# Randomization + Derandomization?

## Today

- randomized complexity classes
- derandomization via enumeration

  $BPP \subseteq EXP$

- pairwise independence & derandomization

  Max Cut Algorithm

  defn. of p.i.

  derandomizing max cut

Some Complexity Classes:

def. a **language** $L$ is a subset of $\{0,1\}^*$

e.g. $\{X \mid X$ is a graph with a hamilton path$\}$
$\{X \mid X$ is a collection of sets that have a proper 2-coloring$\}$

def $P$ is class of languages $L$
with ptime **deterministic** algorithms $A$
s.t. $x \in L \Rightarrow A(x)$ accepts
$x \notin L \Rightarrow A(x)$ rejects

def $RP$ is class of languages $L$
with ptime probabilistic algorithm $A$
s.t. $x \in L \Rightarrow \Pr[A(x) \text{ accepts}] \geq 1/2$
$x \notin L \Rightarrow \Pr[A(x) \text{ accepts}] = 0$
} 1-sided error

def. $BPP$ is class of languages $L$
with ptime probabilistic algorithm $A$
s.t. $x \in L \Rightarrow \Pr[A(x) \text{ accepts}] \geq 2/3$
$x \notin L \Rightarrow \Pr[A(x) \text{ accepts}] \leq 1/3$
} 2-sided error

## Comments

- constants arbitrary -
  with mult cost of $O(\log 1/\beta)$ can get error $\leq \beta$

- Clearly $P \subseteq RP \subseteq BPP$

## Big Open Question:

is $P = BPP$?

do we need random coins for efficient algorithms?

## Derandomization via enumeration

- Given probabilistic algorithm $\mathcal{A}$ & input $x$
- Run $\mathcal{A}$ on every possible random string of length $r(n)$
  at most time bound of $\mathcal{A}$.
  Is there a better bound?
- output majority answer

**Behavior**

if $x \in L$, $\geq \frac{2}{3}$ of random strings cause $\mathcal{A}$ to accept $\Rightarrow$ majority answer is ACCEPT

if $x \notin L$ " " " " " " " " reject $\Rightarrow$ " " " REJECT

**runtime**

$$O(2^{r(n)} \cdot \underbrace{t(n)}_{\text{time bound of } \mathcal{A}}) \leq O(2^{t(n)} t(n))$$

**Corollary**

$$BPP \subseteq EXP$$

$$EXP \equiv DTIME\left(\bigcup_c 2^{n^c}\right)$$

**Comments**: $r(n) \leq t(n)$ since can use at most 1 bit per step

if can get better bound on $r(n)$, can improve runtime

e.g. if $r(n) = O(\log n)$,

runtime is poly$(n)$ for ptime $\mathcal{A}$

- Given a problem with a randomized ptime algorithm, 1-sided error

Homework problem 3

$\Rightarrow \exists$ one random string that works for all inputs of size $n$

i.e. $\exists$ ckt (with no random bits) that work for all inputs of size $n$.

- What about 2-sided error?

also true!

# Pairwise independence & derandomization

- a simple randomized algorithm for MaxCut
- pairwise independent sample spaces
- derandomization

## Max Cut:

given: $G=(V,E)$

output: partition $V$ into $S,T$ to maximize $\underbrace{\{(u,v) \mid u\in S, v\in T\}}_{\text{size of } S,T \text{ cut}}$ } NP-hard

## A randomized algorithm:

Flip $n$ coins $r_1 \ldots r_n$

put vertex $i$ on side $r_i$ to get $S,T$ ← i.e. add $i$ to $S$ if $r_i=0$ + to $T$ o.w.

## Analysis:

let $1_{u,v} \equiv \begin{cases} 1 & \text{if } r_u \neq r_v \\ 0 & \text{o.w.} \end{cases}$ (i.e. placed on different sides so $(u,v)$ crosses cut) ⇐ so cut size $= \sum_{(u,v)\in E} 1_{u,v}$

$$E[\text{cut}] = E\left[\sum_{(u,v)\in E} 1_{u,v}\right]$$

$$= \sum_{(u,v)\in E} E[1_{u,v}] = \sum_{(u,v)\in E} \Pr[1_{u,v}=1]$$

$$= \sum_{(u,v)\in E} \Pr[(r_u=1 \,\&\, r_v=0) \text{ or } (r_u=0 \,\&\, r_v=1)]$$

$$= \sum_{(u,v)} \left(\Pr[\underbrace{r_u=1 \,\&\, r_v=0}_{1/4}] + \Pr[\underbrace{r_u=0 \,\&\, r_v=1}_{1/4}]\right) = \frac{|E|}{2}$$

## Pairwise independent random variables : definition

Pick $n$ values $X_1 \cdots X_n$

each $X_i \in T$ (domain) s.t. $|T| = t$ (size of domain)

in some way

def. $X_1 \cdots X_n$ independent if $\forall\ b_1 \cdots b_n \in T^n$

$$Pr[X_1 \cdots X_n = b_1 \ldots b_n] = \frac{1}{t^n}$$

pairwise independent if $\forall\ i \neq j \quad b_i, b_j \in T^2$

$$Pr[X_i X_j = b_i b_j] = \frac{1}{t^2}$$

k-wise independent if $\forall \underset{i_1 \cdots i_k}{\text{distinct}} \quad b_1 \cdots b_k \in T^k$

$$Pr[X_{i_1} \cdots X_{i_k} = b_1 \cdots b_k] = \frac{1}{t^k}$$

Main point:

Only use pairwise independence in max-cut algorithm

(i.e., algorithm analysis still works if random bits are only pairwise indep).

# Derandomization of max-cut

both work pretty well!

## Full enumeration :

$n$ fully random bits $\rightarrow \boxed{\text{Algorithm}} \rightarrow$ cut

try all $2^n$ possible coin tosses
pick best cut } gets very best cut, not just $\frac{|E|}{2}$

## "Partial enumeration":

$m$ pairwise indep random bits $\rightarrow \boxed{\text{Algorithm}} \rightarrow$ cut

don't try all possible coin tosses
just a subset that satisfies pairwise independence

e.g.

pick a row uniformly

| $r_1$ | $r_2$ | $r_3$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$\underbrace{\text{for } i \neq j,\ \forall b_1, b_2 \in \{0,1\}^2 \quad \Pr[r_i = b_1 \text{ \& } r_j = b_2] = \frac{1}{4}}_{\text{good enough to give}}$$

$$E[\text{cut}] = \frac{|E|}{2}$$

for 3 node graphs,
only need to enumerate over 4 rows
instead of 8 rows.

## Another picture

enumerate all choices of $r_1 \cdots r_n$

$\boxed{b_1 \cdots b_m} \rightarrow \boxed{\text{"randomness generator"}} \rightarrow \boxed{r_1 r_2 \cdots r_n}$

$\underbrace{\text{totally independent}}_{\text{enumerate all } 2^m \text{ choices}}$

$\uparrow$ pick a random row

pairwise independent + good enough for our algorithm!

CAN WE MAKE $n \gg m$?

above example: $m=2$, $n=3$

derandomize Max-Cut, given "randomness generator" taking $(\log n + 1) \Rightarrow n$ bits

- First: construct new randomized MC alg MC':
  - given $\log n$ truly random bits $b_1 \dots b_{\log n + 1}$
  - use generator to construct $n$ p.i. random bits $r_1 \dots r_n$
  - use $r_i$'s in MC alg + evaluate cutsize

- Then: derandomize via enumeration

Deterministic M-C alg:

For all choices of $b_1 \dots b_{\log n + 1}$

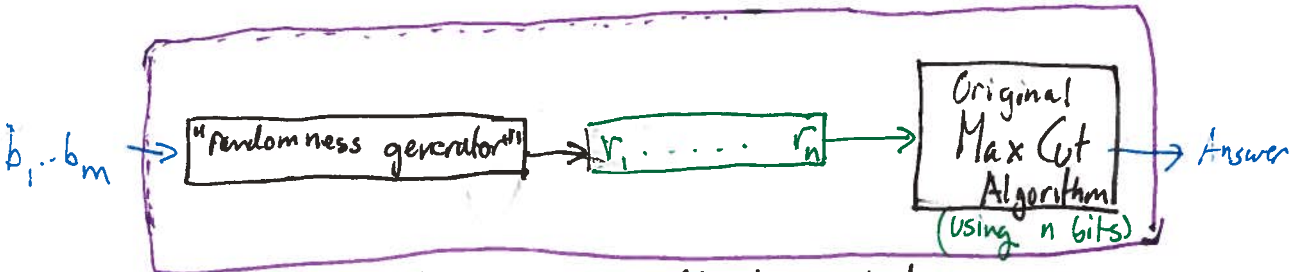
run MC' on $b_1 \dots b_{\log n + 1}$ + evaluate cutsize

pick best cutsize

Runtime: $\underbrace{\left(2^{c \log n}\right)}_{\substack{\#\text{choices} \\ \text{of } b_i\text{'s}}} \times (\text{time for generator} + \text{time to run MC}) = \text{poly}(n)$

Comments

- no guarantee of getting OPT cut as in basic enumeration method
- generator determines a very small set of random strings, atleast one of which gives a good cut




New Max Cut Algorithm MC'
(using $m \ll n$ bits)

do "full enumeration" derandomization on this in $O(2^m) \times$ [time to generate + time to run Max Cut] dri 8

# How to generate pairwise independent random variables?

1) Bits

- choose $k$ truly random bits $b_1 \ldots b_k$
  $\forall S \subseteq [k]$ s.t. $S \neq \emptyset$ set $C_S \equiv \bigoplus_{i \in S} b_i$
- output all $C_S$

Generates $2^k - 1$ bits from $k$ truly random bits
i.e. $m = \log n$

Generated bits are pairwise independent

proof: exercise

2) Integers in $[0, \ldots, q-1]$ ($q$ prime)

trivial method that works for $q = 2^l$ (note that $q$ is not prime)

- repeat "bits" construction independently for each position in $1..l$

uses $O(\log n \cdot \log q) = O(\log n)$ bits of true randomness

Somewhat better construction:

(when $n \approx q$ needs $O(\log q)$ bits of randomness)

- pick $a, b \in \mathbb{Z}_q$
- $r_i \leftarrow a \cdot i + b \mod q \quad \forall i \in \{0..q-1\}$
- output $r_1 \cdots r_q$

useful to think of as input/output description of a fctn from

$$h_{a,b} : [0..q-1] \to \mathbb{Z}_q$$

note: $|\mathcal{H}| = q^2$

Family of fctns $\mathcal{H} = \{h_1, h_2, \ldots\}$ for $h_i : [N] \to [M]$ is "pairwise independent" if:

notation: "$x \in_u D$" means $x$ chosen uniformly at random from $D$

when $H \in_u \mathcal{H}$

1) $\forall x \in [N]$, $H(x) \in_u [M]$ ← any one location distributed uniformly

2) $\forall x_1 \neq x_2 \in [N]$, $H(x_1)$ + $H(x_2)$ independent ← any 2 are indep

equivalently: $\forall x_1 \neq x_2 \in [N]$

$\forall y_1, y_2 \in [M]$

$$\Pr_{H \in \mathcal{H}}[H(x_1) = y_1 \wedge H(x_2) = y_2] = \frac{1}{M^2}$$

## Comments

- no single fctn is p.i. - have to pick a random fctn from a family
- given $H$ & $x \in [N]$ $H(x)$ should be computable in time poly$(\log N, \log M)$ } don't have to compute "all at once"
- also called "strongly 2-universal hash fctns"

Why is our example p.i.?

$$\mathcal{H} = \{h_{a,b} \mid \mathbb{Z}_q \to \mathbb{Z}_q\}$$ (recall $q$ is prime)

$$h_{a,b} = ax+b \mod q$$

fix any $x \neq w$, $c, d$

$$\Pr_{a,b}[\overset{h_{a,b}(x)}{ax+b} = c \wedge \overset{h_{a,b}(w)}{aw+b} = d] = \frac{1}{q^2}$$

$$\underbrace{\begin{pmatrix} x & 1 \\ w & 1 \end{pmatrix}}_{\substack{w \neq x \text{ so} \\ \text{nonsingular}}} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$$

$\Rightarrow$ unique soln

how many truly random bits?

$2 \log q$ yields $q$ p.i. random field elts.

## More Comments

- can construct for all finite fields, even when domain + range have different sizes

- original motivation : hashing

  hash fctns chosen from p.i. family instead of random fctns.

  Why is this good?

  how would you store a random fctn on a domain of size $2^{100000000\,00000\,0000\,00\ldots00}$?