Probabilistically Checkable Proofs

Computability & Complexity

May 24, 2019

NP Verifier

Definition (NP)

A language $L \in NP$ iff there exists a TM P (verifier), that reads the input string x and a proof string π such that:

- **Completeness** : $x \in L \implies \exists \pi V(x, \pi) = 1$
- **Soundness** : $x \notin L \implies \forall \pi V(x, \pi) = 0$

Examples:

- 3-SAT
- 3-colorable graphs

What happens if we limit the verifier but allow mistakes?

PCP Verifier

Definition (PCP)

A language $L \in PCP(r, q)$ iff there exists a TM P (verifier), that reads the input string x, r random bits and q bits from a proof string π such that:

- **Completeness** : $x \in L \implies \exists \pi Pr_r[V(x, \pi) = 1] = 1$
- **Soundness** : $x \in L \implies \forall \pi Pr_r[V(x, \pi) = 0] > 1/2$

Examples:

- 3-SAT instances where either all or less than half clauses can be satisfied
- 3-colorable graphs where either all or less than half edges can be bicolor

i.e : constraint satisfaction is either perfect or very bad

Known Classes In The PCP Format

Theorem

$PCP[0, 0] =$

Known Classes In The PCP Format

Theorem

$PCP[0, 0] = P$

The verifier runs its algorithm, no proof to look at. No randomness.

Theorem

$PCP[0, poly(n)] =$

Known Classes In The PCP Format

Theorem

$PCP[0, 0] = P$

The verifier runs its algorithm, no proof to look at. No randomness.

Theorem

$PCP[0, poly(n)] = NP$

The verifier looks at proof and decides. No randomness.

Theorem

$PCP[poly(n), 0] =$

Known Classes In The PCP Format

Theorem

$PCP[0, 0] = P$

The verifier runs its algorithm, no proof to look at. No randomness.

Theorem

$PCP[0, poly(n)] = NP$

The verifier looks at proof and decides. No randomness.

Theorem

$PCP[poly(n), 0] = coRP$

The verifier runs the randomized algorithm in poly time, no proof to look at.

Known Classes In The PCP Format

Theorem

$PCP[0, 0] = P$

The verifier runs its algorithm, no proof to look at. No randomness.

Theorem

$PCP[0, poly(n)] = NP$

The verifier looks at proof and decides. No randomness.

Theorem

$PCP[poly(n), 0] = coRP$

The verifier runs the randomized algorithm in poly time, no proof to look at.

Theorem

$PCP[log(n), O(1)] =?$

Known Classes In The PCP Format

Theorem

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

Proof.

We must show that for every $PCP[log(n), O(1)]$ verifier there is an NP verifier. For every random string of bits it can read $O(1)$ bits, so in all possible runs it can read at most $2^{log(n)} * O(1)$ bits. We can make an NP verifier that reads all possible inputs of the PCP verifier as the proof. □

Known Classes In The PCP Format

Theorem

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

Proof.

We must show that for every $PCP[log(n), O(1)]$ verifier there is an NP verifier. For every random string of bits it can read $O(1)$ bits, so in all possible runs it can read at most $2^{log(n)} * O(1)$ bits. We can make an NP verifier that reads all possible inputs of the PCP verifier as the proof. □

Why wouldn't $PCP[poly(n), O(1)] \in NP$ work? Because the verifier can read too much and the equivalent input would be exponential. In fact $PCP[poly(n), O(1)] = NEXP$

Known Classes In The PCP Format

Theorem

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

Proof.

We must show that for every $PCP[log(n), O(1)]$ verifier there is an NP verifier. For every random string of bits it can read $O(1)$ bits, so in all possible runs it can read at most $2^{log(n)} * O(1)$ bits. We can make an NP verifier that reads all possible inputs of the PCP verifier as the proof. □

Why wouldn't $PCP[poly(n), O(1)] \in NP$ work? Because the verifier can read too much and the equivalent input would be exponential. In fact $PCP[poly(n), O(1)] = NEXP$
What about the reverse?

Classical PCP Theorem

We know that $P \subseteq PCP(log(n), O(1) \subseteq NP$. Is PCP weaker than NP?

Classical PCP Theorem

We know that $P \subseteq PCP(log(n), O(1) \subseteq NP$. Is PCP weaker than NP?

Theorem (PCP theorem)

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

Classical PCP Theorem

We know that $P \subseteq PCP(log(n), O(1) \subseteq NP$. Is PCP weaker than NP?

Theorem (PCP theorem)

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

This means that every NP problem has a probabilistic checking scheme. i.e we can demand the proof in specific format (PCP format) such that the mistake can be found everywhere.

Initial Proof **PCP transformation** **PCP format**

Another View of PCP

What is a "mistake"?

Another View of PCP

What is a "mistake"?
Its a violated constraint of the problem.

Another View of PCP

What is a "mistake"?
Its a violated constraint of the problem.
Normally an assignment (proof) to a 3SAT instance (problem) would be able to possibly satisfy all except one clauses.
In this case the "mistake" is very localized and random sampling would only have $1/m$ chance to find it.

Another View of PCP

What is a "mistake"?
Its a violated constraint of the problem.
Normally an assignment (proof) to a 3SAT instance (problem) would be able to possibly satisfy all except one clauses.
In this case the "mistake" is very localized and random sampling would only have $1/m$ chance to find it.
To get a assignment in PCP format we would need to force the problem to be only able to be either 100% satisfied or at most 50% satisfied.

Another View of PCP

Definition (q-CSP)

A q-CSP problem is a set of m q-ary constaints. It is a YES instance when there exists an assignment that satisfies all of them an NO otherwise.

Examples: 3SAT, 3COL etc

Another View of PCP

Definition (q-CSP)

A q-CSP problem is a set of m q-ary constaints. It is a YES instance when there exists an assignment that satisfies all of them an NO otherwise.

Examples: 3SAT, 3COL etc

Definition (ρ-Gap-q-CSP)

A ρ-Gap-q-CSP problem is a q-CSP problem such that either all constaint can be satisfied or at most $\rho * m$ of them.

Robust version of q-CSPs.

Another View of PCP

Definition (q-CSP)

A q-CSP problem is a set of m q-ary constaints. It is a YES instance when there exists an assignment that satisfies all of them an NO otherwise.

Examples: 3SAT, 3COL etc

Definition (ρ-Gap-q-CSP)

A ρ-Gap-q-CSP problem is a q-CSP problem such that either all constaint can be satisfied or at most $\rho * m$ of them.

Robust version of q-CSPs.

Theorem (Equivalent of PCP theorem)

There exist ρ and q such that ρ-Gap-q-CSP is NP-hard.

This shows the hardness of approximation connection!

Another View of PCP

Theorem

The two statements are equivalent:

- *There exist ρ and q such that ρ-Gap-q-CSP is NP-hard.*
- $NP \subseteq PCP(log(n), O(1))$

Proof.

- ($\rightarrow$) Any NP problem can be reduced to the ρ-Gap-q-CSP which will either be satisfied or at most ρ constraints are satisfied. The PCP samples a constraint to check.
- ($\leftarrow$) We make 1 constraint for each run of the PCP verifier. The constraint is true if the verifier accepts. Hence at most ρ clauses are satisfied if its a NO instance.

Robustification

To prove the PCP theorem we would need to **reduce** from an NP-hard problem to a ρ-Gap-q-CSP problem.

Robustification

To prove the PCP theorem we would need to **reduce** from an NP-hard problem to a ρ-Gap-q-CSP problem.
Goal: efficient algorithm that maps almost satisfiable instances to barely satisfiable (robustifier).

Robustification

To prove the PCP theorem we would need to **reduce** from an NP-hard problem to a ρ-Gap-q-CSP problem.
Goal: efficient algorithm that maps almost satisfiable instances to barely satisfiable (robustifier).
1st proof by Arora et al (1998).
2nd proof by Dinur (2007) using expander graphs. 2019 Goedel Prize.

Dinur's Approach

We can begin from any NP-hard problem and robustify it to prove the PCP theorem.

Dinur's Approach

We can begin from any NP-hard problem and robustify it to prove the PCP theorem.
We choose 3COL.

Dinur's Approach

We can begin from any NP-hard problem and robustify it to prove the PCP theorem.
We choose 3COL.
Call $UNSAT(G)$ the minimum numbeer of edges that are violated by a coloring. Obviously for an arbitrary graph we have that $UNSAT(G) \geq 1/|E|$.

Dinur's Approach

We can begin from any NP-hard problem and robustify it to prove the PCP theorem.
We choose 3COL.
Call $UNSAT(G)$ the minimum numbeer of edges that are violated by a coloring. Obviously for an arbitrary graph we have that $UNSAT(G) \geq 1/|E|$.
We will define a transformation that increases the size of any given graph by a constant while at least doubling the $UNSAT$ value of the graph.

Dinur's Approach

We can begin from any NP-hard problem and robustify it to prove the PCP theorem.
We choose 3COL.
Call $UNSAT(G)$ the minimum numbeer of edges that are violated by a coloring. Obviously for an arbitrary graph we have that $UNSAT(G) \geq 1/|E|$.
We will define a transformation that increases the size of any given graph by a constant while at least doubling the $UNSAT$ value of the graph.
By applying this transformation $log(E)$ times we obtain a graph with constant $UNSAT$, i.e we robustify it.

Dinur's Approach

This is done in 3 steps

Dinur's Approach

This is done in 3 steps

The first step preprocesses the graph, giving it some "nice" properties

Dinur's Approach

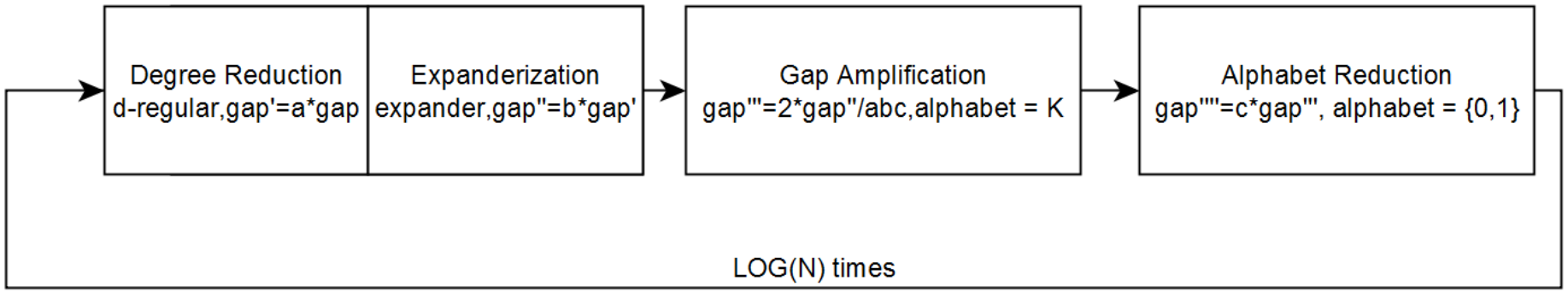
This is done in 3 steps
The first step preprocesses the graph, giving it some "nice" properties
The second step amplifies the UNSAT value, while increasing the alphabet.
Metaphor: "the area of the jam doubles"

Dinur's Approach

This is done in 3 steps
The first step preprocesses the graph, giving it some "nice" properties
The second step amplifies the UNSAT value, while increasing the alphabet.
Metaphor: "the area of the jam doubles"
The third step reduces the alpabet to $\{0, 1\}$ while reducing UNSAT by a constant.

Dinur's Approach

This is done in 3 steps
The first step preprocesses the graph, giving it some "nice" properties
The second step amplifies the UNSAT value, while increasing the alphabet.
Metaphor: "the area of the jam doubles"
The third step reduces the alpabet to $\{0, 1\}$ while reducing UNSAT by a constant.



Main Lemma

Theorem (DIN07)

There exists Σ_0 such that the following holds. For any finite alphabet Σ there exist $C > 0$ and $0 < a < 1$ such that, given a constraint graph $G =< (V, E), \Sigma, C >$, one can construct in polynomial time, a constraint graph $G' =< (V', E'), \Sigma_0, C' >$ such that

- $size(G) < C * size(G')$
- $UNSAT(G) = 0 \implies UNSAT(G') = 0$
- $UNSAT(G') \geq min(2 * UNSAT(G), a)$

Background: Expanders

Definition

The edge boundary of a set S, denoted ∂S, is the edges emanating from S, $\partial S = E(S, \bar{S})$. The edge expansion ratio of G, denoted $h(G)$ is $h(G) = min_S \frac{|\partial S|}{|S|}$

Definition

A d-regular graph with n vertices, $h(G) \geq \eta$ is called an (n,d,η)-expander.

Such graphs exist. Can we construct them?

Background: Expanders

Definition

The edge boundary of a set S, denoted ∂S, is the edges emanating from S, $\partial S = E(S, \bar{S})$. The edge expansion ratio of G, denoted $h(G)$ is $h(G) = min_S \frac{|\partial S|}{|S|}$

Definition

A d-regular graph with n vertices, $h(G) \geq \eta$ is called an (n,d,η)-expander.

Such graphs exist. Can we construct them? Yes!

Theorem

There exists η such that for every n,d there is an efficient to construct a (n,d,η)-expander.

Background: Expanders

Intuitively expanders are highly connected graphs, and they have several useful properties.

Background: Expanders

Intuitively expanders are highly connected graphs, and they have several useful properties.
In particular, a random walk is statistically close to random independent sampling.

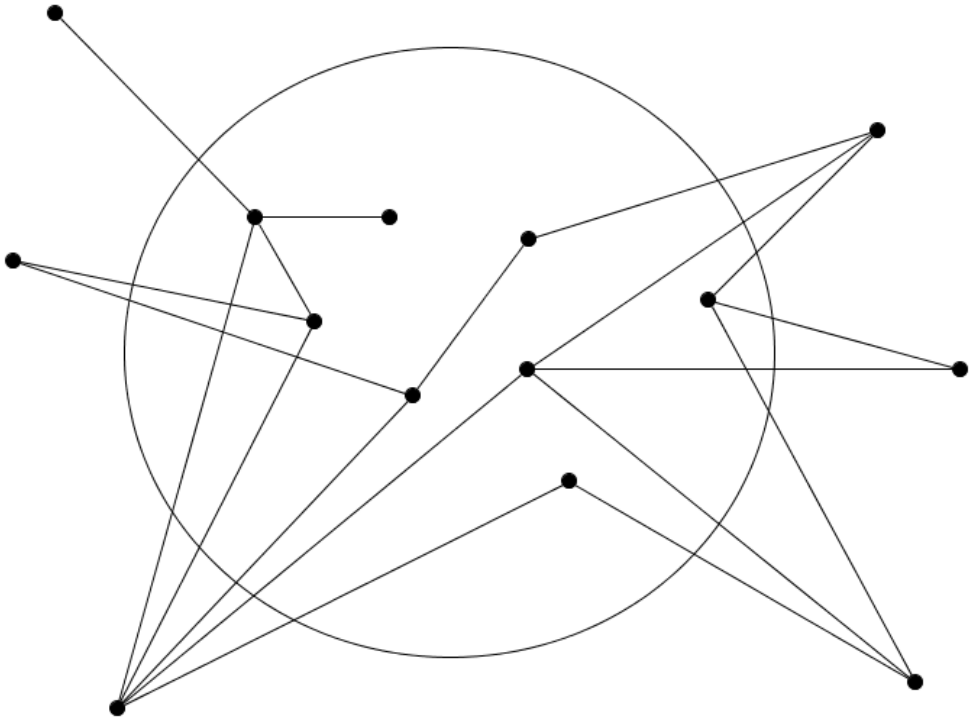
Background: Expanders

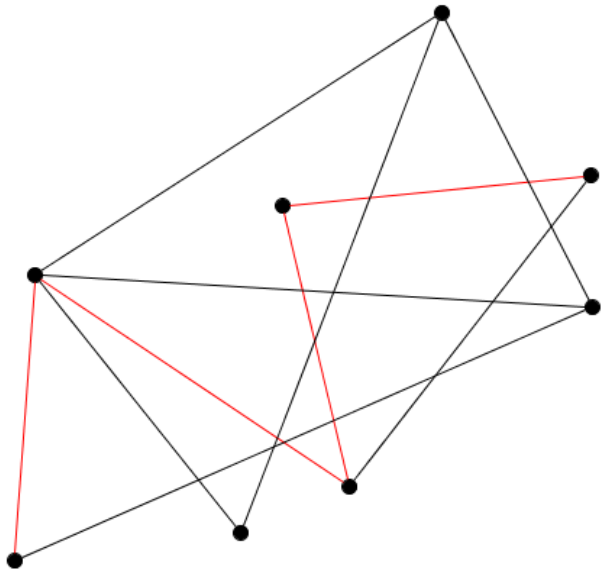
Intuitively expanders are highly connected graphs, and they have several useful properties.
In particular, a random walk is statistically close to random independent sampling.

Theorem

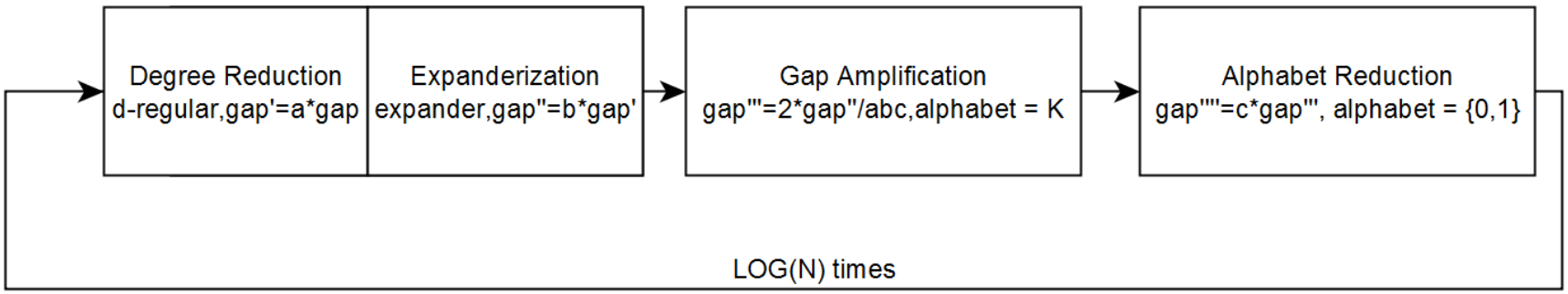
Let $G = (V, E)$ be a d-regular graph with second eigenvalue λ. Let $F \subset E$ be a set of edges without self loop, and let K be the distribution on vertices induced by selecting a random edge in F, and then a random endpoint.
The probability p that a random walk that starts with distribution K takes the $i+1$st step in F, is upper bounded by $\frac{|F|}{|E|} + (\frac{|\lambda|}{d})^i$.

Background: Expanders



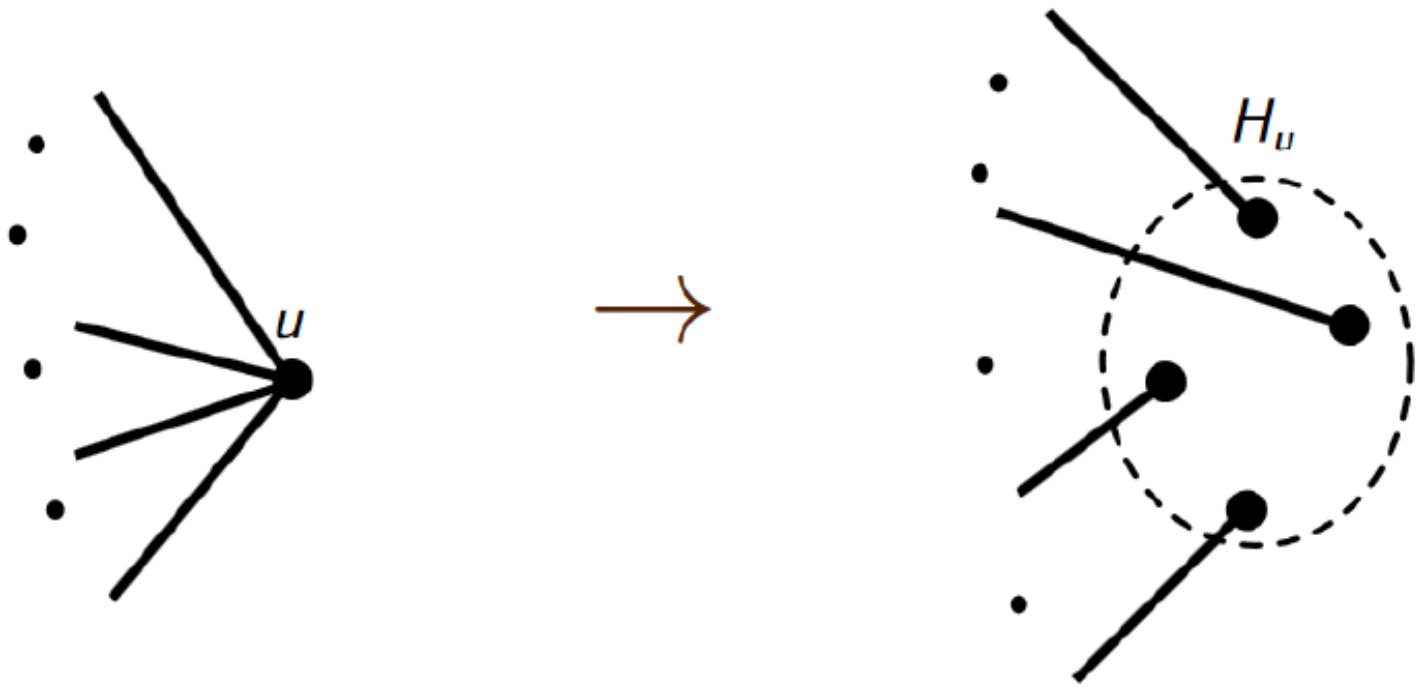


Overview


Degree Reduction
d-regular,gap'=a*gap
Expanderization
expander,gap"=b*gap'
Gap Amplification
gap'"=2*gap"/abc,alphabet = K
Alphabet Reduction
gap""=c*gap'", alphabet = {0,1}
LOG(N) times

Degree Reduction

Here the graph is made regular without making it too much easier to color.
Vertices are replaced with a d-1-regular expander.
(Papadimitriou-Yannakakis gadget)
After this step UNSAT is increased by a constant.

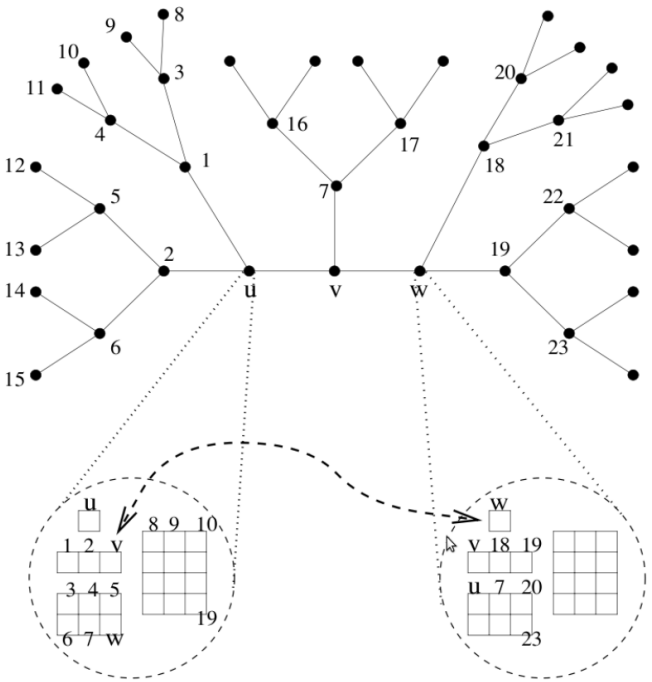


Expanderization

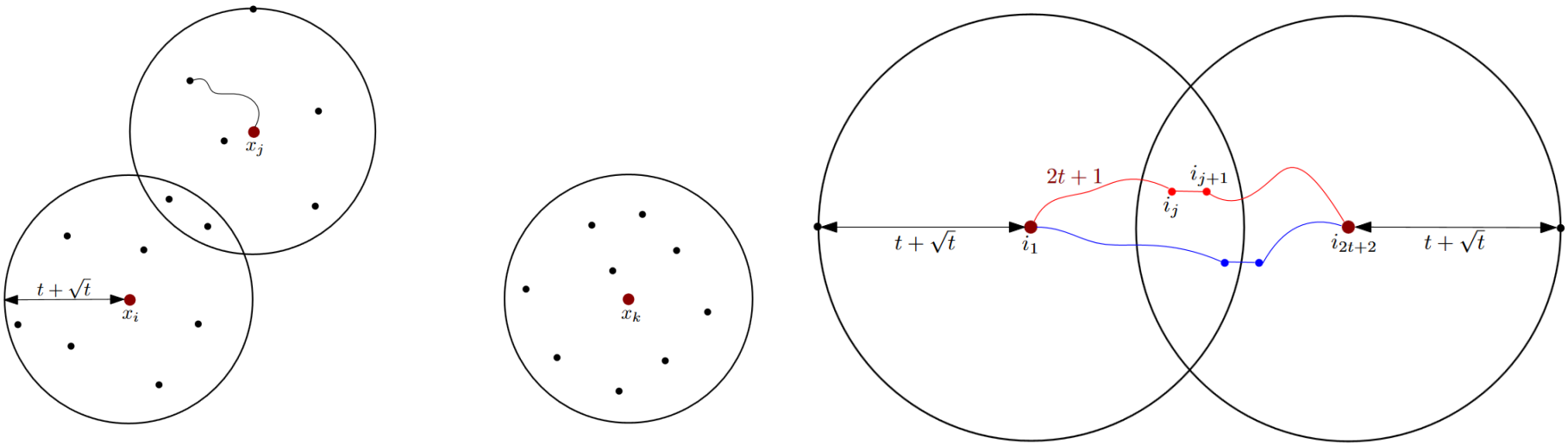
Here certain edges are added to make the graph an expander.
$NewGraph = OldGraph \cup Expander$
The new edges are null constraints so UNSAT can only increase but only by a constant again.

Gap Amplification

A new graph is constructed such that an edge (constraint) between A,B exists iff there was a walk of length t from A to B.
The alphabet(colors) is now $\{0,1\}^{d^t}$. Every vertex now has "opinion" about every other vertex within distance t from it.
The new constraints are satisfied iff the opinions of A,B agree on common nodes and they satisfy the original constraints.



Gap Amplification



Gap Amplification

What happens to UNSAT after this powering operation?

Gap Amplification

What happens to UNSAT after this powering operation?
Assume for simplicity that the assignments all agree, and that they are derived from an underlying assignment to the initial graph.

Gap Amplification

What happens to UNSAT after this powering operation?
Assume for simplicity that the assignments all agree, and that they are derived from an underlying assignment to the initial graph.
The initial graph had UNSAT $\frac{|F|}{|E|}$. The chance that a walk of length t doesn't hit those violating edges is approximately $(1 - \frac{|F|}{|E|})^t$.

Gap Amplification

What happens to UNSAT after this powering operation?
Assume for simplicity that the assignments all agree, and that they are derived from an underlying assignment to the initial graph.
The initial graph had UNSAT $\frac{|F|}{|E|}$. The chance that a walk of length t doesn't hit those violating edges is approximately $(1 - \frac{|F|}{|E|})^t$.
Hence approximately $1 - (1 - \frac{|F|}{|E|})^t$ of the new vertices violate their constraints.

Gap Amplification

What happens to UNSAT after this powering operation?
Assume for simplicity that the assignments all agree, and that they are derived from an underlying assignment to the initial graph.
The initial graph had UNSAT $\frac{|F|}{|E|}$. The chance that a walk of length t doesn't hit those violating edges is approximately $(1 - \frac{|F|}{|E|})^t$.
Hence approximately $1 - (1 - \frac{|F|}{|E|})^t$ of the new vertices violate their constraints.
Hence $1 - (1 - \frac{|F|}{|E|})^t \approx 1 - (1 - t * \frac{|F|}{|E|}) = t * \frac{|F|}{|E|}$
This means UNSAT is increased by a constant of our choice every time we execute this step!

Alphabet Reduction

But the size of the alphabet is now 2^{dt}!

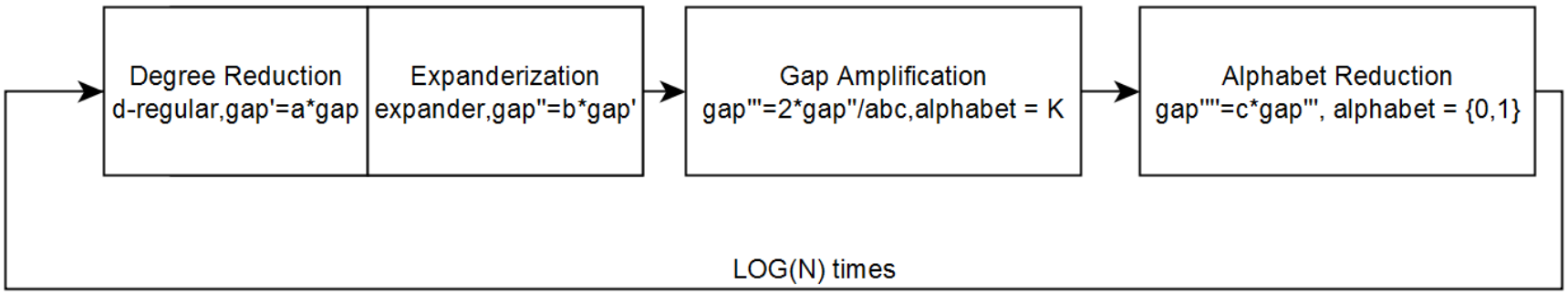
Alphabet Reduction

But the size of the alphabet is now 2^{dt}!
We need to drop it to 2 before we execute the previous step again.

Alphabet Reduction

But the size of the alphabet is now 2^{dt}!
We need to drop it to 2 before we execute the previous step again.
Tools: Hadamard encoding and Linearity testing
For another time...

Overview


Degree Reduction
d-regular,gap'=a*gap
Expanderization
expander,gap"=b*gap'
Gap Amplification
gap'"=2*gap"/abc,alphabet = K
Alphabet Reduction
gap""=c*gap'", alphabet = {0,1}
LOG(N) times

Overview

- According to the PCP theorem we can spread a "mistake" all over the proof.
- It is equivalent to "robustifying" an NP-hard problem.
- To do that Dinur defines a transformation applied Log(n) times that each time doubles the area of the mistake.
- Central to Dinur's proof are expander graphs that permit distributing the mistake everywhere through small walks.

Further Reading

Irit Dinur

The PCP Theorem by Gap Amplification

http://www.wisdom.weizmann.ac.il/ dinuri/mypapers/combpcp.pdf

Jaikumar Radhakrishnan, and Madhu Sudan

On Dinurs Proof of the PCP Theorem

people.csail.mit.edu/dmoshkov/courses/pcp/dinur_pcp_overview.pdf

S. Arora, and B. Barak

Computational Complexity: A Modern Approach, chapter 18

https://theory.cs.princeton.edu/complexity/book.pdf

S. Hoory, N Linial, and A. Wigderson

EXPANDER GRAPHS AND THEIR APPLICATIONS

http://www.cs.huji.ac.il/ nati/PAPERS/expander_survey.pdf