PCPs of sub-constant error via derandomized direct product

Irit Dinur Or Meir

Weizmann Institute
Department of Computer Science and Applied Mathematics

PCP

Recall: A PCP verifier V for a language L is a probabilistic oracle machine that on input x:

- If $x \in L$, then $\exists \pi$ s.t. $V^{\pi}(x)$ accepts w.p. 1.
- If $x \notin L$, then $\forall \pi$: $V^{\pi}(x)$ accepts with small probability.
- V makes few queries to the proof string.

PCP

Recall: A PCP verifier V for a language L is a probabilistic oracle machine that on input x:

- If $x \in L$, then $\exists \pi$ s.t. $V^{\pi}(x)$ accepts w.p. 1.
- If $x \notin L$, then $\forall \pi$: $V^{\pi}(x)$ accepts with small probability.
- V makes few queries to the proof string.

PCP

Recall: A PCP verifier V for a language L is a probabilistic oracle machine that on input x:

- If $x \in L$, then $\exists \pi$ s.t. $V^{\pi}(x)$ accepts w.p. 1.
- If $x \notin L$, then $\forall \pi$: $V^{\pi}(x)$ accepts with small probability.
- V makes few queries to the proof string.

PCP

Recall: A PCP verifier V for a language L is a probabilistic oracle machine that on input x:

- If $x \in L$, then $\exists \pi$ s.t. $V^{\pi}(x)$ accepts w.p. 1.
- If $x \notin L$, then $\forall \pi$: $V^{\pi}(x)$ accepts with small probability.
- V makes few queries to the proof string.

PCP parameters

- q - query complexity.
- s - soundness error.
- ℓ - proof length.
- Σ - proof alphabet.

The PCP theorem [AS92, ALMSS92]

Every $L \in \mathbf{NP}$ has a PCP verifier with constant q, s and $|\Sigma|$, and with $\ell = \mathrm{poly}(n)$.

PCP parameters

- q - query complexity.
- s - soundness error.
- ℓ - proof length.
- Σ - proof alphabet.

The PCP theorem [AS92, ALMSS92]

Every $L \in \mathbf{NP}$ has a PCP verifier with constant q, s and $|\Sigma|$, and with $\ell = \text{poly}(n)$.

Decreasing the soundness error

One research direction, useful for hardness of approximation, is decreasing the soundness error:

- Wish to decrease s as much as possible - ideally to a sub-constant.
- Wish to maintain constant q - ideally 2.
- Wish to maintain polynomial ℓ.

- Since $s \geq 1/|\Sigma|^q$, must have large Σ.

Decreasing the soundness error

One research direction, useful for hardness of approximation, is decreasing the soundness error:

- Wish to decrease s as much as possible - ideally to a sub-constant.
- Wish to maintain constant q - ideally 2.
- Wish to maintain polynomial ℓ.
- Since $s \geq 1/|\Sigma|^q$, must have large Σ.

State of the art

- Via parallel repetition [R95], one can get such a PCP with arbitrarily small contant $s > 0$.
- Folklore (explicit in [MR08]) - using low-degree manifolds: $s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$.
- Recent result of [MR08] (simplification by [DH09]): $\forall s$ have $|\Sigma| = \exp(1/s)$.

State of the art

- Via parallel repetition [R95], one can get such a PCP with arbitrarily small contant $s > 0$.
- Folklore (explicit in [MR08]) - using low-degree manifolds: $s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$.
- Recent result of [MR08] (simplification by [DH09]): $\forall s$ have $|\Sigma| = \exp(1/s)$.

State of the art

- Via parallel repetition [R95], one can get such a PCP with arbitrarily small contant $s > 0$.
- Folklore (explicit in [MR08]) - using low-degree manifolds: $s = 1/\mathrm{poly}\log n$, $|\Sigma| = \exp(\mathrm{poly}\log n)$.
- Recent result of [MR08] (simplification by [DH09]): $\forall s$ have $|\Sigma| = \exp(1/s)$.

Our work

- We show an alternative approach for achieving the folklore result ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- Simpler, more intuitive - using only the sampling properties of linear spaces.
- Our approach is based on derandomized direct product.
- Work in progress: Plugging the construction into the framework of [DH09].

Our work

- We show an alternative approach for achieving the folklore result ($s = 1/\mathrm{poly}\log n$, $|\Sigma| = \exp(\mathrm{poly}\log n)$).
- Simpler, more intuitive - using only the sampling properties of linear spaces.
- Our approach is based on derandomized direct product.
- Work in progress: Plugging the construction into the framework of [DH09].

Our work

- We show an alternative approach for achieving the folklore result ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- Simpler, more intuitive - using only the sampling properties of linear spaces.
- Our approach is based on derandomized direct product.
- Work in progress: Plugging the construction into the framework of [DH09].

Outline

1. Introduction
2. Direct Product PCPs
3. Construction
   - PCP based on Direct Product
   - PCP based on Derandomized Direct Product
   - PCPs and de-Bruijn Graphs

Sequential and Parallel Repetition

- Sequential repetition: Invoking the verifier k times.
- Decreasing s to s^k.
- Increasing q to $k \cdot q$.

- Parallel repetition: Making invocations in parallel.
- Combining $k \cdot q$ queries into q queries.

Sequential and Parallel Repetition

- Sequential repetition: Invoking the verifier k times.
- Decreasing s to s^k.
- Increasing q to $k \cdot q$.

- Parallel repetition: Making invocations in parallel.
- Combining $k \cdot q$ queries into q queries.

Direct Product

- Given a string $w \in \Sigma^\ell$, the k-th direct product (k-DP) of w, denoted $w^{\otimes k}$, is a string of length $\binom{\ell}{k}$ over Σ^k.
- For every $\mathbf{i} = \{i_1, \ldots, i_k\} \subseteq [\ell]$, we define $\left(w^{\otimes k}\right)_{\mathbf{i}} = (w_{i_1}, \ldots, w_{i_k})$.
- In derandomized direct product, we take only some of the k-subsets.

Direct Product

- Given a string $w \in \Sigma^\ell$, the k-th direct product (k-DP) of w, denoted $w^{\otimes k}$, is a string of length $\binom{\ell}{k}$ over Σ^k.
- For every $\mathbf{i} = \{i_1, \ldots, i_k\} \subseteq [\ell]$, we define $\left(w^{\otimes k}\right)_{\mathbf{i}} = (w_{i_1}, \ldots, w_{i_k})$.
- In derandomized direct product, we take only some of the k-subsets.

Parallel Repetition

- The proof strings of the new PCP are k-DPs of the proof strings of the original PCP.
- A query to the new proof simulates k queries to the original proof.
- One test of the new verifier simulates k tests of the original verifier.

Parallel Repetition

- The proof strings of the new PCP are k-DPs of the proof strings of the original PCP.
- A query to the new proof simulates k queries to the original proof.
- One test of the new verifier simulates k tests of the original verifier.

Parallel Repetition

- The proof strings of the new PCP are k-DPs of the proof strings of the original PCP.
- A query to the new proof simulates k queries to the original proof.
- One test of the new verifier simulates k tests of the original verifier.

Parallel Repetition

- Suppose we are given a false claim $x \notin L$ and a proof Π for the new verifier.
- If Π is k-DP (i.e., $\Pi = \pi^{\otimes k}$), the new verifier accepts with probability $\leq s^k$.
- The proof length increases from ℓ to $\approx \ell^k$.
- For super-constant k, the proof length is super-polynomial.
- So, wish to derandomize in order to obtain sub-constant error.

Parallel Repetition

- Suppose we are given a false claim $x \notin L$ and a proof Π for the new verifier.
- If Π is k-DP (i.e., $\Pi = \pi^{\otimes k}$), the new verifier accepts with probability $\leq s^k$.
- The proof length increases from ℓ to $\approx \ell^k$.
- For super-constant k, the proof length is super-polynomial.
- So, wish to derandomize in order to obtain sub-constant error.

Parallel Repetition

- Problem: The proof Π may not be a k-DP,
- Parallel Repetition Theorem [R95]: the new verifier still accepts with probability $\leq \exp(-k)$.
- But much, much more complicated proof.
- Difficult to derandomize.

Parallel Repetition

- Problem: The proof Π may not be a k-DP,
- Parallel Repetition Theorem [R95]: the new verifier still accepts with probability $\leq \exp(-k)$.
- But much, much more complicated proof.
- Difficult to derandomize.

Parallel Repetition

- Problem: The proof Π may not be a k-DP,
- Parallel Repetition Theorem [R95]: the new verifier still accepts with probability $\leq \exp(-k)$.
- But much, much more complicated proof.
- Difficult to derandomize.

PCP based on Direct Product Test

- Natural solution: Direct Product Test.
- Test that the proof string is indeed a direct product.
- A DP test was analyzed by [GS97, DR04, DG08, IKW09].
- [IKW09] used this DP test to construct a PCP.
- This gives a considerably simpler proof for a "parallel repetition"-like theorem.

PCP based on Direct Product Test

- Natural solution: Direct Product Test.
- Test that the proof string is indeed a direct product.
- A DP test was analyzed by [GS97, DR04, DG08, IKW09].

- [IKW09] used this DP test to construct a PCP.
- This gives a considerably simpler proof for a "parallel repetition"-like theorem.

PCP based on Derandomized DP Test

- [IKW09] also suggested a notion of derandomized direct product, and showed it can be tested.
- However, they did not construct a PCP based on this derandomized direct product.
- Our work: Constructing a PCP based on the derandomized direct product of [IKW09].
- Thereby obtaining PCPs of sub-constant error.

PCP based on Derandomized DP Test

- [IKW09] also suggested a notion of derandomized direct product, and showed it can be tested.
- However, they did not construct a PCP based on this derandomized direct product.
- Our work: Constructing a PCP based on the derandomized direct product of [IKW09].
- Thereby obtaining PCPs of sub-constant error.

PCP based on Derandomized DP Test

- [IKW09] also suggested a notion of derandomized direct product, and showed it can be tested.
- However, they did not construct a PCP based on this derandomized direct product.
- Our work: Constructing a PCP based on the derandomized direct product of [IKW09].
- Thereby obtaining PCPs of sub-constant error.

Outline

1. Introduction
2. Direct Product PCPs
3. Construction
    - PCP based on Direct Product
    - PCP based on Derandomized Direct Product
    - PCPs and de-Bruijn Graphs

Outline

1. Introduction
2. Direct Product PCPs
3. Construction
   - PCP based on Direct Product
   - PCP based on Derandomized Direct Product
   - PCPs and de-Bruijn Graphs

Constraint Graphs

- Proof coordinate $\equiv$ Vertex.
- Proof string $\equiv$ Assignment of symbols in Σ to the vertices.
- Possible test $\equiv$ Edge.

- $x \in L \equiv$ Graph s.t. $\exists$ satisfying assignment.
- $x \notin L \equiv$ Graph s.t. $\forall$ assignment satisfies $\leq s$ fraction of the edges.

Rest of the talk

Original verifier viewed as a graph.

Constraint Graphs

- Proof coordinate $\equiv$ Vertex.
- Proof string $\equiv$ Assignment of symbols in Σ to the vertices.
- Possible test $\equiv$ Edge.

- $x \in L \equiv$ Graph s.t. $\exists$ satisfying assignment.
- $x \notin L \equiv$ Graph s.t. $\forall$ assignment satisfies $\leq s$ fraction of the edges.

Rest of the talk

Original verifier viewed as a graph.

Constraint Graphs

- Proof coordinate $\equiv$ Vertex.
- Proof string $\equiv$ Assignment of symbols in Σ to the vertices.
- Possible test $\equiv$ Edge.

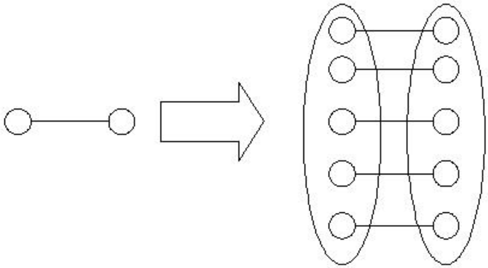
- $x \in L \equiv$ Graph s.t. $\exists$ satisfying assignment.
- $x \notin L \equiv$ Graph s.t. $\forall$ assignment satisfies $\leq s$ fraction of the edges.

Rest of the talk

Original verifier viewed as a graph.

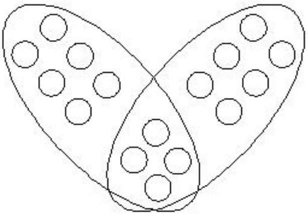
Parallel Repetition on Constraint Graphs

- The verifier chooses k random edges.
- The verifier queries the oracle on the set of left endpoints and on the set of right endpoints.



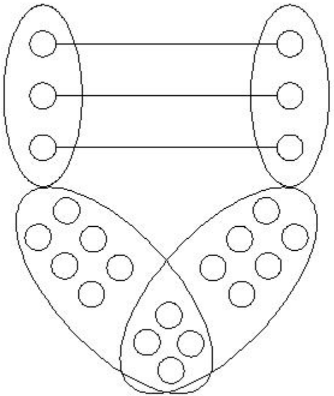
Direct Product Test [GS97, DR04, DG08, IKW09]

- Wish to test that a string Π is a k-DP.
- Choose a k_1-subset $A \subseteq V$.
- Choose k-sets $B_1, B_2 \subseteq V$ containing A.
- Check that Π_{B_1} and Π_{B_2} agree on A.
- If Π is far from any k-DP, the test rejects w.h.p.



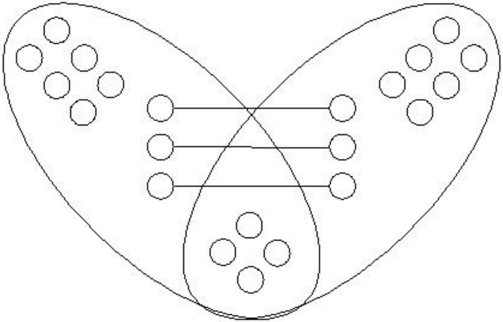
PCP based on DP Test

Natural way to combine parallel repetition with direct product (different than [IKW09]):



PCP based on DP Test

More convenient way to view it.



PCP based on DP Test

Given $G = (V, E)$ and Π:

- Choose k_0-set $E_0 \subseteq E$. Let C_1 and C_2 be the left and right endpoints of E_0.
- Choose a k_1-subset $A \subseteq V$.
- Choose k-sets B_1 and B_2 of V containing $A \cup C_1$ and $A \cup C_2$.
- Check that Π_{B_1} and Π_{B_2} agree on A, and satisfy E_0.

If G has constant soundness, then the probability that the test accepts is $\approx \exp(-k_0)$.

PCP based on DP Test

Given $G = (V, E)$ and Π:

- Choose k_0-set $E_0 \subseteq E$. Let C_1 and C_2 be the left and right endpoints of E_0.
- Choose a k_1-subset $A \subseteq V$.
- Choose k-sets B_1 and B_2 of V containing $A \cup C_1$ and $A \cup C_2$.
- Check that Π_{B_1} and Π_{B_2} agree on A, and satisfy E_0.

If G has constant soundness, then the probability that the test accepts is $\approx \exp(-k_0)$.

Outline

1. Introduction
2. Direct Product PCPs
3. Construction
   - PCP based on Direct Product
   - PCP based on Derandomized Direct Product
   - PCPs and de-Bruijn Graphs

Derandomized Direct Product [IKW09]

- Suppose we want to take the direct product of a string $w \in \Sigma^\ell$.
- Identify coordinates in $[\ell]$ with $\mathbb{F}^m$.
- Instead of taking all k-sets, take only sets that are d-dimensional subspaces of $\mathbb{F}^m$.

Derandomized Direct Product Test [IKW09]

- Wish to test that a string Π is a k-DDP.
- Choose a d_1-subspace A of $\mathbb{F}^m$.
- Choose d-subspaces B_1, B_2 containing A.
- Check that Π_{B_1} and Π_{B_2} agree on A.
- If Π is far from any k-DDP, the test rejects w.h.p. [IKW09].

Why is constructing a PCP difficult?

Imagine the following test:

- Choose k_0-set $E_0 \subseteq E$. Let C_1 and C_2 be the left and right endpoints of E_0.
- Choose a d_1-subspace A.
- Choose d-subspaces B_1, B_2 containing $A \cup C_1$, $A \cup C_2$.
- Check that Π_{B_1} and Π_{B_2} agree on A, and satisfy the edges in E_0.

How do we know that B_1 and B_2 even exist?

Graphs with linear structure

We say that a graph $G = (V, E)$ has linear structure if the following holds:

- The vertices V of G are identified with $\mathbb{F}^m$.
- The edges E of G form a subspace of $\mathbb{F}^{2m}$.
- And:
  - Let E_0 be a random d_0-subspace of E.
  - Let C be either the heads or tails of the edges in E_0.
  - Then, C is a random d_0-subspace of $\mathbb{F}^m$.

Graphs with linear structure

We say that a graph $G = (V, E)$ has linear structure if the following holds:

- The vertices V of G are identified with $\mathbb{F}^m$.
- The edges E of G form a subspace of $\mathbb{F}^{2m}$.
- And:
  - Let E_0 be a random d_0-subspace of E.
  - Let C be either the heads or tails of the edges in E_0.
  - Then, C is a random d_0-subspace of $\mathbb{F}^m$.

Graphs with linear structure

We say that a graph $G = (V, E)$ has linear structure if the following holds:

- The vertices V of G are identified with $\mathbb{F}^m$.
- The edges E of G form a subspace of $\mathbb{F}^{2m}$.
- And:
  - Let E_0 be a random d_0-subspace of E.
  - Let C be either the heads or tails of the edges in E_0.
  - Then, C is a random d_0-subspace of $\mathbb{F}^m$.

PCP based on Derandomized DP Test

Given $G = (V, E)$ with linear structure and Π:

- Choose d_0-subspace $E_0 \subseteq E$. Let C_1 and C_2 be the left and right endpoints of E_0.
- Choose a d_1-subspace A.
- Choose d-subspaces B_1, B_2 containing $A \cup C_1$, $A \cup C_2$.
- Check that Π_{B_1} and Π_{B_2} agree on A, and satisfy E_0.

If G has constant soundness, then the probability that the test accepts is $\approx 1/k_0^{\Omega(1)}$.

PCP based on Derandomized DP Test

Given $G = (V, E)$ with linear structure and Π:

- Choose d_0-subspace $E_0 \subseteq E$. Let C_1 and C_2 be the left and right endpoints of E_0.
- Choose a d_1-subspace A.
- Choose d-subspaces B_1, B_2 containing $A \cup C_1$, $A \cup C_2$.
- Check that Π_{B_1} and Π_{B_2} agree on A, and satisfy E_0.

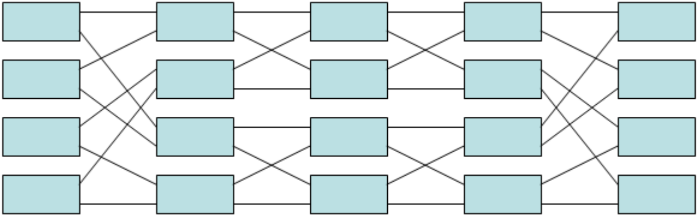
If G has constant soundness, then the probability that the test accepts is $\approx 1/k_0^{\Omega(1)}$.

Outline

de-Bruijn Graphs

A de-Bruijn graph is:

- A layered graph with $\text{poly}(\log n)$ layers.
- The vertices of every layer are identified with $\mathbb{F}^t$.
- The vertex $(\alpha_1, \ldots, \alpha_t) \in \mathbb{F}^t$ in layer i is connected with $(\alpha_2, \ldots, \alpha_t, \beta)$ in layer $i+1$ for every $\beta \in \mathbb{F}$.



(Wikipedia)

de-Bruijn Graphs

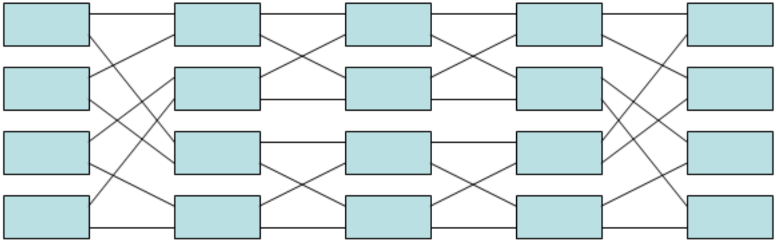
de-Bruijn graphs have linear structure:

- We identify the vertices of the graph with $\mathbb{F}^m$ for $m = t + 1$.
- Let γ be a generator of the multiplicative group of $\mathbb{F}$.
- The vertex $(\alpha_1, \ldots, \alpha_t)$ in layer i is identified with $(\gamma^i, \alpha_1, \ldots, \alpha_t)$.
- Edges are of the form $\left((\gamma^i, \alpha_1, \ldots, \alpha_t), (\gamma^{i+1}, \alpha_2, \ldots, \alpha_t, \beta)\right)$
  - clearly a subspace of $\mathbb{F}^{2m}$.

Routing on de-Bruijn Graphs

de-Brujin Graphs are routing networks:

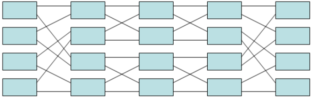
- Given a permutation σ of the first layer to the last layer.
- Can find paths from each vertex v in the first layer to $\sigma(v)$.
- The paths are vertex-disjoint.



Embedding PCPs on de-Bruijn Graphs

- We can use it to embed any constraint graph $G = (V, E)$ in a de-Bruijn graph.
- With loss of generality, constraint graph has constant degree.
- Variant of [BFLS91, PS94].

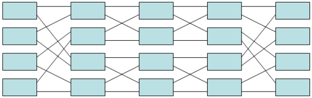
- Assume that each vertex had degree 1, then:
  - Identify the first layer with V, and same for the last layer.
  - Define $\sigma(u) = v$ if v is the neighbor of u in G.
  - Find vertex-disjoint paths for σ.
  - Embed the edges of G on the vertex-disjoint paths.



Embedding PCPs on de-Bruijn Graphs

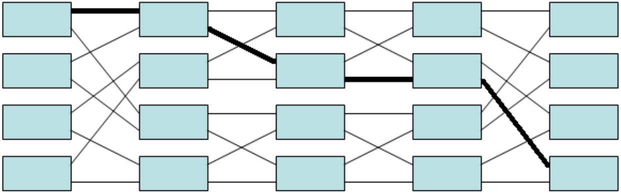
- We can use it to embed any constraint graph $G = (V, E)$ in a de-Bruijn graph.
- With loss of generality, constraint graph has constant degree.
- Variant of [BFLS91, PS94].

- Assume that each vertex had degree 1, then:
  - Identify the first layer with V, and same for the last layer.
  - Define $\sigma(u) = v$ if v is the neighbor of u in G.
  - Find vertex-disjoint paths for σ.
  - Embed the edges of G on the vertex-disjoint paths.



Embedding PCPs on de-Bruijn Graphs

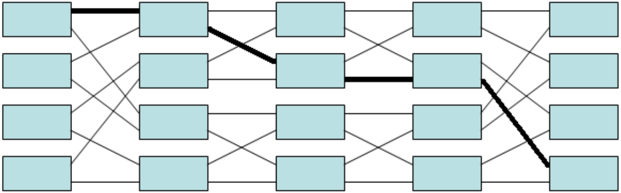
- How do we embed an edge $\mathbf{e}$ of G on a path $e_1, \ldots, e_p$?
- Put equality constraints on $e_1, \ldots, e_{p-1}$.
- Associate e_p with the constraint of $\mathbf{e}$.



- If G has constant degree d, repeat d times.

Embedding PCPs on de-Bruijn Graphs

- How do we embed an edge $\mathbf{e}$ of G on a path $e_1, \ldots, e_p$?
- Put equality constraints on $e_1, \ldots, e_{p-1}$.
- Associate e_p with the constraint of $\mathbf{e}$.



- If G has constant degree d, repeat d times.

Embedding PCPs on de-Bruijn Graphs

- The embedded PCP has soundness error $1 - \frac{1-s}{\text{poly}\log n}$.
- This is affordable.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly} \log n$, $|\Sigma| = \exp(\text{poly} \log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Summary

- We obtain PCPs of sub-constant soundness ($s = 1/\text{poly}\log n$, $|\Sigma| = \exp(\text{poly}\log n)$).
- The construction is based on a direct product approach:
  1. Testing that the proof is a direct product.
  2. Performing parallel repetition.
  3. We use derandomized direct product.
- This is done by:
  1. Showing a test for graphs with linear structure.
  2. Showing that de-Bruijn graphs have linear structure.
  3. Embedding any PCP on a de-Bruijn graph.

Thank you!