>>> On Dinur's Proof of the PCP Theorem

>>> COMP 531: Advanced Theory of Computation

Names: Aly Ibrahim[†], Justin Li[‡], Harley Wiltzer[§]

Institute: McGill University

Date: May 2, 2020

[†] aly.ibrahim@mail.mcgill.ca

[‡] juan.y.li@mail.mcgill.ca

[§] harley.wiltzer@mail.mcgill.ca

>>> Table of Contents

\>>> Theorem Statement

Definition (Non-Deterministic Polynomial (NP))

$\exists$ deterministic polynomial-time verifier for an NP problem S that takes an input instance x and a polynomial-size proof t That:

* Completeness: Accepts ($x \in S$)
* Soundness: Rejects ($x \notin S$)

Definition (Non-Deterministic Polynomial (NP))

$\exists$ deterministic polynomial-time verifier for an NP problem S that takes an input instance x and a polynomial-size proof t
That:

* Completeness: Accepts ($x \in S$)
* Soundness: Rejects ($x \notin S$)

Definition (Probabilistically Checkable Proof (PCP))

$\mathsf{PCP}[r(n), q(n)]$ is the class of languages provable with a PCP system which uses $O(r(n))$ bits of randomness, queries $O(q(n))$ bits in the proof, and has completeness 1, soundness 1/2.

>>> Theorem Statement

Theorem (PCP Theorem)
$\mathsf{NP} \subseteq \mathsf{PCP}[\log n, 1]$

BACKGROUND

>>> Background

* Hardness of Approximation
* Constraints
* Constraint Graphs
* Expander Graphs

>>> Hardness of Approximation

EFFICIENT

EXACT

HARD

>>> Hardness of Approximation

P Complexity Class

EFFICIENT

EXACT

HARD

>>> Hardness of Approximation

Fixed Parameter Algorithms

EFFICIENT | EXACT | HARD

Fixed Parameter Algorithms

EFFICIENT

EXACT

HARD

Imagine *A* solves an NP graph problem in $O(V \cdot 2^E)$ time

Although inefficient, if graph is sparse, it is useful

>>> Hardness of Approximation

Approximation Algorithms

EFFICIENT | EXACT | HARD

>>> Hardness of Approximation

Definition (α-Approximation)

Given a combinatorial optimization maximization problem x with an optimal solution OPT, we say an algorithm A is an α-approximation algorithm with $(0 < \alpha \leq 1)$, if A is guaranteed to return a solution with value $\geq \alpha \cdot \text{OPT}$.

>>> Hardness of Approximation

Definition (α-Approximation)

Given a combinatorial optimization maximization problem x with an optimal solution OPT, we say an algorithm A is an α-approximation algorithm with $(0 < \alpha \leq 1)$, if A is guaranteed to return a solution with value $\geq \alpha \cdot \mathrm{OPT}$.

Definition (Polynomial Time Approximation Scheme (PTAS))

A maximization combinatorial optimization problem is said to have a PTAS if it has a $(1-\epsilon)$-approximation algorithm for every constant $\epsilon > 0$.

>>> Hardness of Approximation

Definition (α-Approximation)

Given a combinatorial optimization maximization problem x with an optimal solution OPT, we say an algorithm A is an α-approximation algorithm with $(0 < \alpha \leq 1)$, if A is guaranteed to return a solution with value $\geq \alpha \cdot \text{OPT}$.

Definition (Polynomial Time Approximation Scheme (PTAS))

A maximization combinatorial optimization problem is said to have a PTAS if it has a $(1-\epsilon)$-approximation algorithm for every constant $\epsilon > 0$.

Definition (Hardness assuming $P \neq NP$)

When a combinatorial optimization problem has no PTAS

>>> Hardness of Approximation

Definition (Combinatorial Optimization to Decision)

Example: GAP-E3SAT$_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES (OPT $\geq c \cdot m$)
* NO (OPT $< s \cdot m$)
* Anything o.w.

Definition (Combinatorial Optimization to Decision)

Example: $\text{GAP-E3SAT}_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES ($\text{OPT} \geq c \cdot m$)
* NO ($\text{OPT} < s \cdot m$)
* Anything o.w.

Theorem (PCP-Theorem ≡ GAP-E3SAT-Hardness)

PCP-*theorem* $\iff$ $\exists$ *universal constant* $s < 1$ *s.t.* $\text{GAP-E3SAT}_{1,s}$ *is* NP-*hard.*

Definition (Combinatorial Optimization to Decision)

Example: GAP-E3SAT$_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES (OPT $\geq c \cdot m$)
* NO (OPT $< s \cdot m$)
* Anything o.w.

Theorem (PCP-Theorem $\equiv$ GAP-E3SAT-Hardness)

PCP-*theorem* $\iff$ $\exists$ *universal constant* $s < 1$ *s.t.* GAP-E3SAT$_{1,s}$ *is* NP-*hard.*

Proof.

$\Leftarrow$ *Build* PCP$[\log n, 1]$ *for* GAP-E3SAT$_{1,s}$

>>> Hardness of Approximation

Definition (Combinatorial Optimization to Decision)

Example: GAP-E3SAT$_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES (OPT $\geq c \cdot m$)
* NO (OPT $< s \cdot m$)
* Anything o.w.

Theorem (PCP-Theorem $\equiv$ GAP-E3SAT-Hardness)

PCP-*theorem* $\iff$ $\exists$ *universal constant* $s < 1$ *s.t.* GAP-E3SAT$_{1,s}$ *is* NP-*hard.*

Proof.

$\Leftarrow$ *Build* PCP$[\log n, 1]$ *for* GAP-E3SAT$_{1,s}$

choose random clause, check if it satisfies.

>>> Hardness of Approximation

Definition (Combinatorial Optimization to Decision)

Example: GAP-E3SAT$_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES (OPT $\geq c \cdot m$)
* NO (OPT $< s \cdot m$)
* Anything o.w.

Theorem (PCP-Theorem $\equiv$ GAP-E3SAT-Hardness)

PCP-*theorem* $\iff$ $\exists$ *universal constant* $s < 1$ *s.t.* GAP-E3SAT$_{1,s}$ *is* NP-*hard.*

Proof.

$\Leftarrow$ *Build* PCP$[\log n, 1]$ *for* GAP-E3SAT$_{1,s}$

choose random clause, check if it satisfies.

If $x \in$ GAP-E3SAT$_{1,s}$ *then all clauses satisfy*

>>> Hardness of Approximation

Definition (Combinatorial Optimization to Decision)

Example: GAP-E3SAT$_{c,s}$ $(0 < s \leq c \leq 1)$

Given an estimate-3SAT (E3SAT) formula on m clauses, output:

* YES (OPT $\geq c \cdot m$)
* NO (OPT $< s \cdot m$)
* Anything o.w.

Theorem (PCP-Theorem $\equiv$ GAP-E3SAT-Hardness)

PCP-*theorem* $\iff$ $\exists$ *universal constant* $s < 1$ *s.t.* GAP-E3SAT$_{1,s}$ *is* NP-*hard.*

Proof.

$\Leftarrow$ *Build* $\mathsf{PCP}[\log n, 1]$ *for* GAP-E3SAT$_{1,s}$

choose random clause, check if it satisfies.

If $x \in$ GAP-E3SAT$_{1,s}$ *then all clauses satisfy*

If $x \notin$ GAP-E3SAT$_{1,s}$ *then at most* s *clauses satisfy*

>>> Constraint Graphs

Definition

$G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ is called a constraint graph, if

1. (V, E) is an undirected graph.
2. V is the set of variables that assumes values over alphabet Σ.
3. Each edge $e \in E$, defines a constraint $c(e) \subseteq \Sigma \times \Sigma$, and $\mathcal{C} = \{c(e)\}_{e \in E}$. A constraint $c(e)$ is said to be satisfied by (a, b) iff $(a, b) \in c(e)$.

>>> Constraint Graphs

Definition

$G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ is called a constraint graph, if

1. (V, E) is an undirected graph.
2. V is the set of variables that assumes values over alphabet Σ.
3. Each edge $e \in E$, defines a constraint $c(e) \subseteq \Sigma \times \Sigma$, and $\mathcal{C} = \{c(e)\}_{e \in E}$. A constraint $c(e)$ is said to be satisfied by (a, b) iff $(a, b) \in c(e)$.

An assignment is a mapping $\sigma : V \to \Sigma$.

>>> Constraint Graphs

Definition

$G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ is called a constraint graph, if

1. (V,E) is an undirected graph.
2. V is the set of variables that assumes values over alphabet Σ.
3. Each edge $e \in E$, defines a constraint $c(e) \subseteq \Sigma \times \Sigma$, and $\mathcal{C} = \{c(e)\}_{e \in E}$. A constraint $c(e)$ is said to be satisfied by (a,b) iff $(a,b) \in c(e)$.

An assignment is a mapping $\sigma : V \to \Sigma$.

$$\mathtt{UNSAT}_\sigma(G) = \Pr_{(u,v)\in E}[\sigma(u), \sigma(v)) \notin c(e)], \quad \mathtt{UNSAT}(G) = \min_\sigma \mathtt{UNSAT}_\sigma(G).$$

>>> Expander Graphs

Definition

Let $G = (V, E)$ be a d-regular graph. Let $E(S, \bar{S})$ = $\left|(S \times \bar{S}) \cap E\right|$ = $|\{(u, v) \in E \mid u \in S$ and $v \in \bar{S}\}|$ equal the number of edges from a subset $S \subseteq V$ to its complement. The edge expansion of G is defined as

$$h(G) = \min_{S:\ |S| \leq \frac{|V|}{2}} \frac{E(S, \bar{S})}{|S|}.$$

>>> Expander Graphs

Definition

Let $G = (V, E)$ be a d-regular graph. Let $E(S, \bar{S}) = \left|(S \times \bar{S}) \cap E\right| = |\{(u, v) \in E \mid u \in S$ and $v \in \bar{S}\}|$ equal the number of edges from a subset $S \subseteq V$ to its complement. The edge expansion of G is defined as

$$h(G) = \min_{S:\ |S| \leq \frac{|V|}{2}} \frac{E(S, \bar{S})}{|S|}.$$

Definition ($h(G) \propto (d - \lambda)$)

The spectral gap $(d - \lambda)$ of G's adjacency matrix corresponds to its edge expansion $h(G)$. Good expanders have large $h(G)$.

>>> Expander Graphs

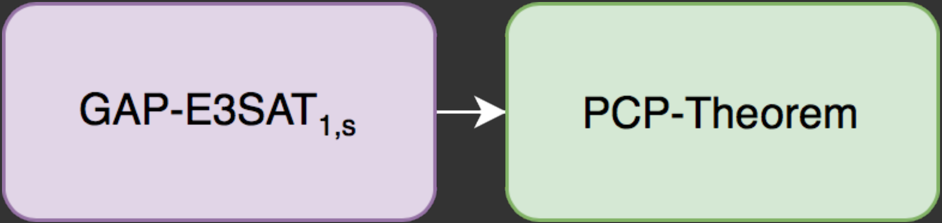
Lemma

There exists $d_0 \in \mathbb{N}$ and $h_0 > 0$, such that there is a polynomial-time constructible family $\{X_n\}_{n\in\mathbb{N}}$ of d_0-regular graphs X_n on n vertices with $h(X_n) \geq h_0$.

PROOF SKETCH

PCP-Theorem

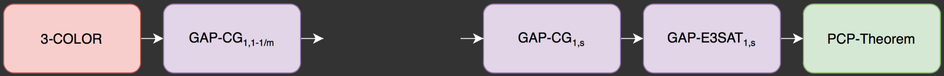
>>> Proof Sketch



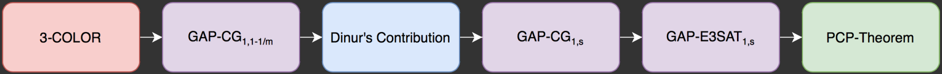
>>> Proof Sketch

>>> Proof Sketch

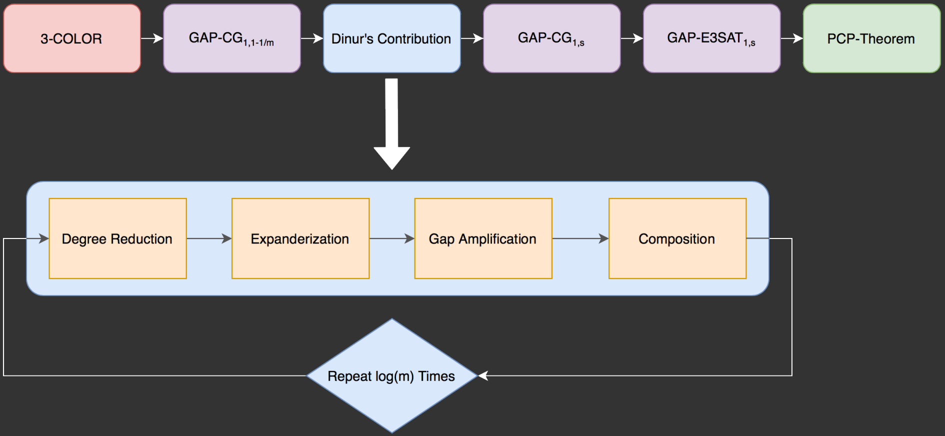
>>> Proof Sketch



>>> Proof Sketch



>>> Proof Sketch



\>>> Dinur's Contribution

Property \ Post-Stage	Init	Pre	Gap Amp (t)	Comp
$\lvert V\rvert$	$\lvert V_0\rvert$	$\leq 2\lvert E_0\rvert$	==	$O(d\lvert V_0\rvert)$
$\lvert E\rvert$	$\lvert E_0\rvert$	$O(\lvert E_0\rvert)$	$O(d^t\lvert E_0\rvert)$	$O(d^t\lvert E_0\rvert)$
$\lvert\Sigma\rvert$	$\tilde{\Sigma}$	Σ_0	$\Sigma_0^{d^t}$	Σ_0
deg(G)	const	$O(d\lvert V\rvert)$	?	-
Regular?	✗	✓	✗	✗
Good Expander?	✗	✓	✗	✗
$\lambda(G)$	-	λ	?	-
NO case gap(G)	gap	$\downarrow$	$O(t \cdot \text{gap})$	$\downarrow$

\>>> Dinur's Contribution

Property \ Post-Stage	Init	Pre	Gap Amp (t)	Comp
$\|V\|$	$\|V_0\|$	$\leq 2\|E_0\|$	==	$O(d\|V_0\|)$
$\|E\|$	$\|E_0\|$	$O(\|E_0\|)$	$O(d^t\|E_0\|)$	$O(d^t\|E_0\|)$
$\|\Sigma\|$	$\tilde{\Sigma}$	Σ_0	$\Sigma_0^{d^t}$	Σ_0
deg(G)	const	$O(d\|V\|)$	?	-
Regular?	✗	✓	✗	✗
Good Expander?	✗	✓	✗	✗
$\lambda(G)$	-	λ	?	-
NO case gap(G)	gap	$\downarrow$	$O(t \cdot \mathsf{gap})$	$\downarrow$

Lemma (Preprocessing Lemma)

There exist constants $0 < \lambda < d$ *and* $\beta_1 > 0$ *such that any constraint graph* G *can be transformed into a constraint graph* G'*, denoted* $G' = \mathrm{prep}(G)$*, such that*

* G' *is* d*-regular with self-loops, and* $\lambda(G') \leq \lambda \leq d$.
* G' *has the same alphabet as* G*, and* $size(G') = \mathcal{O}(size(G))$.
* $\beta_1 \cdot \mathtt{UNSAT}(G) \leq \mathtt{UNSAT}(G') \leq \mathtt{UNSAT}(G)$.

Lemma (Amplification Lemma)

Let $0 < \lambda < d$*, and* $|\Sigma|$ *be constants. There exists a constant* $\beta_2 = \beta_2(\lambda, d, |\Sigma|) > 0$*, such that for every* $t \in \mathbb{N}$ *and for every* d*-regular constraint graph* $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ *with a self-loop on each vertex and* $\lambda(G) \leq \lambda$*,*

$$\mathtt{UNSAT}(G^t) \geq \beta_2 \cdot t \cdot \min\left(\mathtt{UNSAT}(G), \frac{1}{t}\right).$$

Lemma (Amplification Lemma)

Let $0 < \lambda < d$, and $|\Sigma|$ be constants. There exists a constant $\beta_2 = \beta_2(\lambda, d, |\Sigma|) > 0$, such that for every $t \in \mathbb{N}$ and for every d-regular constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ with a self-loop on each vertex and $\lambda(G) \leq \lambda$,

$$\mathtt{UNSAT}(G^t) \geq \beta_2 \cdot t \cdot \min\left(\mathtt{UNSAT}(G), \frac{1}{t}\right).$$

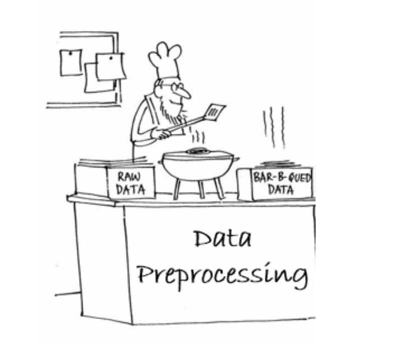
Lemma (Composition Lemma)

Assume the existence of an assignment tester $\mathcal{P}$, with constant rejection probability $\epsilon > 0$, and alphabet $\Sigma_0, |\Sigma_0| = \mathcal{O}(1)$. There exists $\beta_3 > 0$ that depends only on $\mathcal{P}$, such that given any constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$, one can compute, in linear time, the constraint graph $G' = G \circ \mathcal{P}$, such that size$(G') = c(\mathcal{P}, |\Sigma|) \cdot$ size(G), and

$$\beta_3 \cdot \mathtt{UNSAT}(G) \leq \mathtt{UNSAT}(G') \leq \mathtt{UNSAT}(G).$$

PRE-PROCESSING
Degree Reduction + Expanderization

>>> Preprocessing



* Start with a NP-complete problem
* Encode it into constraint graphs
* Transform constraint graph into expander graphs that are easier to work with
* Keep track of the change in gap during transformations

>>> Preprocessing

Two transformations:

1. Degree reduction
    * Blow up number of vertices
    * Transform into a constant degree graph

>>> Preprocessing

Two transformations:

1. Degree reduction
   * Blow up number of vertices
   * Transform into a constant degree graph
2. Expanderize
   * Superimpose a constant degree expander graph
   * Add self-loops to each vertex

>>> Preprocessing

Two transformations:

1. Degree reduction
   * Blow up number of vertices
   * Transform into a constant degree graph
2. Expanderize
   * Superimpose a constant degree expander graph
   * Add self-loops to each vertex

These two steps can be captured by the following:

$$G' = \mathrm{prep}_2(\mathrm{prep}_1(G)).$$

>>> Preprocessing lemma

Lemma

There exist constants $0 < \lambda < d$ and $\beta_1 > 0$ such that any constraint graph G can be transformed into a constraint graph G', denoted $G' = \mathrm{prep}(G)$, such that

* G' *is* d*-regular with self-loops, and* $\lambda(G') \leq \lambda \leq d$.
* G' *has the same alphabet as* G, *and* $size(G') = \mathcal{O}(size(G))$.
* $\beta_1 \cdot \mathtt{UNSAT}(G) \leq \mathtt{UNSAT}(G') \leq \mathtt{UNSAT}(G)$.

>>> Prep_1

Definition

Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\text{prep}_1(G) = \langle (V', E'), \Sigma, \mathcal{C}' \rangle$ is defined as follow:

* For each vertex $v \in V$, $[v] = \{(v, e) | e \in E$ is incident on $v\}$.

>>> Prep_1

Definition

Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\mathrm{prep}_1(G) = \langle (V', E'), \Sigma, \mathcal{C}' \rangle$ is defined as follow:

* For each vertex $v \in V$, $[v] = \{(v, e) | e \in E \text{ is incident on } v\}$.
* Connect all vertices in $[v]$ to form a d_0-regular graph X_v with expansion at least h_0. We denote $E_1 = \cup_{v \in V} E(X_v)$.

$$E_2 = \{\{(v, e), (v', e)\} \mid e = \{v, v'\} \in E\}.$$

Finally, $E' = E_1 \cup E_2$.

>>> Prep_1

Definition

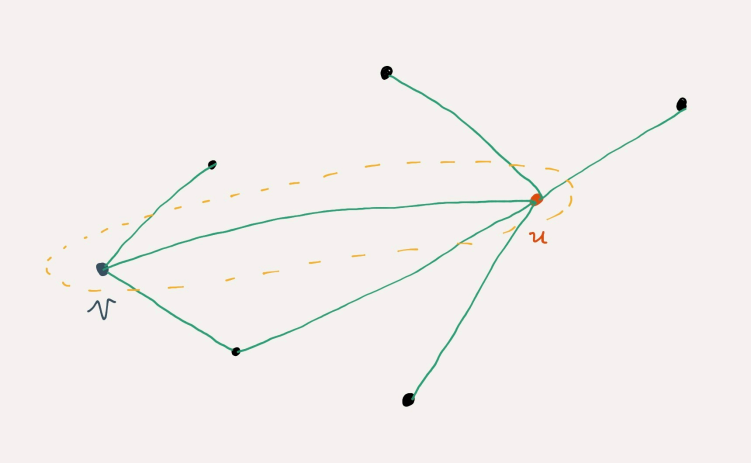
Let $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\text{prep}_1(G) = \langle (V', E'), \Sigma, \mathcal{C}' \rangle$ is defined as follow:

* For each vertex $v \in V$, $[v] = \{(v, e) | e \in E \text{ is incident on } v\}$.
* Connect all vertices in $[v]$ to form a d_0-regular graph X_v with expansion at least h_0. We denote $E_1 = \cup_{v \in V} E(X_v)$.

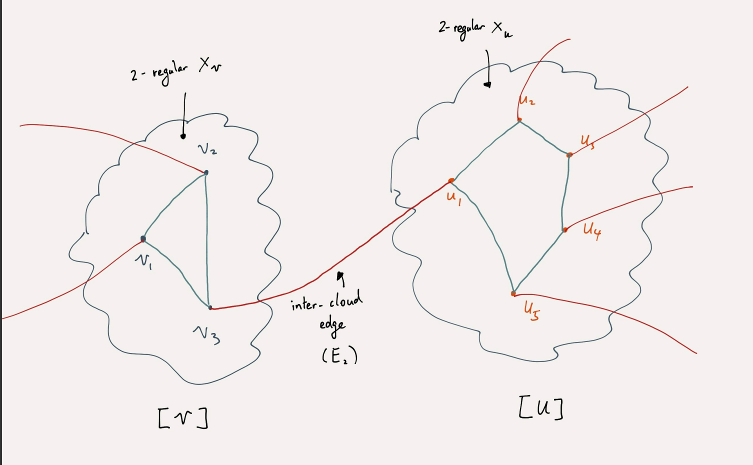
  $$E_2 = \{\{(v, e), (v', e)\} \mid e = \{v, v'\} \in E\}.$$

  Finally, $E' = E_1 \cup E_2$.
* Add equality constraint for each new edge.

>>> Prep_1



>>> Prep_1



2 - regular X_v
v_2
v_1
v_3
[v]
inter- cloud
edge
(E_2)
2- regular X_u
u_2
u_3
u_1
u_4
u_5
[u]

>>> Prep_1

The above transformation gives rise to a constraint graph such that $|V'| < 2|E|$, and

$$c \cdot \texttt{UNSAT}(G) \leq \texttt{UNSAT}(G') \leq \texttt{UNSAT}(G).$$

and moreover, for any assignment $\sigma' : V' \to \Sigma$ let $\sigma : V \to \Sigma$ be defined according to the plurality value,

$$\forall v \in V, \sigma(v) \triangleq \arg\max_{a \in \Sigma} \left\{ \Pr_{(v,e) \in [v]} [\sigma'(v,e) = a] \right\}.$$

Then, $c \cdot \texttt{UNSAT}_\sigma(G) \leq \texttt{UNSAT}(G')$.

>>> Prep$_1$

Proof preliminaries:

* $|E'| \leq d|E|, d = d_0 + 1$
* $\sigma(v)$ is the most popular assigned value in cloud v.

>>> Prep_1

Proof preliminaries:

* $|E'| \leq d|E|, d = d_0 + 1$
* $\sigma(v)$ is the most popular assigned value in cloud v.
* Let $\sigma' : V' \to \Sigma$ be the best assignment for G'.
* $F \subseteq E$ is the set of edges that reject σ.
* $F' \subseteq E'$ is the set of edges that reject σ'.
* $S = \bigcup_{v \in V} \{(v, e) \in [v] \mid \sigma'(v, e) \neq \sigma(v)\}$

>>> Prep_1

Proof preliminaries:

* $|E'| \leq d|E|, d = d_0 + 1$
* $\sigma(v)$ is the most popular assigned value in cloud v.
* Let $\sigma' : V' \to \Sigma$ be the best assignment for G'.
* $F \subseteq E$ is the set of edges that reject σ.
* $F' \subseteq E'$ is the set of edges that reject σ'.
* $S = \bigcup_{v \in V} \{(v, e) \in [v] \mid \sigma'(v, e) \neq \sigma(v)\}$
* Key observation: an edge $e = \{v, v'\} \subseteq F$, the corresponding inter-cloud edge $\{(v, e), (v', e)\} \in E'$ is either in F' or has an end point in S.
* $\implies |F'| + |s| \geq |F| = \alpha \cdot |E|$.
* $\alpha = \mathtt{UNSAT}_\sigma(G) = \frac{|F|}{|E|}, \mathtt{UNSAT}_{\sigma'}(G') = \frac{|F'|}{|E'|}$

>>> Prep_1

Proof preliminaries:

* $|E'| \leq d|E|, d = d_0 + 1$
* $\sigma(v)$ is the most popular assigned value in cloud v.
* Let $\sigma' : V' \to \Sigma$ be the best assignment for G'.
* $F \subseteq E$ is the set of edges that reject σ.
* $F' \subseteq E'$ is the set of edges that reject σ'.
* $S = \bigcup_{v \in V} \{(v, e) \in [v] \mid \sigma'(v, e) \neq \sigma(v)\}$
* Key observation: an edge $e = \{v, v'\} \subseteq F$, the corresponding inter-cloud edge $\{(v, e), (v', e)\} \in E'$ is either in F' or has an end point in S.
* $\implies |F'| + |s| \geq |F| = \alpha \cdot |E|$.
* $\alpha = \text{UNSAT}_\sigma(G) = \frac{|F|}{|E|}, \text{UNSAT}_{\sigma'}(G') = \frac{|F'|}{|E'|}$
* if $\text{gap} = 0, \text{gap}' = 0$
* otherwise,We look at the following two cases:
  1. $|F'| \geq \frac{\alpha}{2}|E|$.
  2. $|F'| < \frac{\alpha}{2}|E|, \implies |S| \geq \frac{\alpha}{2}|E|$.

>>> $\texttt{Prep}_1$

$|F'| \geq \frac{\alpha}{2}|E|$:

$$\implies \alpha' = \frac{|F'|}{|E'|} \geq \frac{\frac{\alpha}{2}|E|}{d|E|} = \frac{\alpha}{2d}$$

$$\implies \texttt{UNSAT}_{\sigma'}(G') \geq \frac{\texttt{UNSAT}_{\sigma}(G)}{2d}$$

>>> Prep_1

$|S| \geq \frac{\alpha}{2}|E|$:

* Let S^v denote the set of vertices in $[v]$ that σ' disagrees with σ.
* $S_a^v = \{(v,e) \in S^u | \sigma'(v,e) = a\}$
* $\implies |S_a^v| \leq \frac{|[v]|}{2}$
* from expander property, $E(S_a^v, [v] \setminus S_a^v) \geq h_0 \cdot |S_a^v|$
* but these are precisely the edges that connect two vertices with a majority value and minority value, meaning it violates the equality constraint for edges within a cloud!

This means we have at least the following amount of edges that belongs to F':

$$\begin{aligned}\sum_{v\in V}\sum_{a\in\Sigma}\frac{h_0}{2}|s_a^v| &= \frac{h_0}{2}\sum_{v\in V}\sum_{a}|s_a^v| \\ &= \frac{h_0}{2}\sum_{v\in V}s^v \\ &= \frac{h_0}{2}|S| \\ &\geq \frac{h_0}{2}\frac{\alpha}{2}|E| \quad \text{(from case 2)} \\ &= \frac{\alpha h_0}{4}|E|\end{aligned}$$

$$\implies \mathtt{UNSAT}_{\sigma'}(G') = \frac{|F'|}{|E'|} \geq \frac{\frac{\alpha h_0}{4}|E|}{|E'|} \geq \frac{\frac{\alpha h_0}{4}|E|}{d|E|} = \frac{h_0}{4d}\alpha = \frac{h_0}{4d}\mathtt{UNSAT}_{\sigma}(G).$$

>>> Prep_2

Definition

Let $G = \langle (V,E), \Sigma, \mathcal{C} \rangle$ be a constraint graph. The constraint graph $\text{prep}_2(G) = \langle (V,E'), \Sigma, \mathcal{C}' \rangle$ as follows.

* Vertices remain the same.
* Let X be a d'_o- regular graph on V and edge set E_1 such that $\lambda(X) < \lambda_0 < d'_0$ (expander graph).Let $E_2 = \{\{v,v\} \mid v \in V\}$ (self-loops). $E' = E \cup E_1 \cup E_2$.
* For constraints, we just add null constrains (always satisfied) for each new edge.

>>> Prep_2

The above transformation leads to the following lemma:

Lemma

There exists global constants $d'_o > \lambda_0 > 0$ such that for any $d-$regular constraint graph G, the constrain graph $G' = prep_2(G)$ has the following properties:

* *G' is $(d + d_0 + 1)$-regular expander with self-loop on every vertex, and $\lambda(G') \leq d + d + \lambda_0 + 1 < \deg(G')$,*
* *$size(G') = \mathcal{O}(size(G))$,*
* *For every $\sigma : V \to \Sigma, \frac{d}{d+d'_0+1} \cdot \mathtt{UNSAT}_\sigma(G) \leq \mathtt{UNSAT}_\sigma(G') \leq \mathtt{UNSAT}_\sigma(G)$.*

>>> Prep_2

* The degree of each vertex is increased by $d'_0 + 1$.
* $|E'|$ is gone up by at most $c' \leq (d + d'_0 + 1)/d$
* constant size increase.

>>> $\texttt{Prep}_2$

* The degree of each vertex is increased by $d'_0 + 1$.
* $|E'|$ is gone up by at most $c' \leq (d + d'_0 + 1)/d$
* constant size increase.
* Fix an assignment $\sigma : V \to \Sigma$,
*

$$\begin{aligned} \texttt{UNSAT}_\sigma(G') &= \frac{\#\ \texttt{edges violated by}\ \sigma}{|E'|} \\ &\geq \frac{|E| \cdot \texttt{UNSAT}_\sigma(G)}{c' \cdot |E|} \\ &= \frac{d}{d + d'_0 + 1} \cdot \texttt{UNSAT}_\sigma(G) \end{aligned}$$

>>> Preprocessing Lemma

Finally, combining the previous two transformation, we have:

$$G' = \texttt{prep}_2(\texttt{prep}_1(G)$$

with the following gap:

$$\texttt{UNSAT}(\texttt{Prep}_1(G)) \leq c \cdot \texttt{UNSAT}(G),$$

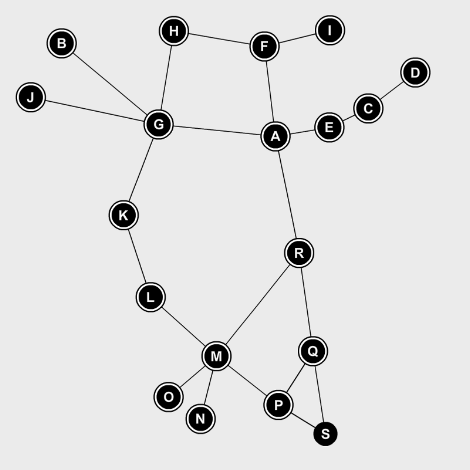
and

$$\texttt{UNSAT}(\texttt{Prep}_2(\texttt{Prep}_1(G))) \leq \frac{d}{d + d_0' + 1} \cdot c \cdot \texttt{UNSAT}(G)$$
$$\implies \beta_1 = \frac{cd}{d + d_0' + 1}$$

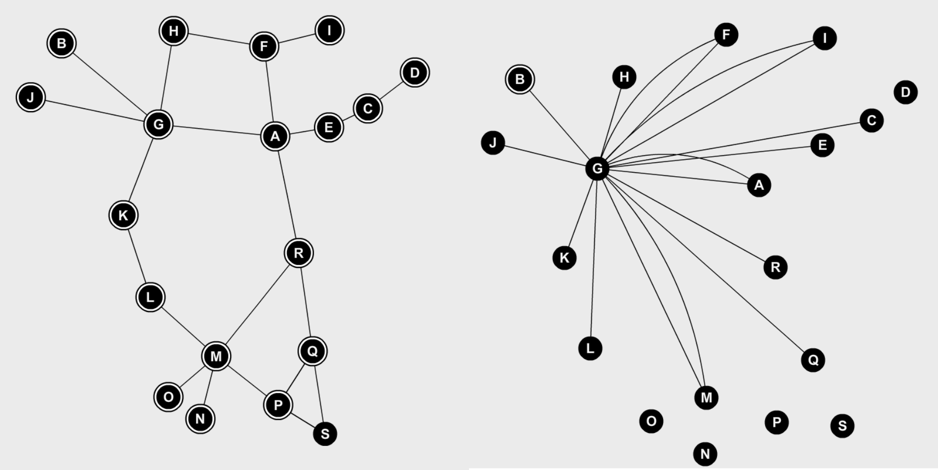
■

GAP AMPLIFICATION

>>> Idea



>>> Idea



\>>> Where were we and where are we going?

INPUT: $(G,\mathcal{C})$ a d-regular constraint (n,d,λ)-expander.

>>> Where were we and where are we going?

INPUT: $(G, \mathcal{C})$ a d-regular constraint (n, d, λ)-expander.
DENOTE: GAP = UNSAT(G)

>>> Where were we and where are we going?

INPUT: $(G, \mathcal{C})$ a d-regular constraint (n, d, λ)-expander.

DENOTE: GAP = UNSAT(G)

INTRODUCE: a fixed constant parameter t.

>>> Where were we and where are we going?

INPUT: $(G, \mathcal{C})$ a d-regular constraint (n, d, λ)-expander.

DENOTE: GAP = UNSAT(G)

INTRODUCE: a fixed constant parameter t.

RECALL: $F \subseteq E$ is the set of edges failing $\mathcal{C}$

>>> Where were we and where are we going?

INPUT: $(G, \mathcal{C})$ a d-regular constraint (n, d, λ)-expander.
DENOTE: GAP = UNSAT(G)
INTRODUCE: a fixed constant parameter t.
RECALL: $F \subseteq E$ is the set of edges failing $\mathcal{C}$
AFTERMATH:

* Polynomial increase in graph size.
* GAP$'$ increases $\begin{cases} 0 & \text{If } \mathtt{GAP} = 0 \\ \geq \frac{t}{O(1)} \times \min(\mathtt{GAP}, \frac{1}{t}) & \text{Else} \end{cases}$
* Alphabet blows up $\Sigma' = \Sigma^{d^t}$.
* λ' and d' decrease in value (ignore).

Definition

An ``One or More Random Walk'' (OoM) in a regular graph $G = (V, E)$:

1. Picks a random vertex $a \in V$ to start at
2. Takes a step along a random edge of current vertex
3. Decides to stop with probability $1/t$. Otherwise step 2
4. Names the final vertex b.

Definition

An ``One or More Random Walk'' (OoM) in a regular graph $G = (V, E)$:

1. Picks a random vertex $a \in V$ to start at
2. Takes a step along a random edge of current vertex
3. Decides to stop with probability $1/t$. Otherwise step 2
4. Names the final vertex b.

Definition

A ``Zero or More Random Walk'' (ZoM) in a regular graph $G = (V, E)$, starting from a vertex v:

1. Stop with probability $1/t$
2. Take a step along a random edge of current vertex
3. Go to step 1

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$
* $(a, b) \in E'$ correspond to *walks* in G of $\texttt{dist}_G(a, b) \approx t$

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$
* $(a, b) \in E'$ correspond to *walks* in G of $\text{dist}_G(a, b) \approx t$
* $\Sigma' = \Sigma^{1+d+d^2+..+d^t}$
    * upper bound on the number of vertices at distance at most t from a given vertex in d-regular graph G

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$
* $(a,b) \in E'$ correspond to *walks* in G of $\text{dist}_G(a,b) \approx t$
* $\Sigma' = \Sigma^{1+d+d^2+..+d^t}$
  * upper bound on the number of vertices at distance at most t from a given vertex in d-regular graph G
* $\sigma' : V' \to \Sigma'$ (each vertex is mapped to a string)
  * a has an opinion $(\sigma'(a)_b \in \Sigma)$ on the value for each $b \in V_{t\text{-neighbourhood } a}$ in its string
  * σ' hereafter is optimal assignment for G'

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$
* $(a,b) \in E'$ correspond to *walks* in G of $\mathrm{dist}_G(a,b) \approx t$
* $\Sigma' = \Sigma^{1+d+d^2+..+d^t}$
  * upper bound on the number of vertices at distance at most t from a given vertex in d-regular graph G
* $\sigma' : V' \to \Sigma'$ (each vertex is mapped to a string)
  * a has an opinion $(\sigma'(a)_b \in \Sigma)$ on the value for each $b \in V_{t\text{-neighbourhood } a}$ in its string
  * σ' hereafter is optimal assignment for G'
* $\mathcal{C}'(a,b)$ $\begin{cases} (\sigma'(a)_u, \sigma'(b)_v) \in \mathcal{C} & \forall (u,v) \in E_{\mathtt{walk}(a,b)} \\ \sigma'(a)_v = \sigma'(b)_v \in \mathcal{C} & \forall v \in V_{\mathtt{walk}(a,b)} \end{cases}$

>>> Graph Powering $G' = G^{(t)}$

* $V' = V$
* $(a, b) \in E'$ correspond to *walks* in G of $\mathtt{dist}_G(a, b) \approx t$
* $\Sigma' = \Sigma^{1+d+d^2+..+d^t}$
  * upper bound on the number of vertices at distance at most t from a given vertex in d-regular graph G
* $\sigma' : V' \to \Sigma'$ (each vertex is mapped to a string)
  * a has an opinion ($\sigma'(a)_b \in \Sigma$) on the value for each $b \in V_{t\text{-neighbourhood } a}$ in its string
  * σ' hereafter is optimal assignment for G'
* $\mathcal{C}'(a, b) \begin{cases} (\sigma'(a)_u, \sigma'(b)_v) \in \mathcal{C} & \forall (u, v) \in E_{\mathtt{walk}(a,b)} \\ \sigma'(a)_v = \sigma'(b)_v \in \mathcal{C} & \forall v \in V_{\mathtt{walk}(a,b)} \end{cases}$
* $\sigma(v) \doteq$ $\mathtt{maxarg}_{a \in \Sigma} \{ \ \mathbb{P}[\mathtt{ZoM}\ v \to w \ \mathtt{s.t.}\ \sigma'(w)_v = a \mid \mathtt{stops} \leq t \ \mathtt{steps}] \ \}$

>>> How do we generate Edges?

Perform the following Random Process/Verifier

1. Perform a One or More Random Walk (OoM)
2. Denote the start vertex by a and the end vertex by b
3. For each $u \to v$ in path from a to b
   * Reject if $\text{dist}_G(u, a) \leq t$
     and $\text{dist}_G(v, b) \leq t$
     and $(\sigma'(a)_u, \sigma'(b)_v) \notin \mathcal{C}(u, v)$
   * Accept o.w.

>>> How do we generate Edges?

BAD SIDE EFFECTS:

* $|E'| = \Omega(|E|^2)$
* produce a *probability distribution over all possible edges* making the resulting graph a weighted constraint graph with possible parallel edges

>>> How do we generate Edges?

BAD SIDE EFFECTS:

* $|E'| = \Omega(|E|^2)$
* produce a *probability distribution over all possible edges* making the resulting graph a weighted constraint graph with possible parallel edges

FIX:

* throws away any $(a \to b) \in E'$ if $\text{dist}_G(a, b) > 10\log(|\Sigma|)t \doteq B$.
    * Why can we throw these edges? Verifier always Accepts them.
    * Effect? reduce graph size and gap $\uparrow \approx \frac{|F'|}{|E'|\downarrow}$
* replace each weighted edge with multiple parallel edges appropriately
    * Effect? back to an unweighted constraint graph

Definition (Faulty Step)

Within the verifier' s OoM random walk, we say a particular step $u \to v$ is faulty if:

1. $(u \to v) \in F$ (recall $F \subseteq E$ is the edges failing $\mathcal{C}$)
2. $\mathtt{dist}_G(u \to a) \le t$ and $\sigma'(a)_u = \sigma(u)$
3. $\mathtt{dist}_G(v \to b) \le t$ and $\sigma'(b)_v = \sigma(v)$

Definition (Faulty Step)

Within the verifier' s OoM random walk, we say a particular step $u \to v$ is faulty if:

1. $(u \to v) \in F$ (recall $F \subseteq E$ is the edges failing $\mathcal{C}$)
2. $\mathrm{dist}_G(u \to a) \leq t$ and $\sigma'(a)_u = \sigma(u)$
3. $\mathrm{dist}_G(v \to b) \leq t$ and $\sigma'(b)_v = \sigma(v)$

Definition (# Faulty Steps)

Let N be the r.v. counting the number of faulty steps. If $N > 0$, the verifier rejects

$\mathbb{P}[N > 0]$ is large $\implies$ GAP' large

Definition (Faulty* Step)

A step $u \to v$ is faulty* if (1) it is faulty, and (2) $\mathtt{dist}_G(a, b) \leq B$

Definition (Faulty* Step)

A step $u \to v$ is faulty* if (1) it is faulty, and (2) $\texttt{dist}_G(a, b) \leq B$

Definition (# Faulty* Steps)

Let N^* be a r.v. denoting the number of faulty* steps in the $a \to b$ walk.

Definition (Faulty* Step)

A step $u \to v$ is faulty* if (1) it is faulty, and (2) $\texttt{dist}_G(a, b) \leq B$

Definition (# Faulty* Steps)

Let N^* be a r.v. denoting the number of faulty* steps in the $a \to b$ walk.

Definition (# Violating Steps)

Let N_F be a r.v. denoting the number of steps $\in F$

Definition (Faulty* Step)

A step $u \to v$ is faulty* if (1) it is faulty, and (2) $\texttt{dist}_G(a, b) \leq B$

Definition (# Faulty* Steps)

Let N^* be a r.v. denoting the number of faulty* steps in the $a \to b$ walk.

Definition (# Violating Steps)

Let N_F be a r.v. denoting the number of steps $\in F$

Definition (# Steps)

Let L be a r.v. denoting the number of steps the $a \to b$ walk.

We need to show that $\mathrm{GAP}' \geq \frac{t}{O(1)} \cdot \frac{|F|}{|E|}$

Lemma

For any non-negative r.v. N, $\mathbb{P}[N > 0] \geq \frac{\mathbb{E}[N]^2}{\mathbb{E}[N^2]}$

Proof.

Using Cauchy-Schwarz

$$\mathbb{E}[N] = \mathbb{E}[N \cdot \mathbb{1}[N > 0]] \leq \sqrt{\mathbb{E}[N^2]}\sqrt{\mathbb{E}[\mathbb{1}[N > 0]]^2} = \sqrt{\mathbb{E}[N^2]}\sqrt{\mathbb{P}[N > 0]}$$

We need to show that $\mathrm{GAP}' \geq \frac{t}{O(1)} \cdot \frac{|F|}{|E|}$

Lemma

For any non-negative r.v. N, $\mathbb{P}[N > 0] \geq \frac{\mathbb{E}[N]^2}{\mathbb{E}[N^2]}$

Proof.

Using Cauchy-Schwarz

$$\mathbb{E}[N] = \mathbb{E}[N \cdot \mathbb{1}[N > 0]] \leq \sqrt{\mathbb{E}[N^2]}\sqrt{\mathbb{E}[\mathbb{1}[N > 0]]^2} = \sqrt{\mathbb{E}[N^2]}\sqrt{\mathbb{P}[N > 0]}$$

Lemma

$\mathbb{E}[N] \geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Lemma

$\mathbb{E}[N^*] \geq \frac{t}{8|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Lemma

$\mathbb{E}[N^{*2}] \leq O(1) \cdot t \cdot \frac{|F|}{|E|}$

Lemma

Let (u, v) be a fixed edge in the regular graph $G = (V, E)$. Do a OoM in G, conditioned on making exactly k $u \to v$ steps. Then:

* *The distribution on the final vertex b is the same as if we did a ZoM starting from v.*
* *The distribution on the initial vertex a is same as if we did an ZoM starting from u.*
* *a and b are independent.*

Lemma
$\mathbb{E}[N] \geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Proof.

$$
\begin{aligned}
\mathbb{E}[N_{u\to v}] &= \sum_{k\geq 1} \mathbb{E}[N_{u\to v} \mid \texttt{exactly } k \ \ u \to v \ \texttt{ steps}] &(1)\\
&\quad \times \mathbb{P}[\texttt{exactly } k \ \ u \to v \ \texttt{ steps}] &(2)\\
&= \sum_{k\geq 1} k \cdot \mathbb{P}[u \to v \ \texttt{ is faulty} \mid \texttt{exactly } k \ \ u \to v \ \texttt{ steps}] &(3)\\
&\quad \times \mathbb{P}[\texttt{exactly } k \ \ u \to v \ \texttt{ steps}] &(4)
\end{aligned}
$$

Lemma

$\mathbb{E}[N] \geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Proof.

$$\mathbb{E}[N_{u\to v}] = \sum_{k\geq 1} \mathbb{E}[N_{u\to v} \mid \texttt{exactly } k \ u \to v \texttt{ steps}] \tag{1}$$
$$\times \mathbb{P}[\texttt{exactly } k \ u \to v \texttt{ steps}] \tag{2}$$
$$= \sum_{k\geq 1} k \cdot \mathbb{P}[u \to v \texttt{ is faulty} \mid \texttt{exactly } k \ u \to v \texttt{ steps}] \tag{3}$$
$$\times \mathbb{P}[\texttt{exactly } k \ u \to v \texttt{ steps}] \tag{4}$$

$$\mathbb{P}[u \to v \texttt{ is faulty} \mid \texttt{exactly } k \ u \to v \texttt{ steps}] \tag{5}$$
$$= \left(\mathbb{P}[\texttt{dist}_G(w \to a) \leq t \texttt{ and } \sigma'(w)_u = \sigma(u)]\right)^2 \tag{6}$$

Lemma

$$\mathbb{E}[N^*] \geq \frac{t}{8|\Sigma|^2} \cdot \frac{|F|}{|E|}$$

Proof.

Using $\mathbb{E}[N] \geq \frac{t}{4|\Sigma|^2} \cdot \frac{|F|}{|E|}$

Lemma

Let G be an (n, d, λ)-expander and $F \subset E(G) = E$, then the probability that a random walk, starting in the zero$^{\text{th}}$ step from a random edge in F, passes through F on its t^{th} step, is bounded by

$$\frac{|F|}{|E|} + \left(\frac{\lambda}{d}\right)^{t-1}$$

ENGLISH: Vertices from a Random Walk in an Expander are as if picked Independently from V.

Lemma
$\mathbb{E}[N^{*2}] \leq O(1) \cdot t \cdot \frac{|F|}{|E|}$

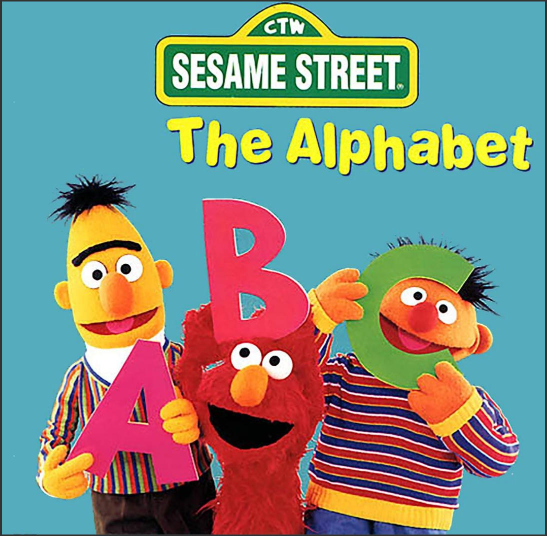
Proof.
Let $N_F = \sum_{i=1}^{\infty} \mathbb{1}[i^{\text{th}}\texttt{ step in } F] = \sum_{i=1}^{\infty} \zeta_i$.

$$\mathbb{E}[N^{*2}] \leq \mathbb{E}[N_F^2] = \sum_{i,j=1}^{\infty} \mathbb{E}[\zeta_i \cdot \zeta_j] \leq 2 \sum_{i=1}^{\infty} \mathbb{P}[\zeta_i = 1] \cdot \sum_{j \geq i} \mathbb{P}[\zeta_j = 1 \mid \zeta_i = 1] \tag{7}$$

$$\mathbb{P}[\zeta_j = 1 \mid \zeta_i = 1] \tag{8}$$
$$= \mathbb{P}[\texttt{the walk is } j - i \texttt{ steps more}] \tag{9}$$
$$\times \mathbb{P}[\texttt{a walk from random vertex} \in F \texttt{ takes } (j-i)^{\text{th}} \texttt{ step in } F] \tag{10}$$
$$\leq (1 - 1/t)^{j-i} \left(\frac{|F|}{|E|} + \left(\frac{\lambda}{d} \right)^{j-i-1} \right) \tag{11}$$

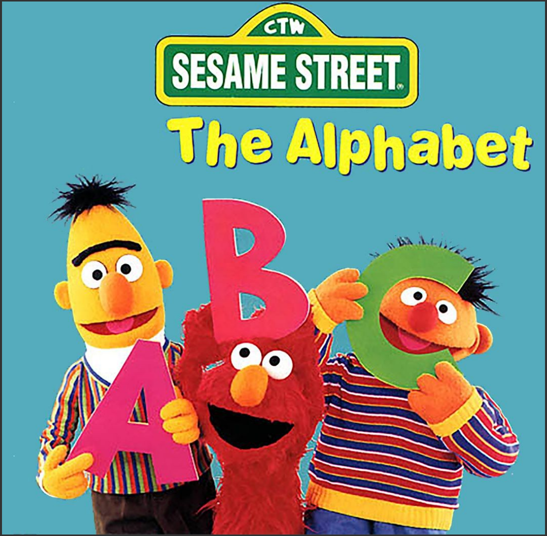
ALPHABET REDUCTION / COMPOSITION

\>>> Alphabet Reduction



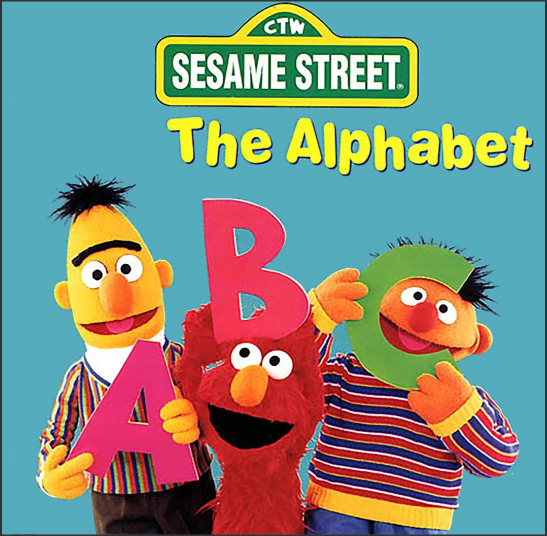
* The amplification step does a great job of increasing the gap

>>> Alphabet Reduction



* The amplification step does a great job of increasing the gap
* ... at the cost of blowing up the assignment alphabet superexponentially

>>> Alphabet Reduction



* The amplification step does a great job of increasing the gap
* ... at the cost of blowing up the assignment alphabet superexponentially
* We must construct a method of constraining the assignment alphabet. This is called the Alphabet Reduction/Composition step.

>>> Assignment Tester: Motivation

* Ultimately, we want to transform a constraint graph output by the amplification step into a new constraint graph on a fixed-size alphabet Σ_0.

>>> Assignment Tester: Motivation

* Ultimately, we want to transform a constraint graph output by the amplification step into a new constraint graph on a fixed-size alphabet Σ_0.
* Due to the recursive nature of the overall gap amplification procedure, individual constraints are aggregated iteratively with an alphabet reduction step after each iteration

>>> Assignment Tester: Motivation

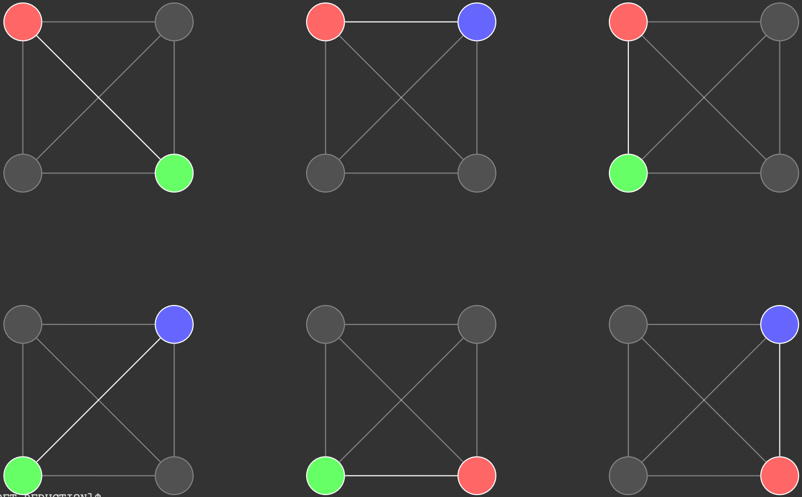
* Ultimately, we want to transform a constraint graph output by the amplification step into a new constraint graph on a fixed-size alphabet Σ_0.
* Due to the recursive nature of the overall gap amplification procedure, individual constraints are aggregated iteratively with an alphabet reduction step after each iteration
* We can therefore condition on the satisfiability of constraints in previous iterations to encode the satisfiability of the next constraints in a fixed alphabet. However, we must maintain two guarantees:

>>> Assignment Tester: Motivation

* Ultimately, we want to transform a constraint graph output by the amplification step into a new constraint graph on a fixed-size alphabet Σ_0.
* Due to the recursive nature of the overall gap amplification procedure, individual constraints are aggregated iteratively with an alphabet reduction step after each iteration
* We can therefore condition on the satisfiability of constraints in previous iterations to encode the satisfiability of the next constraints in a fixed alphabet. However, we must maintain two guarantees:
  1. Completeness: If the constraint graph originally had an unsat value of 0, the alphabet reduction step must output a constraint graph with unsat value 0.
  2. Soundness: If the constraint graph was not originally satisfiable and had gap g, the constraint graph output by the alphabet reduction step has gap ϵg for some $\epsilon > 0$.

\>>> Assignment Tester: It's not so simple

* Unfortunately, these guarantees are not enough.
* The issue: the satisfiability of constraints individually *does not* imply the satisfiability of constraints simultaneously.



>>> Assignment Tester: The definition

Definition (Assignment Tester)

An *Assignment Tester* with alphabet Σ_0 and rejection probability $\epsilon > 0$ is an algorithm $\mathcal{P}$ whose input is a circuit Φ over Boolean variables X, and whose output is a constraint graph $G = \langle (V, E), \Sigma_0, \mathcal{C} \rangle$ such that $V \supset X$, and such that the following hold. Let $V' \triangleq V \setminus X$ and let $a : X \to \{0, 1\}$ be an assignment.

* Completeness: If $a \in \mathsf{SAT}(\Phi)$, there exists $b : V' \to \Sigma_0$ such that $\mathrm{UNSAT}_{a \cup b}(G) = 0$.
* Soundness: If $a \notin \mathsf{SAT}(\Phi)$, then for all $b : V' \to \Sigma_0$, $\mathrm{UNSAT}_{a \cup b}(G) \geq \epsilon r_d(a, \mathsf{SAT}(\Phi))$.

where $r_d : \{0, 1\}^n \times \{0, 1\}^n \to \mathbf{R}_{\geq 0} : (x, y) \mapsto \frac{1}{n} d_{\mathtt{Hamming}}(x, y)$, and $d_{\mathtt{Hamming}}$ denotes the Hamming distance, which trivially satisfies the properties of a metric.

>>> Assignment Tester: What's the damage?

* Good news: Assignment Testers exist

>>> Assignment Tester: What's the damage?

* Good news: Assignment Testers exist
* Ambiguous news: Assignment Testers increase the number of constraints, thereby increasing the constraint graph size

>>> Assignment Tester: What's the damage?

* Good news: Assignment Testers exist
* Ambiguous news: Assignment Testers increase the number of constraints, thereby increasing the constraint graph size
* Best news: *It's not so bad.* We know that the alphabet size of the constraint graph at the beginning of each alphabet reduction step is either some initial arbitrary alphabet Σ or the fixed size ``target" alphabet Σ_0. But, $|\Sigma|, |\Sigma_0| \in O(1)$! Therefore, the size of the constraint graph constructed for each constraint is some function $c(\mathcal{P}, |\Sigma|)$, where $\mathcal{P}$ is the assignment tester. Note, neither $\mathcal{P}$ or $|\Sigma|$ depend on the size of the graph!

>>> Composition: Preliminaries

* So far we've discussed how constraints are transformed into constraint graphs over a fixed alphabet Σ_0, using an assignment tester $\mathcal{P}$.

>>> Composition: Preliminaries

* So far we've discussed how constraints are transformed into constraint graphs over a fixed alphabet Σ_0, using an assignment tester $\mathcal{P}$.
* Now we turn to the problem of using these assignment testers to transform the entire constraint graph into a new one, over alphabet Σ_0. This is called *composition*.

>>> Composition: Preliminaries

* So far we've discussed how constraints are transformed into constraint graphs over a fixed alphabet Σ_0, using an assignment tester $\mathcal{P}$.
* Now we turn to the problem of using these assignment testers to transform the entire constraint graph into a new one, over alphabet Σ_0. This is called *composition*.
* Before we begin, we state a few definitions concerning error-correcting codes:

Definition (Linear Dimension)

An error correcting code $\mathsf{Enc} : \Sigma \to \{0,1\}^\ell$ is said to have *linear dimension* if $\ell \in O(\log_2 |\Sigma|)$.

>>> Composition: Preliminaries

* So far we've discussed how constraints are transformed into constraint graphs over a fixed alphabet Σ_0, using an assignment tester $\mathcal{P}$.
* Now we turn to the problem of using these assignment testers to transform the entire constraint graph into a new one, over alphabet Σ_0. This is called *composition*.
* Before we begin, we state a few definitions concerning error-correcting codes:

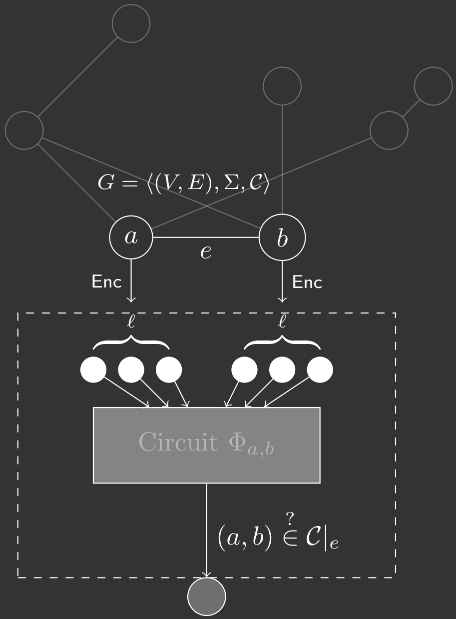
Definition (Linear Dimension)

An error correcting code $\mathsf{Enc} : \Sigma \to \{0,1\}^\ell$ is said to have *linear dimension* if $\ell \in O(\log_2 |\Sigma|)$.

Definition (Relative Distance)

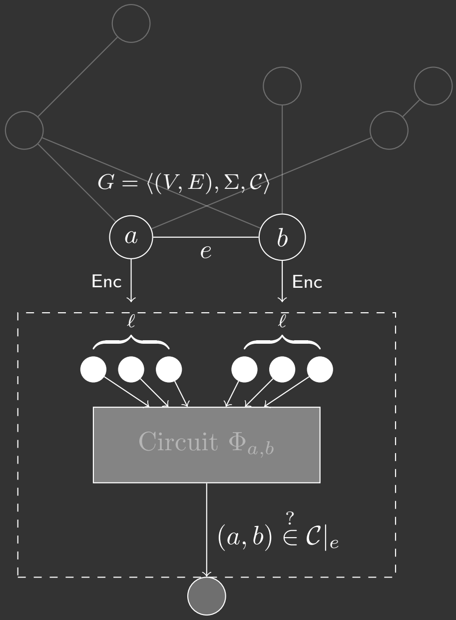
An error correcting code $\mathsf{Enc} : \Sigma \to \{0,1\}^\ell$ is said to *have relative distance* ρ if for every $a_1, a_2 \in \Sigma$ with $a_1 \neq a_2$, $d_{\mathtt{Hamming}}(\mathsf{Enc}(a_1), \mathsf{Enc}(a_2)) \geq \rho\ell$ (or equivalently, $d_r(\mathsf{Enc}(a_1), \mathsf{Enc}(a_2)) \geq \rho$).

>>> Composition: Step 1 -- Robustization



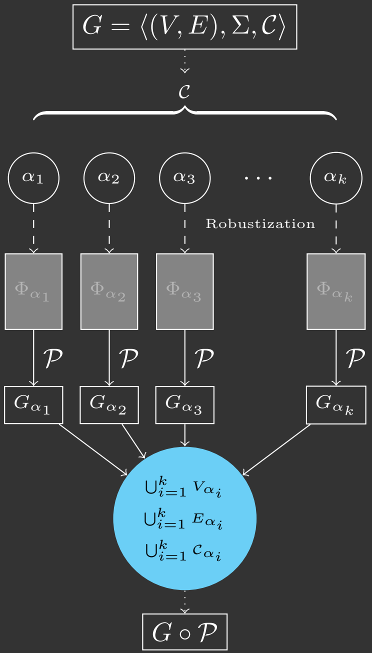
* Input: Constraint graph G
* Given an error correcting code Enc with
  1. Linear dimension, ℓ
  2. Relative distance $\rho > 0$

>>> Composition: Step 1 -- Robustization



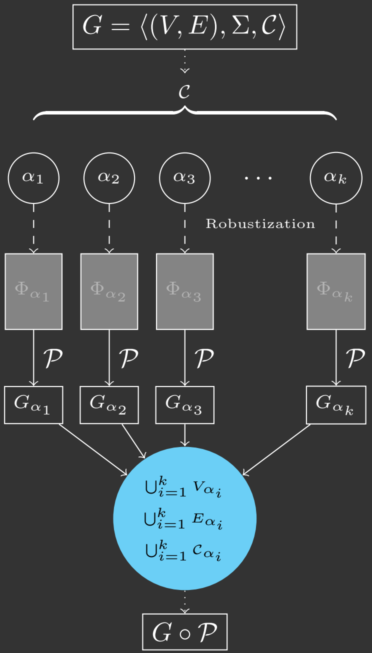
* Input: Constraint graph G
* Given an error correcting code Enc with
  1. Linear dimension, ℓ
  2. Relative distance $\rho > 0$
* Output: For each constraint in $\mathcal{C}$, a circuit on $2\ell \in O(1)$ variables as shown on the left.

\>>> Composition: Step 2 -- Constraint graph composition



* Input: Boolean circuits Φ_{α_i} for each constraint $\alpha_i \in \mathcal{C}$, produced by the robustization
* Given an Assignment Tester $\mathcal{P}$

>>> Composition: Step 2 -- Constraint graph composition



* Input: Boolean circuits Φ_{α_i} for each constraint $\alpha_i \in \mathcal{C}$, produced by the robustization
* Given an Assignment Tester $\mathcal{P}$
* Output: The Σ_0-alphabet constraint graph encoding the original input constraint graph, written $G \circ \mathcal{P}$

>>> Alphabet Reduction Lemma

Given the following:

1. A constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$

>>> Alphabet Reduction Lemma

Given the following:

1. A constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$
2. An Assignment Tester $\mathcal{P}$ with a target fixed-size alphabet Σ_0

>>> Alphabet Reduction Lemma

Given the following:

1. A constraint graph $G = \langle (V, E), \Sigma, \mathcal{C} \rangle$
2. An Assignment Tester $\mathcal{P}$ with a target fixed-size alphabet Σ_0

There exists $\beta_3 \in O(1) > 0$ such that

* $\beta_3 \texttt{UNSAT}(G) \leq \texttt{UNSAT}(G \circ \mathcal{P}) \leq \texttt{UNSAT}(G)$
* $G \circ \mathcal{P}$ can be computed in $O(|G|)$ time, and $|G \circ \mathcal{P}| \in O(|G|)$

>>> Alphabet Reduction Lemma: Proof of complexity

* Each circuit Φ_{α_i} is over 2ℓ nodes. Therefore, the circuits are simulated in $2^{O(2\ell)}$ time.
* However, ℓ is constant, so the circuit are simulated in constant time.

>>> Alphabet Reduction Lemma: Proof of complexity

* Each circuit Φ_{α_i} is over 2ℓ nodes. Therefore, the circuits are simulated in $2^{O(2\ell)}$ time.
* However, ℓ is constant, so the circuit are simulated in constant time.
* As explained earlier, the size of each constraint graph G_{α_i} is $c(\mathcal{P}, |\Sigma|) \in O(1)$. Therefore, the size of $G \circ \mathcal{P}$ is $c(\mathcal{P}, |\Sigma|)|G| \in O(|G|)$.

>>> Alphabet Reduction Lemma: Proof of complexity

* Each circuit Φ_{α_i} is over 2ℓ nodes. Therefore, the circuits are simulated in $2^{O(2\ell)}$ time.
* However, ℓ is constant, so the circuit are simulated in constant time.
* As explained earlier, the size of each constraint graph G_{α_i} is $c(\mathcal{P}, |\Sigma|) \in O(1)$. Therefore, the size of $G \circ \mathcal{P}$ is $c(\mathcal{P}, |\Sigma|)|G| \in O(|G|)$.
* Likewise, union linearly-many constraint graphs (since $|G \circ \mathcal{P}| \in O(|G|)$), so the overall time complexity is $O(|G|)$.

>>> Alphabet Reduction Lemma: Proof of inequality (RHS)

We will now prove that $\mathtt{UNSAT}(G \circ \mathcal{P}) \leq \mathtt{UNSAT}(G)$.

* Let $\sigma : V \to \Sigma$ be an optimal assignment for G, such that $\mathtt{UNSAT}_\sigma(G) = \mathtt{UNSAT}(G)$
* We define an assignment $\sigma' : V' \to \Sigma_0$ on the variables of $G \circ \mathcal{P}$ such that $\sigma'([v]) = \mathsf{Enc}(\sigma(v)) \in \{0, 1\}^\ell$.

>>> Alphabet Reduction Lemma: Proof of inequality (RHS)

We will now prove that $\texttt{UNSAT}(G \circ \mathcal{P}) \leq \texttt{UNSAT}(G)$.

* Let $\sigma : V \to \Sigma$ be an optimal assignment for G, such that $\texttt{UNSAT}_{\sigma}(G) = \texttt{UNSAT}(G)$
* We define an assignment $\sigma' : V' \to \Sigma_0$ on the variables of $G \circ \mathcal{P}$ such that $\sigma'([v]) = \mathsf{Enc}(\sigma(v)) \in \{0,1\}^{\ell}$.
* Since $[v] \cup [w] \subset V_{(v,w)}$, it remains to define the assignment σ' on $\cup_{e=(v,w)\in E}(V_e \setminus ([v] \cup [w])) \triangleq B$.

>>> Alphabet Reduction Lemma: Proof of inequality (RHS)

We will now prove that $\mathtt{UNSAT}(G \circ \mathcal{P}) \leq \mathtt{UNSAT}(G)$.

* Let $\sigma : V \to \Sigma$ be an optimal assignment for G, such that $\mathtt{UNSAT}_\sigma(G) = \mathtt{UNSAT}(G)$
* We define an assignment $\sigma' : V' \to \Sigma_0$ on the variables of $G \circ \mathcal{P}$ such that $\sigma'([v]) = \mathsf{Enc}(\sigma(v)) \in \{0,1\}^\ell$.
* Since $[v] \cup [w] \subset V_{(v,w)}$, it remains to define the assignment σ' on $\cup_{e=(v,w)\in E}(V_e \setminus ([v] \cup [w])) \triangleq B$.
* If σ satisfies a constraint $c(e) \in \mathcal{C}$ for some edge $e = (v,w)$,
    * $\implies$ there must be a satisfying assignment over $V_e \setminus ([v] \cup [w])$ for all constraints in G_e by the completeness of $\mathcal{P}$

>>> Alphabet Reduction Lemma: Proof of inequality (RHS)

We will now prove that $\text{UNSAT}(G \circ \mathcal{P}) \leq \text{UNSAT}(G)$.

* Let $\sigma : V \to \Sigma$ be an optimal assignment for G, such that $\text{UNSAT}_\sigma(G) = \text{UNSAT}(G)$
* We define an assignment $\sigma' : V' \to \Sigma_0$ on the variables of $G \circ \mathcal{P}$ such that $\sigma'([v]) = \mathsf{Enc}(\sigma(v)) \in \{0,1\}^\ell$.
* Since $[v] \cup [w] \subset V_{(v,w)}$, it remains to define the assignment σ' on $\cup_{e=(v,w)\in E}(V_e \setminus ([v] \cup [w])) \triangleq B$.
* If σ satisfies a constraint $c(e) \in \mathcal{C}$ for some edge $e = (v,w)$,
  * $\implies$ there must be a satisfying assignment over $V_e \setminus ([v] \cup [w])$ for all constraints in G_e by the completeness of $\mathcal{P}$
* Else,
  * We define σ' over these inputs arbitrarily. Then, $\text{UNSAT}_{\sigma'}(G') \leq \text{UNSAT}_\sigma(G)$.

>>> Alphabet Reduction Lemma: Proof of inequality (RHS)

We will now prove that $\mathtt{UNSAT}(G \circ \mathcal{P}) \le \mathtt{UNSAT}(G)$.

* Let $\sigma : V \to \Sigma$ be an optimal assignment for G, such that $\mathtt{UNSAT}_\sigma(G) = \mathtt{UNSAT}(G)$
* We define an assignment $\sigma' : V' \to \Sigma_0$ on the variables of $G \circ \mathcal{P}$ such that $\sigma'([v]) = \mathsf{Enc}(\sigma(v)) \in \{0,1\}^\ell$.
* Since $[v] \cup [w] \subset V_{(v,w)}$, it remains to define the assignment σ' on $\cup_{e=(v,w)\in E}(V_e \setminus ([v] \cup [w])) \triangleq B$.
* If σ satisfies a constraint $c(e) \in \mathcal{C}$ for some edge $e = (v, w)$,
  * $\implies$ there must be a satisfying assignment over $V_e \setminus ([v] \cup [w])$ for all constraints in G_e by the completeness of $\mathcal{P}$
* Else,
  * We define σ' over these inputs arbitrarily. Then, $\mathtt{UNSAT}_{\sigma'}(G') \le \mathtt{UNSAT}_\sigma(G)$.

So, we have shown that

$$\mathtt{UNSAT}(G') = \min_{\tilde{\sigma}'} \mathtt{UNSAT}_{\tilde{\sigma}'}(G') \le \mathtt{UNSAT}_{\sigma'}(G') \le \mathtt{UNSAT}_\sigma(G) = \mathtt{UNSAT}(G)$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS)

It remains to show that $\beta_3 \texttt{UNSAT}(G) \leq \texttt{UNSAT}(G \circ \mathcal{P})$ for some $\beta_3 > 0$.

* Let $\sigma' : V' \to \Sigma_0$ be an optimal assignment for $G' = G \circ \mathcal{P}$.

>>> Alphabet Reduction Lemma: Proof of inequality (LHS)

It remains to show that $\beta_3 \mathtt{UNSAT}(G) \leq \mathtt{UNSAT}(G \circ \mathcal{P})$ for some $\beta_3 > 0$.

* Let $\sigma' : V' \to \Sigma_0$ be an optimal assignment for $G' = G \circ \mathcal{P}$.
* From σ' we construct an assignment on G, $\sigma : V \to \Sigma$, such that $\sigma(v) = \min_{s \in \Sigma} d_r(\sigma'([v]), \mathsf{Enc}(s))$. Denote by F the set of edges whose constraints are falsified by σ.

>>> Alphabet Reduction Lemma: Proof of inequality (LHS)

It remains to show that $\beta_3 \texttt{UNSAT}(G) \leq \texttt{UNSAT}(G \circ \mathcal{P})$ for some $\beta_3 > 0$.

* Let $\sigma' : V' \to \Sigma_0$ be an optimal assignment for $G' = G \circ \mathcal{P}$.
* From σ' we construct an assignment on G, $\sigma : V \to \Sigma$, such that $\sigma(v) = \min_{s \in \Sigma} d_r(\sigma'([v]), \mathsf{Enc}(s))$. Denote by F the set of edges whose constraints are falsified by σ.
* For $(v, w) \in F$, $c((v, w))$ is falsified by σ, so the restriction of σ' to $[v] \cup [w]$ must be the closest assignment to $\mathsf{SAT}(\tilde{c}((v, w)))$ by the construction of σ.

>>> Alphabet Reduction Lemma: Proof of inequality (LHS)

It remains to show that $\beta_3 \mathtt{UNSAT}(G) \leq \mathtt{UNSAT}(G \circ \mathcal{P})$ for some $\beta_3 > 0$.

* Let $\sigma' : V' \to \Sigma_0$ be an optimal assignment for $G' = G \circ \mathcal{P}$.
* From σ' we construct an assignment on G, $\sigma : V \to \Sigma$, such that $\sigma(v) = \min_{s \in \Sigma} d_r(\sigma'([v]), \mathsf{Enc}(s))$. Denote by F the set of edges whose constraints are falsified by σ.
* For $(v, w) \in F$, $c((v, w))$ is falsified by σ, so the restriction of σ' to $[v] \cup [w]$ must be the closest assignment to $\mathsf{SAT}(\tilde{c}((v, w)))$ by the construction of σ.
* However, since Enc has relative distance $\rho > 0$, we must change up to a fraction $\rho/2$ of the bits in $[v]$ or $[w]$ (if not both) to satisfy $c((v, w))$.
    * $\implies d_r(\sigma'|_{[v]\cup[w]}, \mathsf{SAT}(\tilde{c}(e))) \geq \frac{1}{2}\frac{\rho}{2} = \frac{\rho}{4}$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}\mathtt{UNSAT}(G') &= \mathtt{UNSAT}_{\sigma'}(G') \\ &= \frac{1}{|E|}\sum_{e\in E}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e)\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\mathtt{UNSAT}(G') &= \mathtt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|} \sum_{e \in E} \mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|} \sum_{e \in F} \mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e)
\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\texttt{UNSAT}(G') &= \texttt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|}\sum_{e \in E} \texttt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e \in F} \texttt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e \in F} \epsilon \frac{\rho}{4} && \text{By Soundness}
\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\mathtt{UNSAT}(G') &= \mathtt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|}\sum_{e\in E}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\epsilon\frac{\rho}{4} && \text{By Soundness} \\
&= \epsilon\frac{\rho}{4}\frac{|F|}{|E|}
\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\texttt{UNSAT}(G') &= \texttt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|}\sum_{e\in E}\texttt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\texttt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\epsilon\frac{\rho}{4} && \text{By Soundness} \\
&= \epsilon\frac{\rho}{4}\frac{|F|}{|E|} \\
&= \beta_3\texttt{unsat}_{\sigma}(G)
\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\mathtt{UNSAT}(G') &= \mathtt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|}\sum_{e\in E}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\epsilon\frac{\rho}{4} && \text{By Soundness} \\
&= \epsilon\frac{\rho}{4}\frac{|F|}{|E|} \\
&= \beta_3\mathtt{unsat}_{\sigma}(G) \\
&\geq \beta_3\min_{\tilde{\sigma}}\mathtt{unsat}_{\tilde{\sigma}}(G) \\
&= \beta_3\mathtt{unsat}_{\sigma}(G)
\end{aligned}$$

>>> Alphabet Reduction Lemma: Proof of inequality (LHS) Pt. 2

Recall that $\mathcal{P}$ satisfies the soundness probability, with rejection probability ϵ.

$$\begin{aligned}
\mathtt{UNSAT}(G') &= \mathtt{UNSAT}_{\sigma'}(G') \\
&= \frac{1}{|E|}\sum_{e\in E}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\mathtt{UNSAT}_{\sigma'|_{V_e}}(G_e) \\
&\geq \frac{1}{|E|}\sum_{e\in F}\epsilon\frac{\rho}{4} && \text{By Soundness} \\
&= \epsilon\frac{\rho}{4}\frac{|F|}{|E|} \\
&= \beta_3 \mathtt{unsat}_{\sigma}(G) \\
&\geq \beta_3 \min_{\tilde{\sigma}} \mathtt{unsat}_{\tilde{\sigma}}(G) \\
&= \beta_3 \mathtt{unsat}_{\sigma}(G) \quad \blacksquare
\end{aligned}$$

THANK YOU FOR LISTENING! WE HOPE IT WAS FUN

HAPPY TO ANSWER ANY QUESTIONS