Pairwise independence & de-randomization

- 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 \( \sum_{(u,v) \in E} I_{u \in S, v \in T} \)

Analysis:

- Let \( 1_{u,v} = 1 \) if \( r_u \neq r_v \) (i.e., placed on different sides)
- \( \mathbb{E}[\text{cut}] = \sum_{(u,v) \in E} 1_{u,v} \)
- \( = \sum_{(u,v) \in E} \mathbb{E}[1_{u,v}] = \sum_{(u,v) \in E} \mathbb{P}_r[1_{u,v} = 1] \)
- \( = \sum_{(u,v) \in E} \mathbb{P}_r[r_u = 1 + r_v = 0] \) or \( r_u = 0 + r_v = 1 \)
- \( = \sum_{(u,v) \in E} \left( \frac{1}{2} \mathbb{P}_r[r_u = 1 + r_v = 0] + \frac{1}{2} \mathbb{P}_r[r_u = 0 + r_v = 1] \right) = \frac{|E|}{2} \)
if $E[\text{cut}] = \frac{|E|}{a}$ then $\exists$ cut of size $\frac{|E|}{a}$

**Why?**

- $E[\text{cut}]$ is just ave value of cuts coming from random process.
- must be at least one cut which is as big as average value.
Pairwise independent random variables: definition

Pick \( n \) values \( X_1 \ldots X_n \) each \( X_i \in \mathcal{X} \) (domain) s.t. \(|\mathcal{X}|=\ell \) (size of domain) in some way.

\[ \text{def. } X_1, \ldots, X_n \text{ independent } \iff \forall b_1, \ldots, b_n \in \mathcal{X}^n \]
\[ \Pr[X_1, \ldots, X_n = b_1, \ldots, b_n] = \frac{1}{\ell^n} \]

pairwise independent \( \iff \forall \ i \neq j \quad b_i, b_j \in \mathcal{T} \)
\[ \Pr[X_i, X_j = b_i, b_j] = \frac{1}{\ell^2} \]

\( k \)-wise independent \( \iff \forall \ \text{distinct } b_1, \ldots, b_k \in \mathcal{T}^k \)
\[ \Pr[X_1, \ldots, X_k = b_1, \ldots, b_k] = \frac{1}{\ell^k} \]

Main point:

1. Only use pairwise independence in max-cut algorithm (i.e., algorithm analysis still works if random bits are only pairwise indep).
   \[ \Rightarrow \text{if random bits p.i. then } E[\text{cut}] = \frac{|E|}{\ell} \]
   \[ \Rightarrow \exists \text{ cut chosen by p.i. bits which has size } \geq \frac{|E|}{\ell} \]

2. Can enumerate over fewer options!!
Derandomization of max-cut

**Full enumeration**: 
- \( n \) fully random bits → Algorithm → cut
- try all \( 2^n \) possible coin tosses
- pick best cut

**Partial enumeration**: 
- \( m \) pairwise independent random bits → Algorithm → cut
- don't try all possible coin tosses
- just a subset that satisfies pairwise independence

\[ \begin{align*}
\text{e.g.} & \quad \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{pmatrix} \\
\text{pick a row uniformly} & \quad \begin{pmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{pmatrix}
\end{align*} \]

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

\[ \text{for } i \neq j, b_i, b_j \in \{0,1\} \]
\[ \Pr[b_i = b_j] = \frac{1}{2} \]
\[ \text{good enough to give} \]
\[ E[\text{cut}] = \frac{\left| E \right|}{2} \]

Another picture

\[ b_1 \ldots b_m \]
- totally independent
- enumerate all \( 2^m \) choices

```
randomness generator
```
- pick a random row

```
good enough for our algorithm!
```

\( \text{Can we make } n \geq m? \)
Max Cut:

\[
\begin{align*}
0 & \\
2 & \quad \text{Value} = 2 \\
3 & \quad \text{Value} = 2
\end{align*}
\]

but we are just claiming to find cut of size \(\frac{|E|}{2} = \frac{3}{2} = 1\)

All cuts:

\[
\begin{align*}
\{0\} & \quad \text{Value} = 0 \\
\{0, 2\} & \quad \text{Value} = 2 \\
\{0, 3\} & \quad \text{Value} = 2 \\
\{0, 1, 2\} & \quad \text{Value} = 1 \\
\{0, 1, 3\} & \quad \text{Value} = 1 \\
\{0, 1, 2, 3\} & \quad \text{Value} = 1
\end{align*}
\]

\[\frac{|E|}{2} = \text{Average value: } \frac{2 \cdot 0 + 2 \cdot 2 + 2 \cdot 1 + 2 \cdot 1}{8} = 1 \]

\[\Rightarrow \exists \text{ cut of value 1}\]

\(r_{11}\) cuts:

\[
\begin{align*}
r_1 = r_2 = r_3 = 0 & \quad \text{Value} = 0 \\
1 & \\
2 & \\
3 & \quad \text{Value} = 0
\end{align*}
\]

\[
\begin{align*}
r_1 = 0 & \quad r_2 = r_3 = 1 \quad \text{Value} = 1 \\
0 & \\
2 & \\
3 & \quad \text{Value} = 1
\end{align*}
\]

\[
\begin{align*}
r_1 = r_3 = 1 & \quad r_2 = 0 \quad \text{Value} = 2 \\
0 & \quad r_1 = 0 \\
2 & \\
3 & \quad r_3 = 0
\end{align*}
\]

\[\frac{|E|}{2} = \text{Average value: } \frac{0 + 1 + 2 + 1}{4} = 1 \]

(same) Analysis \[\Rightarrow \exists \text{ cut of value 1}\]
derandomize Max-Cut is given “randomness generator” taking \((\log n)^3\) \(\Rightarrow n\) bits

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

Then: derandomize via enumeration

Deterministic MC alg:
For all choices of \(b_1 \ldots b_{\log n}\)
run \(MC'\) on \(b_1 \ldots b_{\log n}\) & evaluate cutsize
pick best cutsize

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

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
do "full enumeration" derandomization
on this in $O(2^n \times \text{time to generate} + \text{time to run MaxCut})$
How to generate pairwise independent random variables?

1) Bits

- Choose \( k \) truly random bits \( b_1, \ldots, b_k \)
  \[ \forall S \subseteq [k] \text{ s.t. } S \neq \emptyset \text{ set } C_S = \{ b_i \}_{i \in S} \]
- Output all \( C_S \)

Generates \( 2^k - 1 \) bits from \( k \) truly random bits

**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 \( 1 \ldots l \)

Uses \( O(\log n \cdot \log q) = O(\log n) \cdot \text{bits of free randomness} \)
Somewhat better construction:
(when \( m \leq q \) needs \( O(\log q) \) bits of randomness)

- Pick \( a, b \in \mathbb{Z}_q \)
- \( r_i \leq a \cdot i + b \mod q \quad \forall \ i \in \{0, \ldots, q\} \)
- Output \( r_1, \ldots, r_q \)

Useful to think of as \( \text{input/output description of a} \) taken from \( \text{indep/output description of a} \)

\[ h_{a,b} : \{0, \ldots, q\} \to \mathbb{Z}_q \]

Family of fns \( \mathcal{H} = \{ h_1, h_2, \ldots \} \) for \( h_i : \{N\} \to \{M\} \)

is "pairwise independent" if:

1. \( \forall x \in \{N\}, \quad H(x) \in \{M\} \)
2. \( \forall x_1 \neq x_2 \in \{N\}, \quad H(x_1), H(x_2) \) independent

Equivalently:
\( \forall x_1 \neq x_2 \in \{N\} \)
\( \forall y_1, y_2 \in \{M\} \)
\[ \Pr_{H \in \mathcal{H}} \left[ H(x_1) = y_1, \quad H(x_2) = y_2 \right] = \frac{1}{M^2} \]
no single fn is p.i. - have to pick a random fn from a family

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

also called "strongly 2-universal hash fnqs"

Why is our example p.i.?

\[ H = \{ h_{a,b} \mid \mathbb{Z}_q \rightarrow \mathbb{Z}_q \} \quad \text{(recall } q \text{ is prime)} \]

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

fix any \( x \neq w, c, d \)

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

\[ \begin{pmatrix} x \ 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix} \]

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

how many truly random bits?

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

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

- Original motivation: hashing
  hash fcn chosen from p.i.i. family
  instead of random fcn.

  Why is this good?

  how would you store a
  random fcn on a domain
  of size \( \ldots \)