Probabilistically Checkable Proofs

Lecture 16

Probability (and techniques)

Random Walks

Stationary Distribution

Perron-Frobenius Theorem

Algorithms (models and applications)

Pagerank

Lecture 18

Probability (and techniques)

Hitting times

Cover times

Electrical networks

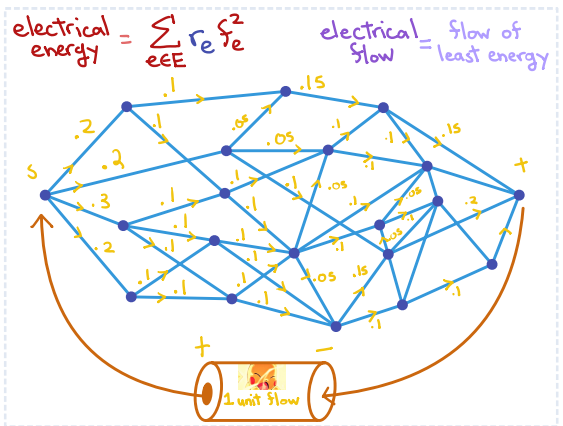
Algorithms (models and applications)

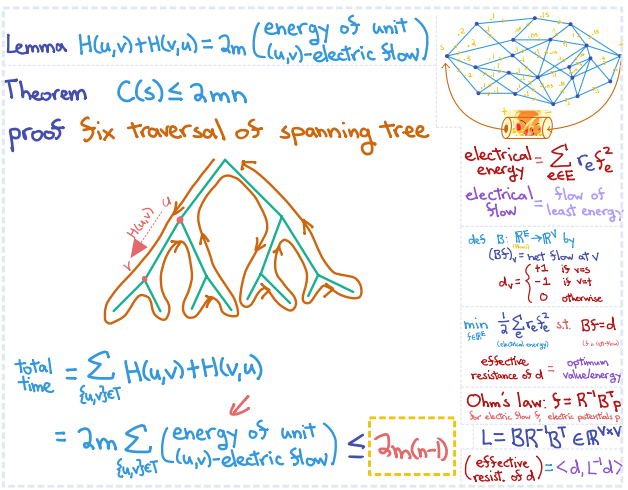
Connectivity in $O(\log n)$ space

random walk terminology:

Hitting time $H(s,t)$ = expected # steps until a random walk from s reaches t

Cover time $C(s)$ = expected # steps until a random walk from s visits every vertex





Lecture 19

Probability (and techniques)

Spectral theorem for symmetric maps

Spectral analysis of random walks

Algorithms (models and applications)

Lecture 20

Probability (and techniques)

Conductance

Cheeger's inequality

Conductance-based mixing time

Algorithms (models and applications)

Fiedler's algorithm for low-conductance cuts

Lecture 21

Probability (and techniques)

Expanders

Zig zag product

tensors

Algorithms (models and applications)

Deterministic Connectivity in $O(\log n)$ space

Lecture 22

Probability (and techniques)

classes RP and BPP

tensors of graphs

Algorithms (models and applications)

More efficient amplification

Efficient construction of exponentially large expanders

A language L is in...

no error

P (polynomial time) if there is a (det.) polytime algo to decide if $x \in L$

one-sided error

RP (randomized polynomial time) if there's a random. polytime algo that, on input x:

- if $x \in L$, decides $x \in L$ w/ constant prob.
- if $x \notin L$, always decides $x \notin L$

two-sided error

BPP (bounded error probabilistic polynomial time) if there's a random. polytime algo that, on input x:

- if $x \in L$, decides $x \in L$ w/ prob $\geq 2/3$
- if $x \notin L$, decides $x \notin L$ w/ prob. $\geq 2/3$

Amplification

RP (randomized polynomial time) if there's a random. polytime algo that, on input x:
- if $x \in L$, decides $x \in L$ w/ constant prob.
- if $x \notin L$, always decides $x \notin L$

let LERP

⇒ algo w/ false-negative error prob. (say) $1/2$
using (say) m random bits

how to reduce error prob. to δ?

1. run algo $O(\log(1/\delta))$ times
2. output disjunction (or) of all trials

total # of random bits?

$O(m \log(1/\delta))$

Better?

RP (randomized polynomial time) if there's a random. polytime algo that, on input x:

- if $x \in L$, decides $x \in L$ w/ constant prob.
- if $x \notin L$, always decides $x \notin L$

BPP (bounded error probabilistic polynomial time) if there's a random. polytime algo that, on input x:

- if $x \in L$, decides $x \in L$ w/ prob $\geq 2/3$
- if $x \notin L$, decides $x \notin L$ w/ prob. $\geq 2/3$

Theorem

Given RP/BPP algo, one can get error prob δ w/

- $O(\log(1/\delta))$-factor increase in running time
- $O(\log(1/\delta))$ **additional** random bits

ideas?

Theorem

Given RP/BPP algo, one can get error prob δ w/

- $O(\log(1/\delta))$-factor increase in running time
- $O(\log(1/\delta))$ additional random bits

RP (randomized polynomial time) if there's a random. polytime algo that, on input x:

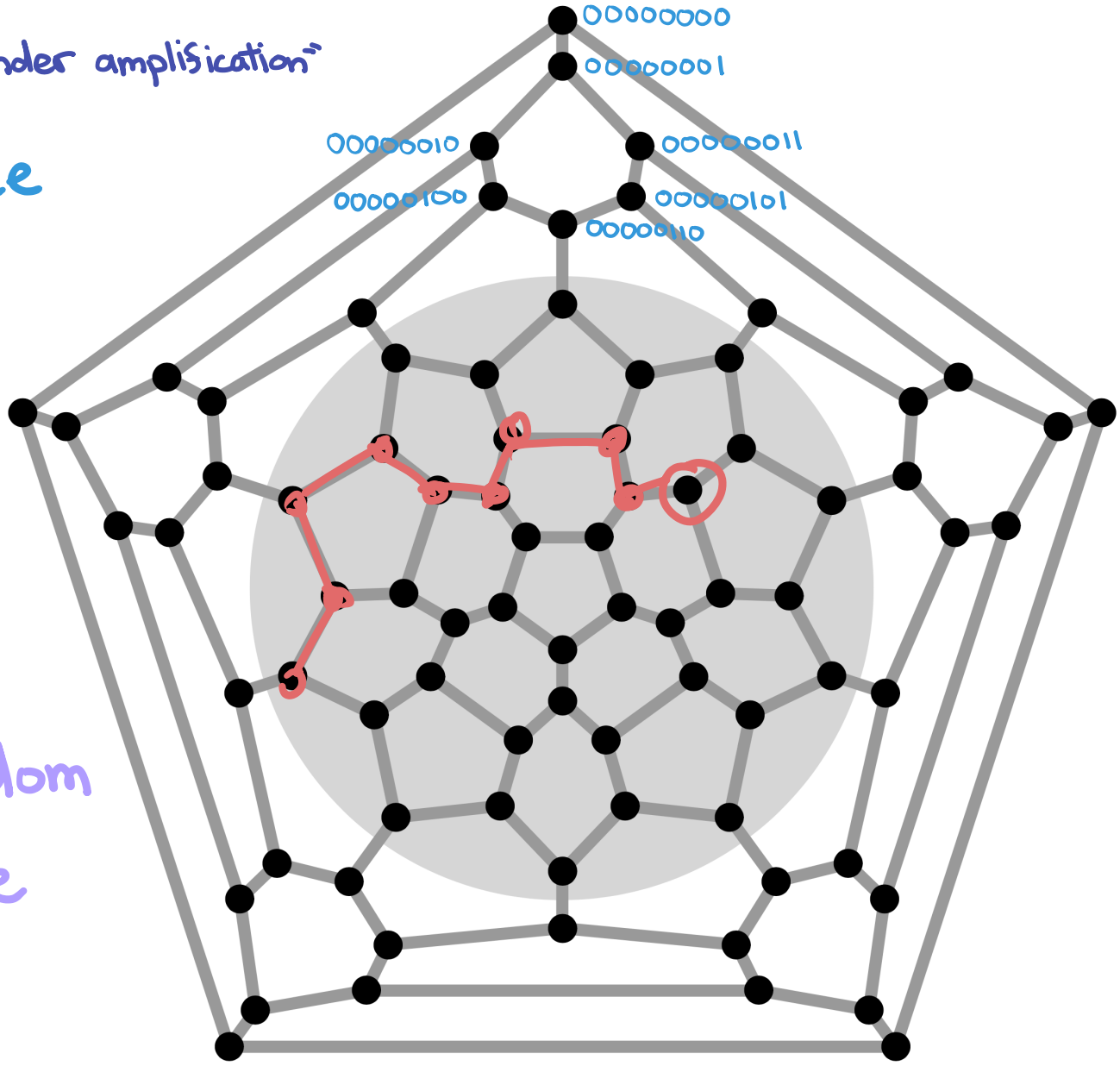
- if $x \in L$, decides $x \in L$ w/ constant prob.
- if $x \notin L$, always decides $x \notin L$

BPP (bounded error probabilistic polynomial time) if there's a random. polytime algo that, on input x:

- if $x \in L$, decides $x \in L$ w/ prob $\geq 2/3$
- if $x \notin L$, decides $x \notin L$ w/ prob. $\geq 2/3$

High level idea "expander amplification"

- make constant degree expander $G=(V,E)$ w/ $V = \{0,1\}^m$
- take random walk $v_1, \sim, v_t \in \{0,1\}^m$ w/ $t = O(\log(1/\delta))$
- use each v_i as random bits for ith instance of algo



Lecture 23/24

Probability (and techniques)

Algorithms (models and applications)

Probabilistically checkable proofs

A language L is in...

P (polytime) if there is a det. polytime decision algo

NP (nondeterministic polytime) if there is a polytime "verifier" that takes as input $x \in \{0,1\}^n$ and $y \in \{0,1\}^{poly(n)}$ and:

- if $x \in L$, answers "$x \in L$" for *some* y
- if $x \notin L$, answers "$x \notin L$" for *all* y

PCP(r,q): (nonadaptive) probabilistically checkable proof w/ r random bits and q queries

Oracle model: query ith bit of proof y

Verifier:

Given $x \in \{0,1\}^n$ and oracle access to $y \in \{0,1\}^{poly(n)}$:

1. select $O(q)$ queries based on x, $O(r)$ random bits
2. make $O(q)$ queries to oracle
3. accept or reject x
   - if $x \in L$, always accept x
   - if $x \notin L$, accept x w/ prob. $\leq \frac{1}{2}$

PCP theorem [90's]

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

random bits, queries

leads to:

Theorem. [Håstad 1997]

$\forall \varepsilon > 0$, $(7/8+\varepsilon)$-APX for 3SAT is NP-Hard

A language L is in...

- **P** (polytime): if there is a det. polytime decision algo
- **NP** (nondeterministic polytime): if there is a polytime "verifier" that takes as input $x \in \{0,1\}^n$ and $y \in \{0,1\}^{poly(n)}$ and:
  - if $x \in L$, answers "$x \in L$" for *some* y
  - if $x \notin L$, answers "$x \notin L$" for *all* y

PCP(r,q): (nonadaptive) probabilistically checkable proof w/ r random bits and q queries

Oracle model: query ith bit of proof y

Verifier:

Given $x \in \{0,1\}^n$ and oracle access to $y \in \{0,1\}^{poly(n)}$:

1. select $O(q)$ queries based on x, $O(r)$ random bits
2. make $O(q)$ queries to oracle
3. accept or reject x
   - if $x \in L$, always accept x
   - if $x \notin L$, accept x w/ prob. $\leq 1/2$

Lemma:

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

(random bits, queries)

why?

A language L is in...

- P (polytime): if there is a det. polytime decision algo
- NP (nondeterministic polytime): if there is a polytime "verifier" that takes as input $x \in \{0,1\}^n$ and $y \in \{0,1\}^{poly(n)}$ and:
  - if $x \in L$, answers "$x \in L$" for *some* y
  - if $x \notin L$, answers "$x \notin L$" for *all* y

$PCP(r,q)$: (nonadaptive) probabilistically checkable proof w/ r random bits and q queries

Oracle model: query ith bit of proof y

Verifier:

Given $x \in \{0,1\}^n$ and oracle access to $y \in \{0,1\}^{poly(n)}$:

1. select $O(q)$ queries based on x, $O(r)$ random bits
2. make $O(q)$ queries to oracle
3. accept or reject x
   - if $x \in L$, always accept x
   - if $x \notin L$, accept x w/ prob. $\leq 1/2$

PCP theorem $NP = PCP(O(1), O(\log n))$ (random bits, queries)

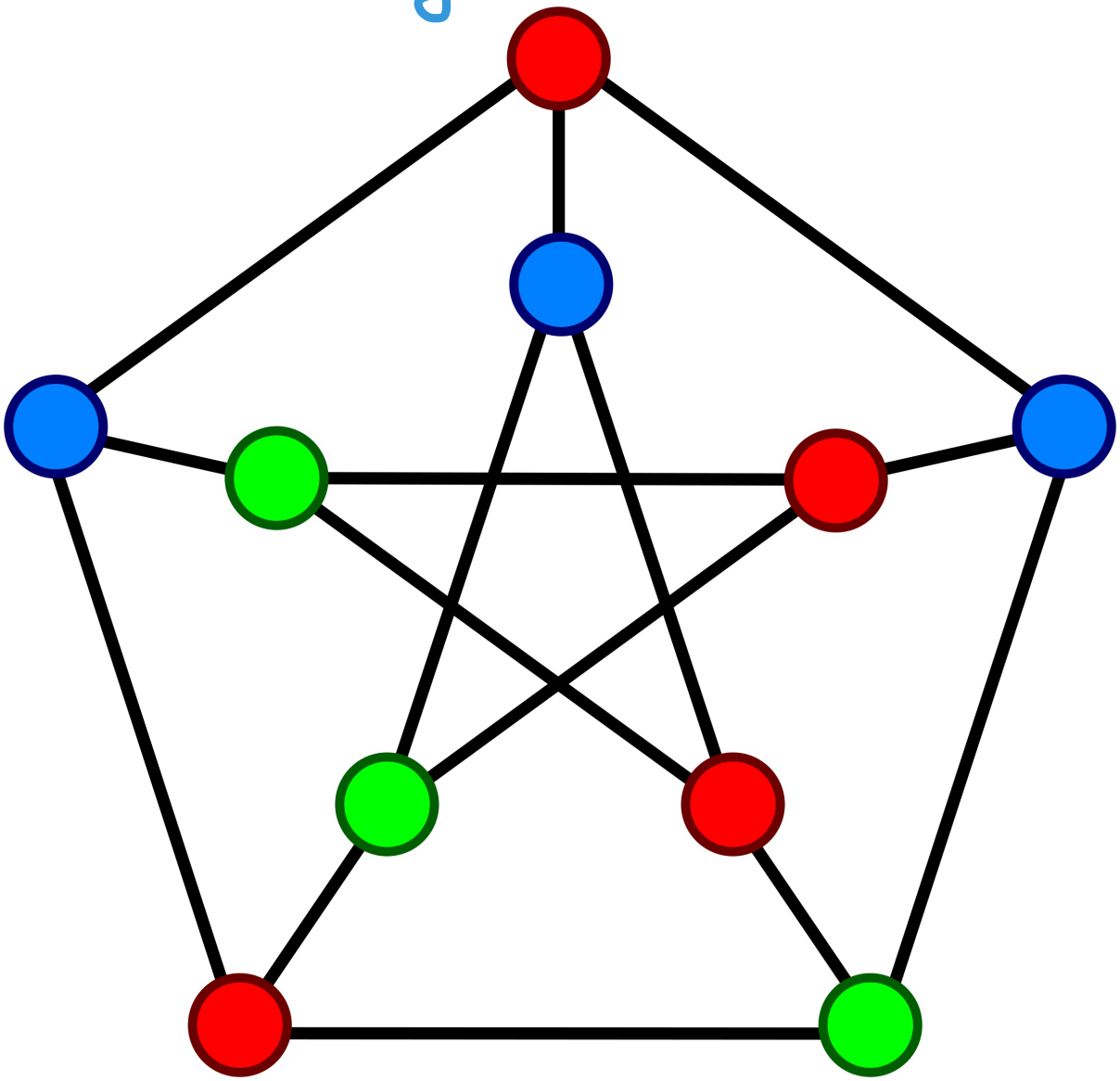
Theorem. [Håstad 1997]

$\forall \varepsilon > 0$, $(7/8+\varepsilon)$-APX for 3SAT is NP-Hard

Harder direction is $NP \subseteq PCP(O(\log n), O(1))$ (random bits, queries)

3-colorability

can vertices of G be colored in 3 colors s.t. there are no monochromatic edges?



q-ary constraint satisfaction problems

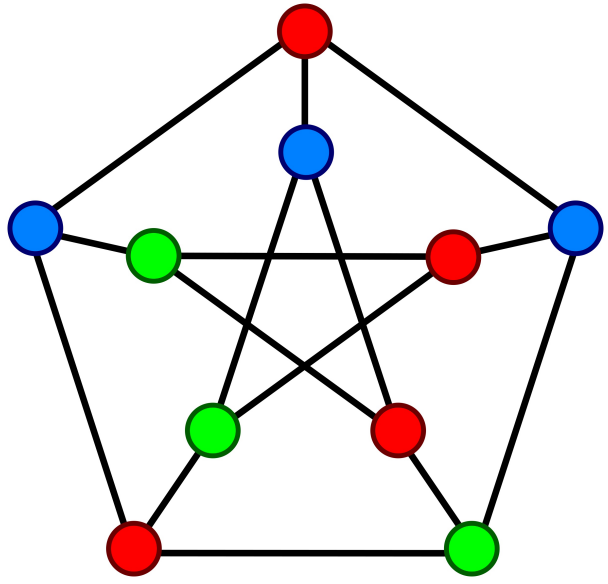
- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

V = vertices

$A = \{\text{red}, \text{blue}, \text{green}\}$

clauses = edges



q-ary constraint satisfaction problems

- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \sim, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \sim, \pi(v_k)) \in S$

given CSP P, let

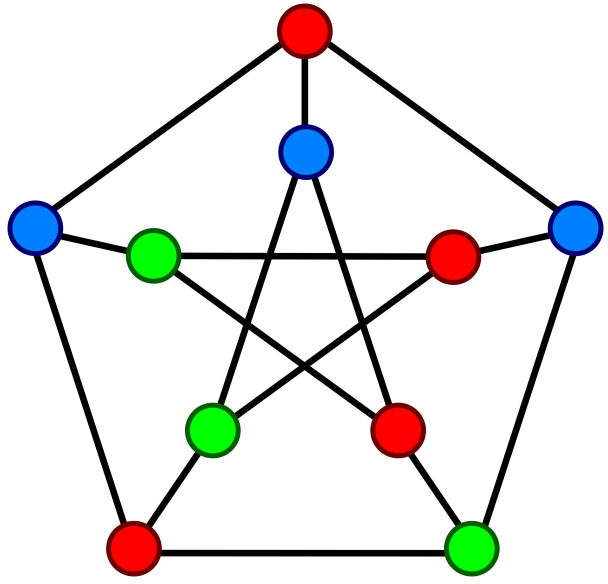
UNSAT(P) = min fraction of unsatisfied clauses

(e.g., UNSAT(P)=0 means P satisfiable)

V = vertices

A = {red, blue, green}

clauses = edges



q-ary constraint satisfaction problems

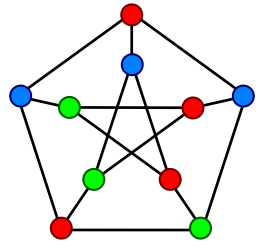
- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

V = vertices

A = {red, blue, green}

clauses = edges



Theorem:

there are integers $q > 1$, $|A| > 1$ s.t.

given q-ary CSP P over alphabet A,

it is NP-Hard to decide whether:

- all clauses can be satisfied ($\text{UNSAT}(P) = 0$)
- $\leq \frac{1}{2}$ of clauses can be satisfied ($\text{UNSAT}(P) \geq \frac{1}{2}$)

PCP theorem

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

random bits — queries

Theorem:

there are integers $q > 1$, $|A| > 1$ s.t.
given q-ary CSP over alphabet A,
it is NP-Hard to decide whether:

- all clauses can be satisfied
- $\leq \frac{1}{2}$ of clauses can be satisfied

equivalent
(exercise)

Graph CSP

2-ary CSP

q-ary constraint satisfaction problems

- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

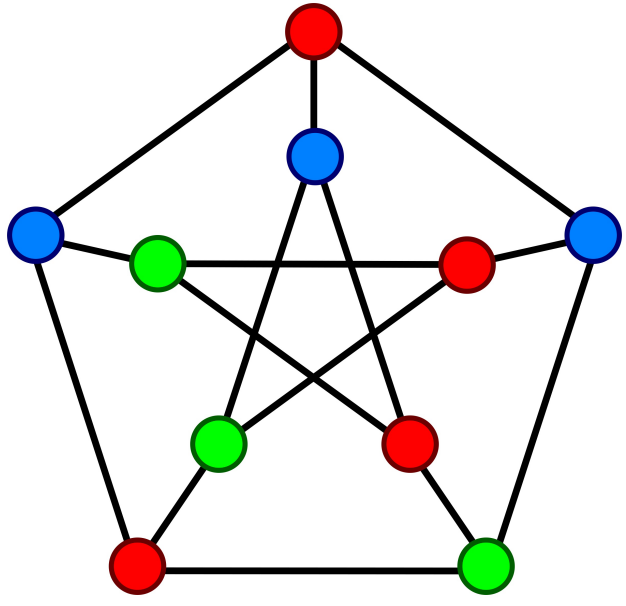
Variables = vertices

clause over $u, v \in V$ $\approx$ edge between u and v

V = vertices

A = {red, blue, green}

clauses = edges



there are integers $q>1$, $|A|>1$ s.t. given q-ary CSP P over alphabet A, it is NP-Hard to decide whether:

- $UNSAT(P)=0$
- $UNSAT(P)\geq 1/2$

q-ary constraint satisfaction problems

- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

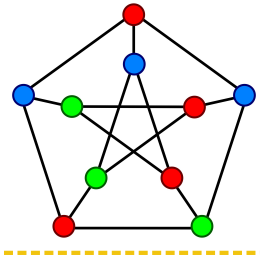
Graph CSP: 2-ary CSP
variables = vertices
clause over $u, v \in V$ $\approx$ edge between u, v

given graph-CSP G, it is NP-Hard to distinguish:

- $UNSAT(G)=0$
- $UNSAT(G)\geq 1/m$

Goal: amplify ↑ to a constant

V = vertices
A = {red, blue, green}
clauses = edges



given graph-CSP G, it is NP-Hard

to distinguish:
- $UNSAT(G) = 0$
- $UNSAT(G) \geq 1/m$

Goal: amplify to a constant

q-ary constraint satisfaction problems
- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \ldots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

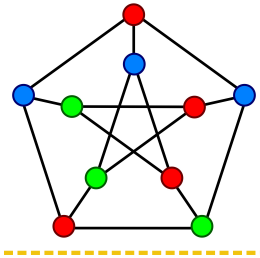
$\pi: V \to A$ satisfies clause if $(\pi(v_1), \ldots, \pi(v_k)) \in S$

Graph CSP: 2-ary CSP
variables = vertices
clause over $u, v \in V$ ≈ edge between u, v

Three transformations:

1. "Expander-ification" make G a constant degree expander
2. Powering raises $UNSAT(G)$ but also increases $|A|$
3. Alphabet reduction

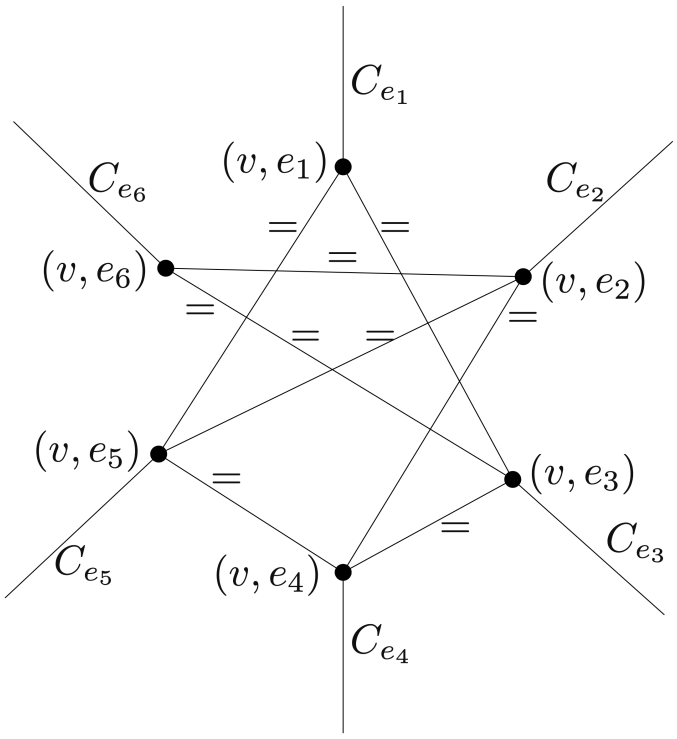
V = vertices
A = {red, blue, green}
clauses = edges



1. "Expander-ification" make G a constant degree expander

idea: replace each $v \in V$ w/ $O(1)$-degree expander

1. for each edge $e=\{u,v\}$, make new vertices (u,e) and (v,e)
2. for each edge $e=(u,v)$, make edge $((u,e),(v,e))$ w/ same constraint
3. for each vertex v, add $O(1)$-degree expander over $\{(v,e): v \in e\}$. each of these edges get "equality constraint"



claim let $G=(V,E)$ be old graph, $G'=(V',E')$ be new graph.

(a) $|E'| \le O(|E|)$ (b) G' is $O(1)$-regular

(c) $\text{UNSAT}(G) \ge \text{UNSAT}(G') \ge \Omega(\text{UNSAT}(G))$

(excercise)

given graph-CSP G, it is NP-Hard

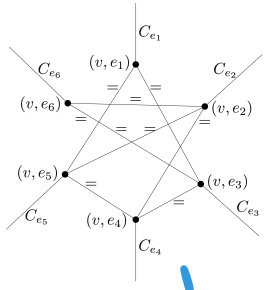
to distinguish:
- UNSAT(G) = 0
- UNSAT(G) ≥ 1/m

Goal: amplify to a constant

q-ary constraint satisfaction problems
- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$

$\pi: V \to A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

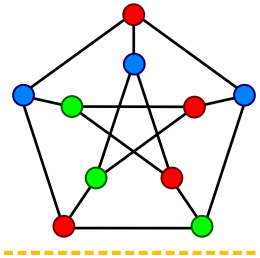
Graph CSP: 2-ary CSP
variables = vertices
clause over $u, v \in V$ ≈ edge between u, v



Three transformations:

1. ✓ "Expander-ification" make G a constant degree expander
2. Powering raises UNSAT(G) but also increases |A|
3. Alphabet reduction

V = vertices
A = {red, blue, green}
clauses = edges



thought experiment: let $t \in \mathbb{N}$

given graph-CSP $G=(V,E)$

for $e_1, \sim, e_t \in E$, let $C_{e_1,\sim,e_t} = C_{e_1} \wedge C_{e_2} \wedge \sim \wedge C_{e_t}$

(C_{e_1}: constraint for e_1)

let $P = \{C_{e_1,\sim,e_t} : e_1, \sim, e_t\}$

UNSAT(P) = 

|P| = 

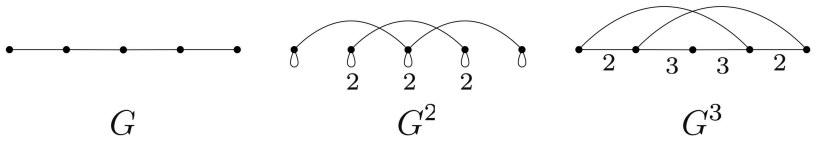
is P a graph CSP?

how to do something similar but graphically?

2. Powering raises UNSAT(G) but also increases |A|

Idea: one constraint for each t-step walk

let $G^t = (V, E_t) =$ tth graph power



for $v \in V$, let $N_t(v) = \{u$ within t steps of $v\}$

$$|N_t(v)| \le D \text{ where } D = \sum_{i=1}^{t} d^i = (1+d)^t$$

alphabet is A^D. for $\bar{\pi}: V \to A^D$, interpret $\bar{\pi}(v) \in A^D$ as an assignment $\bar{\pi}_v : N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$, edge $e_W = \{v_0, v_t\}$ in G_t is satisfied iff

(a) $\bar{\pi}_{v_0} : N_t(v_0) \to A$ satisfies $e_1, \dots, e_t$

(b) $\bar{\pi}_{v_t} : N_t(v_t) \to A$ satisfies $e_1, \dots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \le D$ where $D = \sum_{i=1}^{t} d^i = (1+d)^t$

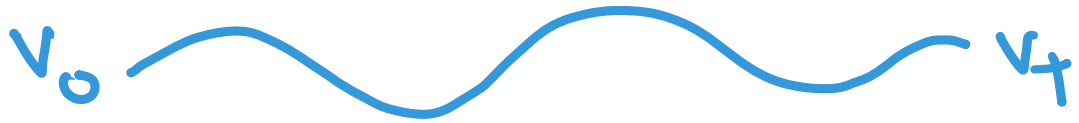
alphabet is A^D. for $\bar{\pi}: V \to A^D$, interpret $\bar{\pi}(v) \in A^D$ as an assignment $\bar{\pi}_v : N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$, edge $e_W = \{v_0, v_t\}$ in G_t is satisfied iff

(a) $\bar{\pi}_{v_0} : N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t} : N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$



claim:

$\mathrm{UNSAT}(G^t) \geq$

$\Omega(\sqrt{t}) \min\{\mathrm{UNSAT}(G), \frac{1}{t}\}$

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \leq D$ w/ $D = \sum_{i=1}^{t} d^i = (1+d)^t$

alphabet is A^D. for $\bar{\pi}: V \to A^D$, interpret $\bar{\pi}(v) \in A^D$ as an assignment $\bar{\pi}_v: N_t(v) \to A$

for walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$, edge $e_W = \{v_0, v_t\}$ in G_t is satisfied iff

(a) $\bar{\pi}_{v_0}: N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t}: N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$

t suff. large constant $\Rightarrow$

either $\mathrm{UNSAT}(G) = \Omega(1)$ or $\mathrm{UNSAT}(G^t) \geq 2\,\mathrm{UNSAT}(G)$

how to set up a proof?

let $\bar{\pi}: V \to A^D$ be optimal for G

define $\pi: V \to A$ by

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \le D$ w/ $D = (1+d)^t$

alphabet is A^D. for $\bar{\pi}: V \to A^D$,

interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$,

edge $e_W = \{v_0, v_t\}$ is satisfied iff

(a) $\bar{\pi}_{v_0}: N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t}: N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \le D$ w/ $D = (1+d)^t$

alphabet is A^D. for $\bar{\pi}: V \to A^D$,

interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v : N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$,

edge $e_W = \{v_0, v_t\}$ is satisfied iff

(a) $\bar{\pi}_{v_0} : N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t} : N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \le D$ w/ $D = (1+d)^t$

alphabet is A^D. for $\bar{\pi}: V \to A^D$,

interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v : N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$,

edge $e_W = \{v_0, v_t\}$ is satisfied iff

(a) $\bar{\pi}_{v_0} : N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t} : N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \le D$ w/ $D = (1+d)^t$

alphabet is A^D. for $\bar{\pi}: V \to A^D$,

interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_t(v) \to A$

for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_t} v_t)$,

edge $e_W = \{v_0, v_t\}$ is satisfied iff

(a) $\bar{\pi}_{v_0}: N_t(v_0) \to A$ satisfies $e_1, \ldots, e_t$

(b) $\bar{\pi}_{v_t}: N_t(v_t) \to A$ satisfies $e_1, \ldots, e_t$

(c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_t(v_0) \cap N_t(v_t)$