A landscape sketch of Quantum Complexity

Niel de Beaudrap
Dept. Computer Science, Oxford

Oxford Cryptography Day
Cryptography and Quantum Computing
17 March 2016

A view of
Randomised Computation

Distributions are data,
and transform linearly

A view of

Randomised Computation

Distributions are data, and transform linearly

$x \in \{0,1\}^k$

$p(x_1, \ldots, x_k)$

$y \in \{0,1\}^r$

$p'(y_1, \ldots, y_r)$

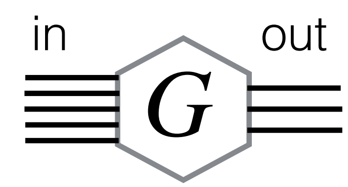
A view of

Randomised Computation

Distributions are data, and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$



$y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$

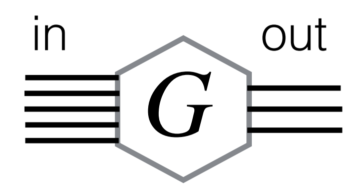
$p'(y_1, \ldots, y_r)$

$$p' = Gp$$

A view of
Randomised Computation

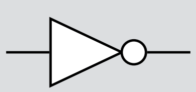
Distributions are data,
and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

NOT p  p' $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_0 \end{bmatrix}$

A view of Randomised Computation

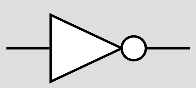
Distributions are data, and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $\quad$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

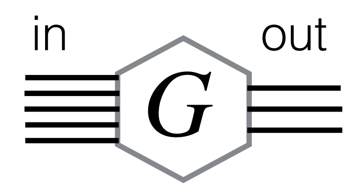
NOT  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

A view of

Randomised Computation

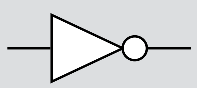
Distributions are data, and transform linearly

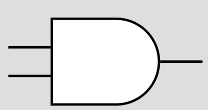
- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $\qquad$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

NOT  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

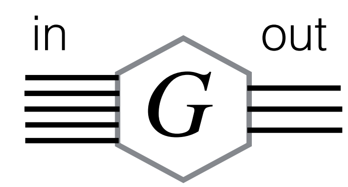
AND  $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{bmatrix} = \begin{bmatrix} p_{00}+p_{01}+p_{10} \\ p_{11} \end{bmatrix}$

A view of

Randomised Computation

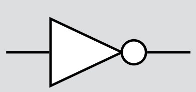
Distributions are data, and transform linearly

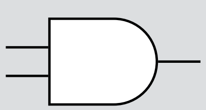
- Computations on random bits = linear (stochastic) transformations of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

NOT  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

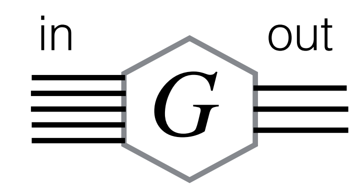
AND  $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

A view of

Randomised Computation

Distributions are data, and transform linearly

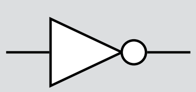
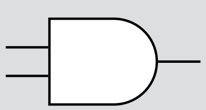
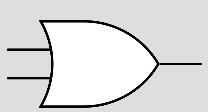
- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$ 

$y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $p'(y_1, \ldots, y_r)$

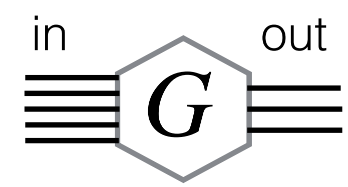
$$p' = Gp$$

NOT		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
AND		$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
OR		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$

A view of
Randomised Computation

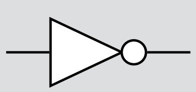
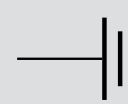
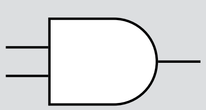
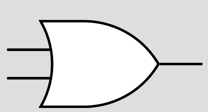
Distributions are data, and transform linearly

- Computations on random bits = linear (stochastic) transformations of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1,\dots,x_k)$ $\quad$ $p'(y_1,\dots,y_r)$

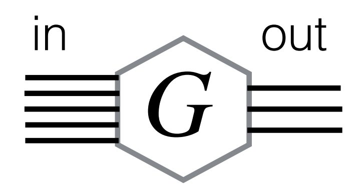
$$p' = Gp$$

NOT		$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$		ERASE		$\begin{bmatrix}1 & 1\end{bmatrix}$
AND		$\begin{bmatrix}1 & 1 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$				
OR		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 1 & 1\end{bmatrix}$				

A view of
Randomised Computation

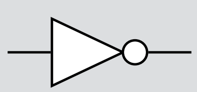
Distributions are data, and transform linearly

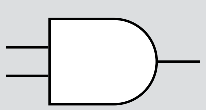
- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

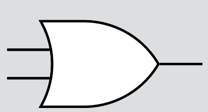
$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

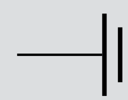
$p(x_1, \ldots, x_k)$ $\qquad$ $p'(y_1, \ldots, y_r)$

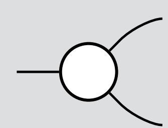
$$p' = Gp$$

NOT  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

AND  $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

OR  $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$

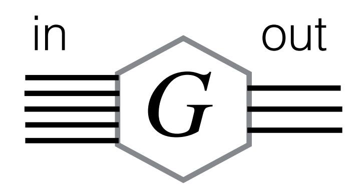
ERASE  $\begin{bmatrix} 1 & 1 \end{bmatrix}$

FANOUT  $\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$

A view of
Randomised Computation

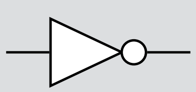
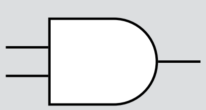
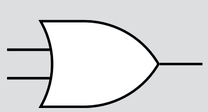
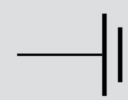
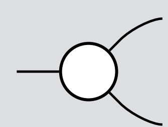
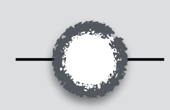
Distributions are data, and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $\quad$ $p'(y_1, \ldots, y_r)$

$$\boxed{p' = Gp}$$

NOT		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
AND		$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
OR		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$
ERASE		$\begin{bmatrix} 1 & 1 \end{bmatrix}$
FANOUT		$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$
MIX		$\begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$

A view of Randomised Computation

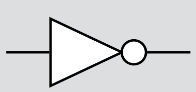
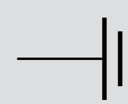
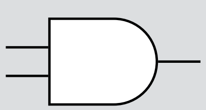
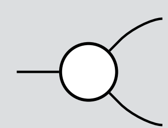
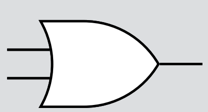
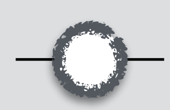
Distributions are data, and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $\quad$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

Gate	Symbol	Matrix	Gate	Symbol	Matrix
NOT		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	ERASE		$\begin{bmatrix} 1 & 1 \end{bmatrix}$
AND		$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	FANOUT		$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$
OR		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$	MIX		$\begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$

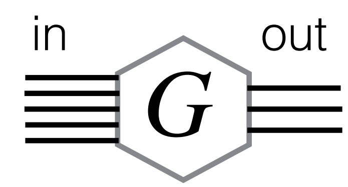
- Polynomial-size randomised circuits
  = efficiently describable (stochastic) tensor networks

A view of

Randomised Computation

Distributions are data, and transform linearly

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1, \ldots, x_k)$ $\qquad$ $p'(y_1, \ldots, y_r)$

$$p' = Gp$$

NOT	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	ERASE	$\begin{bmatrix} 1 & 1 \end{bmatrix}$
AND	$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	FANOUT	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$
OR	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$	MIX	$\begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$

- Polynomial-size randomised circuits
  = efficiently describable (stochastic) tensor networks

Exponentially long real-valued vectors?

Exponentially small amplitudes?

A view of
Randomised Computation

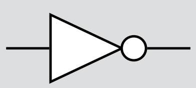
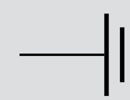
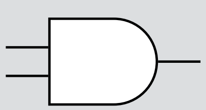
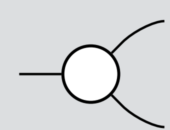
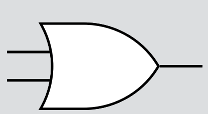
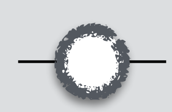
Distributions are data, and transform linearly

$x \in \{0,1\}^k$  $y \in \{0,1\}^r$

$p(x_1,\ldots,x_k)$ $\qquad$ $p'(y_1,\ldots,y_r)$

$$p' = Gp$$

- Computations on random bits
  = linear (stochastic) transformations
  of probability distributions

NOT		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	ERASE		$\begin{bmatrix} 1 & 1 \end{bmatrix}$
AND		$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	FANOUT		$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$
OR		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$	MIX		$\begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$

- Polynomial-size randomised circuits
  = efficiently describable (stochastic) tensor networks

Exponentially long real-valued vectors?

Exponentially small amplitudes?

Not a problem!
(these are *descriptions* of algorithms, not *products* of them)

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit 'distributions'
  over bit-strings

A view of Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit ‘distributions’ over bit-strings

$\longrightarrow$ $\uparrow$

$|0\rangle$ $|1\rangle$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

A view of Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit 'distributions' over bit-strings

$\rightarrow$	$\uparrow$	$\nearrow$	$\searrow$			
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert +\rangle$	$\lvert -\rangle$	$\psi(x)$	$\psi(x_1 x_2)$	*etc.*
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix}$	$\begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$	

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit 'distributions' over bit-strings

$\rightarrow$	$\uparrow$	$\nearrow$	$\searrow$			
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert +\rangle$	$\lvert -\rangle$	$\psi(x)$	$\psi(x_1 x_2)$	*etc.*
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix}$	$\begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$	

$$\mathbf{Pr}[x_n{=}0] = \sum_{y\in\{0,1\}^{n-1}} \left|\psi_{y;0}\right|^2$$

A view of Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit ‘distributions’ over bit-strings
- Valid operations = linear (ℓ_2-norm preserving) transformations

$\rightarrow$	$\uparrow$	$\nearrow$	$\searrow$			
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert +\rangle$	$\lvert -\rangle$	$\psi(x)$	$\psi(x_1 x_2)$	*etc.*
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix}$	$\begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$	

$$\mathbf{Pr}[x_n{=}0] = \sum_{y\in\{0,1\}^{n-1}} \left|\psi_{y;0}\right|^2$$

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit 'distributions' over bit-strings

$|0\rangle$ $|1\rangle$ $|+\rangle$ $|-\rangle$ $\psi(x)$ $\psi(x_1x_2)$ *etc.*

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}\psi_0\\\psi_1\end{bmatrix} \quad \begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$$

$$\mathbf{Pr}[x_n{=}0] = \sum_{y\in\{0,1\}^{n-1}} \left|\psi_{y;0}\right|^2$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit ‘distributions’ over bit-strings

$\rightarrow$	$\uparrow$	$\nearrow$	$\searrow$			
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert +\rangle$	$\lvert -\rangle$	$\psi(x)$	$\psi(x_1x_2)$	*etc.*
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix}$	$\begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$	

$$\mathbf{Pr}[x_n{=}0] = \sum_{y\in\{0,1\}^{n-1}} \left|\psi_{y;0}\right|^2$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix} \ \textit{s.t.}\ U^\dagger U = I_2$$

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit ‘distributions’ over bit-strings

$$\begin{array}{ccccccc} \rightarrow & \uparrow & \nearrow & \searrow & & & \\ |0\rangle & |1\rangle & |+\rangle & |-\rangle & \psi(x) & \psi(x_1x_2) & \textit{etc.} \\ \begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix} & \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} & \begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix} & \begin{bmatrix}\psi_0\\\psi_1\end{bmatrix} & \begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix} & \end{array}$$

$$\mathbf{Pr}[x_n\!=\!0] = \sum_{y\in\{0,1\}^{n-1}} \left|\psi_{y;0}\right|^2$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix} \quad \textit{s.t. } U^\dagger U = I_2$$

reversible computations
(*i.e.* permutation operators)

A view of
Quantum Computation

Distributions are data, and transform linearly

- Multi-qubit states = ℓ_2-unit 'distributions' over bit-strings

$\rightarrow$ $\uparrow$ $\nearrow$ $\searrow$

$|0\rangle$ $|1\rangle$ $|+\rangle$ $|-\rangle$ $\psi(x)$ $\psi(x_1x_2)$ *etc.*

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}\psi_0\\\psi_1\end{bmatrix} \quad \begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix} \quad s.t.\ U^\dagger U = I_2$$

reversible computations (*i.e.* permutation operators)

A view of
Quantum Computation

Distributions are data, and transform linearly

- Pure n-qubit states = ℓ_2-unit 'distributions' over bit-strings

$\rightarrow$	$\uparrow$	$\nearrow$	$\searrow$			
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert +\rangle$	$\lvert -\rangle$	$\psi(x)$	$\psi(x_1 x_2)$	*etc.*
$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix}$	$\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix}$	$\begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$	

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix}$$

reversible computations
(*i.e.* permutation operators)

A view of
Quantum Computation

Distributions are data, and transform linearly

- Pure n-qubit states = ℓ_2-unit 'distributions' over bit-strings

$\rightarrow$ $\uparrow$ $\nearrow$ $\searrow$

$|0\rangle$ $|1\rangle$ $|+\rangle$ $|-\rangle$ $\psi(x)$ $\psi(x_1 x_2)$ *etc.*

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}\psi_0\\\psi_1\end{bmatrix} \quad \begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \tfrac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix}$$

reversible computations
(*i.e.* permutation operators)

- Polynomial-size quantum circuits
  = efficiently describable (unitary) tensor networks

A view of Quantum Computation

Distributions are data, and transform linearly

- Pure n-qubit states = ℓ_2-unit 'distributions' over bit-strings

$$\rightarrow \quad \uparrow \quad \nearrow \quad \searrow$$

$$|0\rangle \quad |1\rangle \quad |+\rangle \quad |-\rangle \quad \psi(x) \quad \psi(x_1 x_2) \quad etc.$$

$$\begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{bmatrix} \quad \begin{bmatrix}\psi_0\\\psi_1\end{bmatrix} \quad \begin{bmatrix}\psi_{00}\\\psi_{01}\\\psi_{10}\\\psi_{11}\end{bmatrix}$$

- Valid operations = linear (ℓ_2-norm preserving) transformations

$$H = \tfrac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

$$U = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix}$$

reversible computations (*i.e.* permutation operators)

- Polynomial-size quantum circuits
  = efficiently describable (unitary) tensor networks

"Exponentially long vectors?"… no problem

Commonalities

between randomised and quantum algorithms

- Algorithms represented by tensor networks (“*circuits*”)
  (linear transformations on 1, 2, or 3 bits at a time)

Commonalities

between randomised and quantum algorithms

- Algorithms represented by tensor networks (“$circuits$”)
  (linear transformations on 1, 2, or 3 bits at a time)

- Space of distributions on n bits is compact
  (norm-bounded; transformations have bounded singular values)

Commonalities

between randomised and quantum algorithms

- Algorithms represented by tensor networks ("*circuits*")
  (linear transformations on 1, 2, or 3 bits at a time)

- Space of distributions on n bits is compact
  (norm-bounded; transformations have bounded singular values)

- Minute **individual** coefficients are not significant
  (*i.e.* unstructured search appears to require exponential time)

Example:
quantum *lower* bound for search

1996 Grover demonstrates an $O(2^{n/2})$-time quantum algorithm for search among n-bit strings

Example: quantum *lower* bound for search

1995 Bennett, Bernstein, Brassard, and Vazirani demonstrate an $\Omega(2^{n/2})$-time **lower** bound for search on n-bit strings via quantum algorithms

1996 Grover demonstrates an $O(2^{n/2})$-time quantum algorithm for search among n-bit strings

Example:
quantum *lower* bound for search

1995 Bennett, Bernstein, Brassard, and Vazirani demonstrate an $\Omega(2^{n/2})$-time **lower** bound for search on n-bit strings via quantum algorithms

1996 Grover demonstrates an $O(2^{n/2})$-time quantum algorithm for search among n-bit strings

Proof idea: Bound the effect of cumulative "oracle" queries on distributions, in a protocol with a generic input state

Example: quantum *lower* bound for search

1995 Bennett, Bernstein, Brassard, and Vazirani demonstrate an $\Omega(2^{n/2})$-time **lower** bound for search on n-bit strings via quantum algorithms

1996 Grover demonstrates an $O(2^{n/2})$-time quantum algorithm for search among n-bit strings

Proof idea: Bound the effect of cumulative "oracle" queries on distributions, in a protocol with a generic input state

$\implies$ quantum "parallelism" cannot directly simulate nondeterministic "parallelism"

One proof technique:

The polynomial method

An oracle $A : \{0,1\}^n \to \{0,1\}$ is represented by an (exponentially long) boolean string $A \in \{0,1\}^{2^n}$

One proof technique:
The polynomial method

An oracle $A : \{0,1\}^n \to \{0,1\}$ is represented by an (exponentially long) boolean string $A \in \{0,1\}^{2^n}$

- For any total function $\mathbf{P} : \{0,1\}^N \to \{0,1\}$ (where $N = 2^n$), how many queries to $A \in \{0,1\}^N$ does a quantum algorithm need, to compute $\mathbf{P}(A)$?

One proof technique:

The polynomial method

An oracle $A : \{0,1\}^n \to \{0,1\}$ is represented by an (exponentially long) boolean string $A \in \{0,1\}^{2^n}$

- For any total function $\mathbf{P} : \{0,1\}^N \to \{0,1\}$ (where $N = 2^n$), how many queries to $A \in \{0,1\}^N$ does a quantum algorithm need, to compute $\mathbf{P}(A)$?

Search is represented by the symmetric function $\mathbf{OR}(A) = \big[\exists x.\ A(x) = 1\big]$

One proof technique:

The polynomial method

An oracle $A : \{0,1\}^n \to \{0,1\}$ is represented by an (exponentially long) boolean string $A \in \{0,1\}^{2^n}$

q For any total function $\mathbf{P} : \{0,1\}^N \to \{0,1\}$ (where $N = 2^n$), how many queries to $A \in \{0,1\}^N$ does a quantum algorithm need, to compute $\mathbf{P}(A)$?

Search is represented by the symmetric function $\mathbf{OR}(A) = \big[\exists x.\ A(x) = 1\big]$

a at least $\frac{1}{2}\widetilde{\deg}(\mathbf{P})$ queries — where $\widetilde{\deg}$ is the minimum degree of a polynomial $f \in \mathbb{R}[x]$ approximating $\mathbf{P}$

One proof technique:

The polynomial method

An oracle $A : \{0,1\}^n \to \{0,1\}$ is represented by an (exponentially long) boolean string $A \in \{0,1\}^{2^n}$

q For any total function $\mathbf{P} : \{0,1\}^N \to \{0,1\}$ (where $N = 2^n$), how many queries to $A \in \{0,1\}^N$ does a quantum algorithm need, to compute $\mathbf{P}(A)$?

Search is represented by the symmetric function $\mathbf{OR}(A) = \big[\exists x.\ A(x) = 1\big]$

$\widetilde{\deg}(\mathbf{OR}) \sim \sqrt{N}$

a at least $\frac{1}{2}\widetilde{\deg}(\mathbf{P})$ queries — where $\widetilde{\deg}$ is the minimum degree of a polynomial $f \in \mathbb{R}[x]$ approximating $\mathbf{P}$

The polynomial method

arXiv:quant-ph/9802049

Lemma 4.1 *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X. Then there exist complex-valued N-variate multilinear polynomials $p_0, \ldots, p_{2^m-1}$, each of degree at most T, such that the final state of the network is the superposition*

$$\sum_{k \in K} p_k(X)|k\rangle,$$

for any black-box X.

The polynomial method

- Each query increases the polynomial degrees of the state-vector's coefficients by ≤ 1

arXiv:quant-ph/9802049

> **Lemma 4.1** *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X. Then there exist complex-valued N-variate multilinear polynomials $p_0, \ldots, p_{2^m-1}$, each of degree at most T, such that the final state of the network is the superposition*
>
> $$\sum_{k \in K} p_k(X)|k\rangle,$$
>
> *for any black-box X.*

The polynomial method

- Each query increases the polynomial degrees of the state-vector's coefficients by ≤ 1
- Probabilities are a quadratic polynomial in the state coefficients (factor of 2 in the degree)

arXiv:quant-ph/9802049

Lemma 4.1 *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X. Then there exist complex-valued N-variate multilinear polynomials $p_0, \ldots, p_{2^m-1}$, each of degree at most T, such that the final state of the network is the superposition*

$$\sum_{k \in K} p_k(X)|k\rangle,$$

for any black-box X.

The polynomial method

- Each query increases the polynomial degrees of the state-vector's coefficients by ≤ 1
- Probabilities are a quadratic polynomial in the state coefficients (factor of 2 in the degree)
- Queries of bounded-error algorithms scale with the degree of a bounded-error approximating polynomial

arXiv:quant-ph/9802049

Lemma 4.1 *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X. Then there exist complex-valued N-variate multilinear polynomials $p_0, \ldots, p_{2^m-1}$, each of degree at most T, such that the final state of the network is the superposition*

$$\sum_{k \in K} p_k(X)|k\rangle,$$

for any black-box X.

The polynomial method

- Each query increases the polynomial degrees of the state-vector's coefficients by ≤ 1
- Probabilities are a quadratic polynomial in the state coefficients (factor of 2 in the degree)
- Queries of bounded-error algorithms scale with the degree of a bounded-error approximating polynomial

arXiv:quant-ph/9802049

Lemma 4.1 *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X. Then there exist complex-valued N-variate multilinear polynomials $p_0, \ldots, p_{2^m-1}$, each of degree at most T, such that the final state of the network is the superposition*

$$\sum_{k \in K} p_k(X)|k\rangle,$$

for any black-box X.

— similar results hold for randomised algorithms as well

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

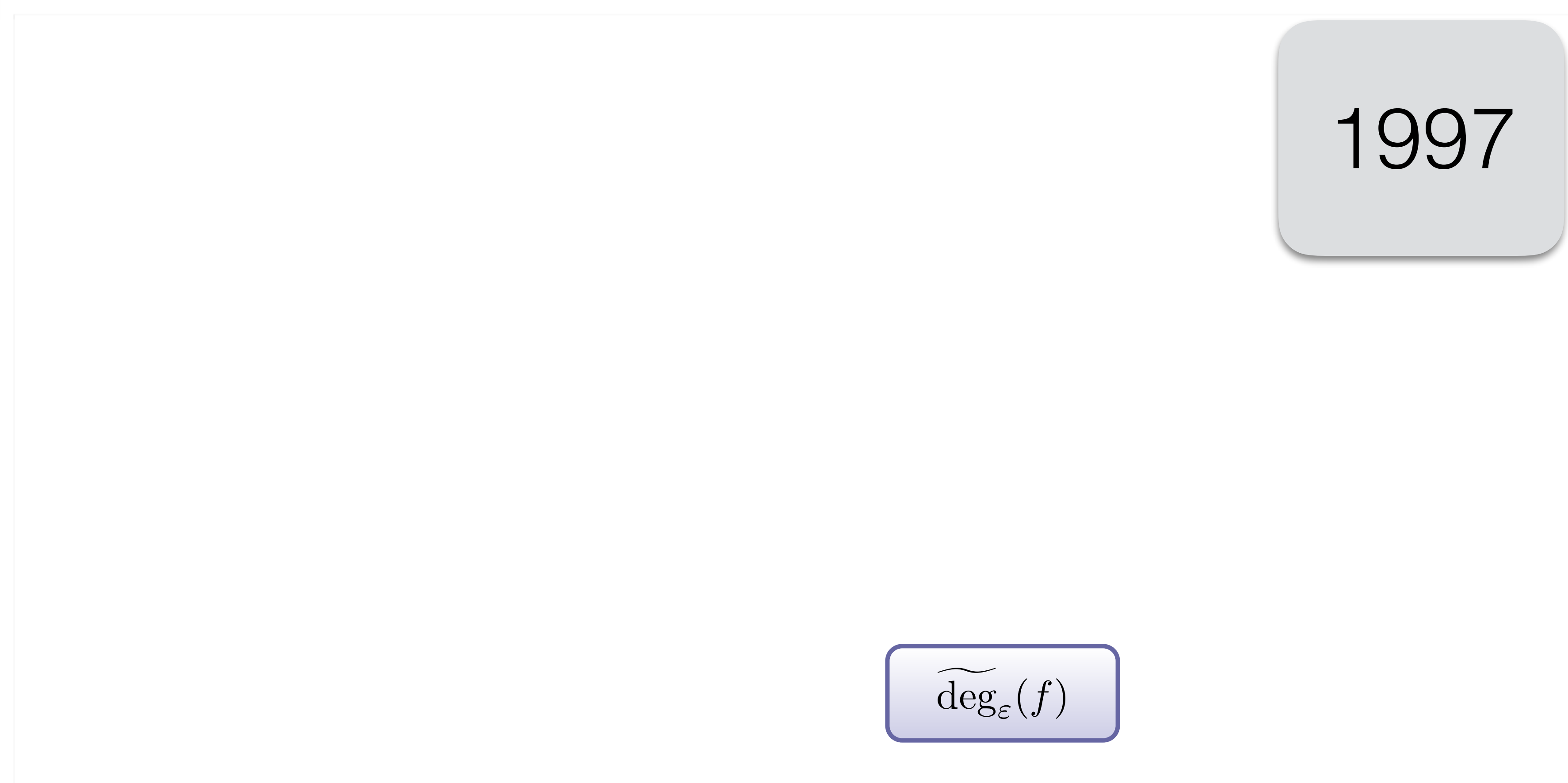


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

“Adversary” methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

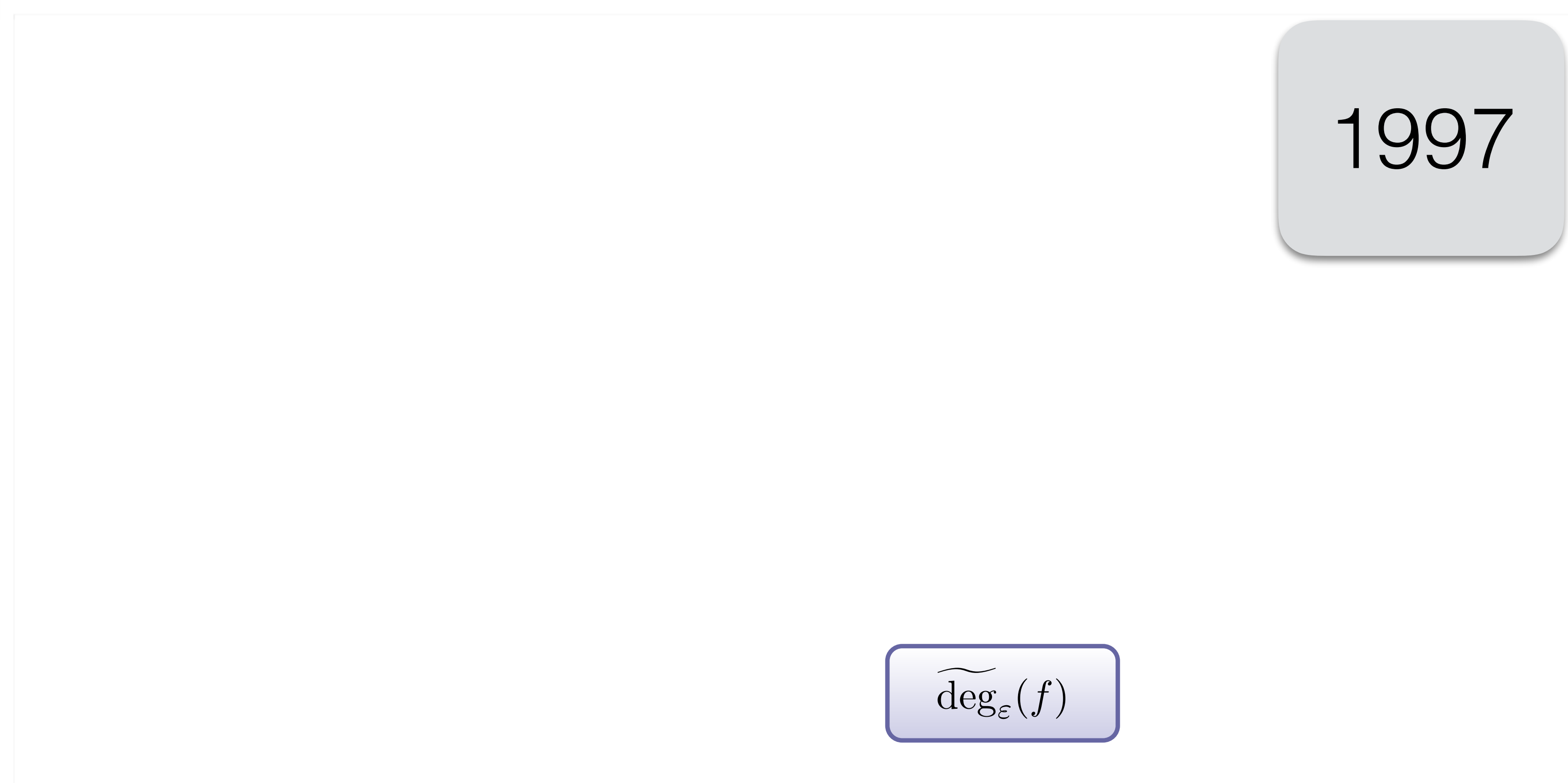


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

"Adversary" methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

- additive (2000)

2000

$$\boxed{\mathrm{ADV}_\varepsilon(f)} \gtrsim \boxed{\widetilde{\deg}_\varepsilon(f)}$$

Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

"Adversary" methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

- additive (2000)
- negative (2007)

2007

(arrows: constructive reductions)

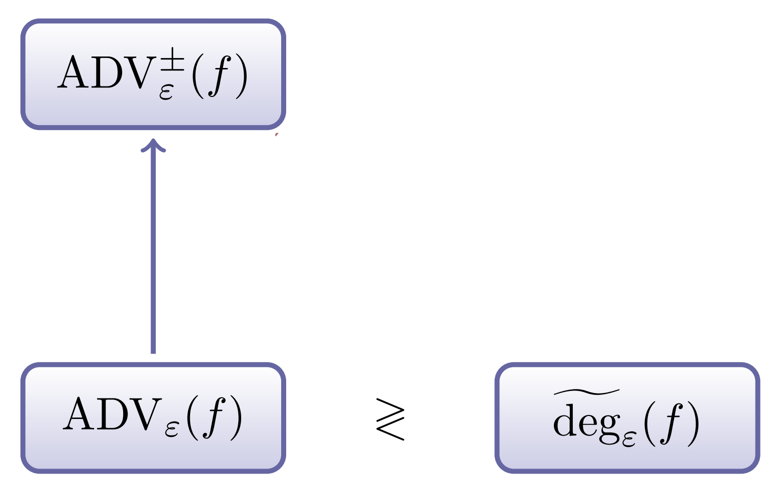


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

“Adversary” methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

- additive (2000)
- negative (2007)
- multiplicative (2011)

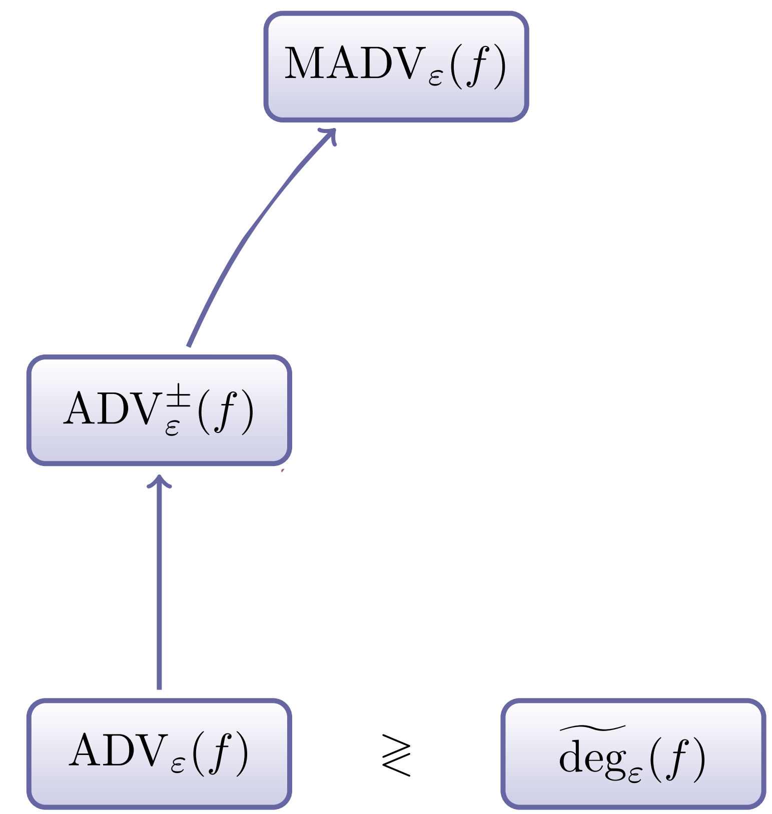


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

2011

(arrows: constructive reductions)

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

“Adversary” methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

- additive (2000)
- negative (2007)
- multiplicative (2011)

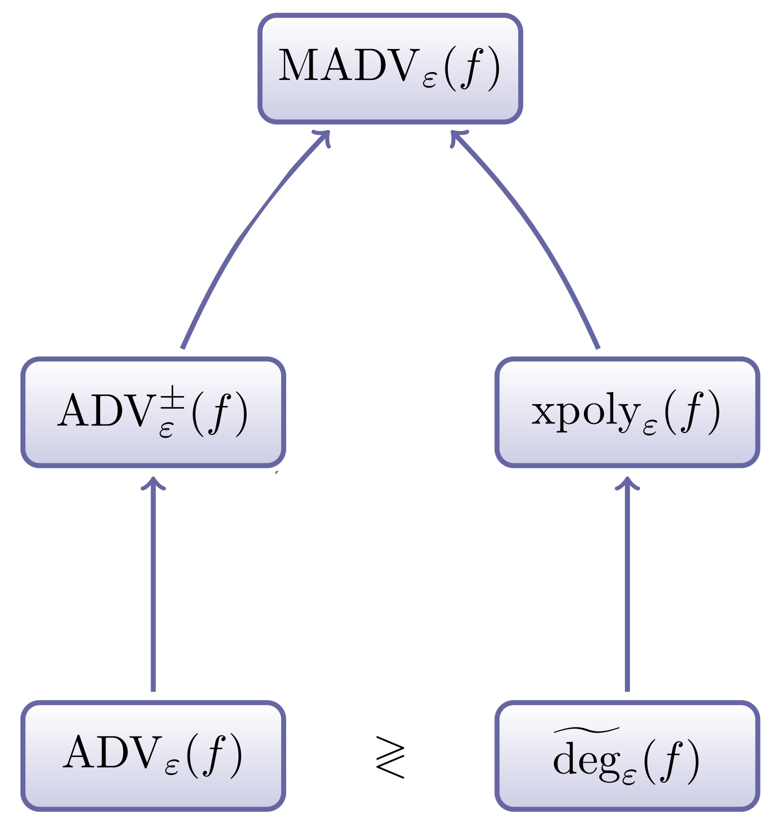


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

2012

(arrows: constructive reductions)

Overview of lower bound techniques

in quantum *query* complexity

(adapted from `arXiv:1209.2713`)

"Adversary" methods — lower-bound the query complexity, by measuring amount of work needed to distinguish inputs with different outputs

- additive (2000)
- negative (2007)
- multiplicative (2011)

Both the "negative" and "multiplicative" adversary methods characterise quantum query complexity [`arXiv:0904.2759`]

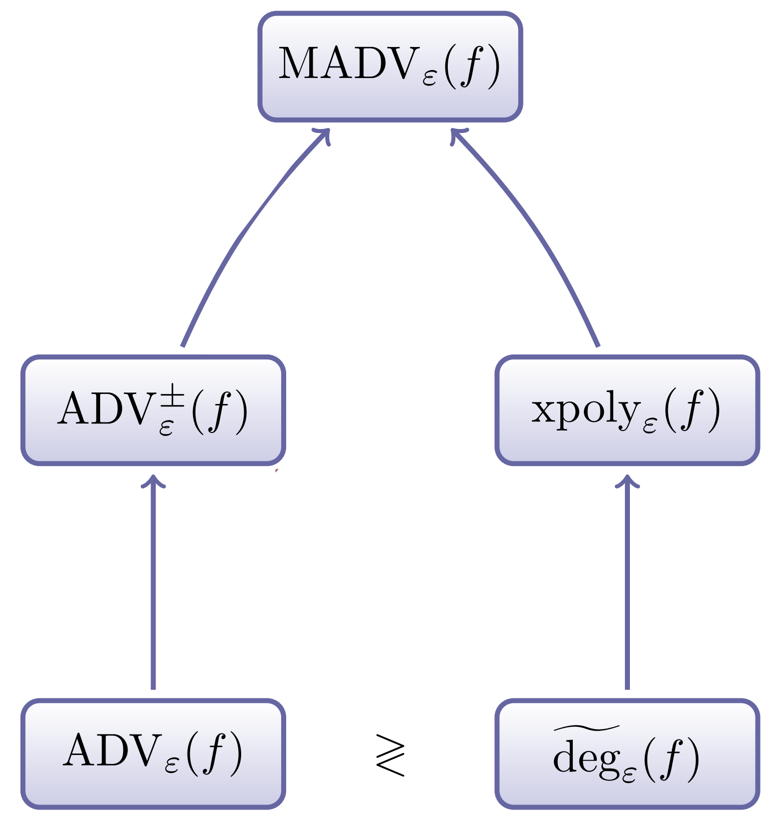


Figure 1: Relations between the different methods to prove lower bounds for quantum query complexity. An arrow from method A to method B implies that any lower bound that can be proved with A can also be proved with B (i.e., B is stronger than A).

2012

(arrows: constructive reductions)

What is “the source” of the quantum advantage?

- Why do we suspect a speed-up?
  (Why is there not a randomised factoring algorithm?)

What is “the source” of the quantum advantage?

q Why do we suspect a speed-up?
(Why is there not a randomised factoring algorithm?)

a The barrier is not the **number** of queries, but
the sort of information you can get **with** those queries

What is “the source” of the quantum advantage?

- **q** Why do we suspect a speed-up?
  (Why is there not a randomised factoring algorithm?)

- **a** The barrier is not the **number** of queries, but
  the sort of information you can get **with** those queries

- **q** What resources provide the “quantum advantage”?

What is "the source" of the quantum advantage?

q Why do we suspect a speed-up?
(Why is there not a randomised factoring algorithm?)

a The barrier is not the **number** of queries, but
the sort of information you can get **with** those queries

q ~~What resources provide the "quantum advantage"?~~
What **computational tricks** provide the
" quantum advantage " ?

What is “the source” of the quantum advantage?

q Why do we suspect a speed-up?
(Why is there not a randomised factoring algorithm?)

a The barrier is not the **number** of queries, but
the sort of information you can get **with** those queries

q ~~What resources provide the “quantum advantage”?~~
What **computational tricks** provide the
“quantum advantage” ?

eg. the “eigenspace trick” (as one may call it)

The “eigenspace trick”?

RSA
Factoring
Discrete Log
Pell’s Equation

$\rightarrow$ Order finding

The “eigenspace trick”?

RSA
Factoring
Discrete Log
Pell’s Equation

Order finding

Fourier sampling

The “eigenspace trick”?

RSA
Factoring
Discrete Log
Pell’s Equation

Order finding

Eigenvalue estimation

Fourier sampling

The “eigenspace trick”?

RSA
Factoring
Discrete Log
Pell’s Equation

Order finding

Eigenvalue estimation

‘Solving’ linear equations

error ~ poly(1/ε)
[arXiv:0811.3171]
[arXiv:1010.4458]

Fourier sampling

The “eigenspace trick”?

RSA
Factoring
Discrete Log
Pell’s Equation

Order finding

Eigenvalue estimation

Fourier sampling

‘Solving’ linear equations

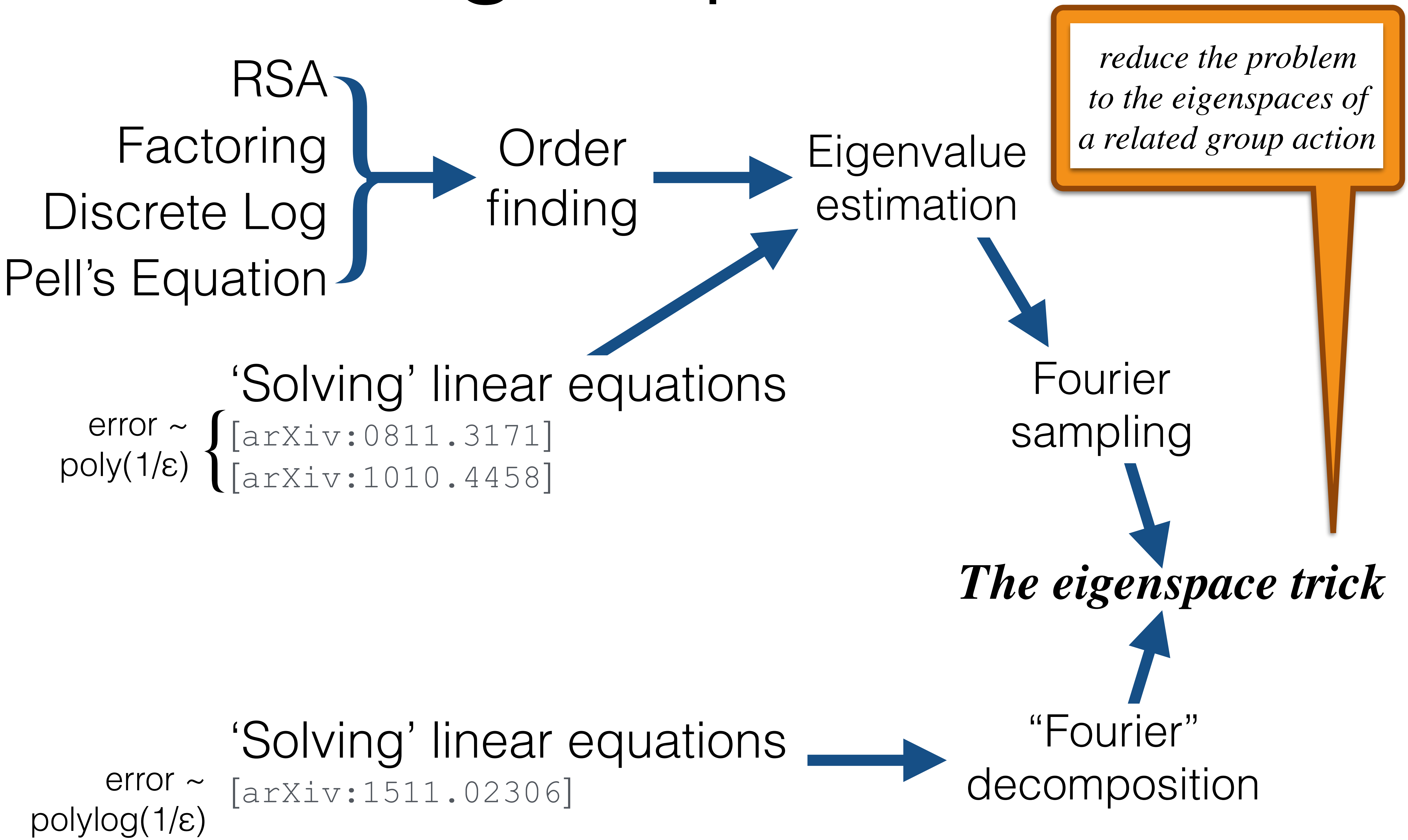
error ~ poly(1/ε)
[arXiv:0811.3171]
[arXiv:1010.4458]

‘Solving’ linear equations

error ~ polylog(1/ε)
[arXiv:1511.02306]

“Fourier” decomposition

The "eigenspace trick"?



A quick description of

The "eigenspace trick" (#1)

reduce the problem to the eigenspaces of a related group action

A quick description of

The "eigenspace trick" (#1)

reduce the problem to the eigenspaces of a related group action

Eigenvalue estimation

A quick description of

The "eigenspace trick" (#1)

reduce the problem to the eigenspaces of a related group action

Eigenvalue estimation

1. Prepare a "probe" system in a uniform distribution over strings $\{0,1\}^m$

A quick description of

The "eigenspace trick" (#1)

reduce the problem to the eigenspaces of a related group action

Eigenvalue estimation

1. Prepare a "probe" system in a uniform distribution over strings $\{0,1\}^m$
2. Use the probe to (coherently) control how many times $0 \leq t \leq \log(m)$ some **group action** A is performed on an input state

A quick description of

The "eigenspace trick" (#1)

reduce the problem to the eigenspaces of a related group action

Eigenvalue estimation

1. Prepare a "probe" system in a uniform distribution over strings $\{0,1\}^m$

2. Use the probe to (coherently) control how many times $0 \le t \le \log(m)$ some **group action A** is performed on an input state

3. Perform an inverse-Fourier transform on the probe, to obtain **a distribution on** (estimates of) **eigenvalues of A**

A quick description of

The “eigenspace trick” (#1)

reduce the problem to the eigenspaces of a related group action

Eigenvalue estimation

1. Prepare a “probe” system in a uniform distribution over strings $\{0,1\}^m$

2. Use the probe to (coherently) control how many times $0 \leq t \leq \log(m)$ some **group action A** is performed on an input state

3. Perform an inverse-Fourier transform on the probe, to obtain **a distribution on** (estimates of) **eigenvalues of A**

(4. Perform further operations conditioned on eigenvalue estimates)

(e.g. obtain a rational estimate, and find the order of A)

A quick description of

The "eigenspace trick" (#2)

reduce the problem to the eigenspaces of a related group action

A quick description of

The “eigenspace trick” (#2)

reduce the problem to the eigenspaces of a related group action

Fourier decomposition

Take a ‘Fourier’ decomposition of a group action:

- Decompose as a series of **commuting operators**
- Constraint: individual operators are efficient to perform
- Truncate so that the error is bounded on **eigenvalues of interest**

A quick description of

The "eigenspace trick" (#2)

reduce the problem to the eigenspaces of a related group action

Fourier decomposition

Take a 'Fourier' decomposition of a group action:

- Decompose as a series of **commuting operators**
- Constraint: individual operators are efficient to perform
- Truncate so that the error is bounded on **eigenvalues of interest**

1. Prepare a "control" system in a distribution, proportional to the Fourier decomposition of the group action

A quick description of

The "eigenspace trick" (#2)

reduce the problem to the eigenspaces of a related group action

Fourier decomposition

Take a 'Fourier' decomposition of a group action:

- Decompose as a series of **commuting operators**
- Constraint: individual operators are efficient to perform
- Truncate so that the error is bounded on **eigenvalues of interest**

1. Prepare a "control" system in a distribution, proportional to the Fourier decomposition of the group action
2. Use the control to (coherently) perform operations from the family of commuting operators

A quick description of

The "eigenspace trick" (#2)

reduce the problem to the eigenspaces of a related group action

Fourier decomposition

Take a 'Fourier' decomposition of a group action:

- Decompose as a series of **commuting operators**
- Constraint: individual operators are efficient to perform
- Truncate so that the error is bounded on **eigenvalues of interest**

1. Prepare a "control" system in a distribution, proportional to the Fourier decomposition of the group action
2. Use the control to (coherently) perform operations from the family of commuting operators
3. Perform a transformation to 'erase' the control (with high probability)

A quick description of

The "eigenspace trick" (#2)

reduce the problem to the eigenspaces of a related group action

Fourier decomposition

Take a 'Fourier' decomposition of a group action:

- Decompose as a series of **commuting operators**
- Constraint: individual operators are efficient to perform
- Truncate so that the error is bounded on **eigenvalues of interest**

1. Prepare a "control" system in a distribution, proportional to the Fourier decomposition of the group action
2. Use the control to (coherently) perform operations from the family of commuting operators
3. Perform a transformation to 'erase' the control (with high probability)

(4. Condition on success of erasure)

The crux of
the eigenspace trick

The crux of the eigenspace trick

- Permutations, etc. can have many stationary distributions
  (though many of them may not be probability distributions)

The crux of the eigenspace trick

- Permutations, etc. can have many stationary distributions
  (though many of them may not be probability distributions)

- Ability of quantum computers to access eigenvectors
  $\implies$ greater versatility

The crux of the eigenspace trick

- Permutations, etc. can have many stationary distributions
  (though many of them may not be probability distributions)

- Ability of quantum computers to access eigenvectors
  $\implies$ greater versatility

Caveat scriptor

If problem **X** has more convenient structure than problem **Y**, useful group actions for **X** may be easier to access

Quantum resistant problems?

Approach: consider variants of NP-hard problems, which seem likely not to be susceptible to the eigenspace trick — *for example:*

Quantum resistant problems?

Approach: consider variants of NP-hard problems, which seem likely not to be susceptible to the eigenspace trick — *for example:*

Lattice-based: derived from NP-hard problems about lattices

Quantum resistant problems?

Approach: consider variants of NP-hard problems, which seem likely not to be susceptible to the eigenspace trick — *for example:*

Lattice-based: derived from NP-hard problems about lattices

Code-based: based on the difficulty of decoding linear error-correcting codes

Quantum resistant problems?

Approach: consider variants of NP-hard problems, which seem likely not to be susceptible to the eigenspace trick — *for example:*

Lattice-based: derived from NP-hard problems about lattices

Code-based: based on the difficulty of decoding linear error-correcting codes

Multivariate equations: based on the difficulty of solving systems of polynomial equations in many variables

Lattice-based problems

Lattice: finitely generated subgroup of $\mathbb{R}^n$
given by a minimal set of generators

Lattice-based problems

Lattice: finitely generated subgroup of $\mathbb{R}^n$
given by a minimal set of generators

SVP: does the shortest vector in the lattice have
length at most 1, or greater than 1 (ℓ_2-norm)?
— **NP-hard**

Lattice-based problems

Lattice: finitely generated subgroup of $\mathbb{R}^n$ given by a minimal set of generators

SVP: does the shortest vector in the lattice have length at most 1, or greater than 1 (ℓ_2-norm)? — **NP-hard**

SVP$_\gamma$: does the shortest vector in the lattice have length at most 1, or greater than $\gamma \in \omega(1)$? — basis of NTRU, Ring-LWE

Lattice-based problems

Lattice: finitely generated subgroup of $\mathbb{R}^n$ given by a minimal set of generators

SVP: does the shortest vector in the lattice have length at most 1, or greater than 1 (ℓ_2-norm)? — **NP-hard**

SVP_γ : does the shortest vector in the lattice have length at most 1, or greater than $\gamma \in \omega(1)$? — basis of NTRU, Ring-LWE

Post-quantum question: for a given γ, are "useful" group actions hard to access for quantum computers?

Code-based problems

Linear code: finitely generated subgroup of $\mathbb{Z}_2^n$ ~ $\{0,1\}^n$ given by a minimal set of generators

Code-based problems

Linear code: finitely generated subgroup of $\mathbb{Z}_2^n \sim \{0,1\}^n$ given by a minimal set of generators

Decoding: what is the nearest "codeword" to a given n bit string? — **NP-hard**

Code-based problems

Linear code: finitely generated subgroup of $\mathbb{Z}_2^n \sim \{0,1\}^n$ given by a minimal set of generators

Decoding: what is the nearest "codeword" to a given n bit string? — **NP-hard**

***eg.* McEliece problem:** given a cyphertext $x \in \{0,1\}^n$, and a (public) generator $\hat{G} = PGS$ of some efficiently decodable linear code (for some private obfuscating operations P and S), find the codeword or plaintext which corresponds to x.

Code-based problems

Linear code: finitely generated subgroup of $\mathbb{Z}_2^n \sim \{0,1\}^n$ given by a minimal set of generators

Decoding: what is the nearest “codeword” to a given n bit string? — **NP-hard**

***eg.* McEliece problem:** given a cyphertext $x \in \{0,1\}^n$, and a (public) generator $\hat{G} = PGS$ of some efficiently decodable linear code (for some private obfuscating operations P and S), find the codeword or plaintext which corresponds to x .

Post-quantum question: Does restricting to efficiently decodable linear codes make “useful” group actions accessible to a quantum attacker?

Problems based on multivariate polynomials

Polynomial eqn. system: Find a solution to a system of poly(n) equations in n unknowns over a finite field — **NP-hard**

Problems based on multivariate polynomials

Polynomial eqn. system: Find a solution to a system of poly(n) equations in n unknowns over a finite field — **NP-hard**

UOV problem: Compute a `[signature]` for a given `[message]`, such that `[message]` = **F**(`[signature]`), where **F** is a (public) system of multivariate polynomials

(and **F** is related to an easily solved private system, by a privately held linear transformation of `[signature]`)

Problems based on multivariate polynomials

Polynomial eqn. system: Find a solution to a system of poly(n) equations in n unknowns over a finite field — **NP-hard**

UOV problem: Compute a `[signature]` for a given `[message]`, such that `[message]` = **F**(`[signature]`), where **F** is a (public) system of multivariate polynomials

(and **F** is related to an easily solved private system, by a privately held linear transformation of `[signature]`)

Post-quantum question: Does the privately held similarity transform suffice, to hide the privately held system of equations?

Overview / Outlook

Overview / Outlook

- What cryptographic techniques can be motivated by similarities between randomised and quantum computation?

Overview / Outlook

- What cryptographic techniques can be motivated by similarities between randomised and quantum computation?

  — $eg.$ lower bounds on query complexity

Overview / Outlook

- What cryptographic techniques can be motivated by similarities between randomised and quantum computation?

  — *eg.* lower bounds on query complexity

- What trapdoor problems can we devise which are immune against the eigenspace trick, and yet efficient to perform?

Overview / Outlook

- What cryptographic techniques can be motivated by similarities between randomised and quantum computation?

  — *eg.* lower bounds on query complexity

- What trapdoor problems can we devise which are immune against the eigenspace trick, and yet efficient to perform?

  — *eg.* involving non-abelian group actions

Overview / Outlook

- What cryptographic techniques can be motivated by similarities between randomised and quantum computation?

  — *eg*. lower bounds on query complexity

- What trapdoor problems can we devise which are immune against the eigenspace trick, and yet efficient to perform?

  — *eg*. involving non-abelian group actions

- What other strategies (beyond the eigenspace trick) may form the basis of useful quantum algorithms?