PCP Theorem

Naman Gupta

June 22, 2021

Overview

- Motivation: Approximate Solutions to NP-hard problems
- PCP Theorem: Equivalence of two views
- Answer Reduction: Preliminaries and Intuition
- Succinct Representation of Deciders
- PCP for normal form deciders

Overview

- Motivation: Approximate Solutions to NP-hard problems
- PCP Theorem: Equivalence of two views
- Answer Reduction: Preliminaries and Intuition
- Succinct Representation of Deciders
- PCP for normal form deciders

Overview

- Motivation: Approximate Solutions to NP-hard problems
- PCP Theorem: Equivalence of two views
- Answer Reduction: Preliminaries and Intuition
- Succinct Representation of Deciders
- PCP for normal form deciders

Overview

- Motivation: Approximate Solutions to NP-hard problems
- PCP Theorem: Equivalence of two views
- Answer Reduction: Preliminaries and Intuition
- Succinct Representation of Deciders
- PCP for normal form deciders

Overview

- Motivation: Approximate Solutions to NP-hard problems
- PCP Theorem: Equivalence of two views
- Answer Reduction: Preliminaries and Intuition
- Succinct Representation of Deciders
- PCP for normal form deciders

Motivation: Approximate Solutions to NP-hard problems

Let MAX-3SAT be the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.

- MAX-3SAT is NP-hard.

We define an approximation of MAX-3SAT as follows –
For every 3CNF formula φ let $val(\varphi)$ be the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

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

- $1/2$ Approximation for MAX-3SAT

Motivation: Approximate Solutions to NP-hard problems

Let MAX-3SAT be the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.

- MAX-3SAT is NP-hard.

We define an approximation of MAX-3SAT as follows –
For every 3CNF formula φ let $val(\varphi)$ be the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

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

- $1/2$ Approximation for MAX-3SAT

Motivation: Approximate Solutions to NP-hard problems

Let MAX-3SAT be the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.

- MAX-3SAT is NP-hard.

We define an approximation of MAX-3SAT as follows –
For every 3CNF formula φ let $val(\varphi)$ be the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

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

- $1/2$ Approximation for MAX-3SAT

Motivation: Approximate Solutions to NP-hard problems

Let MAX-3SAT be the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.

- MAX-3SAT is NP-hard.

We define an approximation of MAX-3SAT as follows –
For every 3CNF formula φ let $val(\varphi)$ be the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

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

- 1/2 Approximation for MAX-3SAT

Motivation: Approximate Solutions to NP-hard problems

Let MAX-3SAT be the problem of finding, given a 3CNF Boolean formula φ as input, an assignment that maximizes the number of satisfied clauses.

- MAX-3SAT is NP-hard.

We define an approximation of MAX-3SAT as follows –
For every 3CNF formula φ let $val(\varphi)$ be the maximum fraction of clauses that can be satisfied by any assignment to φ's variables.

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

- $1/2$ Approximation for MAX-3SAT

PCP Verifier: Intuition

- Recall the definition of NP languages
- L $\in$ NP if there is a polynomial time Turing Machine (verifier) V s.t.,

$$x \in L \implies \exists \pi \text{ s.t. } V^{\pi}(x) = 1$$
$$x \notin L \implies \forall \pi \quad V^{\pi}(x) = 0$$

- Probabilistic verification of certificates
- Probabilistically Checkable Proofs

PCP Verifier: Intuition

- Recall the definition of NP languages
- L $\in$ NP if there is a polynomial time Turing Machine (verifier) V s.t.,

$$x \in L \implies \exists \pi \text{ s.t. } V^{\pi}(x) = 1$$
$$x \notin L \implies \forall \pi \quad V^{\pi}(x) = 0$$

- Probabilistic verification of certificates
- Probabilistically Checkable Proofs

PCP Verifier: Intuition

- Recall the definition of NP languages
- L $\in$ NP if there is a polynomial time Turing Machine (verifier) V s.t.,

$$x \in L \implies \exists \pi \text{ s.t. } V^{\pi}(x) = 1$$
$$x \notin L \implies \forall \pi \quad V^{\pi}(x) = 0$$

- Probabilistic verification of certificates
- Probabilistically Checkable Proofs

PCP Verifier: Intuition

- Recall the definition of NP languages
- L $\in$ NP if there is a polynomial time Turing Machine (verifier) V s.t.,

$$x \in L \implies \exists \pi \text{ s.t. } V^{\pi}(x) = 1$$
$$x \notin L \implies \forall \pi \quad V^{\pi}(x) = 0$$

- Probabilistic verification of certificates
- Probabilistically Checkable Proofs

PCP Verifier: Definition

- Probabilistic Verifier having random access to the proof string π
- Verifiers can be adaptive or nonadaptive

L has an $(r(n), q(n))$-PCP verifier if there's a polynomial-time probabilistic algorithm V satisfying:

- **Efficiency** At most $q(n)$ nonadaptive queries to a string $\pi \in \{0,1\}^*$ using $r(n)$ length random strings as query address, such that V outputs 1 (accept) or 0 (reject).
- **Completeness** $x \in L \implies \exists \pi$ s.t. $\Pr[V^{\pi}(x) = 1] = 1$
- **Soundness** $x \notin L \implies \forall \pi \Pr[V^{\pi}(x) = 1] \leq 1/2$
  $1/2$ is arbitrary and can be any constant < 1

L is in PCP$(r(n), q(n))$ if there are $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP Verifier.

Example: The language GNI of pairs of nonisomorphic graphs is in **PCP**$(poly(n), 1)$

PCP Verifier: Definition

- Probabilistic Verifier having random access to the proof string π
- Verifiers can be adaptive or nonadaptive

L has an $(r(n), q(n))$-PCP verifier if there's a polynomial-time probabilistic algorithm V satisfying:

- **Efficiency** At most $q(n)$ nonadaptive queries to a string $\pi \in \{0,1\}^*$ using $r(n)$ length random strings as query address, such that V outputs 1 (accept) or 0 (reject).
- **Completeness** $x \in L \implies \exists \pi$ s.t. $\Pr[V^\pi(x) = 1] = 1$
- **Soundness** $x \notin L \implies \forall \pi \Pr[V^\pi(x) = 1] \leq 1/2$
  1/2 is arbitrary and can be any constant < 1

L is in PCP$(r(n), q(n))$ if there are $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP Verifier.

Example: The language GNI of pairs of nonisomorphic graphs is in **PCP**$(poly(n), 1)$

PCP Verifier: Definition

- Probabilistic Verifier having random access to the proof string π
- Verifiers can be adaptive or nonadaptive

L has an $(r(n), q(n))$-PCP verifier if there's a polynomial-time probabilistic algorithm V satisfying:

- **Efficiency** At most $q(n)$ nonadaptive queries to a string $\pi \in \{0,1\}^*$ using $r(n)$ length random strings as query address, such that V outputs 1 (accept) or 0 (reject).
- **Completeness** $x \in L \implies \exists \pi$ s.t. $\Pr[V^\pi(x) = 1] = 1$
- **Soundness** $x \notin L \implies \forall \pi \Pr[V^\pi(x) = 1] \leq 1/2$
  1/2 is arbitrary and can be any constant < 1

L is in PCP$(r(n), q(n))$ if there are $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP Verifier.

Example: The language GNI of pairs of nonisomorphic graphs is in **PCP**$(poly(n), 1)$

PCP Verifier: Definition

- Probabilistic Verifier having random access to the proof string π
- Verifiers can be adaptive or nonadaptive

L has an $(r(n), q(n))$-PCP verifier if there's a polynomial-time probabilistic algorithm V satisfying:

- **Efficiency** At most $q(n)$ nonadaptive queries to a string $\pi \in \{0,1\}^*$ using $r(n)$ length random strings as query address, such that V outputs 1 (accept) or 0 (reject).
- **Completeness** $x \in L \implies \exists \pi$ s.t. $\Pr[V^\pi(x) = 1] = 1$
- **Soundness** $x \notin L \implies \forall \pi \Pr[V^\pi(x) = 1] \leq 1/2$
  1/2 is arbitrary and can be any constant < 1

L is in PCP$(r(n), q(n))$ if there are $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP Verifier.

Example: The language GNI of pairs of nonisomorphic graphs is in **PCP**$(poly(n), 1)$

PCP Verifier: Definition

- Probabilistic Verifier having random access to the proof string π
- Verifiers can be adaptive or nonadaptive

L has an $(r(n), q(n))$-PCP verifier if there's a polynomial-time probabilistic algorithm V satisfying:

- **Efficiency** At most $q(n)$ nonadaptive queries to a string $\pi \in \{0,1\}^*$ using $r(n)$ length random strings as query address, such that V outputs 1 (accept) or 0 (reject).
- **Completeness** $x \in L \implies \exists \pi$ s.t. $\Pr[V^\pi(x) = 1] = 1$
- **Soundness** $x \notin L \implies \forall \pi \Pr[V^\pi(x) = 1] \leq 1/2$
  1/2 is arbitrary and can be any constant < 1

L is in PCP$(r(n), q(n))$ if there are $c, d > 0$ such that L has a $(cr(n), dq(n))$-PCP Verifier.
Example: The language GNI of pairs of nonisomorphic graphs is in **PCP**$(poly(n), 1)$

PCP Theorem

- $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$: Constant number of random queries to verify the certificate for a NP language

PCP Theorem: Hardness of Approximation View

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

$$x \in L \Rightarrow \textsf{val}(f(x)) = 1$$
$$x \notin L \Rightarrow \textsf{val}(f(x)) < \rho$$

Corollary – There exists some constant $\rho < 1$ such that if there is a polynomial time ρ-approximation algorithm for MAX-3SAT, then **P=NP**.

PCP Theorem: Hardness of Approximation View

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

$$x \in L \Rightarrow \ \mathsf{val}(f(x)) = 1$$
$$x \notin L \Rightarrow \ \mathsf{val}(f(x)) < \rho$$

Corollary – There exists some constant $\rho < 1$ such that if there is a polynomial time ρ-approximation algorithm for MAX-3SAT, then **P=NP**.

Answer Reduction: Preliminaries

Some notations and definitions to begin with –

- Let $V = \mathbb{F}^n$ be a vector space. A register subspace S of V is a subspace that is the span of the standard basis of V.
- Two subspaces are complimentary if $S \cap T = \{0\}, S + T = V$
- Let $x \in V$. x^S denotes the projection of x onto the subspace S parallel to T.

Answer Reduction: Preliminaries

Some notations and definitions to begin with –

- Let $V = \mathbb{F}^n$ be a vector space. A register subspace S of V is a subspace that is the span of the standard basis of V.
- Two subspaces are complimentary if $S \cap T = \{0\}, S + T = V$
- Let $x \in V$. x^S denotes the projection of x onto the subspace S parallel to T.

Answer Reduction: Preliminaries

Some notations and definitions to begin with –

- Let $V = \mathbb{F}^n$ be a vector space. A register subspace S of V is a subspace that is the span of the standard basis of V.
- Two subspaces are complimentary if $S \cap T = \{0\}, S + T = V$
- Let $x \in V$. x^S denotes the projection of x onto the subspace S parallel to T.

Answer Reduction: Preliminaries

Conditionally Linear Functions –

Intuitively, a conditionally linear function (L) takes as input an element $x \in V = \mathbb{F}^n$ for some $n \geq 0$, and applies linear maps L_j sequentially on x^{V_j} where $V_1, V_2, \ldots$ are a sequence of complementary register subspaces such that both the linear maps L_j and the subspace V_j may depend on the values taken by previous linear maps $L_1(x^{V_1}), L_2(x^{V_2})$, etc.

Let x^{L_1} denote $L_1(x^{V_1})$.

- k-th marginal of L – $L_{\leq k} : V \to V = \sum_{i=1}^{k} x^{L_i}$
- k-th factor space with prefix $u \in L_{<k}(V)$ – $V_{k,u}$
- k-th linear map $L_{k,u} : V_{k,u} \to V_{k,u}$ with prefix u

Answer Reduction: Preliminaries

Conditionally Linear Functions –
Intuitively, a conditionally linear function (L) takes as input an element $x \in V = \mathbb{F}^n$ for some $n \geq 0$, and applies linear maps L_j sequentially on x^{V_j} where $V_1, V_2, \ldots$ are a sequence of complementary register subspaces such that both the linear maps L_j and the subspace V_j may depend on the values taken by previous linear maps $L_1(x^{V_1}), L_2(x^{V_2})$, etc.
Let x^{L_1} denote $L_1(x^{V_1})$.

- k-th marginal of L – $L_{\leq k} : V \to V = \sum_{i=1}^{k} x^{L_i}$
- k-th factor space with prefix $u \in L_{<k}(V)$ – $V_{k,u}$
- k-th linear map $L_{k,u} : V_{k,u} \to V_{k,u}$ with prefix u

Answer Reduction: Preliminaries

Conditionally Linear Functions –
Intuitively, a conditionally linear function (L) takes as input an element $x \in V = \mathbb{F}^n$ for some $n \geq 0$, and applies linear maps L_j sequentially on x^{V_j} where $V_1, V_2, \ldots$ are a sequence of complementary register subspaces such that both the linear maps L_j and the subspace V_j may depend on the values taken by previous linear maps $L_1(x^{V_1}), L_2(x^{V_2})$, etc.
Let x^{L_1} denote $L_1(x^{V_1})$.

- k-th marginal of L – $L_{\leq k} : V \to V = \sum_{i=1}^{k} x^{L_i}$
- k-th factor space with prefix $u \in L_{<k}(V)$ – $V_{k,u}$
- k-th linear map $L_{k,u} : V_{k,u} \to V_{k,u}$ with prefix u

Answer Reduction: Preliminaries

CL Samplers – Turing Machines that perform computations corresponding to CL functions.
A function $q : \mathbb{N} \to \mathbb{N}$ is an admissible field size function if $\forall n \; q(n)$ is of the form 2^d where d is odd.

A 6-input TM $\mathcal{S}$ is a l-level CL Sampler with field size $q(n)$ and dimenstion $s(n)$ if for all n there exist l-level CL functions satisfying certain conditions, such that $\mathcal{S}$ can output the binary representation of marginal, factor space and linear maps.

Answer Reduction: Preliminaries

CL Samplers – Turing Machines that perform computations corresponding to CL functions.
A function $q : \mathbb{N} \to \mathbb{N}$ is an admissible field size function if $\forall n \; q(n)$ is of the form 2^d where d is odd.

A 6-input TM $\mathcal{S}$ is a l-level CL Sampler with field size $q(n)$ and dimenstion $s(n)$ if for all n there exist l-level CL functions satisfying certain conditions, such that $\mathcal{S}$ can output the binary representation of marginal, factor space and linear maps.

Answer Reduction: Preliminaries

Decider – A 5-input TM $\mathcal{D}$ that on all inputs of the form (n, x, y, a, b) where $n \in \mathbb{N}$ and $x, y, a, b \in \{0,1\}^*$ halts and returns a single bit.

Normal Form Verifier – $\mathcal{V} = (\mathcal{S}, \mathcal{D})$

Answer Reduction: Preliminaries

Decider – A 5-input TM $\mathcal{D}$ that on all inputs of the form (n, x, y, a, b) where $n \in \mathbb{N}$ and $x, y, a, b \in \{0, 1\}^*$ halts and returns a single bit.

Normal Form Verifier – $\mathcal{V} = (\mathcal{S}, \mathcal{D})$

Answer Reduction: Intuition

- We transform a normal form verifier $\mathcal{V}$ into an answer reduced verifier $\mathcal{V}^{AR} = (\mathcal{S}^{AR}, \mathcal{D}^{AR})$ such that the answer reduced decider is only polylog in the answer length. This is achieved by composing a PCP.
- Similar to Interactive Proofs, the verifier samples questions x and y and distributes them to players which return answers a and b. The decider than takes the questions and answers as input and outputs a single bit. The answer reduced verifier asks the prover to provide a certificate for $\mathcal{D}(N, x, y, a, b)$, then executes a PCP Verifier after sampling part of that certificate Π.
- The original decider runs in poly(N), but due to the PCP formulation the decision can be executed in poly($n = \log N$).

Answer Reduction: Intuition

- We transform a normal form verifier $\mathcal{V}$ into an answer reduced verifier $\mathcal{V}^{AR} = (\mathcal{S}^{AR}, \mathcal{D}^{AR})$ such that the answer reduced decider is only polylog in the answer length. This is achieved by composing a PCP.
- Similar to Interactive Proofs, the verifier samples questions x and y and distributes them to players which return answers a and b. The decider than takes the questions and answers as input and outputs a single bit. The answer reduced verifier asks the prover to provide a certificate for $\mathcal{D}(N, x, y, a, b)$, then executes a PCP Verifier after sampling part of that certificate Π.
- The original decider runs in poly(N), but due to the PCP formulation the decision can be executed in poly($n = \log N$).

Answer Reduction: Intuition

- We transform a normal form verifier $\mathcal{V}$ into an answer reduced verifier $\mathcal{V}^{AR} = (\mathcal{S}^{AR}, \mathcal{D}^{AR})$ such that the answer reduced decider is only polylog in the answer length. This is achieved by composing a PCP.
- Similar to Interactive Proofs, the verifier samples questions x and y and distributes them to players which return answers a and b. The decider than takes the questions and answers as input and outputs a single bit. The answer reduced verifier asks the prover to provide a certificate for $\mathcal{D}(N, x, y, a, b)$, then executes a PCP Verifier after sampling part of that certificate Π.
- The original decider runs in poly(N), but due to the PCP formulation the decision can be executed in poly($n = \log N$).

Succinct representation of Deciders

Succinct description of 3SAT formulas

Let $N = 2^n$, and let φ be a 3 SAT formula on N variables named $x_0, \ldots, x_{N-1}$. Let $\mathcal{C}$ be a Boolean circuit with 3 inputs of length n and three single-bit inputs. Then $\mathcal{C}$ is a succinct description of φ if for each $i_1, i_2, i_3 \in \{0, 1, \ldots, N-1\}$ and $o_1, o_2, o_3 \in \{0, 1\}$,

$$\mathcal{C}\left(i_1, i_2, i_3, o_1, o_2, o_3\right) = 1$$

if and only if $x_{i_1}^{o_1} \vee x_{i_2}^{o_2} \vee x_{i_2}^{o_3}$ is a clause in φ.
Here x^o denotes x if $o = 1$, $\neg x$ otherwise.

Succinct representation of Deciders

Informally, a decider $\mathcal{D}$ can be converted into a circuit $\mathcal{C}$ which represents a 3SAT formula φ which carries out the time T computation of $\mathcal{D}$. Additionally, $\mathcal{C}$ is of size poly $\log(T)$ rather than poly(T).

Decider Encodings – Encode tape alphabet $= \{0, 1, \sqcup\}$ and set of states K into a binary representation.

$$enc(0, 1, \sqcup, \square) = (00, 01, 10, 11)$$

Succinct representation of Deciders

Informally, a decider $\mathcal{D}$ can be converted into a circuit $\mathcal{C}$ which represents a 3SAT formula φ which carries out the time T computation of $\mathcal{D}$. Additionally, $\mathcal{C}$ is of size poly log(T) rather than poly(T).

Decider Encodings – Encode tape alphabet $= \{0, 1, \sqcup\}$ and set of states K into a binary representation.

$$enc(0, 1, \sqcup, \Box) = (00, 01, 10, 11)$$

Succinct representation of Deciders

Proposition – Let $\mathcal{D}$ be a decider, let n, T, Q, and σ be integers with $Q \leq T$ and $|\mathcal{D}| \leq \sigma$, and let x and y be strings of length at most Q. Then on input $(\mathcal{D}, n, T, Q, \sigma, x, y)$, there is an algorithm that outputs a circuit $\mathcal{C}$ on $3m + 3$ inputs which succinctly describes a 3SAT formula φ_{3SAT} on $M = 2^m$ variables.

φ_{3SAT} has the following property:

- For all $a, b \in \{0,1\}^{2T}$, there exists a $c \in \{0,1\}^{M-4T}$ such that $w = (a, b, c)$ satisfies φ_{3SAT} if and only if there exist $a_{\text{prefix}}, b_{\text{prefix}} \in \{0,1\}^*$ of lengths $\ell_a, \ell_b \leq T$, respectively, such that

$$a = \mathrm{enc}_\Gamma\left(a_{\text{prefix}}, \sqcup^{T-\ell_a}\right) \quad \text{and} \quad b = \mathrm{enc}_\Gamma\left(b_{\text{prefix}}, \sqcup^{T-\ell_b}\right)$$

  and $\mathcal{D}$ accepts $\left(n, x, y, a_{\text{prefix}}, b_{\text{prefix}}\right)$ in time T.

Succinct representation of Deciders

- $m = O(\log T + \log \sigma)$
- $\mathcal{C}$ has $poly(\log n, \log T, Q, \sigma)$ gates
- The algorithm runs in time $poly(\log n, \log T, Q, \sigma)$

Succinct representation of Deciders

- $m = O(\log T + \log \sigma)$
- $\mathcal{C}$ has $poly(\log n, \log T, Q, \sigma)$ gates
- The algorithm runs in time $poly(\log n, \log T, Q, \sigma)$

Succinct representation of Deciders

- $m = O(\log T + \log \sigma)$
- $\mathcal{C}$ has $poly(\log n, \log T, Q, \sigma)$ gates
- The algorithm runs in time $poly(\log n, \log T, Q, \sigma)$

Succinct representation of Deciders

Proof Sketch –

- Consider the execution table of $\mathcal{D}$, that is, record the position of tape heads, values in all tape cells, decider state at every time step $t \leq T$
- As in proof of Cook Levin Theorem, construct a 3SAT formula which defines the execution and boundary constraints ($\varphi_{\text{boundary}}$) for $\mathcal{D}$.
- Define a local check circuit $\mathcal{C}_{\text{Check}}$ having size $poly(|\mathcal{D}|)$ that essentially checks an entry (row) from the execution table. Convert this circuit to corresponding 3SAT formulas.
- Obtain a final 3SAT reduction after considering every check circuit, $\varphi := \varphi_{\text{boundary}} \wedge \left(\bigwedge_i \varphi_{\text{Check}}\right)$
- Construct the final circuit $\mathcal{C}$ which succinctly describes φ

Succinct representation of Deciders

Proof Sketch –

- Consider the execution table of $\mathcal{D}$, that is, record the position of tape heads, values in all tape cells, decider state at every time step $t \leq T$
- As in proof of Cook Levin Theorem, construct a 3SAT formula which defines the execution and boundary constraints ($\varphi_{\text{boundary}}$) for $\mathcal{D}$.
- Define a local check circuit $\mathcal{C}_{\text{Check}}$ having size $poly(|\mathcal{D}|)$ that essentially checks an entry (row) from the execution table. Convert this circuit to corresponding 3SAT formulas.
- Obtain a final 3SAT reduction after considering every check circuit, $\varphi := \varphi_{\text{boundary}} \wedge (\wedge_i \varphi_{\text{Check}})$
- Construct the final circuit $\mathcal{C}$ which succinctly describes φ

Succinct representation of Deciders

Proof Sketch –

- Consider the execution table of $\mathcal{D}$, that is, record the position of tape heads, values in all tape cells, decider state at every time step $t \leq T$
- As in proof of Cook Levin Theorem, construct a 3SAT formula which defines the execution and boundary constraints ($\varphi_{\text{boundary}}$) for $\mathcal{D}$.
- Define a local check circuit $\mathcal{C}_{\text{Check}}$ having size $poly(|\mathcal{D}|)$ that essentially checks an entry (row) from the execution table. Convert this circuit to corresponding 3SAT formulas.
- Obtain a final 3SAT reduction after considering every check circuit, $\varphi := \varphi_{\text{boundary}} \wedge \left(\wedge_i \varphi_{\text{Check}}\right)$
- Construct the final circuit $\mathcal{C}$ which succinctly describes φ

Succinct representation of Deciders

Proof Sketch –

- Consider the execution table of $\mathcal{D}$, that is, record the position of tape heads, values in all tape cells, decider state at every time step $t \leq T$
- As in proof of Cook Levin Theorem, construct a 3SAT formula which defines the execution and boundary constraints ($\varphi_{\text{boundary}}$) for $\mathcal{D}$.
- Define a local check circuit $\mathcal{C}_{\text{Check}}$ having size $poly(|\mathcal{D}|)$ that essentially checks an entry (row) from the execution table. Convert this circuit to corresponding 3SAT formulas.
- Obtain a final 3SAT reduction after considering every check circuit, $\varphi := \varphi_{\text{boundary}} \wedge \left(\wedge_i \varphi_{\text{Check}}\right)$
- Construct the final circuit $\mathcal{C}$ which succinctly describes φ

Succinct representation of Deciders

Proof Sketch –

- Consider the execution table of $\mathcal{D}$, that is, record the position of tape heads, values in all tape cells, decider state at every time step $t \leq T$
- As in proof of Cook Levin Theorem, construct a 3SAT formula which defines the execution and boundary constraints ($\varphi_{\text{boundary}}$) for $\mathcal{D}$.
- Define a local check circuit $\mathcal{C}_{\text{Check}}$ having size $poly(|\mathcal{D}|)$ that essentially checks an entry (row) from the execution table. Convert this circuit to corresponding 3SAT formulas.
- Obtain a final 3SAT reduction after considering every check circuit, $\varphi := \varphi_{\text{boundary}} \wedge \left(\wedge_i \varphi_{\text{Check}}\right)$
- Construct the final circuit $\mathcal{C}$ which succinctly describes φ

Decoupled 5SAT

Previous algorithm allows us to convert any decider $\mathcal{D}$ and inputs n, x, y into a 3SAT formula φ_{3SAT} succinctly described by $\mathcal{C}$.

We modify φ_{3SAT} into φ_{5SAT} such that every clause is made up of different assignments $(a_{i_1}^{o_{i_1}} \vee b_{i_2}^{o_{i_2}} \vee w_{1,i_3}^{o_{i_3}} \vee w_{2,i_4}^{o_{i_4}} \vee w_{3,i_5}^{o_{i_5}})$ instead of $(w_{i_1}^{o_{i_1}} \vee w_{i_2}^{o_{i_2}} \vee w_{i_2}^{o_{i_2}})$ such that (a, b, w_1, w_2, w_3) satisfies φ_{5SAT} iff $\mathcal{D}$ accepts (n, x, y, a, b).

Decoupled 5SAT

Previous algorithm allows us to convert any decider $\mathcal{D}$ and inputs n, x, y into a 3SAT formula φ_{3SAT} succinctly described by $\mathcal{C}$.

We modify φ_{3SAT} into φ_{5SAT} such that every clause is made up of different assignments $(a_{i_1}^{o_{i_1}} \vee b_{i_2}^{o_{i_2}} \vee w_{1,i_3}^{o_{i_3}} \vee w_{2,i_4}^{o_{i_4}} \vee w_{3,i_5}^{o_{i_5}})$ instead of $(w_{i_1}^{o_{i_1}} \vee w_{i_2}^{o_{i_2}} \vee w_{i_2}^{o_{i_2}})$ such that (a, b, w_1, w_2, w_3) satisfies φ_{5SAT} iff $\mathcal{D}$ accepts (n, x, y, a, b).

Explicit Succinct 5SAT Description of Deciders

Let $\mathcal{D}, n, T, Q, \sigma, x, y$ be defined as before. There is a TM SuccinctDecider which outputs a circuit $\mathcal{C}$ with two inputs of length $l_0(T)$, three inputs of length $r_0(T, \sigma)$ and five single bit inputs which succinctly describes $\mathcal{D}$ on inputs n, x, y and time T.

$$
\begin{aligned}
l_0 &= O(\log T) \\
r_0 &= O(\log T + \log \sigma) \\
s &= poly(\log T, \log n, Q, \sigma)
\end{aligned}
$$

PaddedSuccinctDecider

Explicit Succinct 5SAT Description of Deciders

Let $\mathcal{D}, n, T, Q, \sigma, x, y$ be defined as before. There is a TM SuccinctDecider which outputs a circuit $\mathcal{C}$ with two inputs of length $l_0(T)$, three inputs of length $r_0(T, \sigma)$ and five single bit inputs which succinctly describes $\mathcal{D}$ on inputs n, x, y and time T.

$$\begin{aligned} l_0 &= O(\log T) \\ r_0 &= O(\log T + \log \sigma) \\ s &= poly(\log T, \log n, Q, \sigma) \end{aligned}$$

PaddedSuccinctDecider

Explicit Succinct 5SAT Description of Deciders

Let $\mathcal{D}, n, T, Q, \sigma, x, y$ be defined as before. There is a TM SuccinctDecider which outputs a circuit $\mathcal{C}$ with two inputs of length $l_0(T)$, three inputs of length $r_0(T, \sigma)$ and five single bit inputs which succinctly describes $\mathcal{D}$ on inputs n, x, y and time T.

$$\begin{aligned} l_0 &= O(\log T) \\ r_0 &= O(\log T + \log \sigma) \\ s &= poly(\log T, \log n, Q, \sigma) \end{aligned}$$

PaddedSuccinctDecider

PCP for Normal Form Deciders

Preliminaries – Polynomials over finite fields

An m-variate polynomial f over $\mathbb{F}_q$ is a function of the form

$$f(x_1, \ldots, x_m) = \sum_{\alpha \in \{0,1,\ldots,q-1\}^m} c_\alpha x_1^{\alpha_1} \ldots x_m^{\alpha_m}$$

Individual degree $d \implies c_\alpha = 0$ if $\alpha_i > d$
Total degree $d \implies c_\alpha = 0$ if $\sum \alpha_i > d$

PCP for Normal Form Deciders

Preliminaries – Polynomials over finite fields

An m-variate polynomial f over $\mathbb{F}_q$ is a function of the form

$$f(x_1, \ldots, x_m) = \sum_{\alpha \in \{0,1,\ldots,q-1\}^m} c_\alpha x_1^{\alpha_1} \ldots x_m^{\alpha_m}$$

Individual degree $d \implies c_\alpha = 0$ if $\alpha_i > d$

Total degree $d \implies c_\alpha = 0$ if $\sum \alpha_i > d$

PCP for Normal Form Deciders

Preliminaries – Polynomials over finite fields

An m-variate polynomial f over $\mathbb{F}_q$ is a function of the form

$$f(x_1, \ldots, x_m) = \sum_{\alpha \in \{0,1,\ldots,q-1\}^m} c_\alpha x_1^{\alpha_1} \ldots x_m^{\alpha_m}$$

Individual degree $d \implies c_\alpha = 0$ if $\alpha_i > d$
Total degree $d \implies c_\alpha = 0$ if $\sum \alpha_i > d$

PCP for Normal Form Deciders

Preliminaries - Low Degree Code

[Schwartz Zippel Lemma] Let $f, g : \mathbb{F}_q^m \to F_q$ be two unequal polynomials with total degree at most d. Then,

$$\Pr_{x \sim \mathbb{F}_q^m}[f(x) = g(x)] \leq d/q$$

This lemma implies that the set of low degree polynomials form an error correcting code called as low-degree code.

PCP for Normal Form Deciders

Preliminaries – Polynomial basis of zero functions

Let $f : \mathbb{F}^n \to \mathbb{F}$ is an individual degree d polynomial such that $f(x) = 0$ for $x \in \{0,1\}^n$. Define $zero : \mathbb{F} \to \mathbb{F}$ as the univariate polynomial $x \to x(1-x)$. Then there exist polynomials $c_1, c_2, \ldots c_n : \mathbb{F}^n \to \mathbb{F}$ such that

$$f(x) = \sum_{i=1}^{n} c_i(x) \cdot zero(x_i)$$

Proof. Polynomial division and induction

PCP for Normal Form Deciders

PCP Verifier $V^{\Pi} : (\mathcal{D}, n, T, Q, \sigma, \gamma, x, y) \to \{0, 1\}$.
It checks if there exists strings $a_{\text{prefix}}, b_{\text{prefix}}$ of length $O(T)$ such that $\mathcal{D}$ halts on $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$

Parameters for the PCP – (q, m, d, m', s) which are computed from the input $(m' = 5m + 5 + s)$.

PCP for Normal Form Deciders

PCP Verifier $V^{\Pi} : (\mathcal{D}, n, T, Q, \sigma, \gamma, x, y) \to \{0, 1\}$.
It checks if there exists strings $a_{\text{prefix}}, b_{\text{prefix}}$ of length $O(T)$ such that $\mathcal{D}$ halts on $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$

Parameters for the PCP – (q, m, d, m', s) which are computed from the input $(m' = 5m + 5 + s)$.

PCP for Normal Form Deciders

Definition – A low-degree PCP proof is a tuple Π of evaluation tables of polynomials $g_1, \ldots g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ having individual degree at most d.

Definition – $z \in \mathbb{F}_q^{m'} = (x_1, \ldots, x_5, o, w)$ where $x_i \in \mathbb{F}_q^m, o \in \mathbb{F}_q^5, w \in \mathbb{F}_q^s$. The evaluation of Π at z is given by

$$z' = eval_z(\Pi) = (g_i(x_i), c_j(z))$$

PCP for Normal Form Deciders

Definition – A low-degree PCP proof is a tuple Π of evaluation tables of polynomials $g_1, \ldots g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ having individual degree at most d.

Definition – $z \in \mathbb{F}_q^{m'} = (x_1, \ldots, x_5, o, w)$ where $x_i \in \mathbb{F}_q^m, o \in \mathbb{F}_q^5, w \in \mathbb{F}_q^s$. The evaluation of Π at z is given by

$$z' = eval_z(\Pi) = (g_i(x_i), c_j(z))$$

PCP for Normal Form Deciders

Theorem – There exists a TM $\mathcal{M}_{\mathsf{AR}}$ with the following properties:

- Input = Decider Specification + PCP View
  Decider Specification = $(\mathcal{D}, n, T, Q, \sigma, \gamma, x, y)$
  PCP View (random strings) = (z, z') where $z \in \mathbb{F}_q^{m'}, z' \in \mathbb{F}_q^{m'+6}$
- Output is either 1(accept) or 0(reject)
- Complexity = poly$(\log T, \log n, Q, \sigma, \gamma)$
- Completeness: Suppose $\mathcal{D}$ halts on strings a_{prefix} , b_{prefix} then there exists a low-degree proof Π which causes $\mathcal{M}$ to accept with probability 1 over a uniformly random z.
- Soundness: An invalid proof is accepted with less than 1/2 probability

PCP for Normal Form Deciders

Theorem – There exists a TM $\mathcal{M}_{\text{AR}}$ with the following properties:

- Input = Decider Specification + PCP View
  Decider Specification = $(\mathcal{D}, n, T, Q, \sigma, \gamma, x, y)$
  PCP View (random strings) = (z, z') where $z \in \mathbb{F}_q^{m'}, z' \in \mathbb{F}_q^{m'+6}$
- Output is either 1(accept) or 0(reject)
- Complexity = poly$(\log T, \log n, Q, \sigma, \gamma)$
- Completeness: Suppose $\mathcal{D}$ halts on strings a_{prefix} , b_{prefix} then there exists a low-degree proof Π which causes $\mathcal{M}$ to accept with probability 1 over a uniformly random z.
- Soundness: An invalid proof is accepted with less than 1/2 probability

PCP for Normal Form Deciders

Theorem – There exists a TM $\mathcal{M}_{\mathsf{AR}}$ with the following properties:

- Input = Decider Specification + PCP View
  Decider Specification = $(\mathcal{D}, n, T, Q, \sigma, \gamma, x, y)$
  PCP View (random strings) = (z, z') where $z \in \mathbb{F}_q^{m'}, z' \in \mathbb{F}_q^{m'+6}$
- Output is either 1(accept) or 0(reject)
- Complexity = poly$(\log T, \log n, Q, \sigma, \gamma)$
- Completeness: Suppose $\mathcal{D}$ halts on strings $a_{\text{prefix}}, b_{\text{prefix}}$ then there exists a low-degree proof Π which causes $\mathcal{M}$ to accept with probability 1 over a uniformly random z.
- Soundness: An invalid proof is accepted with less than $1/2$ probability

PCP for Normal Form Deciders

Theorem – There exists a TM $\mathcal{M}_{\mathsf{AR}}$ with the following properties:

- Input = Decider Specification + PCP View
  Decider Specification $= (\mathcal{D}, n, T, Q, \sigma, \gamma, x, y)$
  PCP View (random strings) $= (z, z')$ where $z \in \mathbb{F}_q^{m'}, z' \in \mathbb{F}_q^{m'+6}$
- Output is either 1(accept) or 0(reject)
- Complexity $=$ poly$(\log T, \log n, Q, \sigma, \gamma)$
- Completeness: Suppose $\mathcal{D}$ halts on strings a_{prefix} , b_{prefix} then there exists a low-degree proof Π which causes $\mathcal{M}$ to accept with probability 1 over a uniformly random z.
- Soundness: An invalid proof is accepted with less than $1/2$ probability

PCP for Normal Form Deciders

Theorem – There exists a TM $\mathcal{M}_{\text{AR}}$ with the following properties:

- Input = Decider Specification + PCP View
  Decider Specification $= (\mathcal{D}, n, T, Q, \sigma, \gamma, x, y)$
  PCP View (random strings) $= (z, z')$ where $z \in \mathbb{F}_q^{m'}, z' \in \mathbb{F}_q^{m'+6}$
- Output is either 1(accept) or 0(reject)
- Complexity = poly$(\log T, \log n, Q, \sigma, \gamma)$
- Completeness: Suppose $\mathcal{D}$ halts on strings $a_{\text{prefix}}, b_{\text{prefix}}$ then there exists a low-degree proof Π which causes $\mathcal{M}$ to accept with probability 1 over a uniformly random z.
- Soundness: An invalid proof is accepted with less than 1/2 probability

PCP for Normal Form Deciders

The TM $\mathcal{M}$ performs the following steps sequentially:

- Compute $\mathcal{C}$=PaddedSuccinctDecider$(\mathcal{D}, n, T, Q, \sigma, x, y)$. $\mathcal{C}$ has five m-bit inputs and five single-bit inputs, contains at most s gates.
- Compute a boolean formula $\mathcal{F} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that $\mathcal{C}(x, o) = 1$ iff there exists $w \in \{0,1\}^s, \mathcal{F}(x, o, w) = 1$
- Parse $z = (x, o, w)$ and $z' = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$
- Formula Test: Reject if $\beta_0 \neq \mathcal{F}(x, o, w) \cdot (\alpha_1 - o_1) \ldots (\alpha_5 - o_5)$. Else, continue.
- Zero on Subcube Test: Reject if $\beta_0 \neq \sum_{i=1}^{m'} \beta_i \cdot zero(z_i)$. Otherwise, accept.

PCP for Normal Form Deciders

The TM $\mathcal{M}$ performs the following steps sequentially:

- Compute $\mathcal{C}$=PaddedSuccinctDecider$(\mathcal{D}, n, T, Q, \sigma, x, y)$. $\mathcal{C}$ has five m-bit inputs and five single-bit inputs, contains at most s gates.
- Compute a boolean formula $\mathcal{F} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that $\mathcal{C}(x, o) = 1$ iff there exists $w \in \{0,1\}^s, \mathcal{F}(x, o, w) = 1$
- Parse $z = (x, o, w)$ and $z' = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$
- Formula Test: Reject if $\beta_0 \neq \mathcal{F}(x, o, w) \cdot (\alpha_1 - o_1) \ldots (\alpha_5 - o_5)$. Else, continue.
- Zero on Subcube Test: Reject if $\beta_0 \neq \sum_{i=1}^{m'} \beta_i \cdot zero(z_i)$. Otherwise, accept.

PCP for Normal Form Deciders

The TM $\mathcal{M}$ performs the following steps sequentially:

- Compute $\mathcal{C}$=PaddedSuccinctDecider$(\mathcal{D}, n, T, Q, \sigma, x, y)$. $\mathcal{C}$ has five m-bit inputs and five single-bit inputs, contains at most s gates.
- Compute a boolean formula $\mathcal{F} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that $\mathcal{C}(x, o) = 1$ iff there exists $w \in \{0, 1\}^s, \mathcal{F}(x, o, w) = 1$
- Parse $z = (x, o, w)$ and $z' = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$
- Formula Test: Reject if $\beta_0 \neq \mathcal{F}(x, o, w) \cdot (\alpha_1 - o_1) \ldots (\alpha_5 - o_5)$. Else, continue.
- Zero on Subcube Test: Reject if $\beta_0 \neq \sum_{i=1}^{m'} \beta_i \cdot zero(z_i)$. Otherwise, accept.

PCP for Normal Form Deciders

The TM $\mathcal{M}$ performs the following steps sequentially:

- Compute $\mathcal{C}$=PaddedSuccinctDecider$(\mathcal{D}, n, T, Q, \sigma, x, y)$. $\mathcal{C}$ has five m-bit inputs and five single-bit inputs, contains at most s gates.
- Compute a boolean formula $\mathcal{F} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that $\mathcal{C}(x, o) = 1$ iff there exists $w \in \{0, 1\}^s, \mathcal{F}(x, o, w) = 1$
- Parse $z = (x, o, w)$ and $z' = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$
- Formula Test: Reject if $\beta_0 \neq \mathcal{F}(x, o, w) \cdot (\alpha_1 - o_1) \ldots (\alpha_5 - o_5)$. Else, continue.
- Zero on Subcube Test: Reject if $\beta_0 \neq \sum_{i=1}^{m'} \beta_i \cdot zero(z_i)$. Otherwise, accept.

PCP for Normal Form Deciders

The TM $\mathcal{M}$ performs the following steps sequentially:

- Compute $\mathcal{C}$=PaddedSuccinctDecider$(\mathcal{D}, n, T, Q, \sigma, x, y)$. $\mathcal{C}$ has five m-bit inputs and five single-bit inputs, contains at most s gates.
- Compute a boolean formula $\mathcal{F} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that $\mathcal{C}(x, o) = 1$ iff there exists $w \in \{0,1\}^s, \mathcal{F}(x, o, w) = 1$
- Parse $z = (x, o, w)$ and $z' = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$
- Formula Test: Reject if $\beta_0 \neq \mathcal{F}(x, o, w) \cdot (\alpha_1 - o_1) \ldots (\alpha_5 - o_5)$. Else, continue.
- Zero on Subcube Test: Reject if $\beta_0 \neq \sum_{i=1}^{m'} \beta_i \cdot zero(z_i)$. Otherwise, accept.

PCP for Normal Form Deciders

Proof Sketch

- Define $c_0 = \mathcal{F}(x, o, w) \cdot (g_1(x_1) - o_1) \ldots (g_5(x_5) - o_5)$. It is possible to show that $c_0(x, o, w) = 0$ for all z. The PCP view (z, z') then passes the formula test and zero subcube test which establishes completeness.
- Soundness follows from Shwartz Zippel Lemma

PCP for Normal Form Deciders

Proof Sketch

- Define $c_0 = \mathcal{F}(x, o, w) \cdot (g_1(x_1) - o_1) \ldots (g_5(x_5) - o_5)$. It is possible to show that $c_0(x, o, w) = 0$ for all z. The PCP view (z, z') then passes the formula test and zero subcube test which establishes completeness.
- Soundness follows from Shwartz Zippel Lemma