

# CSL853: Complexity Theory

Ragesh Jaiswal, CSE, IIT Delhi

## PCP Theorem and Hardness of Approximation

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- There is a  $7/8$ -approximation algorithm for MAX-3SAT.

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- There is a  $7/8$ -approximation algorithm for MAX-3SAT.
- Is there an efficient algorithm with better approximation factor than  $7/8$ ?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- There is a 2-approximation algorithm for MIN-VC.
- Is there an efficient algorithm with better approximation factor than 2?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- Claim 1:  $C$  is a vertex cover of any graph  $G = (V, E)$  if and only if  $V \setminus C$  is an independent set of  $G$ .
- Does the above claim mean that there is a constant factor approximation algorithm for MAX-IS?
- What is the best approximation algorithm that you can think of for MAX-IS?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- There is an  $O(\log n)$ -approximation algorithm for MIN-SC.
- Is there an efficient algorithm with better approximation factor than  $O(\log n)$ ?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- There is a 2-approximation algorithm for MAX-CUT.
- Is there an efficient algorithm with better approximation factor than 2?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- One way to deal with NP-hard optimization problems is to design approximation algorithms.
- We will work with the following NP-hard optimization problems.
  - MAX-3SAT: Given a Boolean formula in 3 CNF form, output an assignment that makes the maximum number of clauses true.
  - MIN-VC: Given a graph  $G$ , find a vertex cover of  $G$  with minimum cardinality.
  - MAX-IS: Given a graph  $G$ , find an independent set of  $G$  with maximum cardinality.
  - MIN-SC: Given a set  $S$  and subsets  $S_1, \dots, S_m$ , find the set cover with minimum cardinality.
  - MAX-CUT: Given a graph  $G$ , find a partition of the vertices such that the number of edges across the partition is maximized.
- How do we show that there cannot exist an efficient algorithm for a problem that gives approximation factor better than some quantity  $\alpha$ ?

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- How do we show that there cannot exist an efficient algorithm for a problem that gives approximation factor better than some quantity  $\alpha$ ?

### An idea

Suppose there exists a efficiently computable function  $f$  that takes as input Boolean formulas and outputs 3-CNF formulas such that the following holds:

$$\psi \in \text{SAT} \Rightarrow f(\psi) \in \text{SAT}, \text{ and}$$

$$\psi \notin \text{SAT} \Rightarrow < \alpha\text{-fraction of clauses in } f(\psi) \text{ are satisfiable}$$

Then there cannot exist an  $\alpha$ -approximation algorithm for MAX-3SAT, unless  $P = NP$ .

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- How do we show that there cannot exist an efficient algorithm for a problem that gives approximation factor better than some quantity  $\alpha$ ?

### An idea

Suppose there exists a efficiently computable function  $f$  that takes as input Boolean formulas and outputs 3-CNF formulas such that the following holds:

$$\psi \in \text{SAT} \Rightarrow f(\psi) \in \text{SAT}, \text{ and}$$

$$\psi \notin \text{SAT} \Rightarrow < \alpha\text{-fraction of clauses in } f(\psi) \text{ are satisfiable}$$

Then there cannot exist an  $\alpha$ -approximation algorithm for MAX-3SAT, unless  $P = NP$ .

- We may further extend such *inapproximability* to other optimization problems using **gap preserving reductions**.
  - For instance, we can argue that if such a function  $f$  exists, then there cannot exist an  $\alpha$ -approximation algorithm for MAX-IS, unless  $P = NP$ .

# PCP Theorem and Hardness of Approximation

## Hardness of approximation

- How do we show that there cannot exist an efficient algorithm for a problem that gives approximation factor better than some quantity  $\alpha$ ?

### An idea

Suppose there exists a efficiently computable function  $f$  that takes as input Boolean formulas and outputs 3-CNF formulas such that the following holds:

$$\psi \in \text{SAT} \Rightarrow f(\psi) \in \text{SAT}, \text{ and}$$

$$\psi \notin \text{SAT} \Rightarrow < \alpha\text{-fraction of clauses in } f(\psi) \text{ are satisfiable}$$

Then there cannot exist an  $\alpha$ -approximation algorithm for MAX-3SAT, unless  $P = NP$ .

- Does such a function  $f$  exist?

# PCP Theorem and Hardness of Approximation

## PCP Theorem

- PCP stands for Probabilistically Checkable Proofs. This is a proof system in which the probabilistic verifier has *random access* to the proof string and it makes few *queries* to this string to make its decision.

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ 's output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ 's output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

- What is  $\text{PCP}(0, 0)$ ?
- What is  $\text{PCP}(O(\log n), 0)$ ?
- What is  $\text{PCP}(0, O(\log n))$ ?
- What is  $\text{PCP}(\text{poly}(n), 0)$ ?
- What is  $\text{PCP}(0, \text{poly}(n))$ ?

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ ’s output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

- What is  $\text{PCP}(0, 0)$ ? **P**
- What is  $\text{PCP}(O(\log n), 0)$ ? **P**
- What is  $\text{PCP}(0, O(\log n))$ ? **P**
- What is  $\text{PCP}(\text{poly}(n), 0)$ ? **coRP**
- What is  $\text{PCP}(0, \text{poly}(n))$ ? **NP**

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ ’s output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

- Claim:  $\text{PCP}(r(n), q(n)) \subseteq \text{NTIME}(2^{O(r(n))} \cdot q(n))$ .

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ ’s output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

Theorem (The PCP Theorem, Arora-Safra’92,  
Arora-Lund-Motwani-Sudan-Szegedy’92)

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

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (PCP verifier)

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

- Efficiency: On input string  $x \in \{0, 1\}^n$  and given random access to a string  $\pi \in \{0, 1\}^*$  of length at most  $q(n) \cdot 2^{r(n)}$  (which we call the proof),  $V$  uses at most  $r(n)$  random coins and makes at most  $q(n)$  *nonadaptive queries* to locations of  $\pi$ . Then it outputs “accept” or “reject”. We let  $V^\pi(x)$  denote the random variable representing  $V$ ’s output on input  $x$  and with random access to  $\pi$ .
- 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 string  $\pi$  the correct proof for  $x$ .
- Soundness: If  $x \notin L$ , then for every proof  $\pi \in \{0, 1\}^*$ ,  $\Pr[V^\pi(x) = 1] \leq 1/2$ .

### Definition (PCP language)

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

- The PCP theorem turns out to be equivalent to a statement about the inapproximability of MAX-3SAT. We will see this equivalence next.

# PCP Theorem and Hardness of Approximation

## PCP Theorem

- The PCP theorem turns out to be equivalent to a statement about the inapproximability of MAX-3SAT. We will see this equivalence next.
- We will consider the Constraint Satisfaction Problem (CSP) which is a generalization of 3SAT.

### Definition (Constraint Satisfaction Problem (CSP))

Let  $q \in \mathbb{N}$ . A  $q$ CSP instance  $\psi$  is a collection of functions  $\psi_1, \dots, \psi_m$  (called *constraints*) from  $\{0, 1\}^n$  to  $\{0, 1\}$  such that each function  $\psi_i$  depends on at most  $q$  of its input locations. That is, for every  $i \in [m]$  there exists  $j_1, \dots, j_q \in [n]$  and  $f : \{0, 1\}^q \rightarrow \{0, 1\}$  such that  $\psi_i(u) = f(u_{j_1}, u_{j_2}, \dots, u_{j_q})$  for every  $u \in \{0, 1\}^n$ .

We say that an assignment  $u \in \{0, 1\}^n$  satisfies constraint  $\psi_i$  if  $\psi_i(u) = 1$ .

The fraction of constraint satisfied by  $u$  is  $\frac{\sum_{i=1}^m \psi_i(u)}{m}$ , and we let  $val(\psi)$  denote the maximum of this value over all  $u \in \{0, 1\}^n$ . We say that  $\psi$  is satisfiable if  $val(\psi) = 1$ . We call  $q$  the *arity* of  $\psi$ .

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (Constraint Satisfaction Problem (CSP))

Let  $q \in \mathbb{N}$ . A  $q$ CSP instance  $\psi$  is a collection of functions  $\psi_1, \dots, \psi_m$  (called *constraints*) from  $\{0, 1\}^n$  to  $\{0, 1\}$  such that each function  $\psi_i$  depends on at most  $q$  of its input locations. That is, for every  $i \in [m]$  there exists  $j_1, \dots, j_q \in [n]$  and  $f : \{0, 1\}^q \rightarrow \{0, 1\}$  such that  $\psi_i(u) = f(u_{j_1}, u_{j_2}, \dots, u_{j_q})$  for every  $u \in \{0, 1\}^n$ .

We say that an assignment  $u \in \{0, 1\}^n$  satisfies constraint  $\psi_i$  if  $\psi_i(u) = 1$ .

The fraction of constraint satisfied by  $u$  is  $\frac{\sum_{i=1}^m \psi_i(u)}{m}$ , and we let  $\text{val}(\psi)$  denote the maximum of this value over all  $u \in \{0, 1\}^n$ . We say that  $\psi$  is satisfiable if  $\text{val}(\psi) = 1$ . We call  $q$  the *arity* of  $\psi$ .

- Claim 1: 3SAT is a special case of  $q$ CSP.
- We will assume that the number of variables  $n \leq q \cdot m$ .
- Claim 2: A  $q$ CSP instance with  $n$  variables and  $m$  constraints can be described using  $O(m \cdot q \cdot \log n \cdot 2^q)$  bits

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Definition (Constraint Satisfaction Problem (CSP))

Let  $q \in \mathbb{N}$ . A  $q$ CSP instance  $\psi$  is a collection of functions  $\psi_1, \dots, \psi_m$  (called *constraints*) from  $\{0, 1\}^n$  to  $\{0, 1\}$  such that each function  $\psi_i$  depends on at most  $q$  of its input locations. That is, for every  $i \in [m]$  there exists  $j_1, \dots, j_q \in [n]$  and  $f : \{0, 1\}^q \rightarrow \{0, 1\}$  such that  $\psi_i(u) = f(u_{j_1}, u_{j_2}, \dots, u_{j_q})$  for every  $u \in \{0, 1\}^n$ . We say that an assignment  $u \in \{0, 1\}^n$  satisfies constraint  $\psi_i$  if  $\psi_i(u) = 1$ . The fraction of constraint satisfied by  $u$  is  $\frac{\sum_{i=1}^m \psi_i(u)}{m}$ , and we let  $\text{val}(\psi)$  denote the maximum of this value over all  $u \in \{0, 1\}^n$ . We say that  $\psi$  is satisfiable if  $\text{val}(\psi) = 1$ . We call  $q$  the *arity* of  $\psi$ .

### Definition (Gap CSP)

For every  $q \in \mathbb{N}$ ,  $\rho \leq 1$ , define  $\rho$ -GAP $q$ CSP to be the problem of determining for a given CSP instance  $\psi$ , whether (1)  $\text{val}(\psi) = 1$  (in which case we say that  $\psi$  is a YES instance) or (2)  $\text{val}(\psi) < \rho$  (in which case we say that it is a NO instance)

### Definition (NP-hardness of Gap CSP)

We say that  $\rho$ -GAP $q$ CSP is NP-hard if for every language  $L \in \text{NP}$ , there is a polynomial time computable function that maps strings to  $q$ CSP instances such that

$$\text{(Completeness)} \quad x \in L \Rightarrow \text{val}(f(x)) = 1.$$

$$\text{(Soundness)} \quad x \notin L \Rightarrow \text{val}(f(x)) < \rho.$$

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Theorem

*If  $\text{NP} = \text{PCP}(\log n, 1)$ , then there exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard.*

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Theorem

*If  $\text{NP} = \text{PCP}(\log n, 1)$ , then there exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard.*

### Theorem

*If there exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard, then  $\text{NP} = \text{PCP}(\log n, 1)$ .*

# PCP Theorem and Hardness of Approximation

## PCP Theorem

### Theorem

*If  $\text{NP} = \text{PCP}(\log n, 1)$ , then there exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard.*

### Theorem

*If there exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard, then  $\text{NP} = \text{PCP}(\log n, 1)$ .*

### Theorem

*The following two statements are equivalent:*

- ① *There exists constants  $q \in \mathbb{N}, \rho \in (0, 1)$  such that  $\rho\text{-GAP}_q\text{CSP}$  is NP-hard.*
- ② *There exists  $\rho < 1$  such that for every  $L \in \text{NP}$ , there is a polynomial time function mapping strings to 3CNF formulas such that*

$$x \in L \Rightarrow \text{val}(f(x)) = 1$$

$$x \notin L \Rightarrow \text{val}(f(x)) < \rho$$

# PCP Theorem and Hardness of Approximation

## Vertex Cover and Independent Set

### Theorem

*If  $P \neq NP$ , then there are some constants  $\rho, \rho' < 1$  such that the problem MAX-IS cannot be  $\rho$ -approximated and MIN-VC cannot be  $\rho'$ -approximated in polynomial time.*

### Theorem

*For every  $\rho < 1$ , computing a  $\rho$ -approximation to MAX-IS is NP-hard.*

# PCP Theorem and Hardness of Approximation

## Vertex Cover and Independent Set

### Theorem

*If  $P \neq NP$ , then there are some constants  $\rho, \rho' < 1$  such that the problem MAX-IS cannot be  $\rho$ -approximated in polynomial time and MIN-VC cannot be  $\rho'$ -approximated.*

### Theorem

*For every  $\rho < 1$ , computing a  $\rho$ -approximation to MAX-IS is NP-hard.*

- Are these *optimal* inapproximability results?
- What about inapproximability results for other problems?

# PCP Theorem and Hardness of Approximation

## Vertex Cover and Independent Set

### Theorem

*If  $P \neq NP$ , then there are some constants  $\rho, \rho' < 1$  such that the problem MAX-IS cannot be  $\rho$ -approximated in polynomial time and MIN-VC cannot be  $\rho'$ -approximated.*

### Theorem

*For every  $\rho < 1$ , computing a  $\rho$ -approximation to MAX-IS is NP-hard.*

- Are these *optimal* inapproximability results? **No**
- What about inapproximability results for other problems?
  - Various other PCP theorems are used to get (optimal) results for various other problems.
- The kind of results that we get from the  $NP = PCP(\log n, 1)$  is of the form: “*there does not exist a PTAS for the problem, unless  $P = NP$* ”.

End