

Probabilistically Checkable Proofs

Lecture 1

Probability

Events, probabilities,
conditional probabilities,
union bound

Random variables,
expected value,
linearity of expectation
Markov's inequality

Algorithms

Approx. 3-SAT,

Big idea: we only want the average

$$\begin{array}{l} \text{Linearity of Expectation} \\ E[X+Y] = E[X] + E[Y] \quad (\text{stochastic variable}) \\ \text{"average sum = sum of averages"} \end{array}$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$ ($i=1, \dots, m$)

$$E[Y_i] = 7/8 \quad (1 \text{ clause case})$$

$$E[\text{\# clauses satisfied}] = E\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m E[Y_i] = 7/8 m$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$$Z = \sum_{i=1}^m Y_i = \text{\# clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + \frac{1}{2} E[Z | x_1 = \text{false}]$$

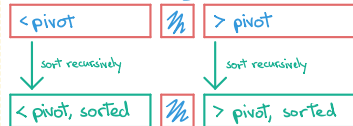
$\Rightarrow$ either $E[Z | x_1 = \text{true}]$ or $E[Z | x_1 = \text{false}]$ is $\geq E[Z]$!

QuickSort,

Random divide and conquer

① pick random 'pivot'

② Divide A into two parts



Expected running time of QuickSort

For each $i, j \in [n]$ w/ $i < j$,

$X_{ij} = \begin{cases} 1 & \text{if } i\text{th smallest element is compared to } j\text{th smallest} \\ 0 & \text{otherwise} \end{cases}$

$\sum_{i,j} X_{ij}$ = total # comparisons made by the algorithm
 $\approx$ total running time

$E[\sum_{i,j} X_{ij}]$ = average running time

$$E[\sum_{i,j} X_{ij}] = \sum_{i,j} E[X_{ij}]$$

now analyze $E[X_{ij}]$ in isolation

$$E[X_{ij}] = \Pr[X_{ij} = 1] = \Pr[i\text{th element is compared to } j\text{th}]$$

i th and j th elem selected before any element in between happens w/ probability $\frac{2}{j-i+1}$

$$E[X_{ij}] = \frac{2}{j-i+1}$$

$$E[\sum_{i,j} X_{ij}] = \sum_{i,j} \frac{2}{j-i+1}$$

$$\leq 2n \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \right)$$

$$\leq O(n \log n) \quad \text{with harmonic series}$$

Quickselect (HW)

Lecture 2

Probability

Hash Functions,

Universal Hash Functions

$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p-1\}$ unif. random

• b in $\{0, \dots, p-1\}$ unif. random



Markov's Inequality

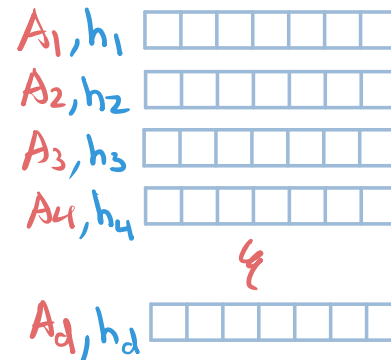
Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Algorithms

Frequency Estimation, Heavy Hitters (in streams)



estimate f_i w/
 $\min_{j=1, \dots, d} A_j[h_j(i)]$

Lecture 3

Probability

k-wise independence,
higher order moments
(generalizing universal hash
functions and Markov's inequality)

Algorithms

Hash tables w/
chaining +
linear probing

k-wise independent hash functions

$$h: [U] \rightarrow [m]$$

s.t. for any k keys $x_1, x_2, \dots, x_k \in [U]$,
 k values $y_1, y_2, \dots, y_k \in [m]$,

$$P[h(x_1)=y_1, h(x_2)=y_2, \dots, h(x_k)=y_k] = \frac{1}{m^k}$$

eg. $h(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \pmod{p}$

where $p \geq U$ prime,

$a_0, a_1, \dots, a_{k-1} \in [p]$ indep. uniformly random

Important lemma

k-wise independent random variables

$$P[X_1=y_1, X_2=y_2, \dots, X_k=y_k]$$

$$= P[X_1=y_1]P[X_2=y_2] \dots P[X_k=y_k]$$

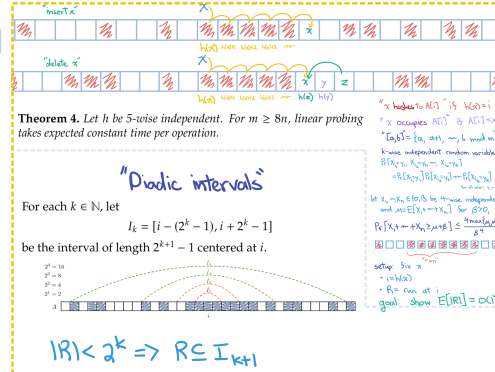
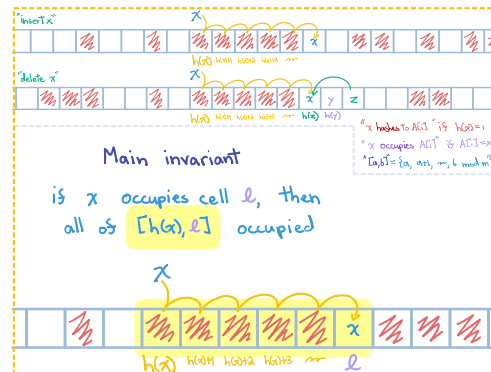
for all values $y_1, \dots, y_k$

let $X_1, \dots, X_n \in \{0, 1\}$ be 4-wise independent
and $\mu = E[X_1 + \dots + X_n]$. then for $\beta > 0$,

$$P[X_1 + \dots + X_n \geq \mu + \beta] \leq \frac{4 \max\{\mu, \mu^3\}}{\beta^4}$$

Main invariant

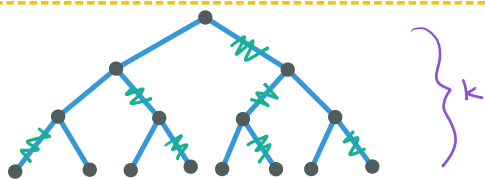
if x occupies cell i , then
all of $[h(x), i]$ occupied



Lecture 4

Probability

Galton-Watson processes



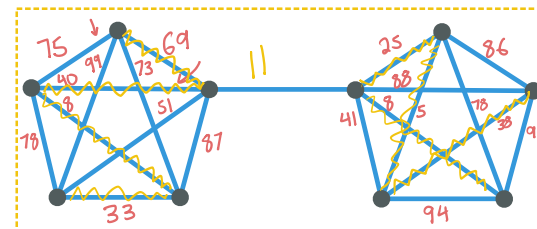
let T = complete binary tree of height k ,
w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

"Galton-Watson process"

Algorithms

Random Contractions for min cut



Randomized contractions

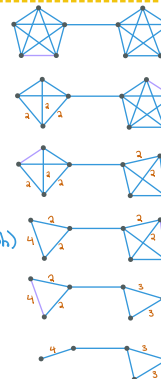
Input: $G=(V,E)$, weights $c \in \mathbb{R}_{\geq 0}^E$
(unweighted)

① while $|E| > 1$

 Ⓐ sample $e \in E$ in proportion to c

 Ⓑ "contract" e

② return set of edges (in original graph)
 contracted to remaining edge



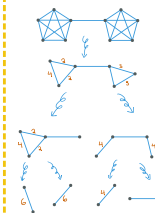
Amplification by branching

Instead of restarting at the beginning, restart partway through

High-level algorithm

① randomly contract until prob.
of ruining the min-cut is $\sim 1/2$

② Branch:
make two copies of current graph
recurse on both copies



Lecture 5

Probability

Multiplicative Chernoff Bound

Multiplicative Chernoff bound

let $X_1, \dots, X_n \in [0, 1]$ be independent

$$\mu = E[X_1 + \dots + X_n]$$

$$\epsilon \in (0, 1)$$

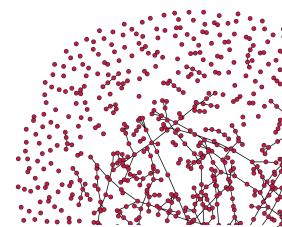
- $P[X_1 + \dots + X_n \geq (1 + \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 3}$
- $P[X_1 + \dots + X_n \leq (1 - \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 2}$

Algorithms (/ results)

Random graphs & the giant component

Erdős-Rényi graphs $G(n, p)$

- n vertices
- each edge sampled independently w/ probability p



Theorem 5.3. Consider a random graph $G \sim \mathcal{G}(n, p)$ for $p = c/n$, where c is a constant.

- If $c > 1$, then with high probability, there is exactly one connected component of G with $\Omega(n)$ vertices, and all other components have size $\leq O(\log n)$.
- For $c < 1$, then with high probability, all connected components of G has size $< O(\log n)$.

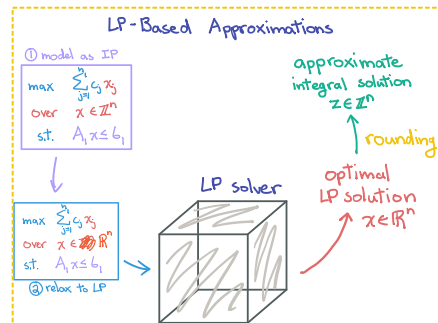
Lecture 6

Probability (& techniques)

Approximations via linear programming & randomized rounding

Linear programs (LPs)
Similar to IPs, except no integrality constraints

min/max $\sum_{j=1}^n c_j x_j$ linear objective
over $x \in \mathbb{R}^n$ linear integral variables
st. $A_1 x \leq b_1$ linear constraints
 $A_2 x = b_2$
 $A_3 x \geq b_3$



Continuous optimization vs Discrete optimization

randomization

Algorithms

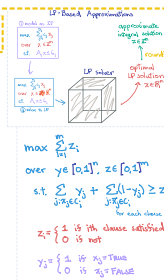
Max Sat Set Cover Covering Integer Programs

For each variable x_j , independently, randomly set $x_j = t$ with probability g_j , and $x_j = f$ otherwise.

$$E[\# \text{ sat. clauses}] \geq (1 - 1/e) \text{OPT}$$

k	oblivious	LP	average
1	1/2	$z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
2	3/4	$(3/4)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
3	7/8	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
4	15/16	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
5	31/32	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
...	...	...	...
k	$1 - 1/2^k$	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$

Can we get $\frac{3}{4}$ -APX?



Rounding LP solution
 $x \in [0,1]^n \rightsquigarrow F \subseteq \{S_1, \dots, S_n\}$

For each set S_j , independently, take S_j with probability x_j .

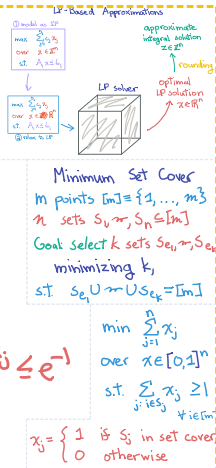
$$E[|F|] = \sum x_j \leq \text{OPT}$$

Fix a point i .

$$E[\# \text{ sets in } F \text{ covering } i] = \sum_{j \in S_i} x_j \geq 1$$

$$P[F \text{ not cover } i] = \prod_{j \in S_i} (1 - x_j) \leq e^{-\sum_{j \in S_i} x_j} \leq e^{-1}$$

$$P[F \text{ cover } i] \geq 1 - e^{-1}$$



Lecture 7

Probability

Continuous Random Variables

Variance, Chebyshev's Variance Reduction

Median trick

Variance

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

Chebyshev's inequality

$$P[|X - E[X]| \geq d] \leq \frac{\text{Var}(X)}{d^2}$$

"median trick", "median of means"

- ① take mean to reduce variance, get constant prob. of error
- ② take median to amplify success prob. exponentially

Algorithms

Distinct Elements

1, 2, 3, ..., 14, 15, 16, ...

Algorithm w/
limited space

Goal: estimate # of distinct elements in the stream

Suppose you had an ideal hash function

$$h: [m] \rightarrow [0, 1]$$

and the following facts:

let $Y_1, \dots, Y_k \in [0, 1]$ independent and uniformly random

$$X = \min \{Y_1, \dots, Y_k\}$$

$$\text{then } E[X] = \frac{1}{k+1}$$

$$\textcircled{1} X \leftarrow 1$$

② for each element e

$$\textcircled{a} X \leftarrow \min\{X, h(e)\}$$

$$/* E[X] = \frac{1}{k+1}, k = (\text{\# distinct elements}) */$$

$$\textcircled{3} \text{ return } \frac{1}{X} - 1$$

$$\text{recall } E[X] = \frac{1}{k+1}$$

$$\text{goal: show } |X - E[X]| \leq \frac{\epsilon}{k+1}$$

w/ good prob.

$$P[|X - E[X]| \geq \frac{\epsilon}{k+1}]$$

$$\leq \text{Var}(X) \cdot \left(\frac{k+1}{\epsilon}\right)^2 \leftarrow \text{Chebyshev's inequality}$$

where

$$\text{Var}(X) = \frac{k}{(k+1)^2(k+2)} \approx \frac{1}{k^2}$$

Q: how to reduce $\text{Var}(X)$?

we can get $(1 \pm \epsilon)$ -APX

w/ 10% error prob. in $O(\frac{1}{\epsilon^2})$ space

what about high probability?

① make q copies $Y_1, \dots, Y_q$ of

② return median of $Y_1, \dots, Y_q$

Algo correct $\Leftrightarrow \geq \frac{1}{2}$ of Y_i 's correct

let $Z = \# \text{correct } Y_i$'s $E[Z] = \frac{q}{2}$

$$P[Z \leq (1 - \frac{\epsilon}{4})E[Z]] \leq e^{-\left(\frac{\epsilon}{4}\right)^2 E[Z]/2} \leq \frac{1}{n^{O(1)}} \text{ for } q = O(\log n)$$

ideal hash $h: [m] \rightarrow [0, 1]$

let $Y_1, \dots, Y_k \in [0, 1]$

indep. and unifs. rand.

$X = \min \{Y_1, \dots, Y_k\}$

then $E[X] = \frac{1}{k+1}$

and $E[X^2] = \frac{2}{(k+1)(k+2)}$

① $X \leftarrow 1$

② for each element e

③ $X \leftarrow \min\{X, h(e)\}$

/* $E[X] = \frac{1}{k+1}, k = (\text{\# distinct elements})$ */

③ return $\frac{1}{X} - 1$

$\text{Var}(X) = E[(X - E[X])^2]$

$X_1, \dots, X_k$ indep. copies of X

$\bar{X} = \frac{1}{k}(X_1 + \dots + X_k)$

$\text{Var}(\bar{X}) = \frac{1}{k} \text{Var}(X)$

$P[X_1 + \dots + X_k \leq (1 - \epsilon)k] \leq e^{-\epsilon^2 k/4}$

Lecture 8

Probability

Gaussian random variables

Gaussian random variables

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

mean variance

Def: the nicest, most pleasant, convenient random variable ever

Fact let $x \sim \mathcal{N}(\mu, \sigma^2)$, $d \in \mathbb{R}$.

$$\text{then } x+d \sim \mathcal{N}(\mu+d, \sigma^2)$$

$$dx \sim \mathcal{N}(d\mu, d^2\sigma^2)$$

Fact let $x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$$\text{then } x_1 + x_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Fact let $x \sim \mathcal{N}(0, 1)$, $t < \frac{1}{2}$

$$\text{then } E[e^{tx^2}] = \frac{1}{\sqrt{1-2t}}$$

Algorithms

Dimensionality reduction

Bag-of-words

literally any sentence is fine

any ↑ is literally sentence English
five encodes text in high dimensions

Lemma 8.8. Let $A \sim \mathcal{N}^{k \times d}$ be a random matrix where each coordinate A_{ij} is an independently drawn sample from $\mathcal{N}$. Let $\epsilon > 0$ be sufficiently small. For any vector x ,

$$\mathbb{P}\left[(1-\epsilon)\|x\|^2 \leq \frac{1}{k}\|Ax\|^2 \leq (1+\epsilon)\|x\|^2\right] \geq 1 - 2e^{-k/8}$$

$$k \left\{ \left[\frac{1}{k} \|Ax\|^2 \right] \leftarrow \alpha_i \right.$$

$$E[\|Ax\|^2] = \sum_{i=1}^k E[(Ax)_i^2] = k\|x\|^2$$

$$E[(\sum_{i=1}^k A_{ij} x_j)^2] = \|x\|^2$$

$$A_{ij} x_j \sim \mathcal{N}(0, x_j^2)$$

$$\sum_{j=1}^n A_{ij} x_j \sim \mathcal{N}(0, \|x\|^2)$$

$$(Ax) \sim \mathcal{N}^k(0, \|x\|^2)$$

$$\Pr[\|Ax\|^2 \geq k\|x\|^2 \cdot (1+\epsilon)] \leq e^{-\epsilon^2 k/8}$$

Dimensionality Reduction - high level

$O(\log n)$ dimensions suffice

where $n = \# \text{ points}$

$$\mathbb{R}^{\text{very large}} \xrightarrow{f(x)} \mathbb{R}^{O(\log n)}$$

One can map n points into $\mathbb{R}^{O(\log n/\epsilon^2)}$ while preserving all distances up to $(1 \pm \epsilon)$ -APX

Lecture 9

Probability

Locality-sensitive hashing

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \text{ w/ } g_i \sim \mathcal{N}^d$$

when querying y :

• if there is $x \in P$ w/ $\angle(x, y) \leq \theta$, then it collides w/ probability $\geq 1/\sqrt{n}$

• expect to collide w/ at most one point $x \in P$ w/ $\angle(x, y) > 2\theta$

Lemma let $\bullet x, y \in S^{d-1}$
 $\bullet g \sim \mathcal{N}^d$
 then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

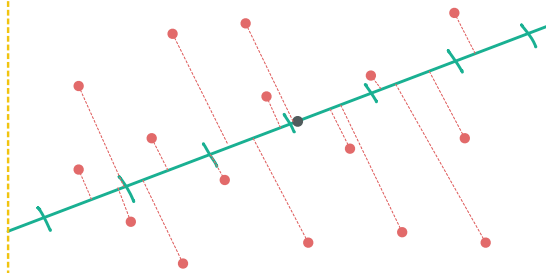
$$k = \pi(\log(n)/2\theta)$$



d-dimensional sphere

Random line embedding & Buckets

$$h(x) = \lfloor \langle g, x \rangle + d \rfloor \text{ w/ } g \sim \mathcal{N}^d, d \in [0, 1] \text{ unifs. rand.}$$

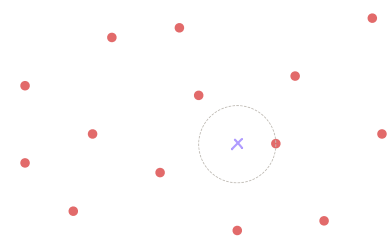


Algorithms

Approximate nearest neighbors

Nearest neighbor search

Input: n points $P \subset \mathbb{R}^d$



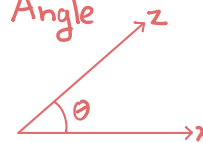
Query: given $q \in \mathbb{R}^d$, find closest point in P to q

Types of distances

• Euclidean distance

$$\|x - z\|_2 = \sqrt{\sum_i (x_i - z_i)^2}$$

• Angle



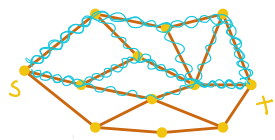
Lecture 10

Probability (and techniques)

LP duality

L_1 -metric embeddings

Max Flow
Min Cut



Path Packing LP

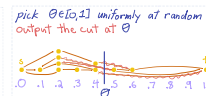
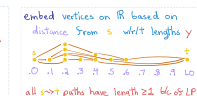
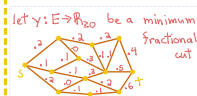
maximize $\sum_{p \in P} x_p$
over $x: P_{st} \rightarrow \mathbb{R}_{\geq 0}$
(all s-t paths)

s.t. $\sum_{p \in P} x_p \leq c_e$ for all edges e
"Ax ≤ c" (capacity of e)

Path Covering LP
(i.e. cut)

minimize $\sum_{e \in E} c_e y_e$
over $y: E \rightarrow \mathbb{R}_{\geq 0}$
(total capacity of "fractional cut")

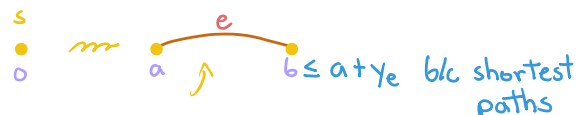
s.t. $\sum_{e \in P} y_e \geq 1$ for all $p \in P_{st}$
"A'y ≥ b" (continuous analog of s-t cut)



$E[\text{capacity of cut}]$

$$\stackrel{\text{(linearity of expectation)}}{=} \sum_{e \in E} c_e \Pr[e \text{ cut}] \leq \sum_{e \in E} c_e y_e$$

minimum capacity
of any fractional cut

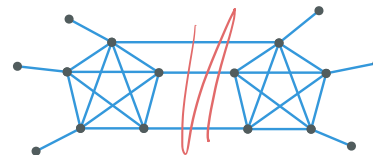


Algorithms

Sparsest Cut

Special case: uniform sparsest cut
w/ uniform demands $b(u,v) = 1 \forall u,v$

$$\text{"sparsity"} = \frac{c(\delta(s))}{|S| |T|}$$



useful primitive for graph partitioning, divide & conquer

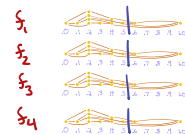
Q: how to round LP to sparse cut?

min $\sum_{e \in E} c_e d(u,v)$ over metrics $d: V \times V \rightarrow \mathbb{R}_{\geq 0}$
s.t. $\sum_{\{s,t\}} b(s,t) d(s,t) \geq 1$

Suppose $d(u,v)$ was of the form

$$d(u,v) = \sum_{i=1}^k |\xi_i(u) - \xi_i(v)| \text{ for } \xi_i: V \rightarrow [0,1]^k$$

(assume $\min \xi_i(a) = 0, \max \xi_i(a) = 1$)



" L_1 -metric"

Sparsest cut
Input: undirected graph $G=(V,E)$
edge capacities $c(e) > 0 \forall e \in E$
demands $b(s,t) > 0 \forall s,t \in V$
Goal: compute SCF minimizing
"sparsity" $\frac{c(\delta(s))}{|S| |T|}$ ← cut has many free
edges

Lecture 11

Probability (and techniques)

Algorithms

Other metric-based problems
(sometimes via correspondence between cuts $\leftrightarrow$ metrics)

Buy-at-bulk network design
k-median clustering
metric labeling
group Steiner tree
metrical task systems
vehicle routing
hierarchical placement
distributed k-server
covering Steiner tree
⋮



these problems are easy for tree metrics

Approximation by tree metrics

Input: undirected $G=(V,E)$ w/ edge weights $w: E \rightarrow \mathbb{R}_{\geq 0}$

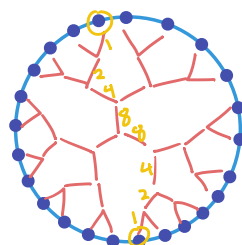
(High-level)
Goal: compute tree T (containing V as nodes)
so that shortest paths on T are close to T

$$d_T(u,v) \geq d_G(u,v)$$

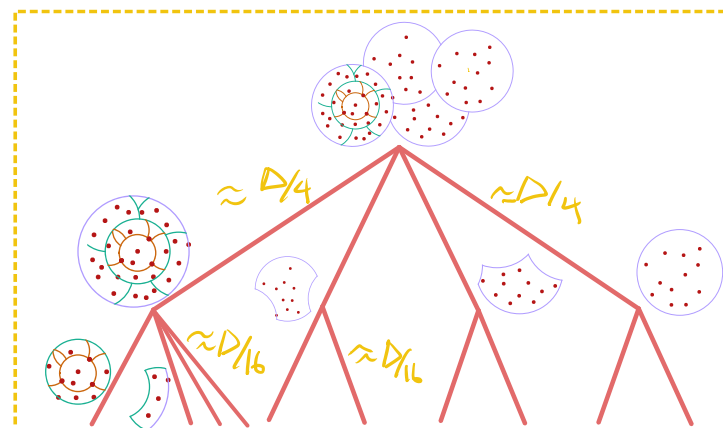
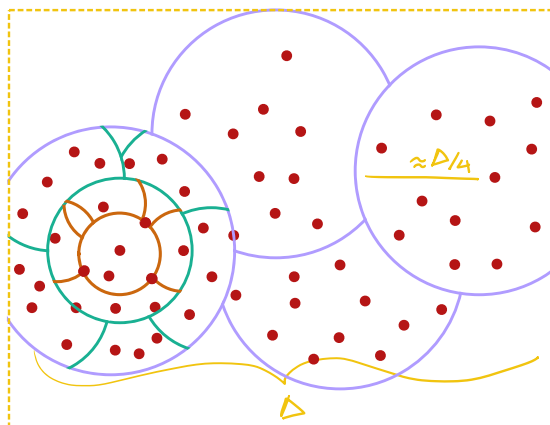
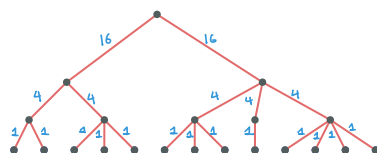
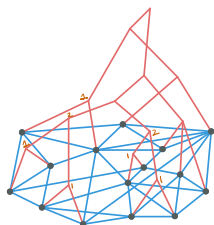
$$\text{For } e=f_u,v \quad \begin{matrix} \text{(distance in } T) \\ \text{(stretch } e) = \frac{d_T(u,v)}{d_G(u,v)} \\ \text{(distance in } G) \end{matrix}$$

ideally

$$(\text{stretch } e) = \text{polylog}(n) \quad \forall e$$



Theorem In randomized polynomial time,
randomized hierarchical dominating tree metric w/
 $E[\text{stretch } e] \leq O(\log n) \quad \forall e \in E$



Randomized Tree metrics

Lecture 12

Probability (and techniques)

Additive Chernoff Inequality

Double-sampling

VC dimension

Additive Chernoff bound

let $X_1, \dots, X_n \in [0, 1]$ be independent

$$\mu = E[X_1 + \dots + X_n] / n$$

$\epsilon \in (0, 1)$

- $P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon] \leq e^{-2\epsilon^2 n}$
- $P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$

Algorithms

ϵ -samples and ϵ -nets

ϵ -sample $|M_Q(r) - \mu(r)| \leq \epsilon \quad \forall r \in R$

let Q sample k points w/ repetition
 $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

ϵ -sample theorem [VC71]
 $k \geq O\left(\frac{\ln(g(2k))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample (w/ constant probability)

e.g. $g(n) = O(n^d)$
 $k \geq O\left(\frac{d \ln(d/\epsilon)}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample
 ↑ independent of $|P|$ (!)

$P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon], P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$ $g(n) = \max\{|RAQ| : a \in P, |a| \leq n\}$

$\mu(r) = \frac{|P \cap r|}{|P|}$
 $M_Q(r) = \frac{|a \cap r|}{|a|}$
 $RAQ = \{r \cap a : r \in R, a \in P\}$

ϵ -sample $|M_Q(r) - \mu(r)| \leq \epsilon \quad \forall r \in R$

let Q sample k points w/ repetition
 $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

ϵ -net theorem [HW87]
 $k \geq O\left(\frac{\ln(g(2k))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

e.g. $g(n) = O(n^d)$
 $k \geq O\left(\frac{d \ln(d/\epsilon)}{\epsilon^2}\right) \Rightarrow \epsilon$ -net
 ↑ independent of $|P|$ (!)

$P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon], P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$ $g(n) = \max\{|RAQ| : a \in P, |a| \leq n\}$

$\mu(r) = \frac{|P \cap r|}{|P|}$
 $M_Q(r) = \frac{|a \cap r|}{|a|}$
 $RAQ = \{r \cap a : r \in R, a \in P\}$

Lecture 13

Probability
(and techniques)

Algorithms
(models and applications)

Consistency model

PAC learning model

Generalization error

Occam's Razor

Generalization by growth

Lecture 14

Probability
(and techniques)

Algorithms
(models and applications)

Entropy

(Optimal)
Error-Correcting
Codes

Lecture 15

Probability
(and techniques)

Lovász Local Lemma

Algorithms
(models and applications)

Moser's resampling
algorithm

Lecture 16

Probability (and techniques)

Algorithms (models and applications)

Random Walks

Stationary Distribution

Perron-Frobenius Theorem

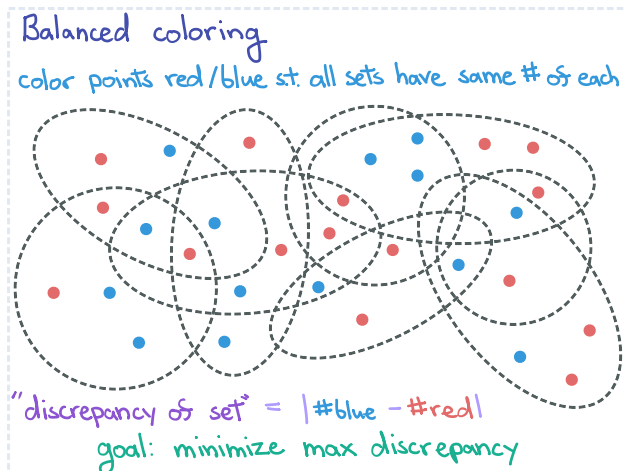
Pagerank

Lecture 17

Probability (and techniques)

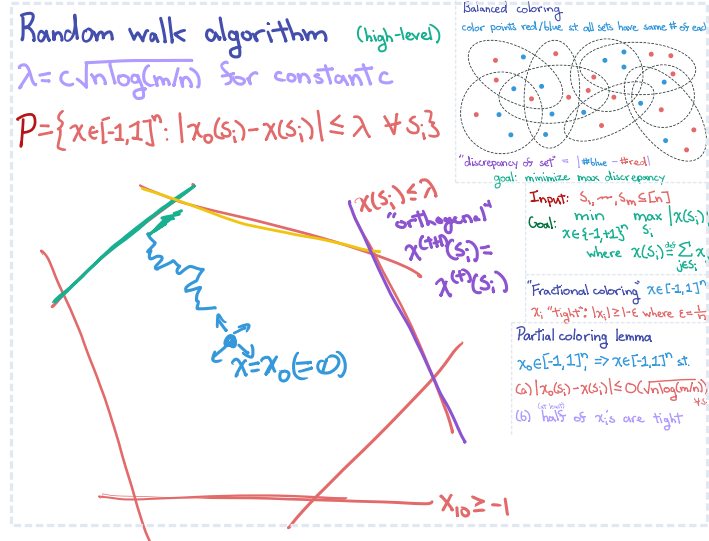
High-dimensional random walks

More fun w/ Gaussians



Algorithms (models and applications)

Discrepancy minimization



Lecture 18

Probability (and techniques)

Algorithms (models and applications)

Hitting times

Cover times

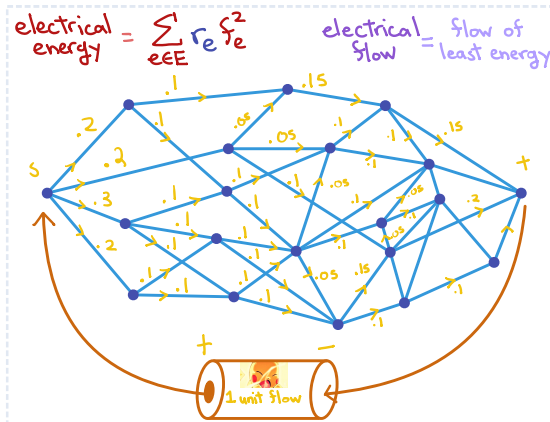
Electrical networks

Connectivity
in $O(\log n)$ space

random walk terminology:

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

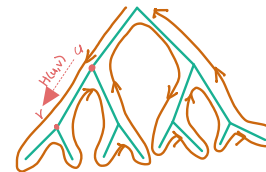
Cover time $C(s)$ = expected # steps until a random walk from s visits every vertex



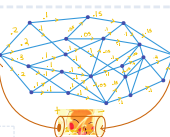
Lemma $H(u, v) + H(v, u) = 2m \cdot (\text{energy of unit flow from } u \text{ to } v)$

Theorem $C(s) \leq 2mn$

proof Six traversal of spanning tree



$$\begin{aligned} \text{total time} &= \sum_{u, v \in V} H(u, v) + H(v, u) \\ &= 2m \sum_{u, v \in V} (\text{energy of unit flow from } u \text{ to } v) \leq 2m(n-1) \end{aligned}$$



electrical energy = $\sum_{e \in E} r_e f_e^2$
electrical flow = least energy

des $B: \mathbb{R}^E \rightarrow \mathbb{R}^V$ by
 $(Bf)_v = \text{net flow at } v$
 $d_v = \begin{cases} -1 & \text{if } v = s \\ 1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases}$

min $\frac{1}{2} \sum_{e \in E} r_e f_e^2$ s.t. $Bf = d$
(electrical energy) (is a quadratic)

effective resistance of d = optimum value/energy
Ohm's law: $f = R^{-1} B^T p$
for electric flow by electric potentials p
 $L = B R^{-1} B^T \in \mathbb{R}^{V \times V}$
(effective resist. of d) $= \langle d, L^{-1} d \rangle$

Lecture 19

Probability (and techniques)

Algorithms (models and applications)

Spectral theorem
for symmetric maps

Spectral analysis of
random walks

Lecture 20

Probability
(and techniques)

Algorithms
(models and applications)

Conductance

Cheeger's inequality

Conductance-based
mixing time

Fiedler's algorithm
for low-conductance
cuts

Lecture 21

Probability
(and techniques)

Algorithms
(models and applications)

Expanders

Zigzag product

tensors

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

one-sided
error

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

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

two-sided
error

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

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

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

let LERP

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

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

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

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

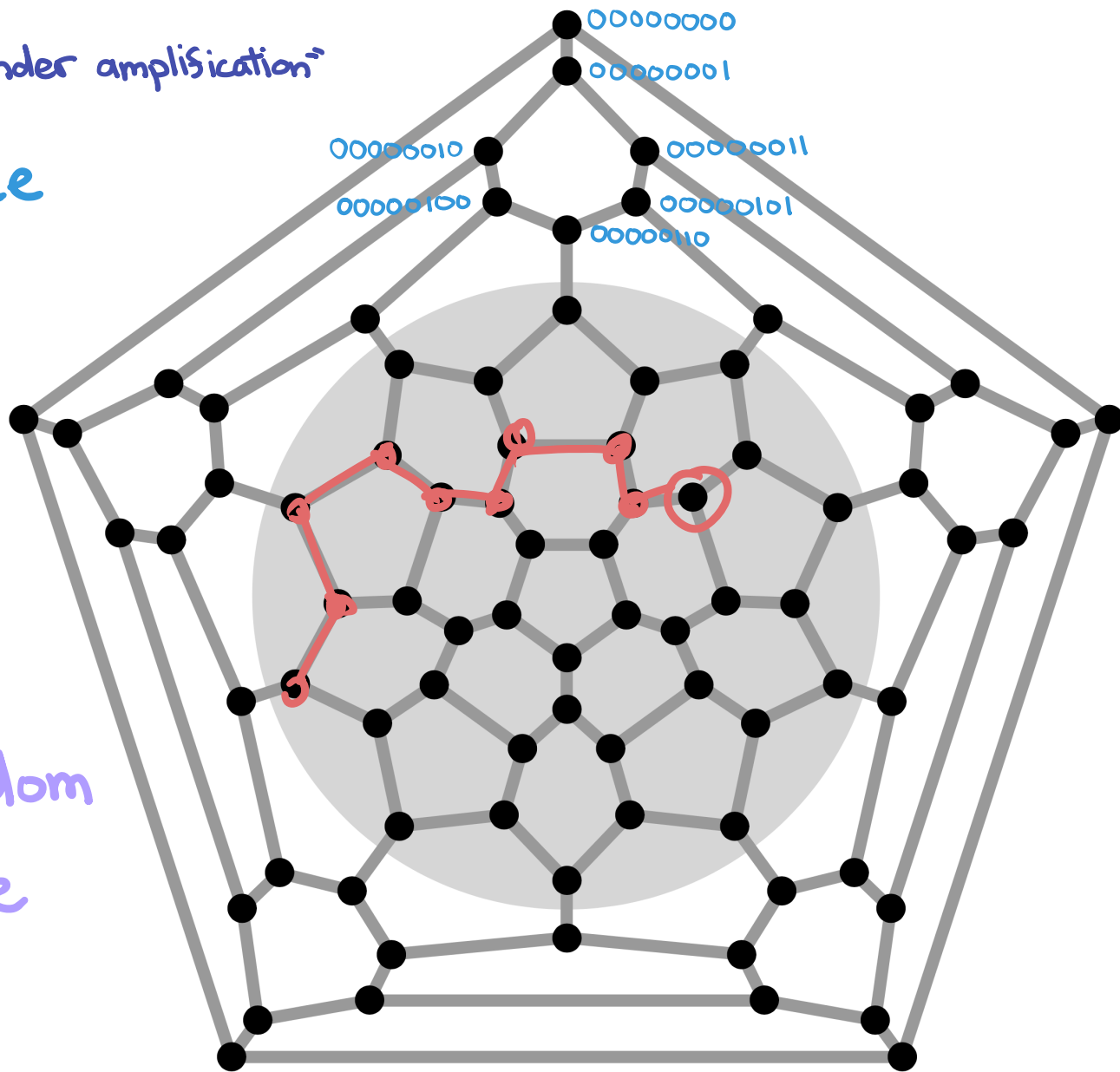
- if $x \in L$, decides $x \in L$ w/ constant prob.
- if $x \notin L$, always decides $x \notin L$

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

- 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
 $v_1, \dots, v_t \in \{0,1\}^m$
w/ $t = O(\log(1/\delta))$
- use each v_i as random bits for i th 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 $y \in \{0,1\}^{\text{poly}(n)}$ 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 i th bit of proof y

Verifier:

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

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 [90's]

$$NP = PCP(\underbrace{O(1)}_{\text{random bits}}, \underbrace{O(\log n)}_{\text{queries}})$$

leads to:

Theorem. [Håstad 1997]

$\forall \epsilon > 0$, $(7/8 + \epsilon)$ -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 $y \in \{0,1\}^{\text{poly}(n)}$ 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 i th bit of proof y

Verifier:

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

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:

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

random bits queries

why?

A language L is in...

$\text{P}_{(\text{polytime})}$ is there is a det. polytime decision algo

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

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

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

Oracle model: query i th bit of proof y

Verifier:

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

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 $\text{NP} = \text{PCP}(O(1), O(\log n))$
random bits queries

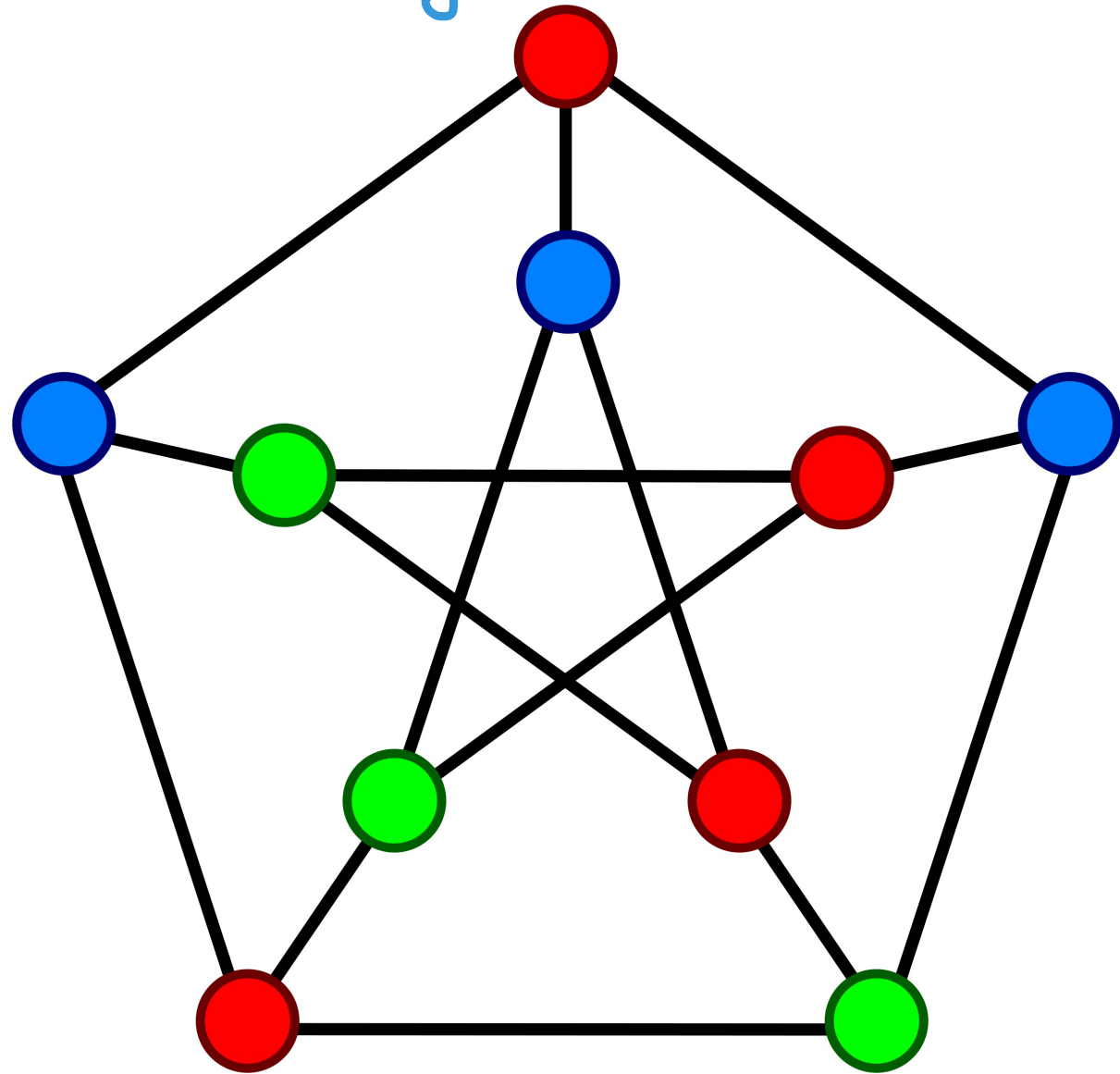
Theorem. [Håstad 1997]

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

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

3-colorability

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



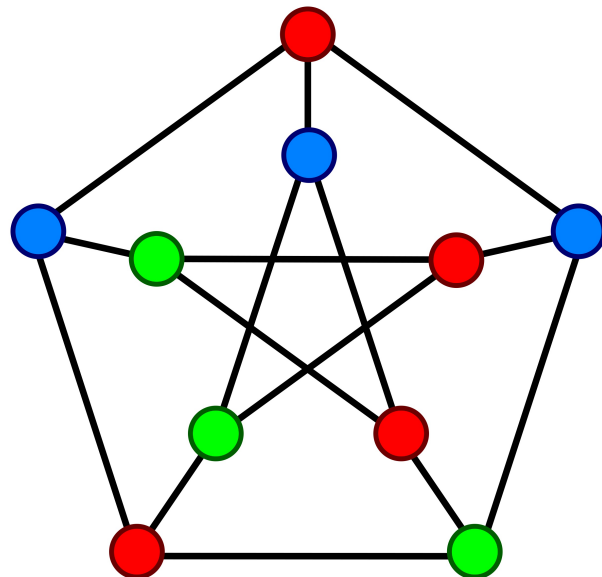
q-ary constraint satisfaction problems

- n variables V • finite alphabet A • m clauses
- a clause consists of q variables $v_1, \dots, v_k \in V$ and "satisfying" set $S \subseteq A^q$
- $\pi: V \rightarrow A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

$V = \text{vertices}$

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

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 $S \subseteq A^q$
- $\pi: V \rightarrow A$ satisfies clause if $(\pi(v_1), \dots, \pi(v_k)) \in S$

given CSP P , let

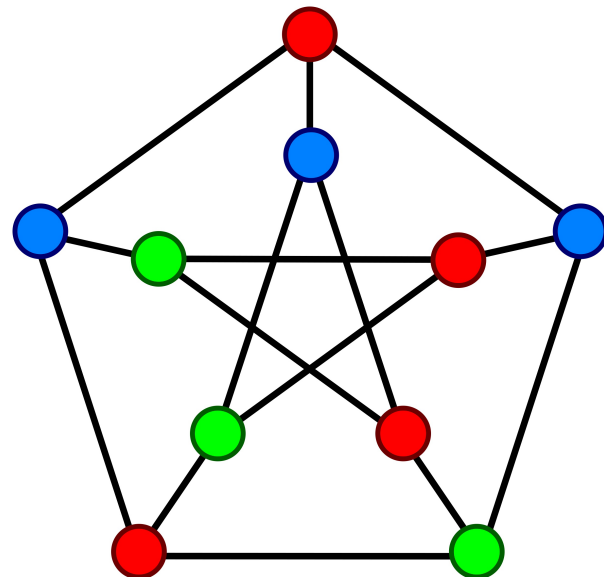
$\text{UNSAT}(P) = \text{min fraction of unsatisfied clauses}$

(e.g., $\text{UNSAT}(P) = 0$ means P satisfiable)

$V = \text{vertices}$

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

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 $S \subseteq A^q$

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

Theorem:

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

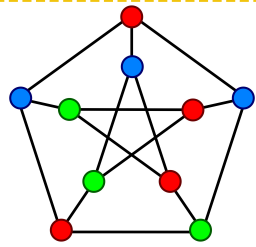
given q-ary CSP P over alphabet A ,
it is NP-Hard to decide whether:

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

$V = \text{vertices}$

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

clauses = edges



PCP theorem

$$NP = PCP(\underbrace{O(1)}_{\text{random bits}}, \underbrace{O(\log n)}_{\text{queries}})$$

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

variables = vertices

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

q -ary constraint satisfaction problems

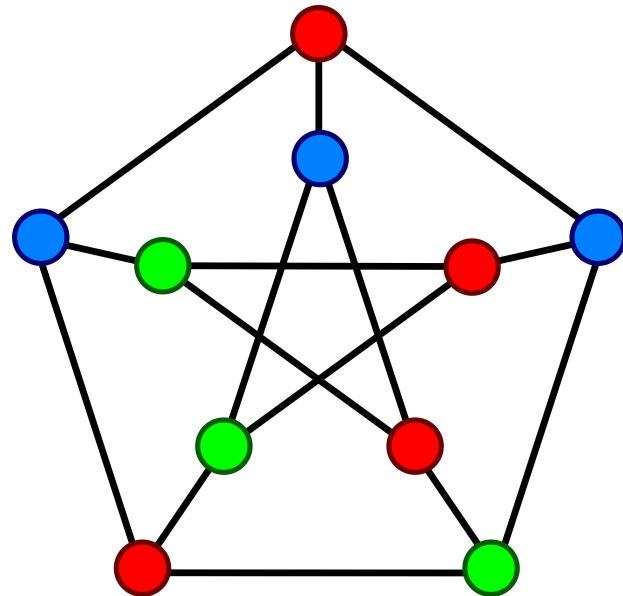
- n variables V
- finite alphabet A
- m clauses
- a clause consists of q variables $v_1, \dots, v_q \in V$ and "satisfying" set $S \subseteq A^q$

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

$V = \text{vertices}$

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

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:

- $\text{UNSAT}(P) = 0$
- $\text{UNSAT}(P) \geq \frac{1}{2}$

q -ary constraint satisfaction problems

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

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

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:

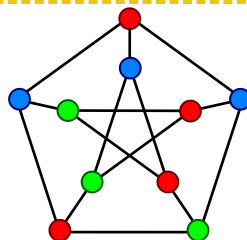
- $\text{UNSAT}(G) = 0$
- $\text{UNSAT}(G) \geq \frac{1}{m}$

Goal: amplify  to a constant

V = vertices

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

clauses = edges



given graph-CSP G , it is NP-Hard

to distinguish:

- $UNSAT(G) = 0$
- $UNSAT(G) \geq \frac{1}{m}$

Goal: amplify $\nearrow$ 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 $S \subseteq A^q$

$\pi: V \rightarrow A$ 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

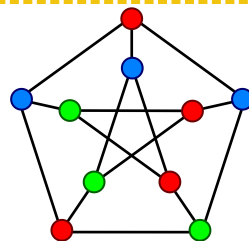
Three transformations:

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

V = vertices

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

clauses = edges



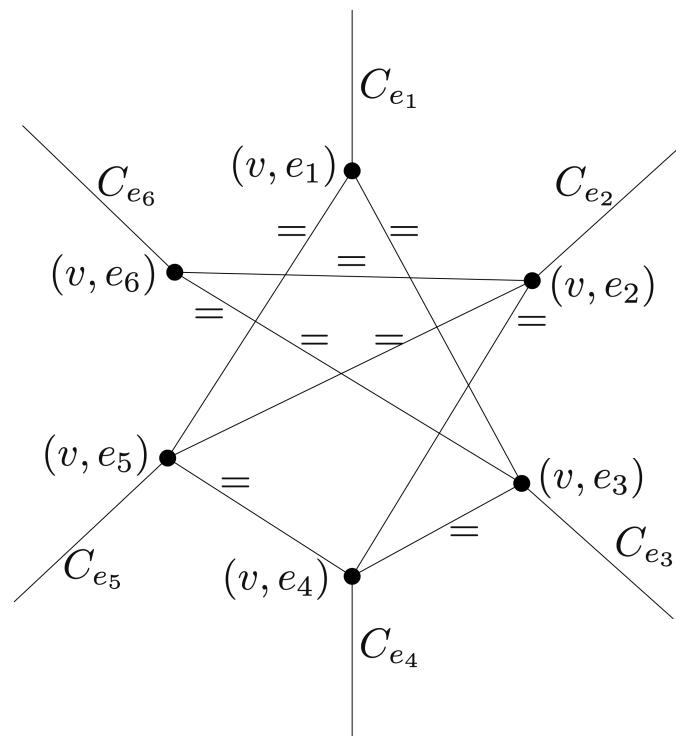
1. "Expander-ification" make G a constant degree expander

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

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

(exercise)

given graph-CSP G , it is NP-Hard

to distinguish:

- $UNSAT(G) = 0$
- $UNSAT(G) \geq \frac{1}{m}$

Goal: amplify $\nearrow$ to a constant

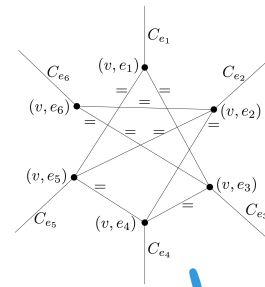
Three transformations:

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

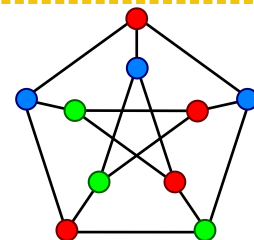
q -ary constraint satisfaction problems

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



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



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

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

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

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

UNSAT(P) = 

$|P| =$ 

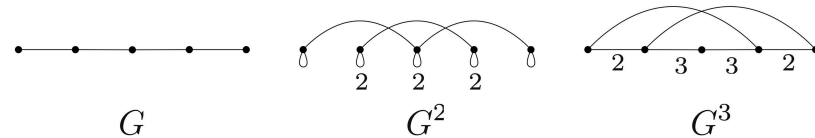
is P a graph CSP?

how to do something
similar but graphically?

2. Powering raises $\text{UNSAT}(G)$ but also increases $|A|$

Idea: one constraint for each t -step walk

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



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

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

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

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

- (a) $\bar{\pi}_{v_0}: N_t(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$ (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree
(b) $\bar{\pi}_{v_t}: N_t(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$ on $N_t(v_0) \cap N_t(v_t)$

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

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

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

- | | |
|---|---|
| (a) $\bar{\pi}_{v_0}: N_+(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$ | (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree |
| (b) $\bar{\pi}_{v_t}: N_+(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$ | on $N_+(v_0) \cap N_+(v_t)$ |
-



claim:

$$\text{UNSAT}(G^+) \geq$$

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

$N_t(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_t(v)| \leq D$ w/ $D = \sum_{i=1}^t d^i = (1+d)^t$
alphabet is A^D . For $\bar{\pi}: V \rightarrow A^D$, interpret $\bar{\pi}(v) \in A^D$
as an assignment $\bar{\pi}_v: N_t(v) \rightarrow A$

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

- | | |
|---|---|
| (a) $\bar{\pi}_{v_0}: N_t(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$ | (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree |
| (b) $\bar{\pi}_{v_t}: N_t(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$ | on $N_t(v_0) \cap N_t(v_t)$ |

t suff. large constant $\Rightarrow$

either $\text{UNSAT}(G) = \Omega(1)$ or $\text{UNSAT}(G^+) \geq 2 \text{UNSAT}(G)$

how to set up a proof?

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

$N_+(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_+(v)| \leq D$
w/ $D = (1+d)^t$
 alphabet is A^D . For $\bar{\pi}: V \rightarrow A^D$,
 interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_+(v) \rightarrow A$
 for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_t} v_t)$,
 edge $e_W = \{v_0, v_t\}$ is satisfied iff
 (a) $\bar{\pi}_{v_0}: N_+(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (b) $\bar{\pi}_{v_t}: N_+(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_+(v_0) \cap N_+(v_t)$

$N_+(v) = \{u \text{ within } t \text{ steps of } v\}$ $\frac{|N_+(v)| \leq D}{w/D = (1+d)^t}$
 alphabet is A^D . For $\bar{\pi}: V \rightarrow A^D$,
 interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_+(v) \rightarrow A$
 for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_t} v_t)$,
 edge $e_W = \{v_0, v_t\}$ is satisfied iff
 (a) $\bar{\pi}_{v_0}: N_+(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (b) $\bar{\pi}_{v_t}: N_+(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_+(v_0) \cap N_+(v_t)$

$N_+(v) = \{u \text{ within } t \text{ steps of } v\}$ $\frac{|N_+(v)| \leq D}{w/D = (1+d)^t}$
 alphabet is A^D . For $\bar{\pi}: V \rightarrow A^D$,
 interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_+(v) \rightarrow A$
 for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_t} v_t)$,
 edge $e_W = \{v_0, v_t\}$ is satisfied iff
 (a) $\bar{\pi}_{v_0}: N_+(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (b) $\bar{\pi}_{v_t}: N_+(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_+(v_0) \cap N_+(v_t)$

$N_+(v) = \{u \text{ within } t \text{ steps of } v\}$ $|N_+(v)| \leq D$
w/ $D = (1+d)^t$
 alphabet is A^D . For $\bar{\pi}: V \rightarrow A^D$,
 interpret $\bar{\pi}(v) \in A^D$ as $\bar{\pi}_v: N_+(v) \rightarrow A$
 for each walk $W = (v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots \xrightarrow{e_t} v_t)$,
 edge $e_W = \{v_0, v_t\}$ is satisfied iff
 (a) $\bar{\pi}_{v_0}: N_+(v_0) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (b) $\bar{\pi}_{v_t}: N_+(v_t) \rightarrow A$ satisfies $e_1, \dots, e_t$
 (c) $\bar{\pi}_{v_0}$ and $\bar{\pi}_{v_t}$ agree on $N_+(v_0) \cap N_+(v_t)$

