The Tale of the PCP Theorem

The PCP Theorem and Hardness of Approximation

Jatin Arora Shaan Vaidya

Computational Complexity

Outline

Approximate solutions to NP-hard optimization problems

- MAX-3SAT is NP-hard
- Can't hope for a fast algorithm that always gets the best solution
- Can we at least guarantee to get a "pretty good" solution efficiently?

Approximate solutions to NP-hard optimization problems

Definition

Approximation of MAX-3SAT:
For every $\rho \leq 1$, an algorithm A is a $\rho -$ approximation algorithm for MAX-3SAT if for every 3CNF formula φ with m clauses, $A(\varphi)$ outputs an assignment satisfying at least $\rho \cdot \mathrm{val}(\varphi)$ of φ's clauses.

- Greedy algorithm - $1/2$-approximation
- Best known - $7/8$ approximation algorithm

Approximate solutions to NP-hard optimization problems

Definition

Approximation of MAX-3SAT:
For every $\rho \leq 1$, an algorithm A is a $\rho -$ approximation algorithm for MAX-3SAT if for every 3CNF formula φ with m clauses, $A(\varphi)$ outputs an assignment satisfying at least $\rho \cdot \text{val}(\varphi)$ of φ's clauses.

- Greedy algorithm - 1/2-approximation
- Best known - 7/8 approximation algorithm
- Can we do better? Or is there a limit to approximation?

GAP problems

- How does one study the hardness of approximation problems?

GAP problems

- MAXCLIQUE: Given a graph G, output the vertices in its largest clique
- SET COVERING PROBLEM: Given a collection of sets $S_1, S_2, \ldots, S_m$ that cover a universe $U = \{1, 2, \ldots, n\}$, find the smallest sub-collection of sets $S_{i_1}, S_{i_2}, \ldots$ that also cover the universe

GAP problems

- MAXCLIQUE: Given a graph G, output the vertices in its largest clique
- CLIQUE $= \{(G, k) \mid$ graph G has a clique of size $\geq k\}$
- SET COVERING PROBLEM: Given a collection of sets $S_1, S_2, \ldots, S_m$ that cover a universe $U = \{1, 2, \ldots, n\}$, find the smallest sub-collection of sets $S_{i_1}, S_{i_2}, \ldots$ that also cover the universe
- SET-COVER $= \{\ (U; \{S_1, S_2, \ldots, S_m\}, k) \mid \exists i_1 \leq i_2 \ldots, i_k \leq m$ st $\bigcup_{j=1}^{m} S_{i_j} = U\}$

GAP problems

- MAXCLIQUE: Given a graph G, output the vertices in its largest clique
- CLIQUE $= \{(G, k) \mid$ graph G has a clique of size $\geq k\}$
- SET COVERING PROBLEM: Given a collection of sets $S_1, S_2, \ldots, S_m$ that cover a universe $U = \{1, 2, \ldots, n\}$, find the smallest sub-collection of sets $S_{i_1}, S_{i_2}, \ldots$ that also cover the universe
- SET-COVER $= \{\ (U; \{S_1, S_2, \ldots, S_m\}, k) \mid \exists i_1 \leq i_2 \ldots, i_k \leq m$ st $\bigcup_{j=1}^{m} S_{i_j} = U\}$

As in the case of computational problems, it would be nice if we could capture the hardness of the approximation problems via decision problems.

GAP problems

Definition

Promise problems

A promise problem $\Pi \subseteq \Sigma^*$ is specified by a pair of sets (YES, NO) such that YES, NO $\subseteq \Sigma^*$ and YES $\cap$ NO $= \emptyset$

GAP problems

Definition

Promise problems
A promise problem $\Pi \subseteq \Sigma^*$ is specified by a pair of sets (YES, NO) such that YES, NO $\subseteq \Sigma^*$ and YES $\cap$ NO $= \emptyset$

Gap problems are promise problems

GAP problems

The gap problem corresponding to the α-approximating MAX3SAT, called gap$_\alpha$-MAX3SAT (for $\alpha \leq 1$)

GAP problems

The gap problem corresponding to the α-approximating MAX3SAT, called gap$_\alpha$-MAX3SAT (for $\alpha \leq 1$)

gap$_\alpha$-MAX3SAT is a promise problem whose (YES, NO) are as follows:

$$\text{YES} = \{\langle \varphi, k \rangle \mid \text{there is an assignment satisfying } \geq k \text{ clauses of } \varphi\}$$
$$\text{NO} = \{\langle \varphi, k \rangle \mid \text{every assignment satisfies } < \alpha k \text{ clauses of } \varphi\}$$

Equivalence of GAP problems and Approximation problems

For any $0 < \alpha < 1$, α-approximating MAX3SAT is polynomially equivalent to solving gap$_\alpha$-MAX3SAT

Equivalence of GAP problems and Approximation problems

Proof. ($\Rightarrow$)

Suppose there is an α-approximation algorithm **A** to MAX3SAT.
Consider the following algorithm **B** for gap$_\alpha$-MAX3SAT

B : On input $\langle \varphi, k \rangle$

1. Run **A** on φ and let $k' = \mathbf{A}(\varphi)$
2. Accept iff $k' \geq \alpha k$.

Equivalence of GAP problems and Approximation problems

Proof. $(\Rightarrow)$

Suppose there is an α-approximation algorithm **A** to MAX3SAT.
Consider the following algorithm **B** for gap$_\alpha$-MAX3SAT

B : On input $\langle \varphi, k \rangle$

1. Run **A** on φ and let $k' = \mathbf{A}(\varphi)$
2. Accept iff $k' \geq \alpha k$.

Equivalence of GAP problems and Approximation problems

Proof. $(\Rightarrow)$

Suppose there is an α-approximation algorithm **A** to MAX3SAT.
Consider the following algorithm **B** for gap_α-MAX3SAT

B : On input $\langle \varphi, k \rangle$

1. Run **A** on φ and let $k' = \mathbf{A}(\varphi)$
2. Accept iff $k' \geq \alpha k$.

$(\Leftarrow)$ Suppose instead there is an algorithm **B** that solves gap_α-MAX3SAT

A : On input φ

1. Let m be the number of clauses of φ
2. Run **B** on $\langle \varphi, 1 \rangle, \langle \varphi, 2 \rangle, \langle \varphi, 3 \rangle, \ldots, \langle \varphi, m \rangle$.
3. Let the largest k such that **B** accepted $\langle \varphi, k \rangle$
4. Output αk

Equivalence of GAP problems and Approximation problems

Thus, to show that approximating MAX3SAT is hard, it suffices (and is necessary) to show that gap_α-MAX3SAT is hard.

Interactive Proof Systems

- A proof system consists of a verifier V and prover P

Interactive Proof Systems

- A proof system consists of a verifier V and prover P
- Given a statement x, such as φ is satisfiable P produces a candidate proof π for the statement φ

Interactive Proof Systems

- A proof system consists of a verifier V and prover P
- Given a statement x, such as φ is satisfiable P produces a candidate proof π for the statement φ
- The verifier V then reads the statement-proof pair (φ, π) and either accepts or rejects the proof π for φ.

Interactive Proof Systems

- A proof system consists of a verifier V and prover P
- Given a statement x, such as φ is satisfiable P produces a candidate proof π for the statement φ
- The verifier V then reads the statement-proof pair (φ, π) and either accepts or rejects the proof π for φ.
- Completeness and Soundness properties define many complexity classes of importance

Probabilistically Checkable Proof Systems

Definition

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 is a polynomial-time probabilistic algorithm $\mathcal{V}$ satisfying:

Probabilistically Checkable Proof Systems

Definition

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 is a polynomial-time probabilistic algorithm $\mathcal{V}$ satisfying:

Efficiency On input $x \in 0, 1n$ and given random access to the proof $\pi \in \{0, 1\}^*$, $\mathcal{V}$ uses at most r(n) random coins and makes at most q(n) queries to locations of π to decide x.

Probabilistically Checkable Proof Systems

Definition

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 is a polynomial-time probabilistic algorithm $\mathcal{V}$ satisfying:

Efficiency On input $x \in 0, 1n$ and given random access to the proof $\pi \in \{0, 1\}^*$, $\mathcal{V}$ uses at most r(n) random coins and makes at most q(n) queries to locations of π to decide x.

Completeness $x \in L \implies \exists \pi \in \{0, 1\},\ Pr[\mathcal{V}^{\pi}(x) = 1] = 1$

Probabilistically Checkable Proof Systems

Definition

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 is a polynomial-time probabilistic algorithm $\mathcal{V}$ satisfying:

Efficiency On input $x \in 0, 1n$ and given random access to the proof $\pi \in \{0,1\}^*$, $\mathcal{V}$ uses at most r(n) random coins and makes at most q(n) queries to locations of π to decide x.

Completeness $x \in L \implies \exists \pi \in \{0,1\},\ Pr[\mathcal{V}^{\pi}(x) = 1] = 1$

Soundness $x \notin L \implies \forall \pi Pr[\mathcal{V}^{\pi}(x) = 1] \leq 1/2$

Probabilistically Checkable Proof Systems

Definition

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 is a polynomial-time probabilistic algorithm $\mathcal{V}$ satisfying:

Efficiency On input $x \in 0, 1n$ and given random access to the proof $\pi \in \{0, 1\}^*$, $\mathcal{V}$ uses at most r(n) random coins and makes at most q(n) queries to locations of π to decide x.

Completeness $x \in L \implies \exists \pi \in \{0, 1\},\ Pr[\mathcal{V}^\pi(x) = 1] = 1$

Soundness $x \notin L \implies \forall \pi Pr[\mathcal{V}^\pi(x) = 1] \leq 1/2$

L is in $\mathrm{PCP}(r(n), q(n))$ if L has a $(O(r(n)), O(q(n)))$-PCP verifier

Some remarks

- The verifier could be either non-adaptive or adaptive (i.e., the locations probed by the verifier could depend on the earlier probes)

Some remarks

- The verifier could be either non-adaptive or adaptive (i.e., the locations probed by the verifier could depend on the earlier probes)
- If the verifier is non-adaptive then the size of the proof is bounded above by $q(n) \cdot 2^{r(n)}$ while if the verifier is adaptive, it is bounded by $2^{r(n)+q(n)}$
- Assume non-adaptive unless otherwise specified

Some remarks

- The verifier could be either non-adaptive or adaptive (i.e., the locations probed by the verifier could depend on the earlier probes)
- If the verifier is non-adaptive then the size of the proof is bounded above by $q(n) \cdot 2^{r(n)}$ while if the verifier is adaptive, it is bounded by $2^{r(n)+q(n)}$
- Assume non-adaptive unless otherwise specified
- $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{r(n)} \cdot q(n))$

Some remarks

- The verifier could be either non-adaptive or adaptive (i.e., the locations probed by the verifier could depend on the earlier probes)
- If the verifier is non-adaptive then the size of the proof is bounded above by $q(n) \cdot 2^{r(n)}$ while if the verifier is adaptive, it is bounded by $2^{r(n)+q(n)}$
- Assume non-adaptive unless otherwise specified
- $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{r(n)} \cdot q(n))$
- It follows from definition that:
  - $\mathrm{NP} = \mathrm{PCP}_{1,0}(0, poly(n))$
  - $\mathrm{BPP} = \mathrm{PCP}_{2/3,1/3}(poly(n), 0)$
  - $\mathrm{P} = \mathrm{PCP}_{1,0}(0, 0)$.

Error reduction in PCP

A PCP verifier with soundness $1/2$ that uses r coins and makes q queries can be converted into a PCP verifier using $c \cdot r$ coins and $c \cdot q$ queries with soundness $1/2^c$ by just repeating its execution c times

The PCP Theorem

PCP Theorem

NP $=$ PCP($\log n$, 1)

Hardness of Approximation Using PCP Theorem

Consider the generalization of the MAX-3SAT problem and its corresponding GAP problem

Constraint satisfaction problems (CSP)

A qCSP instance φ is a collection of boolean constraints $\varphi_1, \ldots, \varphi_m$ such that each φ_i depends on at most q of the input variables
Let $\mathrm{val}(\varphi)$ denote the maximum fraction of constraints that can be satisfied by any assignment

Hardness of Approximation Using PCP Theorem

Consider the generalization of the MAX-3SAT problem and its corresponding GAP problem

Gap CSP

gap_α-qCSP is the promise problem whose (YES, NO) are as follows:

$$\text{YES} = \{\varphi \mid \text{val}(\varphi) = 1\}$$
$$\text{NO} = \{\varphi \mid \text{val}(\varphi) < \alpha\}$$

gap_α-qCSP is NP-hard

Theorem

$\exists q \in \mathbb{N}, \alpha \in (0,1)$: gap_α-qCSP is NP-hard

gap$_\alpha$-qCSP is NP-hard

Proof

- $3\text{SAT} \in \text{NP} \implies 3\text{SAT} \in \text{PCP}(\log n, 1)$

gap$_\alpha$-qCSP is NP-hard

Proof

- 3SAT $\in$ NP $\implies$ 3SAT $\in$ PCP(log n, 1)
- Let $\mathcal{V}$ be the PCP($c\cdot$log n, d) verifier for 3SAT

gap$_\alpha$-qCSP is NP-hard

Proof

- 3SAT $\in$ NP $\implies$ 3SAT $\in$ PCP(log n, 1)
- Let $\mathcal{V}$ be the PCP($c \cdot \log n$, d) verifier for 3SAT
- Let $\mathcal{V}_{x,r}$ be a boolean constraint which evaluates to 1 iff $\mathcal{V}$ accepts π, for input 3SAT instance x and r coins

gap_α-qCSP is NP-hard

Proof

- 3SAT $\in$ NP $\implies$ 3SAT $\in$ PCP($\log n$, 1)
- Let $\mathcal{V}$ be the PCP($c{\cdot}\log n$, d) verifier for 3SAT
- Let $\mathcal{V}_{x,r}$ be a boolean constraint which evaluates to 1 iff $\mathcal{V}$ accepts π, for input 3SAT instance x and r coins
- $\mathcal{V}_{x,r}$ is a formula on d variables which correspond to the queries

gap$_\alpha$-qCSP is NP-hard

Proof

- 3SAT $\in$ NP $\implies$ 3SAT $\in$ PCP($\log n$, 1)
- Let $\mathcal{V}$ be the PCP($c{\cdot}\log n$, d) verifier for 3SAT
- Let $\mathcal{V}_{x,r}$ be a boolean constraint which evaluates to 1 iff $\mathcal{V}$ accepts π, for input 3SAT instance x and r coins
- $\mathcal{V}_{x,r}$ is a formula on d variables which correspond to the queries
- $\mathcal{V}_{x,r}$ is poly-sized since $\mathcal{V}$ is a polynomial verifier

gap_α-qCSP is NP-hard

Proof

- 3SAT $\in$ NP $\implies$ 3SAT $\in$ PCP($\log n$, 1)
- Let $\mathcal{V}$ be the PCP($c{\cdot}\log n$, d) verifier for 3SAT
- Let $\mathcal{V}_{x,r}$ be a boolean constraint which evaluates to 1 iff $\mathcal{V}$ accepts π, for input 3SAT instance x and r coins
- $\mathcal{V}_{x,r}$ is a formula on d variables which correspond to the queries
- $\mathcal{V}_{x,r}$ is poly-sized since $\mathcal{V}$ is a polynomial verifier
- So for any x, we construct the dCSP instance $\varphi = \{\mathcal{V}_{x,r}\}_{r \in \{0,1\}^{c \cdot logn}}$

Hardness of Approximation Using PCP Theorem

Proof

- Since $r \in \{0,1\}^{c \cdot logn}$, φ can be constructed in polynomial time

Hardness of Approximation Using PCP Theorem

Proof

- Since $r \in \{0,1\}^{c \cdot logn}$, φ can be constructed in polynomial time
- $x \in \text{3SAT} \implies \text{val}(\varphi) = 1$

Hardness of Approximation Using PCP Theorem

Proof

- Since $r \in \{0,1\}^{c \cdot logn}$, φ can be constructed in polynomial time
- $x \in \text{3SAT} \implies \text{val}(\varphi) = 1$
- $x \notin \text{3SAT} \implies \text{val}(\varphi) \leq 1/2$

Hardness of Approximation Using PCP Theorem

Proof

- Since $r \in \{0,1\}^{c \cdot logn}$, φ can be constructed in polynomial time
- $x \in \text{3SAT} \implies \text{val}(\varphi) = 1$
- $x \notin \text{3SAT} \implies \text{val}(\varphi) \leq 1/2$
- Hence, φ is a valid instance for the promise problem $\text{gap}_{1/2}\text{-}d\text{CSP}$

Hardness of Approximation Using PCP Theorem

Proof

- Since $r \in \{0,1\}^{c \cdot logn}$, φ can be constructed in polynomial time
- $x \in 3SAT \implies \text{val}(\varphi) = 1$
- $x \notin 3SAT \implies \text{val}(\varphi) \leq 1/2$
- Hence, φ is a valid instance for the promise problem $\text{gap}_{1/2}$-dCSP
- $3SAT \leq_P \text{gap}_{1/2}$-$d$CSP

PCP Theorem Using Hardness of Approximation

We saw that PCP Theorem $\implies$ Gap problem is hard

PCP Theorem Using Hardness of Approximation

We saw that PCP Theorem $\implies$ Gap problem is hard

What about the converse?

PCP Theorem Using Hardness of Approximation

We saw that PCP Theorem $\implies$ Gap problem is hard

What about the converse?

It turns out the converse is also true i.e. hardness of the Gap problem is an alternate view of the PCP theorem

PCP Theorem Using Hardness of Approximation

- We have that $\text{gap}_{1/2}$-qCSP is NP-hard for some q

PCP Theorem Using Hardness of Approximation

- We have that $\text{gap}_{1/2}$-qCSP is NP-hard for some q
- PCP Theorem: $\text{NP} = \text{PCP}(\log n, 1)$

PCP Theorem Using Hardness of Approximation

- We have that $\text{gap}_{1/2}$-qCSP is NP-hard for some q
- PCP Theorem: $\text{NP} = \text{PCP}(\log n, 1)$
- $\text{PCP}(\log n, 1) \subseteq \text{NP}$

PCP Theorem Using Hardness of Approximation

- We have that $\text{gap}_{1/2}$-qCSP is NP-hard for some q
- PCP Theorem: NP $=$ PCP($\log n$, 1)
- PCP($\log n$, 1) $\subseteq$ NP
- If we show that NP $\subseteq$ PCP($\log n$, 1), we are done

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time
- The PCP Verifier($\mathcal{V}$) for φ_x expects a satisfying assignment to the variables from the prover

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time
- The PCP Verifier($\mathcal{V}$) for φ_x expects a satisfying assignment to the variables from the prover
- It then picks a constraint φ_i, at random from φ_x

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time
- The PCP Verifier($\mathcal{V}$) for φ_x expects a satisfying assignment to the variables from the prover
- It then picks a constraint φ_i, at random from φ_x
- This can be done with $O(\log n)$ random bits since φ_x is poly-sized

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time
- The PCP Verifier($\mathcal{V}$) for φ_x expects a satisfying assignment to the variables from the prover
- It then picks a constraint φ_i, at random from φ_x
- This can be done with $O(\log n)$ random bits since φ_x is poly-sized
- Then it queries the assignment of q variables associated with φ_i

PCP Theorem Using Hardness of Approximation

- $L \in NP \implies L \leq_P \text{gap}_{1/2}\text{-}q\text{CSP}$
- for any input $x \in L$, we construct φ_x in polynomial time
- The PCP Verifier($\mathcal{V}$) for φ_x expects a satisfying assignment to the variables from the prover
- It then picks a constraint φ_i, at random from φ_x
- This can be done with $O(\log n)$ random bits since φ_x is poly-sized
- Then it queries the assignment of q variables associated with φ_i
- $\mathcal{V}$ accepts iff φ_i is satisfied

PCP Theorem Using Hardness of Approximation

- $x \in L \implies val(\varphi) = 1$, hence $\mathcal{V}$ will accept with probability 1

PCP Theorem Using Hardness of Approximation

- $x \in L \implies val(\varphi) = 1$, hence $\mathcal{V}$ will accept with probability 1
- $x \notin L \implies val(\varphi) \leq 1/2$, hence $\mathcal{V}$ will accept with probability $\leq 1/2$

Two views of the PCP theorem

Proof View	**Hardness of Approximation View**
PCP Veri er ($\mathcal{V}$)	CSP Instance (φ)
PCP proof (π)	Assignment to variables ($\mathbf{u}$)
Length of proof	Number of variables (n)
Number of queries (q)	Arity of constraints (q)
Number of random bits (r)	Logarithm of number of constraints ($\log m$)
Soundness parameter (typically 1/2)	Maximum of val(φ) for a NO instance
NP $\subseteq$ PCP($\log n$, 1)	$\text{gap}_{1/2}$-qCSP is NP-hard

2-query PCP

Consider a slightly different definition of the PCP classes

Definition

$\text{PCP}^{\Sigma}_{c,s}(r,q)$ is the class of languages that have restricted verifiers that use r random bits, q queries to the proof $\pi \in \Sigma^*$ with

Completeness $x \in L \implies \exists \pi,\ Pr[\mathcal{V}^{\pi}(x) = 1] \geq c$

Soundness $x \notin L \implies \forall \pi,\ Pr[\mathcal{V}^{\pi}(x) = 1] \leq s$

2-query PCP

Theorem

$PCP^{\Sigma}_{c,1-\epsilon}[r,q] \subseteq PCP^{\Sigma}_{c,1-\epsilon/q}[r + log\ q, 2]$

Proof.

- $L \in PCP^{\Sigma}_{c,1-\epsilon}[r,q] \implies L$ has a (r,q)-verifier $\mathcal{V}$
- for $x \in L$, $\exists \pi$ such that $\mathcal{V}$ accepts with probability $\geq$ c
- for $x \notin L$, $\forall \pi$, $\mathcal{V}$ accepts with probability $\leq$ s
- Define $Perm(\pi) : [m]^q \to \Sigma^q$ st $Perm(\pi)(i_1, \ldots, i_q) = \pi(i_1)\pi(i_2)\ldots\pi(i_q)$

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi \rangle$ as proof

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle \mathit{Perm}(\pi), \pi \rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi \rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$
- $\mathcal{V}'$ then guesses the indices of queries $(i_1, \ldots i_q)$ as $\mathcal{V}$ would using r coins

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi \rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$
- $\mathcal{V}'$ then guesses the indices of queries $(i_1, \ldots i_q)$ as $\mathcal{V}$ would using r coins
- $\mathcal{V}'$ queries $\pi_1(i_1, \ldots i_q)$ and proceeds as $\mathcal{V}$ would

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi \rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$
- $\mathcal{V}'$ then guesses the indices of queries $(i_1, \ldots i_q)$ as $\mathcal{V}$ would using r coins
- $\mathcal{V}'$ queries $\pi_1(i_1, \ldots i_q)$ and proceeds as $\mathcal{V}$ would
- If $\mathcal{V}$ would have rejected then $\mathcal{V}'$ will reject

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi \rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$
- $\mathcal{V}'$ then guesses the indices of queries $(i_1, \ldots i_q)$ as $\mathcal{V}$ would using r coins
- $\mathcal{V}'$ queries $\pi_1(i_1, \ldots i_q)$ and proceeds as $\mathcal{V}$ would
- If $\mathcal{V}$ would have rejected then $\mathcal{V}'$ will reject
- Otherwise, $\mathcal{V}'$ selects $j \in [q]$ using $\log q$ coins

2-query PCP

- Consider a verifier $\mathcal{V}'$ that expects $\langle Perm(\pi), \pi\rangle$ as proof
- The prover actually sends $\langle \pi_1, \pi_2 \rangle$
- $\mathcal{V}'$ then guesses the indices of queries $(i_1, \dots i_q)$ as $\mathcal{V}$ would using r coins
- $\mathcal{V}'$ queries $\pi_1(i_1, \dots i_q)$ and proceeds as $\mathcal{V}$ would
- If $\mathcal{V}$ would have rejected then $\mathcal{V}'$ will reject
- Otherwise, $\mathcal{V}'$ selects $j \in [q]$ using $\log q$ coins
- $\mathcal{V}'$ accepts iff jth symbol of $\pi_1(i_1, \dots i_q) = \pi_2(i_j)$

2-query PCP

$x \in L \Rightarrow$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$
- At least *epsilon* fraction of outcomes $\mathcal{V}'_{x,r}(\pi)$ are rejections

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$
- At least *epsilon* fraction of outcomes $\mathcal{V}'_{x,r}(\pi)$ are rejections
- In any proof π_1, π_2, π_1 need not necessarily be $Perm(\pi_2)$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$
- At least *epsilon* fraction of outcomes $\mathcal{V}'_{x,r}(\pi)$ are rejections
- In any proof π_1, π_2, π_1 need not necessarily be $Perm(\pi_2)$
- If $\pi_1(i_1, \ldots, i_q) = \pi_1(i_1, \ldots, i_q)$, for *epsilon* fraction of the runs, it is rejected

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$
- At least *epsilon* fraction of outcomes $\mathcal{V}'_{x,r}(\pi)$ are rejections
- In any proof π_1, π_2, π_1 need not necessarily be $Perm(\pi_2)$
- If $\pi_1(i_1, \ldots, i_q) = \pi_1(i_1, \ldots, i_q)$, for *epsilon* fraction of the runs, it is rejected
- Otherwise, $\pi_1(i_1, \ldots, i_q)$ differs from $\pi_1(i_1, \ldots, i_q)$ in at least one position, and the probability of $\mathcal{V}'$ rejecting it is $1/q$

2-query PCP

$x \in L \Rightarrow$

- $\exists \pi$ that $\mathcal{V}$ accepts with probability $\geq c$
- $\langle Perm(\pi), \pi \rangle$ is a proof that $\mathcal{V}'$ accepts with probability $\geq c$

$x \notin L \Rightarrow$

- For any π, $\mathcal{V}$ rejects with probability $\geq \epsilon$
- At least *epsilon* fraction of outcomes $\mathcal{V}'_{x,r}(\pi)$ are rejections
- In any proof π_1, π_2, π_1 need not necessarily be $Perm(\pi_2)$
- If $\pi_1(i_1, \ldots, i_q) = \pi_1(i_1, \ldots, i_q)$, for *epsilon* fraction of the runs, it is rejected
- Otherwise, $\pi_1(i_1, \ldots, i_q)$ differs from $\pi_1(i_1, \ldots, i_q)$ in at least one position, and the probability of $\mathcal{V}'$ rejecting it is $1/q$
- Hence, overall test rejects with probability at least ϵ/q

Summary

Thank you!