proof system for Max-3SAT-5 (1 / 3)

1. We use a binary code that contains k code words, each of length λ and weight (number of 1's) $\lambda/2$, and Hamming distance at least $\lambda/3$. We choose $\lambda = \Theta(\log \log n)$ and k an arbitrarily large constant. (such a code can be easily found, but it is out of the scope of this presentation)
2. The verifier V selects λ clauses, say $C_1, ..., C_\lambda$ u.i.r.
3. From each C_j V selects a single variable u.i.r. These $x_1, ..., x_\lambda$ are called the *distinguished* variables.
4. Prover P_i receives C_j for those coordinates j in its code word that have the bit 1 and x_j for all the others.
5. Each P_i replies with a string of 2λ bits. This string is interpreted as an assignment to all the variables that the prover received. ($\lambda/2$ distinguished variables plus three variables in each of the $\lambda/2$ clauses)

A k-prover proof system for Max-3SAT-5 (1 / 3)

1. We use a binary code that contains k code words, each of length λ and weight (number of 1's) $\lambda/2$, and Hamming distance at least $\lambda/3$. We choose $\lambda = \Theta(\log \log n)$ and k an arbitrarily large constant. (such a code can be easily found, but it is out of the scope of this presentation)
2. The verifier V selects λ clauses, say $C_1, ..., C_\lambda$ u.i.r.
3. From each C_j V selects a single variable u.i.r. These $x_1, ..., x_\lambda$ are called the *distinguished* variables.
4. Prover P_i receives C_j for those coordinates j in its code word that have the bit 1 and x_j for all the others.
5. Each P_i replies with a string of 2λ bits. This string is interpreted as an assignment to all the variables that the prover received. ($\lambda/2$ distinguished variables plus three variables in each of the $\lambda/2$ clauses)

A k-prover proof system for Max-3SAT-5 (2 / 3)

- For simplicity, assume that the corresponding bits in the prover's answer for each of the $\lambda/2$ clauses received encode a satisfying assignment for that clause.
- Observe that the answer of a prover induces an assignment to the distinguished variables.
- We say that the answers of two provers are *consistent* if the induced assignments to the distinguished variables are identical.
- We now introduce 2 acceptance predicates:
  - *Weak acceptance predicate*: At least one pair of provers is consistent.
  - *Strong acceptance predicate*: Every pair of provers is consistent.

A k-prover proof system for Max-3SAT-5 (2 / 3)

- For simplicity, assume that the corresponding bits in the prover's answer for each of the $\lambda/2$ clauses received encode a satisfying assignment for that clause.
- Observe that the answer of a prover induces an assignment to the distinguished variables.
- We say that the answers of two provers are *consistent* if the induced assignments to the distinguished variables are identical.
- We now introduce 2 acceptance predicates:
  - *Weak acceptance predicate*: At least one pair of provers is consistent.
  - *Strong acceptance predicate*: Every pair of provers is consistent.

A k-prover proof system for Max-3SAT-5 (2 / 3)

- For simplicity, assume that the corresponding bits in the prover's answer for each of the $\lambda/2$ clauses received encode a satisfying assignment for that clause.
- Observe that the answer of a prover induces an assignment to the distinguished variables.
- We say that the answers of two provers are *consistent* if the induced assignments to the distinguished variables are identical.
- We now introduce 2 acceptance predicates:
  - *Weak acceptance predicate*: At least one pair of provers is consistent.
  - *Strong acceptance predicate*: Every pair of provers is consistent.

A k-prover proof system for Max-3SAT-5 (2 / 3)

- For simplicity, assume that the corresponding bits in the prover's answer for each of the $\lambda/2$ clauses received encode a satisfying assignment for that clause.
- Observe that the answer of a prover induces an assignment to the distinguished variables.
- We say that the answers of two provers are *consistent* if the induced assignments to the distinguished variables are identical.
- We now introduce 2 acceptance predicates:
  - *Weak acceptance predicate*: At least one pair of provers is consistent.
  - *Strong acceptance predicate*: Every pair of provers is consistent.

A k-prover proof system for Max-3SAT-5 (3 / 3)

Lemma

Consider the k-prover proof system defined above and a 3CNF-5 formula ϕ. If ϕ is satisfiable, then the provers have a strategy that causes the verifier to always strongly accept. If at most a $(1-\epsilon)$-fraction of the clauses in ϕ are simultaneously satisfiable, then the verifier weakly accepts with probability at most $k^2 \cdot 2^{-c\lambda}$, where $c > 0$ is a constant that depends only on ϵ.

Proof.

On board... □

Construction of Partition Systems

Definition

A partition system $B(m, L, k, d)$ has the following properties:

1. There is a ground set B of m points.
2. There is a collection of L distinct partitions $p_1, ..., p_L$.
3. For $1 \leq i \leq L$, partition p_i is a collection of k disjoint subsets of B whose union is B.
4. Any cover of the m points by subsets that appear in pairwise different partitions requires at least d subsets.

Lemma

For every $c \geq 0$ and m sufficiently large, there is a partition system $B(m, L, k, d)$ whose parameters satisfy the following inequalities:

1. $L \approxeq (\log m)^c$
2. *k can be chosen arbitrarily as long as $k < \ln m / 3 \ln \ln m$*
3. *$d = (1 - f(k))k \ln m$, where $f(k) \to 0$ as $k \to \infty$.*

Construction of Partition Systems

Definition

A partition system $B(m, L, k, d)$ has the following properties:

1. There is a ground set B of m points.
2. There is a collection of L distinct partitions $p_1, ..., p_L$.
3. For $1 \leq i \leq L$, partition p_i is a collection of k disjoint subsets of B whose union is B.
4. Any cover of the m points by subsets that appear in pairwise different partitions requires at least d subsets.

Lemma

For every $c \geq 0$ and m sufficiently large, there is a partition system $B(m, L, k, d)$ whose parameters satisfy the following inequalities:

1. $L \approxeq (\log m)^c$
2. *k can be chosen arbitrarily as long as $k < \ln m / 3 \ln \ln m$*
3. *$d = (1 - f(k))k \ln m$, where $f(k) \to 0$ as $k \to \infty$.*

Construction of Partition Systems

- The above partition system, with $f(k) = 2/k$ can be constructed in $ZTIME(m^{O(\log m)})$. The construction is randomized.
- However, there is a deterministic construction due to Naor, Schulman and Srinivasan (1995) that can be used instead.

Construction of Partition Systems

- The above partition system, with $f(k) = 2/k$ can be constructed in $ZTIME(m^{O(\log m)})$. The construction is randomized.
- However, there is a deterministic construction due to Naor, Schulman and Srinivasan (1995) that can be used instead.

The Reduction to Set Cover

On board...

Outline

Motivation

- The main motivation for the conjecture is to prove inapproximability results.
- The UGC states that a specific problem, called the Unique Game, is inapproximable.
- A **gap-preserving** reduction from the Unique Game then implies inapproximability results for other *NP*-complete problems.

Motivation

- The main motivation for the conjecture is to prove inapproximability results.
- The UGC states that a specific problem, called the Unique Game, is inapproximable.
- A **gap-preserving** reduction from the Unique Game then implies inapproximability results for other *NP*-complete problems.

Motivation

- The main motivation for the conjecture is to prove inapproximability results.
- The UGC states that a specific problem, called the Unique Game, is inapproximable.
- A **gap-preserving** reduction from the Unique Game then implies inapproximability results for other *NP*-complete problems.

Unique Games

Definition (Unique Game)

A Unique Game $\mathcal{U}(G(V,E),[n],\{\pi_e \mid e \in E\})$ is a constraint satisfaction problem defined as follows: $G(V,E)$ is a directed graph whose vertices represent variables and edges represent constraints. The goal is to assign to each vertex a label from the set $[n]$. The constraint on an edge $e=(v,w) \in E$ is described by a bijection $\pi_e : [n] \to [n]$. A labeling $L : V \to [n]$ satisfies the constraint on edge $e=(v,w)$ if and only if $\pi_e(L(v)) = L(w)$. Let $OPT(\mathcal{U})$ denote the maximum fraction of constraints that can be satisfied by any labeling:

$$OPT(\mathcal{U}) := \max_{L:V\to[n]} \frac{1}{|E|} \cdot |\{e \in E \mid L \text{ satisfies } e\}|.$$

The Unique Games Conjecture

Conjecture (UGC, Khot 02)

For every $\epsilon, \delta > 0$, there exists a constant $n = n(\epsilon, \delta)$, such that given a Unique Game instance $\mathcal{U}(G(V, E), [n], \{\pi_e \mid e \in E\})$, it is NP-hard to distinguish between these two cases:

- *YES Case: $OPT(\mathcal{U}) \geq 1 - \epsilon$.*
- *NO Case: $OPT(\mathcal{U}) \leq \delta$.*

Inapproximability results assuming the UGC

Problem	Best Approx. Known	Inapprox. Known Under UGC	Best Inapprox. Known
Vertex Cover	2	$2-\epsilon$	1.36
Max-Cut	$a \approxeq 0.878$ (SDP gap)	$a+\epsilon$	16/17
Any CSP $\mathcal{C}$ with integrality gap $a_{\mathcal{C}}$	$a_{\mathcal{C}}$	$a_{\mathcal{C}}$	-
Non-uniform Sparsest Cut	$\tilde{O}(\sqrt{\log n})$	$\omega(1)$	APX-hard

SDPs and the UGC (1 / 3)

Definition (k-CSP)

A k-ary CSP $\mathcal{C}$ is defined by a collection of finitely many predicates $P : \{0,1\}^k \to \{0,1\}$. An instance is specified as $I(V, E, \{P_e \mid e \in E\})$, where $V = \{x_1, ..., x_N\}$ is a set of boolean variables and E is a collection of constraints, each being a size k subset of V. A constraint $e \in E$ is denoted as $e = (x_{e_1}, ..., x_{e_k})$, $e_i \in [N]$, with a specific order on the variables, and has an associated predicate $P_e \in \mathcal{C}$.

An assignment is a map $p : V \to \{0,1\}$, and satisfies a constraint e if

$$P_e(p(x_{e_1}), ..., p(x_{e_k})) = 1.$$

Let $OPT(I)$ denote the maximum fraction of constraints satisfied by any assignment.

SDPs and the UGC (1 / 3)

Definition (k-CSP)

A k-ary CSP $\mathcal{C}$ is defined by a collection of finitely many predicates $P : \{0,1\}^k \to \{0,1\}$. An instance is specified as $I(V, E, \{P_e \mid e \in E\})$, where $V = \{x_1, ..., x_N\}$ is a set of boolean variables and E is a collection of constraints, each being a size k subset of V. A constraint $e \in E$ is denoted as $e = (x_{e_1}, ..., x_{e_k})$, $e_i \in [N]$, with a specific order on the variables, and has an associated predicate $P_e \in \mathcal{C}$.

An assignment is a map $p : V \to \{0,1\}$, and satisfies a constraint e if

$$P_e(p(x_{e_1}), ..., p(x_{e_k})) = 1.$$

Let $OPT(I)$ denote the maximum fraction of constraints satisfied by any assignment.

SDPs and the UGC (1 / 3)

Definition (k-CSP)

A k-ary CSP $\mathcal{C}$ is defined by a collection of finitely many predicates $P : \{0,1\}^k \to \{0,1\}$. An instance is specified as $I(V, E, \{P_e \mid e \in E\})$, where $V = \{x_1, ..., x_N\}$ is a set of boolean variables and E is a collection of constraints, each being a size k subset of V. A constraint $e \in E$ is denoted as $e = (x_{e_1}, ..., x_{e_k})$, $e_i \in [N]$, with a specific order on the variables, and has an associated predicate $P_e \in \mathcal{C}$.

An assignment is a map $p : V \to \{0,1\}$, and satisfies a constraint e if

$$P_e(p(x_{e_1}), ..., p(x_{e_k})) = 1.$$

Let $OPT(I)$ denote the maximum fraction of constraints satisfied by any assignment.

SDPs and the UGC (2 / 3)

- There is a natural SDP relaxation for the above CSP.
- Let $SDP(I)$ denote the optimum of the SDP relaxation, and define the integrality gap $a_{\mathcal{C}}$ as:

$$a_{\mathcal{C}} := \sup_{I} \frac{SDP(I)}{OPT(I)}$$

- We have $OPT(I) \leq SDP(I) \leq a_{\mathcal{C}} \cdot OPT(I)$, and so we get an $a_{\mathcal{C}}$ approximation from the SDP.
- It has been proved that the UGC captures in a precise way the limitations of SDP. More formally:

SDPs and the UGC (2 / 3)

- There is a natural SDP relaxation for the above CSP.
- Let $SDP(I)$ denote the optimum of the SDP relaxation, and define the integrality gap $a_{\mathcal{C}}$ as:

$$a_{\mathcal{C}} := \sup_{I} \frac{SDP(I)}{OPT(I)}$$

- We have $OPT(I) \leq SDP(I) \leq a_{\mathcal{C}} \cdot OPT(I)$, and so we get an $a_{\mathcal{C}}$ approximation from the SDP.
- It has been proved that the UGC captures in a precise way the limitations of SDP. More formally:

SDPs and the UGC (2 / 3)

- There is a natural SDP relaxation for the above CSP.
- Let $SDP(I)$ denote the optimum of the SDP relaxation, and define the integrality gap $a_{\mathcal{C}}$ as:

$$a_{\mathcal{C}} := \sup_{I} \frac{SDP(I)}{OPT(I)}$$

- We have $OPT(I) \leq SDP(I) \leq a_{\mathcal{C}} \cdot OPT(I)$, and so we get an $a_{\mathcal{C}}$ approximation from the SDP.
- It has been proved that the UGC captures in a precise way the limitations of SDP. More formally:

SDPs and the UGC (2 / 3)

- There is a natural SDP relaxation for the above CSP.
- Let $SDP(I)$ denote the optimum of the SDP relaxation, and define the integrality gap $a_{\mathcal{C}}$ as:

$$a_{\mathcal{C}} := \sup_{I} \frac{SDP(I)}{OPT(I)}$$

- We have $OPT(I) \leq SDP(I) \leq a_{\mathcal{C}} \cdot OPT(I)$, and so we get an $a_{\mathcal{C}}$ approximation from the SDP.
- It has been proved that the UGC captures in a precise way the limitations of SDP. More formally:

SDPs and the UGC (2 / 3)

- There is a natural SDP relaxation for the above CSP.
- Let $SDP(I)$ denote the optimum of the SDP relaxation, and define the integrality gap $a_{\mathcal{C}}$ as:

$$a_{\mathcal{C}} := \sup_{I} \frac{SDP(I)}{OPT(I)}$$

- We have $OPT(I) \leq SDP(I) \leq a_{\mathcal{C}} \cdot OPT(I)$, and so we get an $a_{\mathcal{C}}$ approximation from the SDP.
- It has been proved that the UGC captures in a precise way the limitations of SDP. More formally:

SDPs and the UGC (3 / 3)

Theorem (Raghavendra 08)

Suppose there is an instance I^* of the CSP $\mathcal{C}$ such that $SDP(I^*) \geq c$ and $OPT(I^*) \leq s$. Then for every $\gamma > 0$, there exist $\epsilon, \delta > 0$, and a polynomial time reduction from a Unique Game instance to an instance I of the CSP such that:

- *(YES Case): If $OPT(\mathcal{U}) \geq 1 - \epsilon$, then $OPT(I) \geq c - \gamma$.*
- *(NO Case): If $OPT(\mathcal{U}) \leq \delta$, then $OPT(I) \leq s + \gamma$.*

In particular, assuming the UGC, it is NP-hard to approximate the CSP within any factor strictly less than $a_{\mathcal{C}}$.

SDPs and the UGC (3 / 3)

Theorem (Raghavendra 08)

Suppose there is an instance I^* of the CSP $\mathcal{C}$ such that $SDP(I^*) \geq c$ and $OPT(I^*) \leq s$. Then for every $\gamma > 0$, there exist $\epsilon, \delta > 0$, and a polynomial time reduction from a Unique Game instance to an instance I of the CSP such that:

- *(YES Case): If $OPT(\mathcal{U}) \geq 1 - \epsilon$, then $OPT(I) \geq c - \gamma$.*
- *(NO Case): If $OPT(\mathcal{U}) \leq \delta$, then $OPT(I) \leq s + \gamma$.*

In particular, assuming the UGC, it is NP-hard to approximate the CSP within any factor strictly less than $a_{\mathcal{C}}$.

Bibliography

1. How NP Got a New Definition: A Survey of Probabilistically Checkable Proofs.
   Sanjeev Arora. *ICM*, 2002.
2. Computational Complexity.
   Sanjeev Arora, Boaz Barak. *Cambridge University Press*, 2009.
3. Probabilistically Checkable Proofs.
   Andreas Galanis. *ECE - NTUA thesis*, 2009.
4. Probabilistically Checkable Proofs and Codes.
   Irit Dinur. *ICM*, 2010.
5. A Threshold of $\ln n$ for Approximating Set Cover.
   Uriel Feige. *J. ACM 45(4): 634-652*, 1998. (Preliminary version in STOC 1996)
6. On the Power of Multi-Prover Interactive Protocols.
   Lance Fortnow, John Rompel, Michael Sipser. *Theor. Comput. Sci. 134(2): 545-557*, 1994.

Bibliography

7. On the Unique Games Conjecture.
   Subhash Khot. *CCC*, 2010.
8. Optimal algorithms and inapproximability results for every CSP?
   Prasad Raghavendra. *STOC*, 2008.
9. Inapproximability of Combinatorial Optimization Problems.
   Luca Trevisan. *ECCC*, 2010.

THANK YOU!