Randomised Communication Complexity

$x = x_1x_2x_3 \ldots x_n$

Bob has a bit string $y \in Y$

Alice has a bit string $x \in X$

$y = y_1y_2y_3 \ldots y_n$

Bob has a random bit string

Alice has a random bit string

- $r_a$ and $r_b$ are private, independent and have arbitrary length
- Equivalently they could each have a private coin they can flip
- Each bit Alice sends may now also depend on $r_a$ (and Bob’s on $r_b$)
- Alice and Bob still want to compute $z = f(x,y)$
  but now they are allowed to make occasional mistakes

Protocol Trees for Randomised CC

- We say that protocol $P$ has error 1/3 if
  $\Pr(P(x,y) = f(x,y)) \geq 2/3$
- Here $P(x,y)$ is the value that Alice and Bob decide on for $f(x,y)$
- We can repeat the protocol a constant number of times
  to reduce the error to any constant $0 < \epsilon < 1/2$

Protocol Trees for Randomised CC

- The cost of a protocol on a particular $(x,y)$ and $r_a,r_b$
  is the number of bits sent
- The (overall) cost of a protocol is the maximum cost
  on any $(x,y)$ and $r_a,r_b$
The bit sent dictates the path taken.

Each node $v$ has a function $a_v$ or $b_v$ which dictates who sends a bit (and what to send).

The protocol ends when a leaf node is reached (which gives $P(x,y)$).

$P$ has error $1/3$ if $\Pr(P(x,y) = f(x,y)) \geq 2/3$.

A path tells us the bits sent, so the height of the tree is the cost of the protocol.

For a given $f$ we want to find the tree with the smallest height (the $P$ which uses the fewest bits).
Randomised Communication Complexity

\[ x = x_1 x_2 x_3 \cdots x_n \quad R \]

\[ y = y_1 y_2 y_3 \cdots y_n \quad P \]

- The randomised communication complexity of \( f, R(f) \) is the minimum cost of any protocol \( P \), which computes \( f \) with error 1/3
- We will be interested in proving upper and lower bounds on \( R(f) \)
- The deterministic communication complexity of \( f \) for a given \( f \)
- Note that \( R(f) \leq D(f) \)

Sometimes \( R(f) \ll D(f) \)

Equality again

\[ x = x_1 x_2 x_3 \cdots x_n \]

\[ y = y_1 y_2 y_3 \cdots y_n \]

- Let \( p \) be the smallest prime between \( 3n^2 \) and \( 6n^2 \)
  (for any \( m \geq 1 \), there is always at least one prime between \( m \) and \( 2m \))
- Let \( p \) be the smallest prime between \( 3n^2 \) and \( 6n^2 \)
- Using \( r_a \), Alice picks a random number, \( r \) between 0 and \( p - 1 \)
  which she sends to Bob in \( O(\log n) \) bits
  - Alice computes and sends
    - Bob computes and sends
    - They conclude that \( x = y \) if \( P_x(r) = P_y(r) \)

Equality again

\[ 3n^2 \leq p \leq 6n^2 \quad (p \text{ is prime}) \quad 0 \leq r \leq p - 1 \quad (r \text{ is random}) \]

\[ P_x(r) = \sum_{i=1}^{n} x_i r^i \mod p \quad P_y(r) = \sum_{i=1}^{n} y_i r^i \mod p \]

- What is the probability that the protocol is correct?
- Assume that \( x = y \), then \( P_x(r) = P_y(r) \) for all \( r \)
  \( \ldots \) so \( P \) is always correct if \( x = y \)

Equality again

\[ 3n^2 \leq p \leq 6n^2 \quad (p \text{ is prime}) \quad 0 \leq r \leq p - 1 \quad (r \text{ is random}) \]

\[ P_x(r) = \sum_{i=1}^{n} x_i r^i \mod p \quad P_y(r) = \sum_{i=1}^{n} y_i r^i \mod p \]

- What is the probability that the protocol is correct?
- Assume instead that \( x \neq y \), consider \( P(r) = P_x(r) - P_y(r) \mod p \)
  observe that \( P(r) = 0 \) if \( P_x(r) = P_y(r) \)

\[ P \text{ is a polynomial (in } r \text{) of degree } n \]
  with coefficients \( c_i = (x_i - y_i) \)
  \[ P(r) = \sum_{i=1}^{n} (x_i - y_i) r^i \mod p \]
  so \( P(r) = 0 \) for at most \( n \)
  different \( r \) values

\[ P(r) = \sum_{i=1}^{n} c_i r^i \mod p \]

So \( \Pr(P_x(r) = P_y(r)) \leq \frac{1}{p} \leq \frac{1}{6n^2} \leq \frac{1}{2} \)
### Equality again
\[ x = x_1 x_2 x_3 \ldots x_n \quad \text{and} \quad y = y_1 y_2 y_3 \ldots y_n \]

- So we can solve the Equality problem with error 1/3 using only \( O(\log n) \) bits.
- In fact we proved we could obtain error probability 1/n (with no extra bits).
- Further, we never get false-negative results.
- We can use this protocol as a basis for a GreaterThan protocol using binary search.

### The Disjointness problem
\[ x = x_1 x_2 x_3 \ldots x_n \quad \text{and} \quad y = y_1 y_2 y_3 \ldots y_n \]

- Alice and Bob want to compute \( \text{Dis}(x, y) \)
  \[ \text{Dis}(x, y) = 1 \text{ if there is no index, } i, \text{ such that } x_i = y_i = 1 \]
  \( \text{and} \text{Dis}(x, y) = 0 \text{ otherwise} \)
- So \( \text{Dis}(0110, 0001) = 1, \text{Dis}(0110, 0011) = 0 \text{ and Dis}(0110, 0100) = 0 \)
- For Disjointness, we saw that \( n \leq D(\text{Dis}) \leq n + 1 \)
  \( \text{i.e. there is no protocol which always takes less than } n \text{ bits} \)
- Can we do better using randomisation?

### The InnerProduct problem
\[ x = x_1 x_2 x_3 \ldots x_n \quad \text{and} \quad y = y_1 y_2 y_3 \ldots y_n \]

- Alice and Bob want to compute \( \text{IP}(x, y) \)
  \[ \text{IP}(x, y) = \sum_{i=1}^{n} (x_i \cdot y_i) \mod 2 \]
- IP(0110, 0001) = (0 \cdot 0) + (1 \cdot 0) + (1 \cdot 1) + (0 \cdot 1) \mod 2 = 1 \mod 2
- For InnerProduct, it is also known that
  \( n \leq D(\text{IP}) = R(\text{IP}) \leq n + 1 \)
  \( \text{ (it's also tricky) } \)

### How much better can we do randomised?

**Theorem** Any randomised protocol which uses \( h \)-bits (in the worst case) can be simulated deterministically using \( 2^h \cdot O(h) \) bits.

- As \( D(\text{EQ}) \geq n \), we have that \( R(\text{EQ}) \in \Omega(\log n) \)
  \( \text{i.e. the } O(\log n) \text{ bit randomised protocol for EQ is the best possible} \)

**Key Idea** Alice and Bob compute the probability \( p_\ell \) that \( P_\ell \) reaches leaf \( \ell \)
(only for inputs \( x,y \))

- They do this for each 1 leaf and sum the probabilities which gives \( \Pr(P_\ell(x, y) = 1) \)
- Alice and Bob cannot compute \( p_\ell \) by themselves because they only know their own inputs...
Simulating a randomised protocol

- Let $P_r$ be a randomised protocol which uses $h$ bits

Key Idea

Alice and Bob compute the probability $p_\ell$ that $P_r$ reaches leaf $\ell$ (for inputs $x,y$)

- Alice computes the probability $p_\ell^A$
  This is the probability that she will respond to Bob according to the path to $\ell$

With probability $p_\ell^A$ Alice does not cause the protocol to leave the path to $\ell$

- Alice can compute $p_\ell^A$ for each $\ell$ from $x$ and $r_a$

How many bits were sent?

- Alice sent $\leq 2^h$ values of $p_\ell$ to Bob

How many bits in each $p_\ell$?

Yikes! probabilities are real numbers

Maybe infinity bits!

- By the magic of rounding we can get away with $O(h)$ bits per $p_\ell$

Simulating a randomised protocol

- Let $P_r$ be a randomised protocol which uses $h$ bits

How much better can we do randomised?

- We have proven that, an $h$-bit randomised protocol can be simulated deterministically using $2^h \cdot O(h)$ bits
- We can deduce from this that, $D(f) \leq 2^{R(f)} \cdot O(R(f))$

Theorem

For any $f$, $R(f) \in \Omega(\log D(f))$

- As $D(EQ) \geq n$, we have that $R(EQ) \in \Omega(\log n)$
  i.e. the $O(\log n)$ bit randomised protocol for EQ is the best possible
- More generally if $D(f) \geq n$, then the best randomised protocol for $f$
  communicates $O(\log n)$ bits

Conclusions

- The Disjointness problem requires $\geq n$ bits even randomised
- The InnerProduct problem requires $\geq n$ bits even randomised
  all in the worst case over any input

- Protocols for the Equality problem require
  $\Theta(n)$ bits deterministically
  $\Theta(\log n)$ bits with private randomness

Theorem

For any $f$, $R(f) \in \Omega(\log D(f))$