Today

- Randomized complexity classes
- Derandomization via enumeration
  \[ \text{BPP} \subseteq \text{EXP} \]
- Pairwise independence and derandomization

Max Cut Algorithm

defn. of p.i.

derandomizing max cut
Some Complexity Classes:

**def** a language $L$ is a subset of $\mathbb{Z}_n^*$

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 p-time deterministic algorithms of

$\text{st. } x \in L \implies A(x) \text{ accepts}$

$x \notin L \implies A(x) \text{ rejects}$

**def** $RP$ is class of languages $L$ with p-time probabilistic algorithm of

$\text{st. } x \in L \implies \Pr[A(x) \text{ accepts}] = \frac{1}{2}$

$x \notin L \implies \Pr[A(x) \text{ accepts}] = 0$ $\implies$ 1-sided error

**def** $BPP$ is class of languages $L$ with p-time probabilistic algorithm of

$\text{st. } x \in L \implies \Pr[A(x) \text{ accepts}] \geq \frac{2}{3}$

$x \notin L \implies \Pr[A(x) \text{ accepts}] \leq \frac{1}{3}$ $\implies$ 2-sided error
Comments

- Constants arbitrary -
  with mult rate of \( O(\log \frac{1}{\beta}) \) can get error \( \leq \beta \)

- Clearly \( P \subseteq RP \subseteq \text{BPP} \)

Big Open Question:

- Is \( P = \text{BPP} \)?
- Do we need random coins for efficient algorithms?

Derandomization via enumeration

- Given probabilistic algorithm \( A \) on input \( x \)
- Run \( A \) on every possible random string of length \( r(n) \)
- Output majority answer

Is there a better bound?
Behavior

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

if \( x \notin L \), \( \leq \frac{1}{3} \) of random strings cause \( A \) to accept \( \Rightarrow \) majority answer is \( \text{REJECT} \)

Runtime

\[
\log (2^{2^n} + t(n)) \leq \log (2^{t(n)})
\]

\( t(n) \) is time bound of \( A \)

Corollary

\( \text{BPP} \subseteq \text{EXP} \)

\( \text{EXP} \equiv \text{DTIME} \left( \sum \log 2^n \right) \)

Comments:

- If can get better bound on \( r(n) \), can improve runtime
- E.g. if \( r(n) = O(\log n) \), runtime is \( \text{poly}(n) \) for ptime \( A \)
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 $\sum_{(u,v) \in E, u \in S, v \in T} \mathbb{1}_{u \neq v}$

A randomized algorithm:

Flip $n$ coins $r_1, \ldots, r_n$ and put vertex $i$ on side $S$ to get $S, T$.

Analysis:

Let $\mathbb{1}_{(u,v)} = 1$ if $r_u \neq r_v$ (i.e. placed on different sides, so $(u,v)$ crosses cut).

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

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

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

$$= \sum_{(u,v) \in E} \left(\Pr[r_u = 1 + r_v = 0] + \Pr[r_u = 0 + r_v = 1]\right) = \frac{|E|}{2}$$
Pairwise independent random variables: definition

Pick \( n \) values \( X_1 \ldots X_n \)
each \( X_i \in T \) (domain) \( \text{st. } |T| = t \) (size of domain)
in some way.

def. \( X_1 \ldots X_n \) independent if \( \forall b_1 \ldots b_n \in T^n \)
\[
\Pr[X_1 \ldots X_n = b_1 \ldots b_n] = \frac{1}{t^n}
\]
pairwise independent if \( \forall i \neq j \) \( b_i, b_j \in T^2 \)
\[
\Pr[X_i X_j = b_i, b_j] = \frac{1}{t^2}
\]
k-wise independent if \( \forall i_1 \ldots i_k \) \( b_1 \ldots b_k \in T^k \)
\[
\Pr[X_{i_1} \ldots X_{i_k} = b_1 \ldots b_k] = \frac{1}{t^k}
\]

Main point:
Only use pairwise independence in max-cut algorithm.
Derandomization of max-cut

Full enumeration:
try all \(2^n\) possible coin tosses
pick best cut

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

\[\begin{array}{cccc}
  r_1 & r_2 & r_3 \\
  0 & 0 & 0 \\
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
\end{array}\]

\[\text{for } i \neq j, \forall b_i, b_j \in \{0,1\}, \Pr[r_i = b_i \land r_j = b_j] = \frac{1}{4}\]
ga good enough to give
\[E[\text{cut}] = \frac{1}{2} \cdot \frac{1}{4}\]

only need to enumerate over 4 rows
instead of 8 rows.

Another picture

\[\begin{array}{ccc}
b_1 \ldots b_m & \rightarrow & \text{"randomness generator"} \\
totally independent & \uparrow & \text{pick a random row} \\
enumerate all \(2^m\) choices & & \text{pairwise independent} \\
\end{array}\]

above example: \(m=2, n=3\)

Can we make \(n > m\)?
derandomize Max-Cut given "randomness generator" taking \((\log n + 1)\) \(\Rightarrow\) \(n\) bits

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

2. Then: derandomize via enumeration

Deterministic MC alg:

For all choices of \(b_1...b_{\log n+1}\)
run \(MC'\) on \(b_1...b_{\log n+1}\) + evaluate cutsize
pick best cutsize

Runtime: \((2^{\log n}) \times (\text{time for generator + time to run MC}) = \text{poly}(n)\)

# choices
of \(b_i\)'s

Comments:
- no guarantee of getting OPT cut as in basic enumeration method
  - generator determines a very small set of random strings,
    at least one of which gives a good cut
How to generate pairwise independent random variables?

1) Bits
   
   * choose $k$ truly random bits $b_1 \cdots b_k$
   
   $\forall s \subseteq [k] \text{ st. } s \neq \emptyset \text{ set } C_s = \bigoplus b_i$
   
   * 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 $q=2^6$ (note that this 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 \( m \ll q \) needs \( O(\log q \) bits of randomness)\

- pick \( a, b \in \mathbb{Z}_q \)
- \( r_i \equiv a_i + b \mod q \quad \forall i \in \mathbb{Z}_q \)
- output \( r_1 \ldots r_q \)

Useful to think of as \( \text{foo from } h_{a,b} : [0..q-1] \rightarrow \mathbb{Z}_q \)

Family of \( \text{foo } \mathcal{H} = \{ h_1, h_2, \ldots \} \) \( \text{for } h_i : [N] \rightarrow [M] \) is "pairwise independent" if:

when \( \mathcal{H} \in_u \mathcal{D} \)

1) \( \forall x \in [N] , \mathcal{H}(x) \in_u [M] \)
2) \( \forall x_1 \neq x_2 \in [N] , \mathcal{H}(x_1), \mathcal{H}(x_2) \) independent

Equivalently: \( \forall x_1 \neq x_2 \in [N] \)
\( \forall y_1, y_2 \in [M] \)
\( \Pr \left[ \mathcal{H}(x_1) = y_1, \mathcal{H}(x_2) = y_2 \right] = \frac{1}{M^2} \)
Comments

- no single \( f \)tn is p.i. - have to pick a random \( f \)tn from a \( f \)mily

- given \( H \) \( \in \mathbb{Z} \) \( \mathbb{N} \) \( H(x) \) should be computable in time \( \text{poly} (\log N, \log M) \)

- also called "strongly 2-universal hash \( f \)tns"

Why is our example p.i.?

\[ H = \{(a,b) \in \mathbb{Z}_q \times \mathbb{Z}_q \} \]

\[ h_{a,b} = a \cdot x + b \mod q \]

Fix \( x \neq w, c, d \)

\[ \text{Pr}_{a,b} \left[ ax + b = c \land aw + b = d \right] = \frac{1}{q^2} \]

\[ \binom{x}{w} \binom{a}{b} = \binom{c}{d} \]

\( w \neq x \) so nonsingular \( \exists \) unique soln

how many truly random bits?

\( 2 \log q \) yields \( q \) p.i. random \( \text{field} \) elts.
More Comments

1. Can construct for all finite fields, even when domain and range have different sizes.

2. Original motivation: hashing
   hash texts chosen from p.i.i. family
   instead of random texts.

   Why is this good?

   How would you store a
   random ftn on a domain
   of size \( 2^k \)?

Dr. Li
Sp 2014