Title: Hardness of correcting errors on a stabilizer code

Date: Jan 22, 2014 04:00 PM

URL: http://pirsa.org/14010099

Abstract: Problems in computer science are often classified based on the scaling of the runtimes for algorithms that can solve the problem. Easy problems are efficiently solvable but often in physics we encounter problems that take too long to be solved on a classical computer. Here we look at one such problem in the context of quantum error correction. We will further show that no efficient algorithm for this problem is likely to exist. We will address the computational hardness of a decoding problem, pertaining to quantum stabilizer codes considering independent X and Z errors on each qubit. Much like classical linear codes, errors are detected by measuring certain check operators which yield an error syndrome, and the decoding problem consists of determining the most likely recovery given the syndrome. The corresponding classical problem is known to be NP-Complete, and a similar decoding problem for quantum codes is known to be NP-Complete too. However, this decoding strategy is not optimal in the quantum setting as it does not take into account error degeneracy, which causes distinct errors to have the same effect on the code. Here, we show that optimal decoding of stabilizer codes is computationally much harder than optimal decoding of classical linear codes, it is #P-Complete.

Hardness of correcting errors on a Stabilizer code

Pavithran Iyer, Maîtrise En Physique,

Superviseur: David Poulin, Université de Sherbrooke

Quantum Discussions @ Perimeter Institute, Jan 22th, 2014

In this talk . . .

1. Computational Complexity
2. Classical error correction
3. Quantum error correction
4. Main result
5. Outline of the proof
6. Conclusions

Easy and hard problems in computer science

Some problems are easy $\rightarrow$ we can solve them "efficiently": Ex. Arithmetic operations, ...

P: All problems that can be solved in polynomial-time (polynomial in input size)

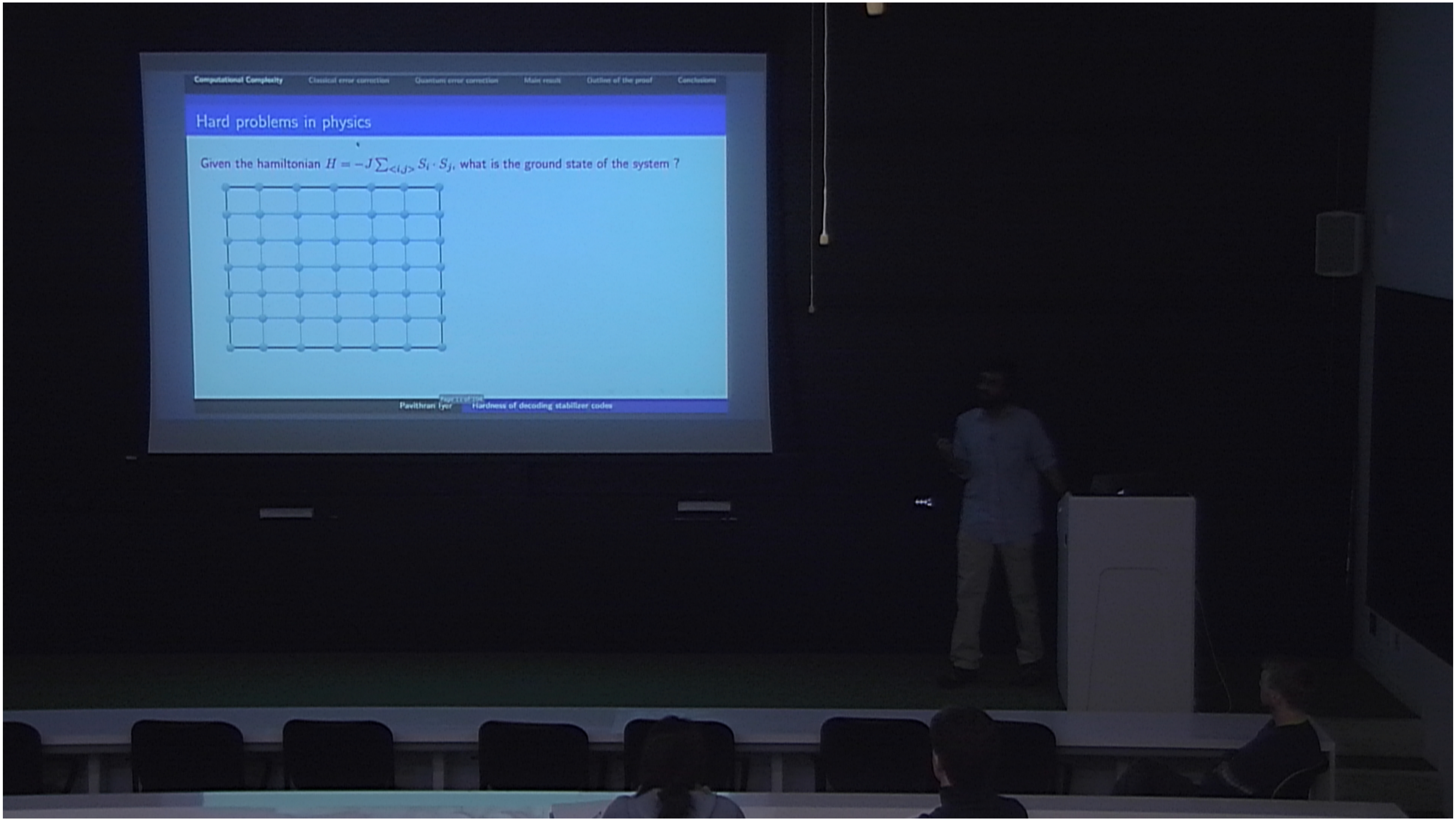
Often, we do not have an efficient solution. But we can verify any proposal in poly-time.

NP: All problems such that any certificate (proposal) can be verified in polynomial-time.

Some problems need a lot of effort $\rightarrow$ if we can solve them, we can solve any NP problem.

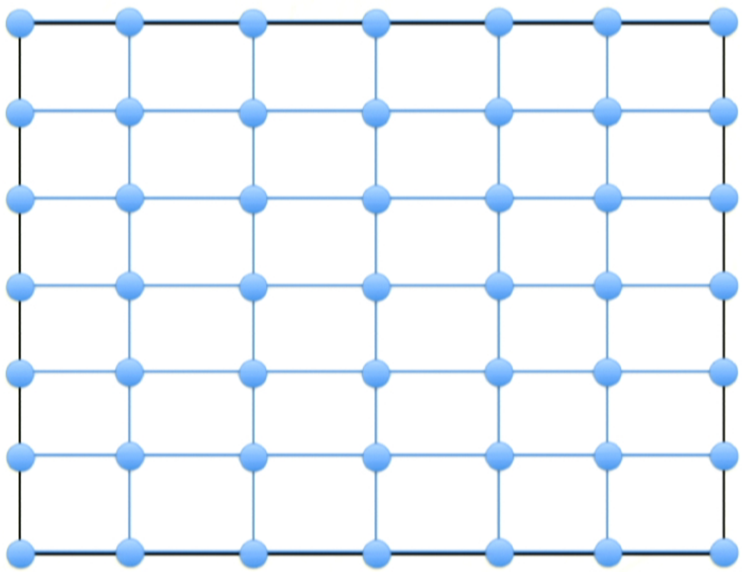
NP-Complete: Problems whose solution can be used to solve any NP problem in poly-time.

Sometimes we are not happy with just one solution ... want to know how many are there ?


Computational Complexity
Classical error correction
Quantum error correction
Main result
Outline of the proof
Conclusions
Hard problems in physics
Given the hamiltonian $H = -J\sum_{<i,j>} S_i \cdot S_j$, what is the ground state of the system ?
Pavithran Iyer
Hardness of decoding stabilizer codes

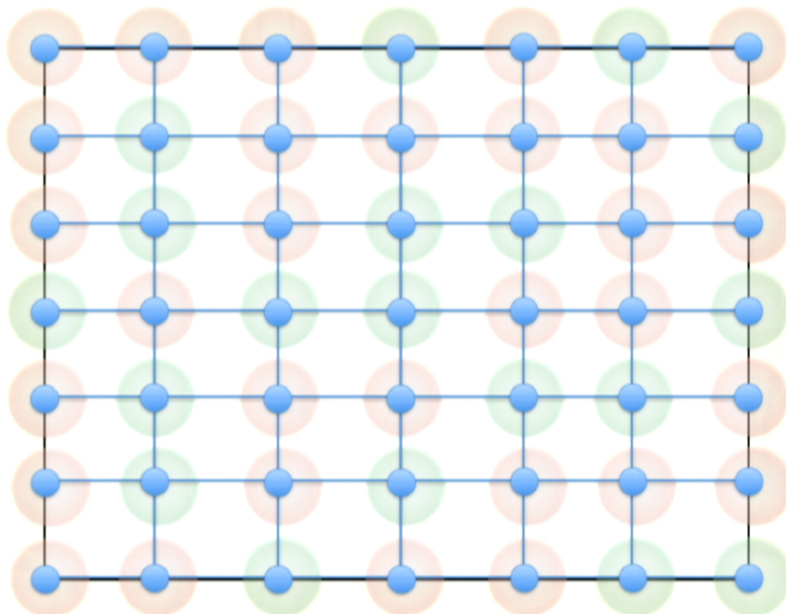
Hard problems in physics

Given the hamiltonian $H = -J\sum_{<i,j>} S_i \cdot S_j$, what is the ground state of the system ?



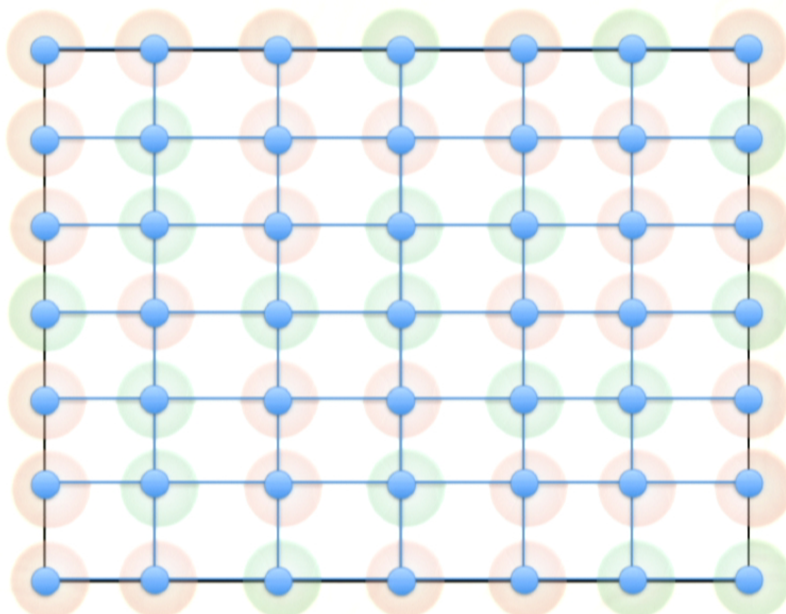
Hard problems in physics

Given $H = -J\sum_{<i,j>} S_i \cdot S_j - \sum_i h_i S_i$, is there a state of the system with energy $\leq E$?



Really hard problems in physics

Given $H = -J\sum_{<i,j>} S_i \cdot S_j - \sum_i h_i S_i$, compute the partition function: $\mathcal{Z}(\beta) = ?$



$$\mathcal{Z} = A_{\epsilon_1}e^{-\epsilon_1} + A_{\epsilon_2}e^{-\epsilon_2} + A_{\epsilon_3}e^{-\epsilon_3} + \dots$$

A_ϵ $\rightarrow$ how many states have energy ϵ

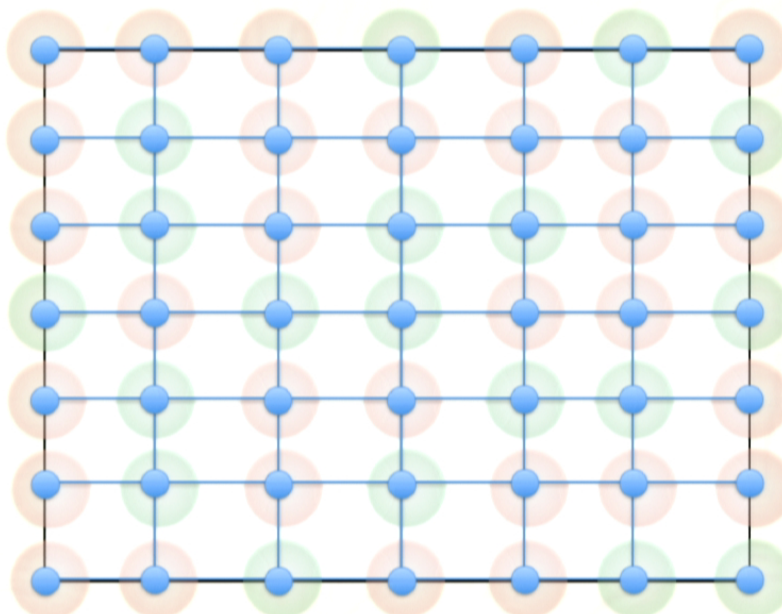
We are now counting solutions to the previous NP problem $\hookrightarrow$ problem $\in \#$P

If we can solve this, we can solve many more hopelessly hard counting problems in computer science ! $\hookrightarrow \in \#$P-Complete

[Goldberg: SIAM J. Com, 39(7), 3336–3402]

Really hard problems in physics

Given $H = -J \sum_{<i,j>} S_i \cdot S_j - \sum_i h_i S_i$, compute the partition function: $\mathcal{Z}(\beta) \;= ?$



$$\mathcal{Z} = A_{\epsilon_1} e^{-\epsilon_1} + A_{\epsilon_2} e^{-\epsilon_2} + A_{\epsilon_3} e^{-\epsilon_3} + \dots$$

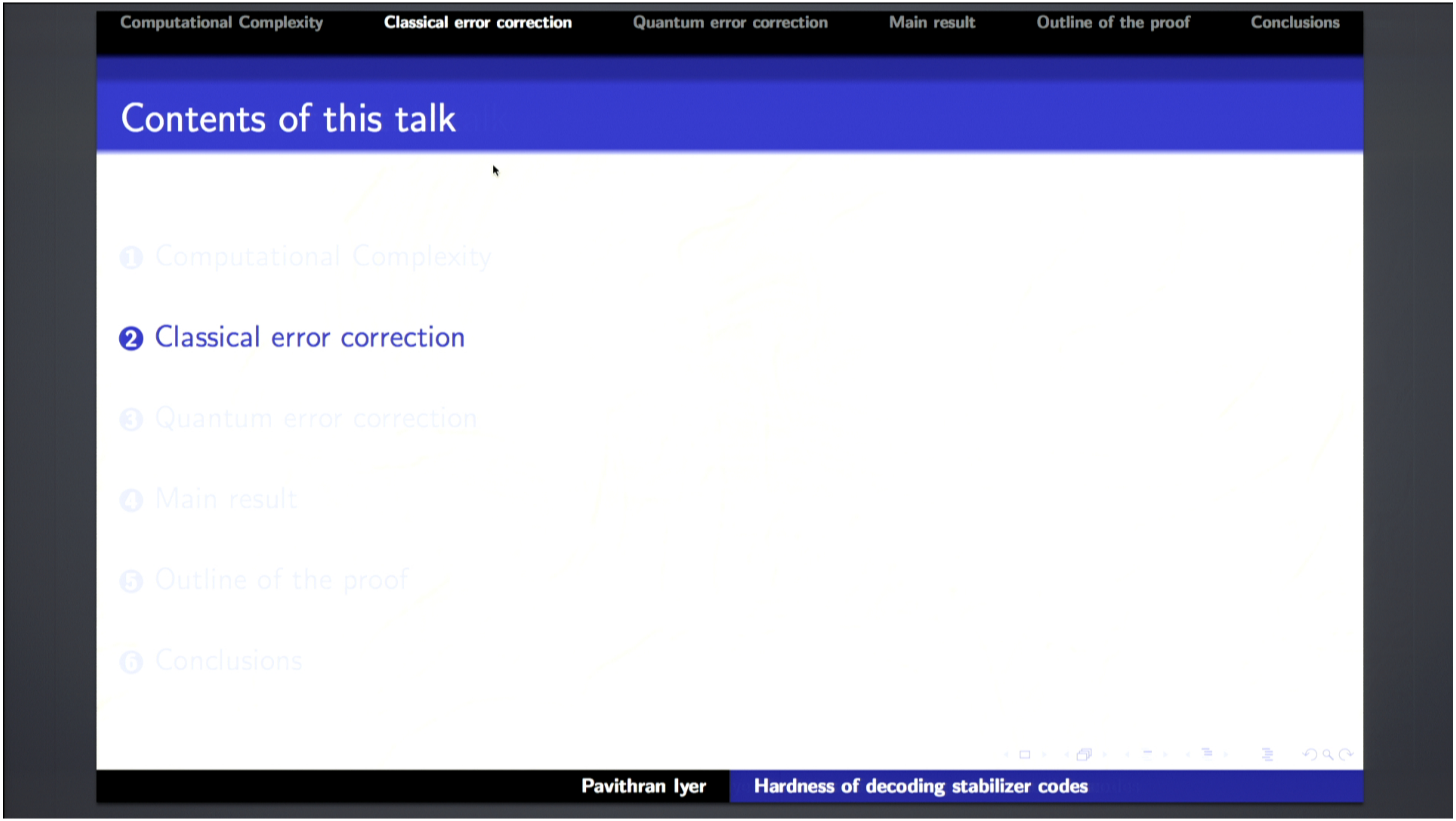
$A_\epsilon \;\rightarrow$ how many states have energy ϵ

We are now counting solutions to the previous NP problem $\hookrightarrow$ problem $\in$ #P

If we can solve this, we can solve many more hopelessly hard counting problems in computer science ! $\hookrightarrow \in$ #P-Complete

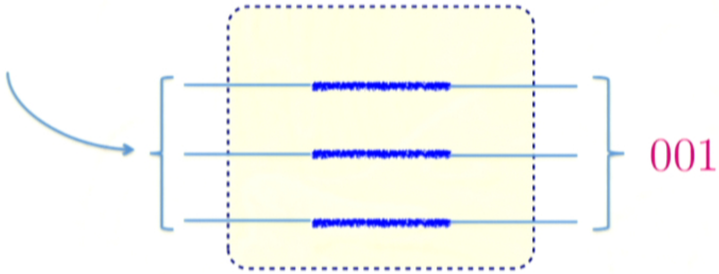
[Goldberg: SIAM J. Com, 39(7), 3336–3402]

It is strongly believed that #P-Complete problems cannot be solved in polynomial time.


Computational Complexity
Classical error correction
Quantum error correction
Main result
Outline of the proof
Conclusions
Contents of this talk
1 Computational Complexity
2 Classical error correction
3 Quantum error correction
4 Main result
5 Outline of the proof
6 Conclusions
Pavithran Iyer
Hardness of decoding stabilizer codes

Hard problems in classical error correction

Classical information is encoded and transmitted in bits $\rightarrow$ strings of 0's and 1's.

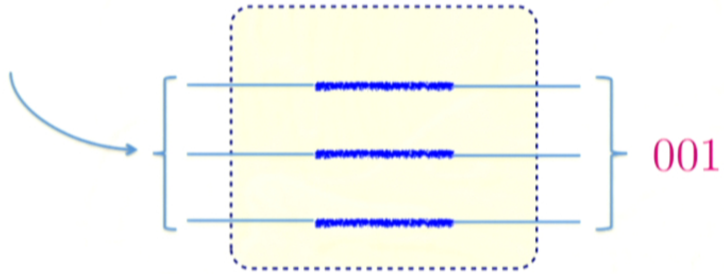


Consider a simple code: $\mathcal{C} = \{000, 111\}$.

If $\vec{r} = 001$ is received $\rightarrow$ some bit(s) were flipped. which ones ? $\leftrightarrow$ what was added ?

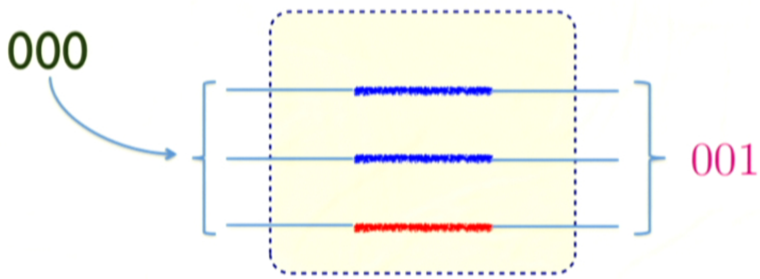
Hard problems in classical error correction

Classical information is encoded and transmitted in bits $\rightarrow$ strings of 0's and 1's.

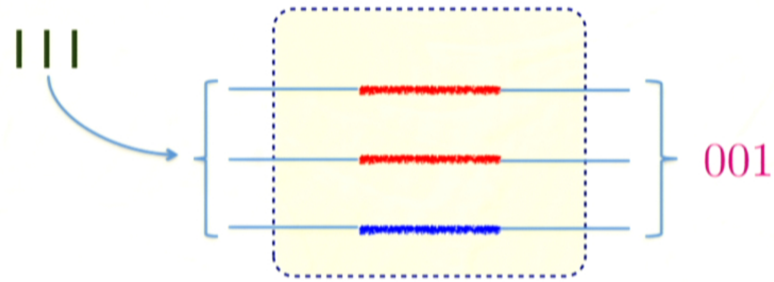


Consider a simple code: $\mathcal{C} = \{000, 111\}$.

If $\vec{r} = 001$ is received $\rightarrow$ some bit(s) were flipped. which ones ? $\leftrightarrow$ what was added ?

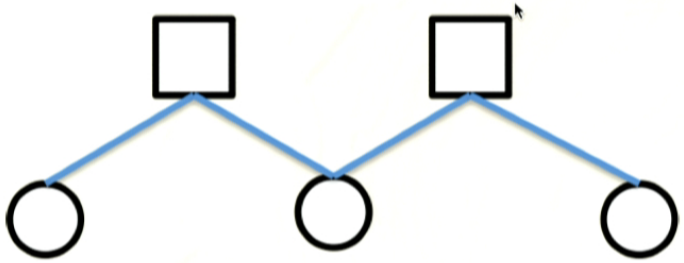


$\vec{e} = 001 \leftrightarrow$ Last bit flipped: $\Pr(\vec{e}) \sim p$



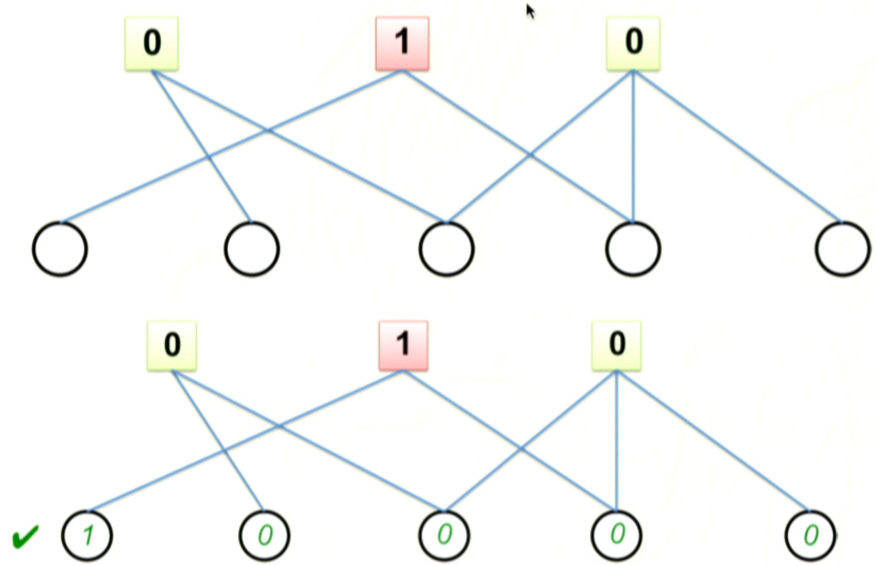
$\vec{e} = 110 \leftrightarrow$ first two bits flipped: $\Pr(\vec{e}) \sim p^2$

A short hand notation . . .



Take the same code: $\mathcal{C} = \{000, 111\}$.

Another example . . .



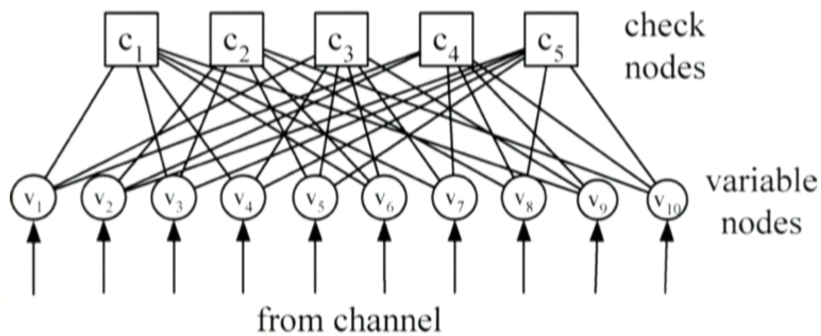
Consider a slightly complicated code:

$\vec{r}$ is received with $s = 010$. What is e ?

$\vec{e} = 10000 \leftrightarrow$ first bit was flipped

A real world example . . .

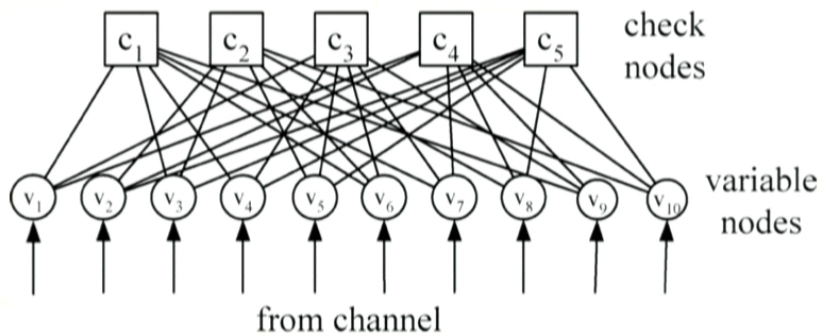
Consider a real-life code. Given the syndrome s, what is the error e ? (min bit flips for $\vec{s}$)



Too many (exponential) errors with the same syndrome s $\rightarrow$ a naive optimisation is hard

A real world example . . .

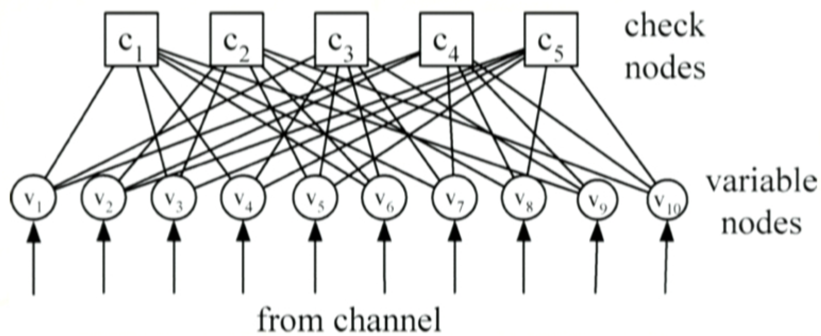
Consider a real-life code. Given the syndrome s, what is the error e ? (min bit flips for $\vec{s}$)



Too many (exponential) errors with the same syndrome s $\rightarrow$ a naive optimisation is hard

A real world example . . .

Consider a real-life code. Given the syndrome s, what is the error e ? (min bit flips for $\vec{s}$)



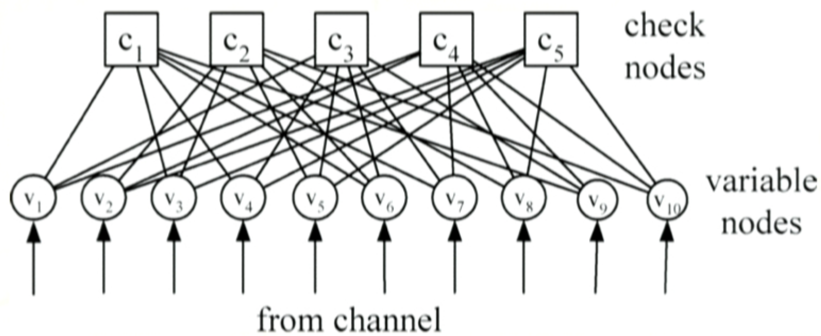
Too many (exponential) errors with the same syndrome s $\rightarrow$ a naive optimisation is hard

What are the problems of interest ?

1. Given a graph G and $\vec{s}$, determine $\vec{e}$ of lowest weight for $\vec{s}$. (NP-Complete)
2. Given a graph G, $\vec{s}$ and i, determine **how many** $\vec{e}$ of weight i for $\vec{s}$. (#P-Complete)

A real world example . . .

Consider a real-life code. Given the syndrome s, what is the error e ? (min bit flips for $\vec{s}$)



Too many (exponential) errors with the same syndrome s $\rightarrow$ a naive optimisation is hard

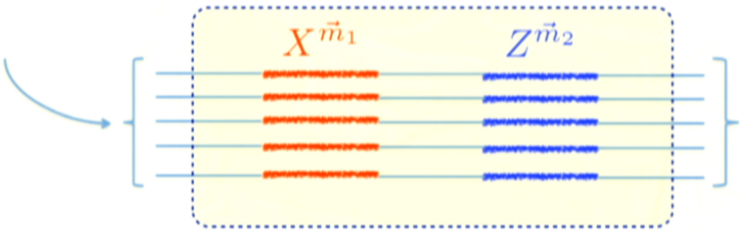
What are the problems of interest ?

1. Given a graph G and $\vec{s}$, determine $\vec{e}$ of lowest weight for $\vec{s}$. (NP-Complete)
2. Given a graph G, $\vec{s}$ and i, determine **how many** $\vec{e}$ of weight i for $\vec{s}$. (#P-Complete)

Decoding Stabilizer codes

Quantum information is encoded and transmitted in qubit states: $\alpha|0001\rangle + \beta|0101\rangle + \cdots$

Errors: independent bit flips X, phase flips Z on each qubit. (Independent $X - Z$ channel)



$$E = Z^{m_2} \cdot X^{m_1}$$

Independent X-Z channel:

$$\Pr(X) = \Pr(Z) = \frac{p}{2}\left(1 - \frac{p}{2}\right)$$

$$\Pr(E) = \left(\frac{p}{2}\right)^{|E|}\left(1 - \frac{p}{2}\right)^{2n-|E|}$$

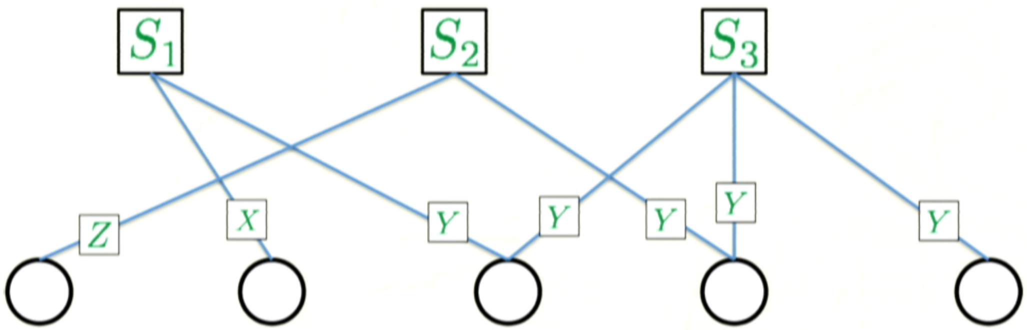
"weight" of E: $|E| = |\vec{m}_1| + |\vec{m}_2|$.

E: $|\vec{m}_1|$ Bit flips $X^{\vec{m}_1}$ then $|\vec{m}_2|$ phase flips $Z^{\vec{m}_2}$.

If $|\Phi\rangle$ is received, what was sent ? $\leftrightarrow$ what is E ?

A short hand notation: store properties, not codewords

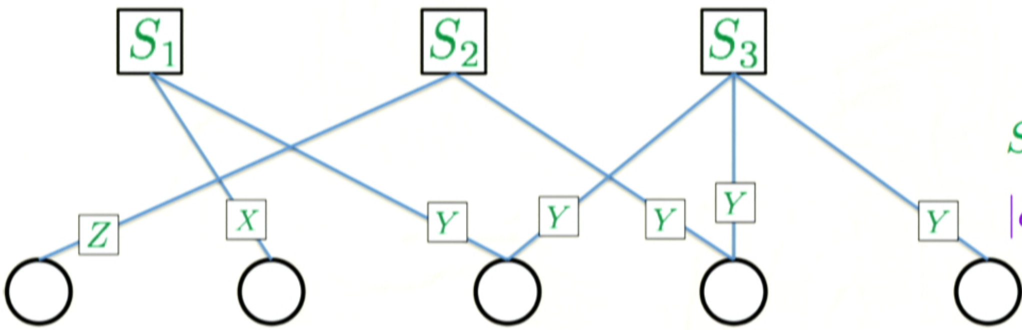
"Checks" are properties we can verify without disturbing the state $\rightarrow$ measurements



$$S_1 = \mathbb{I}XY\mathbb{I}\mathbb{I},\ S_2 = Z\mathbb{I}\mathbb{I}Y\mathbb{I},\ S_3 = \mathbb{I}\mathbb{I}YYY.$$

A short hand notation: store properties, not codewords

"Checks" are properties we can verify without disturbing the state $\rightarrow$ measurements



$|\psi\rangle$ is valid encoding:

$S_1|\psi\rangle = |\psi\rangle, \; S_2|\psi\rangle = |\psi\rangle, \; S_3|\psi\rangle = |\psi\rangle.$

$|\phi\rangle$ isn't a valid encoding: $(|\phi\rangle = E|\psi\rangle)$

$S_i|\phi\rangle = -|\phi\rangle$ (for some i).

$S_1 = \mathbb{I}XY\mathbb{I}\mathbb{I}, \; S_2 = Z\mathbb{I}\mathbb{I}Y\mathbb{I}, \; S_3 = \mathbb{I}\mathbb{I}YYY.$

s: a bit for each $\square$ $\rightarrow$ $\begin{cases} 0 & \text{if } E \cdot S_i = S_i \cdot E \quad \text{(measuring } S_i \text{ on } E|\psi\rangle \text{ results "+1")} \\ 1 & \text{if } E \cdot S_i = -S_i \cdot E \quad \text{(measuring } S_i \text{ on } E|\psi\rangle \text{ results "-1")} \end{cases}.$

Decoding Stabilizer codes

Problem of interest: *Degenerate Quantum Maximum likelihood decoding* (DQMLD)

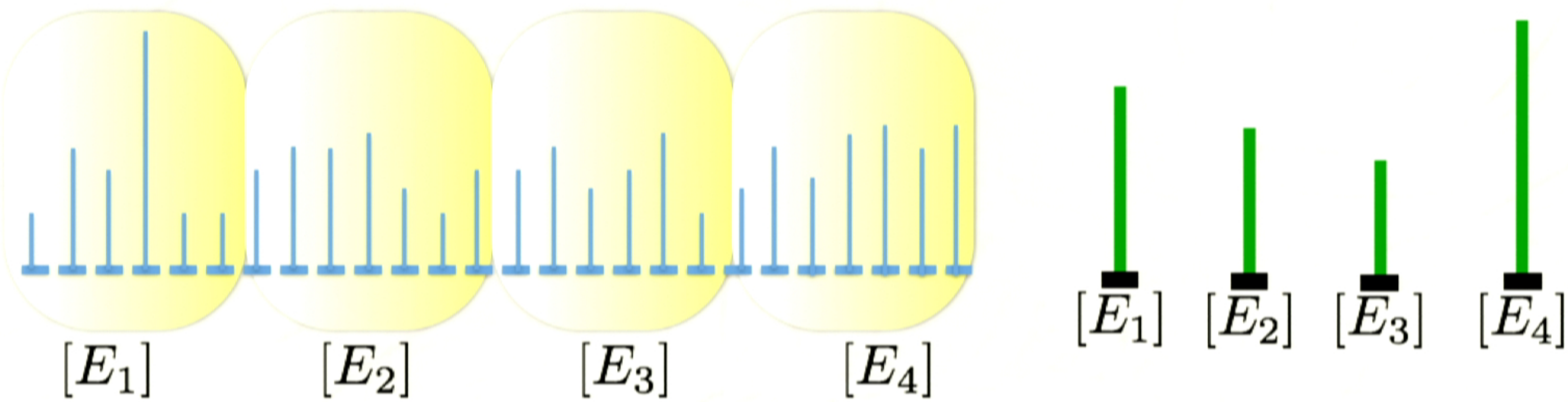
DQMLD: Given the graph and s find the class $[E]$ that has the maximum probability sum.

Decoding Stabilizer codes

Problem of interest: *Degenerate Quantum Maximum likelihood decoding* (DQMLD)

DQMLD: Given the graph and s find the class $[E]$ that has the maximum probability sum.

There are many errors for a syndrome s with different probabilities:



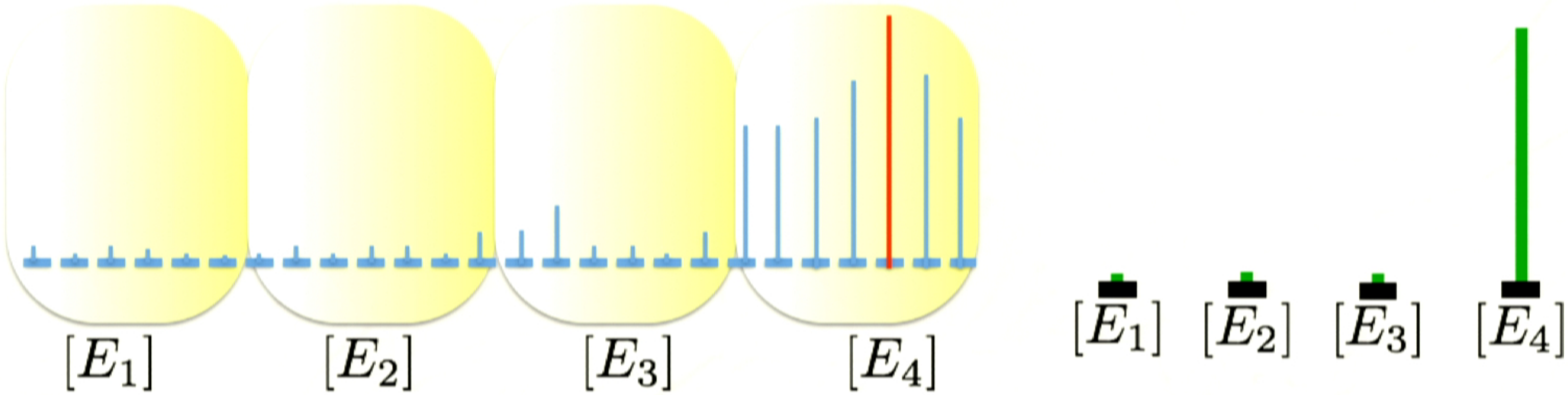
Quantum $\rightarrow$ Group into classes and then find the maximum $\rightarrow$ harder in the quantum case

Decoding Stabilizer codes

Problem of interest: *Degenerate Quantum Maximum likelihood decoding* (DQMLD)

DQMLD: Given the graph and s find the class $[E]$ that has the maximum probability sum.

There are many errors for a syndrome s with different probabilities:



Special case: Large "gap" (Δ) between maximum sum and others (Classical decoding)

Our main result

Decoding a quantum stabilizer code is #P-Complete. (Informal statement)

For a graph with n qubits ◯'s and $n-k$ checks □'s, . . .

Main result: Hardness of DQMLD

DQMLD on $[[n, k = 1]]$ stabilizer code on an independent $X - Z$ channel and with a promise gap $\Delta \leq 2[2 + n^{\lambda}]^{-1}$, with $\lambda = \Omega(\text{polylog}(n))$, is in #P-Complete.

Our main result

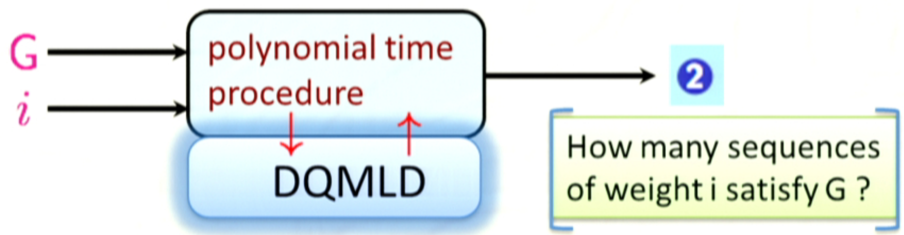
Decoding a quantum stabilizer code is #P-Complete. (Informal statement)

For a graph with n qubits $\bigcirc$'s and $n-k$ checks $\square$'s, ...

Main result: Hardness of DQMLD

DQMLD on $[[n, k=1]]$ stabilizer code on an independent $X-Z$ channel and with a promise gap $\Delta \leq 2[2+n^{\lambda}]^{-1}$, with $\lambda = \Omega(\text{polylog}(n))$, is in #P-Complete.

The proof outline:



Weight enumerator problem $\leq_p$ DQMLD proves DQMLD $\in$ #P-Complete.

Contents of this talk

Preparing to prove

Class of degenerate errors: $E,\ E \cdot S_1,\ E \cdot S_2,\ E \cdot S_3,\ E \cdot S_1S_2,\ E \cdot S_1S_3,\ E \cdot S_2S_3,\ \ldots$

Preparing to prove

Class of degenerate errors: $E,\ E\cdot S_1,\ E\cdot S_2,\ E\cdot S_3,\ E\cdot S_1S_2,\ E\cdot S_1S_3,\ E\cdot S_2S_3,\ \ldots$

Generally: m checks ($\square$'s): $S_1,\ldots,S_m$ produce 2^m degenerate errors in each class.

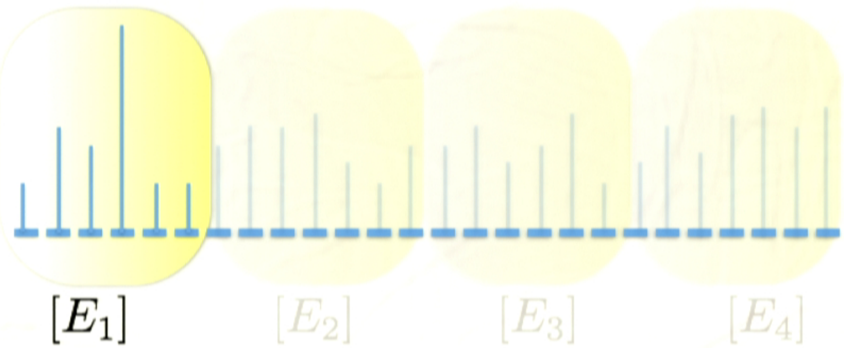
$$\Pr([E]) = \Pr(E) + \Pr(E\cdot S_1) + \Pr(E\cdot S_2) + \Pr(E\cdot S_3) + \cdots = \sum_{S\in <S_1,\cdots,S_m>} \Pr(E\cdot S)$$

Preparing to prove

Class of degenerate errors: $E,\ E\cdot S_1,\ E\cdot S_2,\ E\cdot S_3,\ E\cdot S_1S_2,\ E\cdot S_1S_3,\ E\cdot S_2S_3,\ \ldots$

Generally: m checks ($\square$'s): $S_1,\ldots,S_m$ produce 2^m degenerate errors in each class.

$\Pr([E]) = \Pr(E) + \Pr(E\cdot S_1) + \Pr(E\cdot S_2) + \Pr(E\cdot S_3) + \cdots = \sum_{S\in<S_1,\cdots,S_m>} \Pr(E\cdot S)$



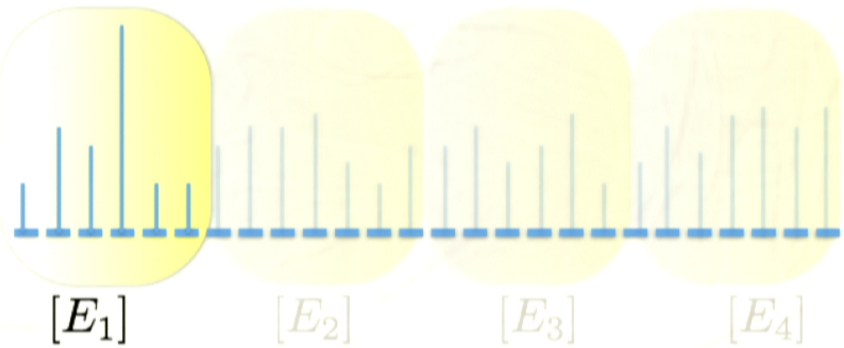
$\Pr(E)\sim p,\ \Pr(E\cdot S_1)\sim p^3,\ \Pr(E\cdot S_2)\sim p^3,\ldots$

Preparing to prove

Class of degenerate errors: $E,\ E\cdot S_1,\ E\cdot S_2,\ E\cdot S_3,\ E\cdot S_1S_2,\ E\cdot S_1S_3,\ E\cdot S_2S_3,\ \ldots$

Generally: m checks ($\square$'s): $S_1,\ldots,S_m$ produce 2^m degenerate errors in each class.

$$\Pr([E]) = \Pr(E) + \Pr(E\cdot S_1) + \Pr(E\cdot S_2) + \Pr(E\cdot S_3) + \cdots = \sum_{S\in<S_1,\cdots,S_m>} \Pr(E\cdot S)$$



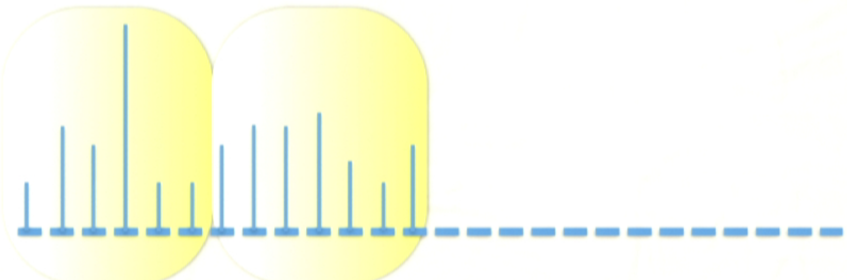
$\Pr(E)\sim p,\ \Pr(E\cdot S_1)\sim p^3,\ \Pr(E\cdot S_2)\sim p^3,\ldots$

In general: $\Pr(E\cdot S_i)\in\{p^0,p^1,\cdots,p^n\}$.

Many errors have equal probabilities $\rightarrow$ group them together

Outlining the technique

Suppose only two classes: Pr(each class) = degree n polynomial (unknown coefficients).



$[E_1]$ $[E_2]$

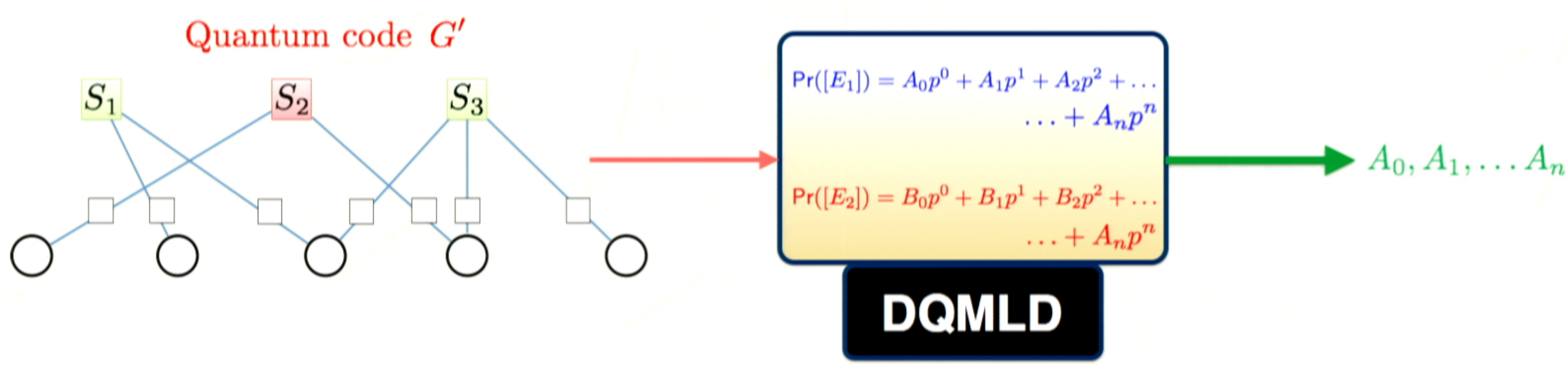
$$B_0 p^0 + B_1 p^1 + \cdots + B_n p^n$$

$$A_0 p^0 + A_1 p^1 + A_2 p^2 + \cdots + A_n p^n$$

Step 1/2: Extracting coefficients

Step 1: Extracting coefficients

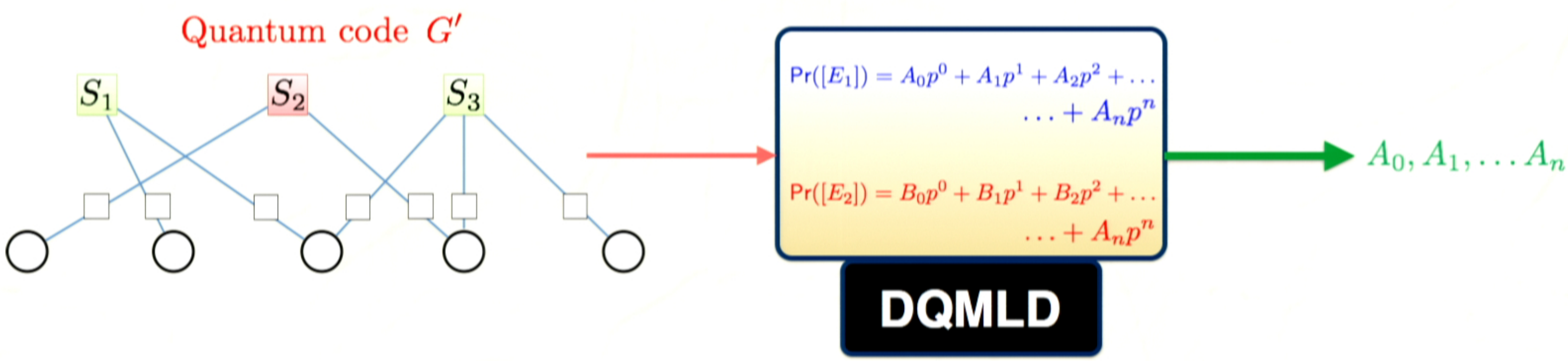
Given access to a decoder, if there are only two possible classes of errors, there is a polynomial time procedure to compute $A_0, \ldots, A_n$.



Step 1/2: Extracting coefficients

Step 1: Extracting coefficients

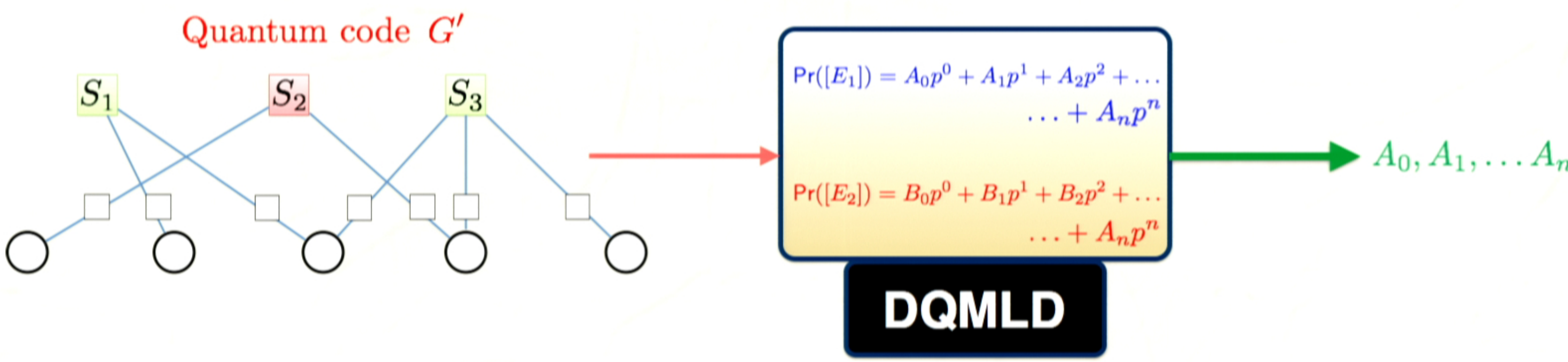
Given access to a decoder, if there are only two possible classes of errors, there is a polynomial time procedure to compute $A_0, \ldots, A_n$.



Step 1/2: Extracting coefficients

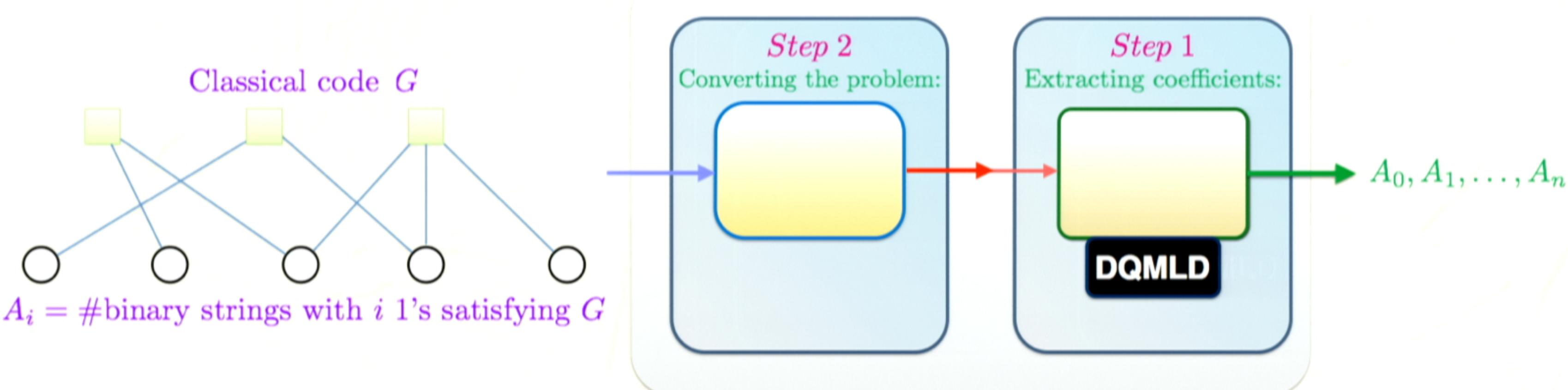
Step 1: Extracting coefficients

Given access to a decoder, if there are only two possible classes of errors, there is a polynomial time procedure to compute $A_0, \ldots, A_n$.



Proof of the main theorem

Recall the hard classical problem which we need to solve: (known #P-Complete)



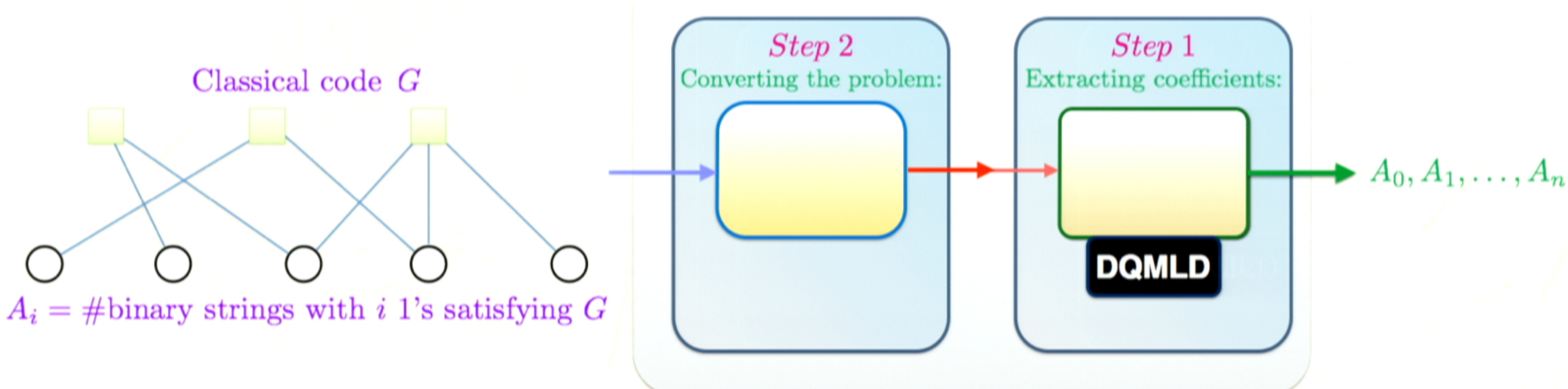
$A_i = \#$binary strings with i 1's satisfying G

Reduction statement [informal]

Given access to an oracle for solving DQMLD with a promise gap $\sim n^{-\lambda}$, it is possible to compute all λ coefficients $\{A_i\}_{i=0}^{\lambda}$, exactly, in polynomial time.

Proof of the main theorem

Recall the hard classical problem which we need to solve: (known #P-Complete)



Reduction statement [informal]

Given access to an oracle for solving DQMLD with a promise gap $\sim n^{-\lambda}$, it is possible to compute all λ coefficients $\{A_i\}_{i=0}^{\lambda}$, exactly, in polynomial time.

If $\lambda > \log_2 n$: A_λ is #P-Complete $\Leftrightarrow$ DQMLD with gap $\sim n^{-\lambda}$ is #P-Complete.

Input classical linear code, but decoder works on a stabilizer code . . .

Let $G_{\mathcal{C}} = (g_1 \; g_2 \; \ldots \; g_k) \hookrightarrow G_{\mathbb{Z}_2^n} = (g_1, g_2, \ldots, g_k, g_{k+1}, \ldots, g_n)$. Let $G^{-1}_{\mathbb{Z}_2^n} = (h_1, \ldots, h_n)$.

Input classical linear code, but decoder works on a stabilizer code . . .

Let $G_{\mathcal{C}} = (g_1\ g_2\ \ldots\ g_k) \hookrightarrow G_{\mathbb{Z}_2^n} = (g_1, g_2, \ldots, g_k, g_{k+1}, \ldots, g_n)$. Let $G^{-1}_{\mathbb{Z}_2^n} = (h_1, \ldots, h_n)$.

	Input		Step 1:
	g_1		Z^{g_1}
Idea:	g_2	$\rightsquigarrow$	Z^{g_2}
	$\vdots$		$\vdots$
	g_k		Z^{g_k}

Input classical linear code, but decoder works on a stabilizer code . . .

Let $G_{\mathcal{C}} = (g_1\ g_2\ \ldots\ g_k) \hookrightarrow G_{\mathbb{Z}_2^n} = (g_1, g_2, \ldots, g_k, g_{k+1}, \ldots, g_n)$. Let $G_{\mathbb{Z}_2^n}^{-1} = (h_1, \ldots, h_n)$.

	Input		Step 1:		Stabilizer generators:
	g_1		Z^{g_1}		$Z^{g_1}, \ldots, Z^{g_k}$
Idea:	g_2	$\rightsquigarrow$	Z^{g_2}	$\rightsquigarrow$	$Z^{g_{k+1}} Z_{n+1}, Z^{g_{k+2}} Z_{n+2}, \ldots, Z^{g_{n-1}} Z_{2n-k-1}$
	$\vdots$		$\vdots$		$X^{h_{k+1}} X_{n+1}, X^{h_{k+2}} X_{n+2}, \ldots, X^{h_{n-1}} X_{2n-k-1}$
	g_k		Z^{g_k}		$Z^{g_n} Z_{2n-k}, X^{h_n} X_{2n-k} X_{2n-k+1}$

We have a $[[2n-k+1, 1]]$ stabilizer code. Logical operators: $\langle Z^{g_n} Z_{2n-k+1}, X^{h_n} X_{2n-k} \rangle$.

ONLY errors: $[\mathbb{1}] = \langle Z^{g_1}, \ldots, Z^{g_k} \rangle$, $[\bar{Z}] = \bar{Z} \cdot [\mathbb{1}]$ iff qubits $n+1, \ldots, 2n-k$ are noiseless.

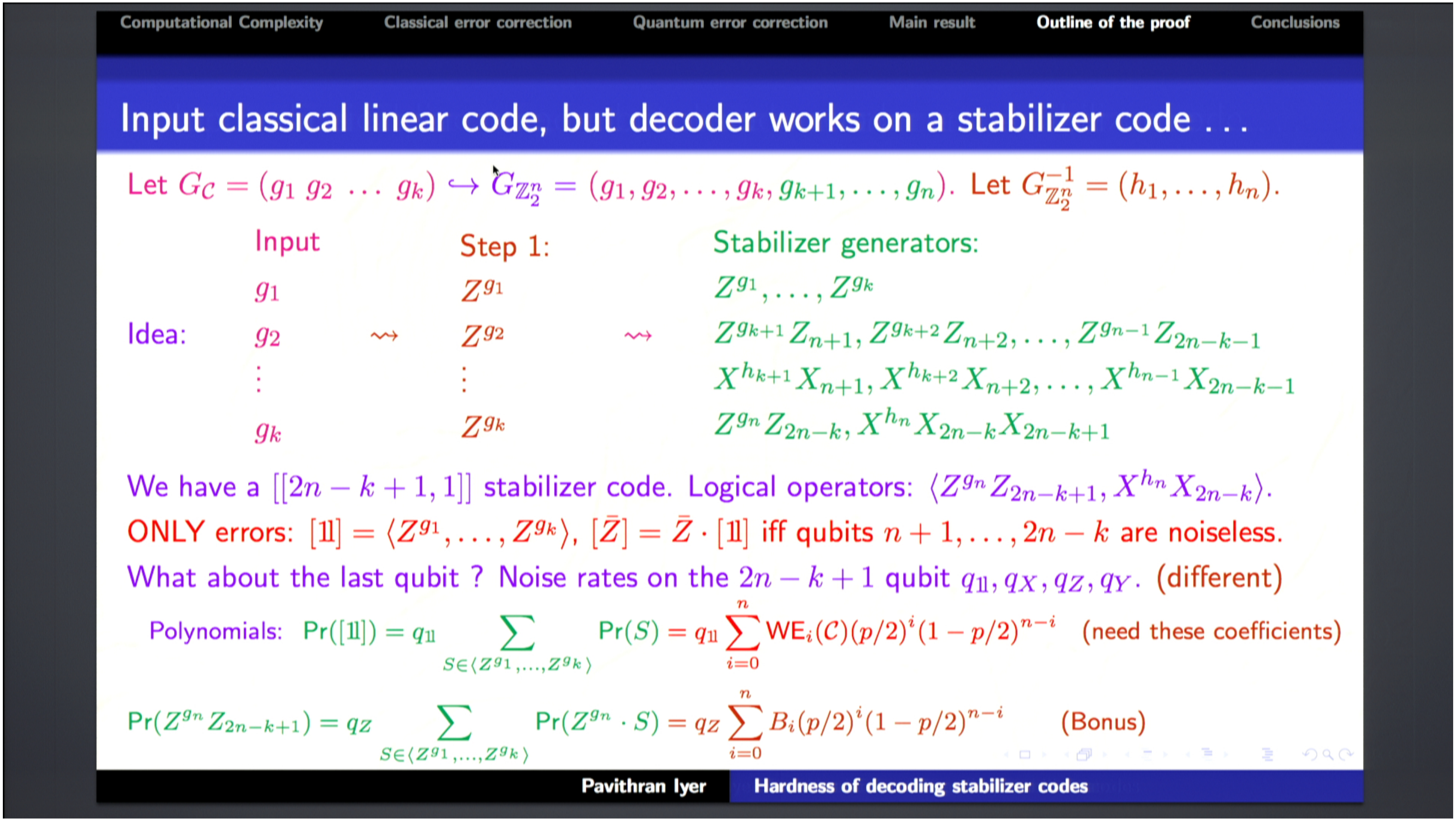
Input classical linear code, but decoder works on a stabilizer code . . .

Let $G_{\mathcal{C}} = (g_1\ g_2\ \dots\ g_k) \hookrightarrow G_{\mathbb{Z}_2^n} = (g_1, g_2, \dots, g_k, g_{k+1}, \dots, g_n)$. Let $G^{-1}_{\mathbb{Z}_2^n} = (h_1, \dots, h_n)$.

	Input		Step 1:		Stabilizer generators:
	g_1		Z^{g_1}		$Z^{g_1}, \dots, Z^{g_k}$
Idea:	g_2	$\rightsquigarrow$	Z^{g_2}	$\rightsquigarrow$	$Z^{g_{k+1}} Z_{n+1}, Z^{g_{k+2}} Z_{n+2}, \dots, Z^{g_{n-1}} Z_{2n-k-1}$
	$\vdots$		$\vdots$		$X^{h_{k+1}} X_{n+1}, X^{h_{k+2}} X_{n+2}, \dots, X^{h_{n-1}} X_{2n-k-1}$
	g_k		Z^{g_k}		$Z^{g_n} Z_{2n-k}, X^{h_n} X_{2n-k} X_{2n-k+1}$

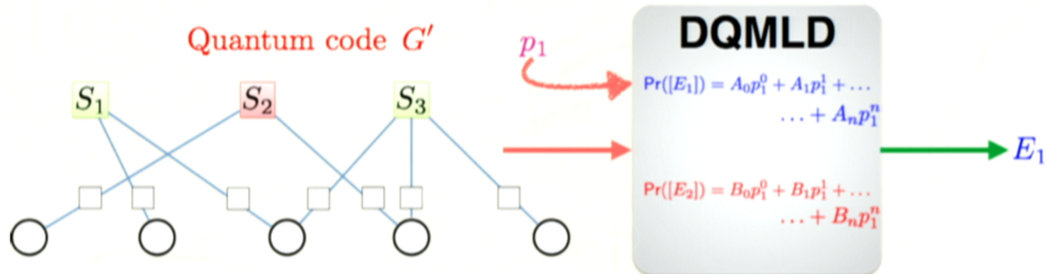
We have a $[[2n-k+1, 1]]$ stabilizer code. Logical operators: $\langle Z^{g_n} Z_{2n-k+1}, X^{h_n} X_{2n-k} \rangle$.

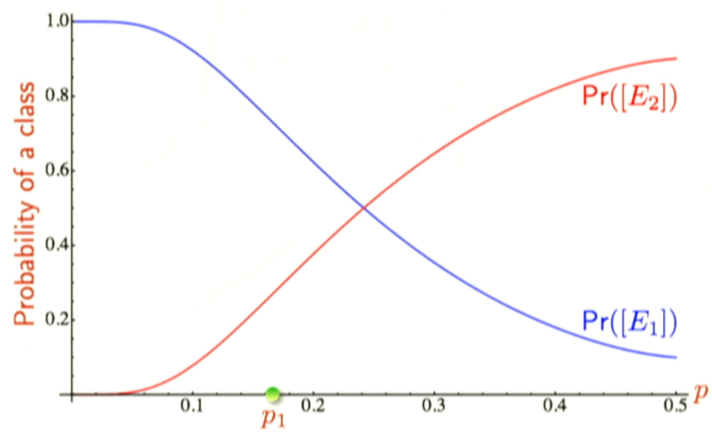
ONLY errors: $[\mathbb{1}] = \langle Z^{g_1}, \dots, Z^{g_k} \rangle$, $[\bar{Z}] = \bar{Z} \cdot [\mathbb{1}]$ iff qubits $n+1, \dots, 2n-k$ are noiseless.


Computational Complexity
Classical error correction
Quantum error correction
Main result
Outline of the proof
Conclusions
Input classical linear code, but decoder works on a stabilizer code . . .
Let $G_{\mathcal{C}} = (g_1\ g_2\ \ldots\ g_k) \hookrightarrow G_{\mathbb{Z}_2^n} = (g_1, g_2, \ldots, g_k, g_{k+1}, \ldots, g_n)$. Let $G^{-1}_{\mathbb{Z}_2^n} = (h_1, \ldots, h_n)$.
Input
Step 1:
Stabilizer generators:
g_1
Z^{g_1}
$Z^{g_1}, \ldots, Z^{g_k}$
Idea:
$g_2 \rightsquigarrow Z^{g_2} \rightsquigarrow$
$Z^{g_{k+1}} Z_{n+1}, Z^{g_{k+2}} Z_{n+2}, \ldots, Z^{g_{n-1}} Z_{2n-k-1}$
$X^{h_{k+1}} X_{n+1}, X^{h_{k+2}} X_{n+2}, \ldots, X^{h_{n-1}} X_{2n-k-1}$
g_k
Z^{g_k}
$Z^{g_n} Z_{2n-k}, X^{h_n} X_{2n-k} X_{2n-k+1}$
We have a $[[2n-k+1, 1]]$ stabilizer code. Logical operators: $\langle Z^{g_n} Z_{2n-k+1}, X^{h_n} X_{2n-k} \rangle$.
ONLY errors: $[\mathbb{1}] = \langle Z^{g_1}, \ldots, Z^{g_k} \rangle$, $[\bar{Z}] = \bar{Z} \cdot [\mathbb{1}]$ iff qubits $n+1, \ldots, 2n-k$ are noiseless.
What about the last qubit ? Noise rates on the $2n-k+1$ qubit $q_{\mathbb{1}}, q_X, q_Z, q_Y$. (different)
Polynomials: $\Pr([\mathbb{1}]) = q_{\mathbb{1}} \sum_{S \in \langle Z^{g_1}, \ldots, Z^{g_k} \rangle} \Pr(S) = q_{\mathbb{1}} \sum_{i=0}^{n} \mathrm{WE}_i(\mathcal{C}) (p/2)^i (1-p/2)^{n-i}$ (need these coefficients)
$\Pr(Z^{g_n} Z_{2n-k+1}) = q_Z \sum_{S \in \langle Z^{g_1}, \ldots, Z^{g_k} \rangle} \Pr(Z^{g_n} \cdot S) = q_Z \sum_{i=0}^{n} B_i (p/2)^i (1-p/2)^{n-i}$ (Bonus)
Pavithran Iyer
Hardness of decoding stabilizer codes

Extracting coefficients from the stabilizer code

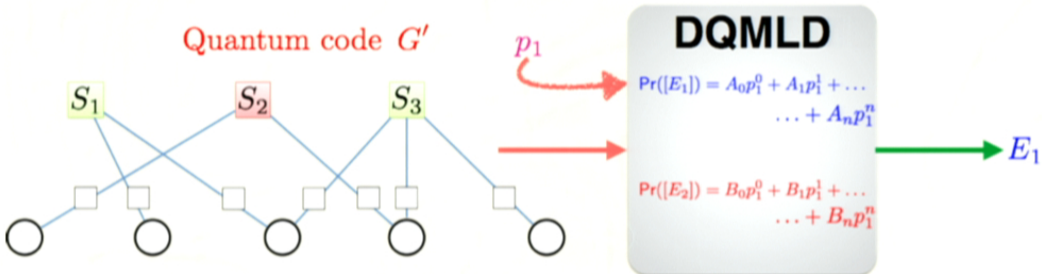
Decoder inputs can be tuned to switch between outputs:

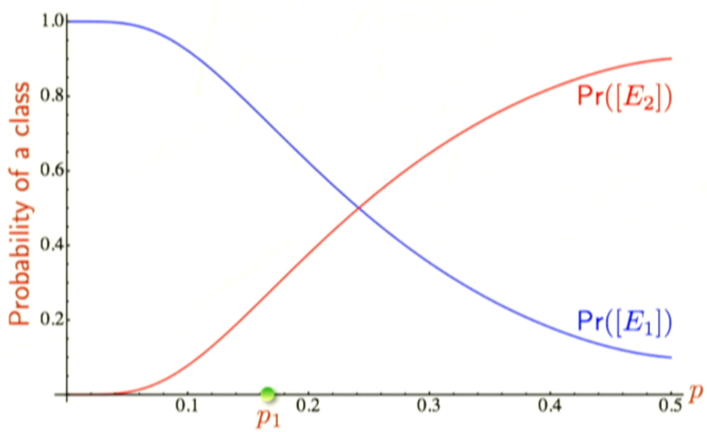




Extracting coefficients from the stabilizer code

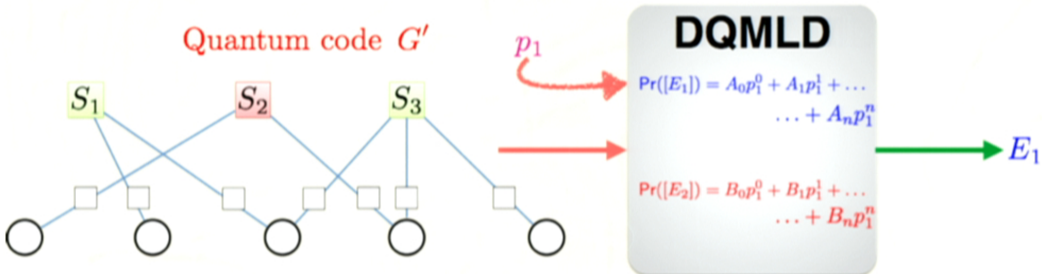
Decoder inputs can be tuned to switch between outputs:

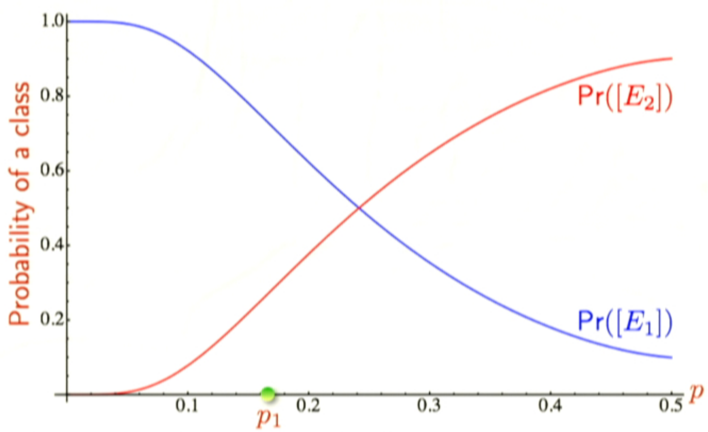




Extracting coefficients from the stabilizer code

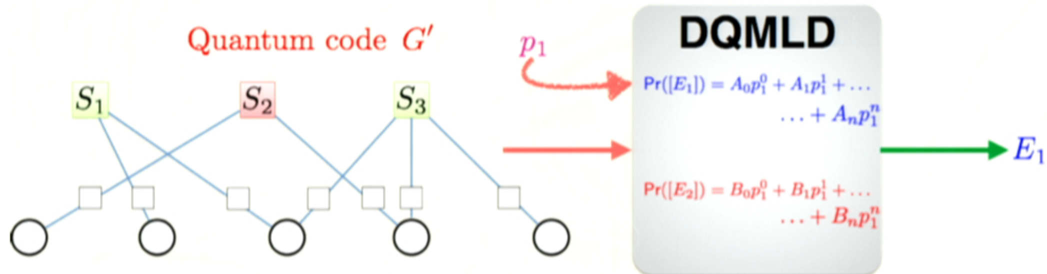
Decoder inputs can be tuned to switch between outputs:

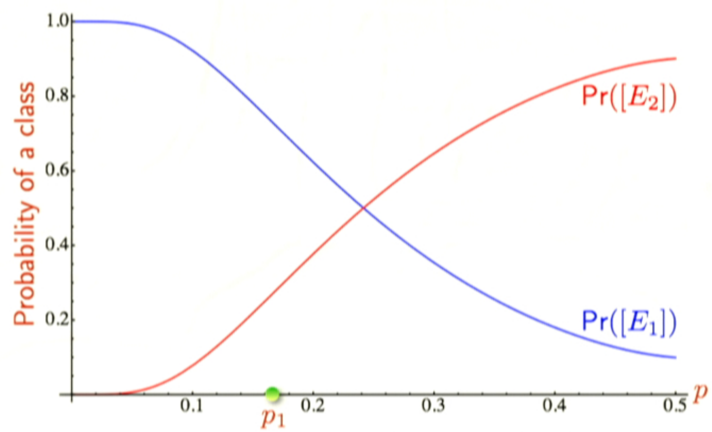




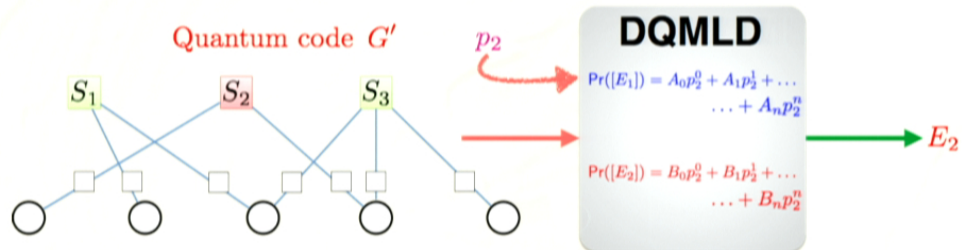
Extracting coefficients from the stabilizer code

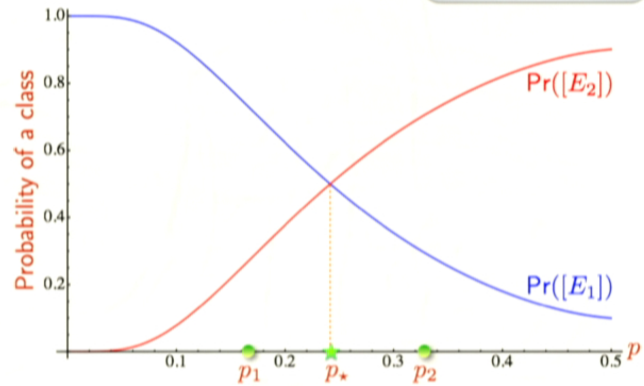
Decoder inputs can be tuned to switch between outputs:





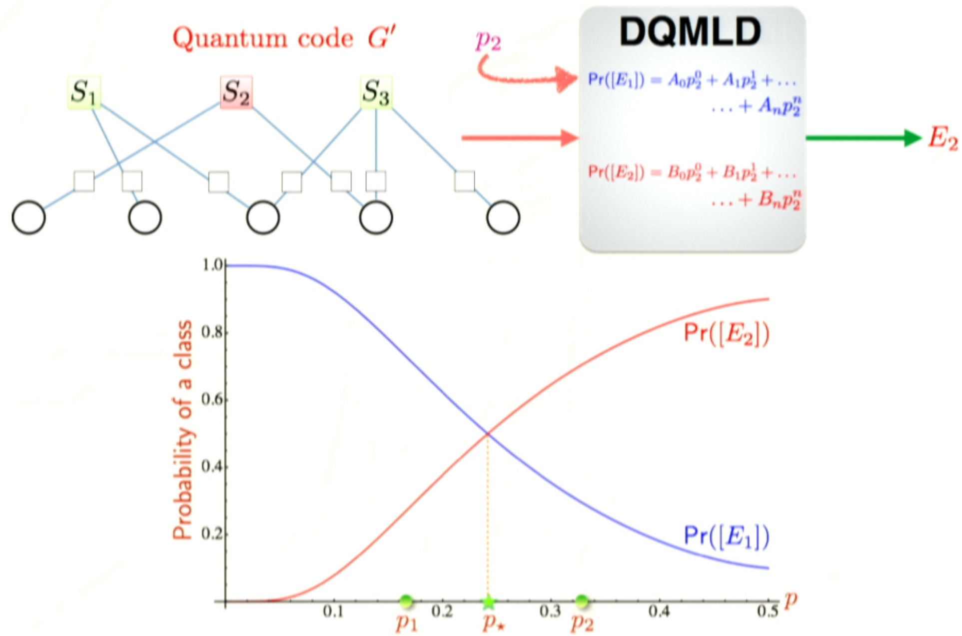
Extracting coefficients from the stabilizer code





Cross: $v \sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C})(p_\star/2)^i(1-p_\star/2)^{n-i} = \sum_{i=0}^{n} B_i (p_\star/2)^i(1-p_\star/2)^{n-i}$, $v = q_{\mathbb{1}}/q_Z$.

Extracting coefficients from the stabilizer code



Cross: $v \sum_{i=0}^{n} \mathrm{WE}_i(\mathcal{C})(p_\star/2)^i(1-p_\star/2)^{n-i} = \sum_{i=0}^{n} B_i(p_\star/2)^i(1-p_\star/2)^{n-i}$, $v = q_{\mathbb{1}}/q_Z$.

The last step – solving the constraints

$2n + 2$ constraints can be constructed in polynomial time:

$$\begin{bmatrix} (1-\Delta)v_1 & (1-\Delta)v_1\tilde{p}_1 & \dots & (1-\Delta)v_1\tilde{p}_1^n & -1 & -\tilde{p}_1 & \dots & -\tilde{p}_1^n \\ (1-\Delta)v_2 & (1-\Delta)v_2\tilde{p}_2 & \dots & (1-\Delta)v_2\tilde{p}_2^n & -1 & -\tilde{p}_2 & \dots & -\tilde{p}_2^n \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ (1-\Delta)v_{2n+1} & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1} & \dots & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1}^n & -1 & -\tilde{p}_{2n+1} & \dots & -\tilde{p}_{2n+1}^n \\ 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \cdot \begin{pmatrix} \mathsf{WE}_0(\mathcal{C}) \\ \mathsf{WE}_n(\mathcal{C}) \\ \vdots \\ B_0 \\ \vdots \\ B_n \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 2^n \end{pmatrix}$$

Can we assume them to be equalities ? Yes ! (Lemma. 6.2) Iff $\Delta \leq 1/\text{polylog}(n)$

Are these constraints all linearly independent ? Yes ! (Lemma. 6.3)

The last step – solving the constraints

$2n+2$ constraints can be constructed in polynomial time:

$$\begin{bmatrix} (1-\Delta)v_1 & (1-\Delta)v_1\tilde{p}_1 & \ldots & (1-\Delta)v_1\tilde{p}_1^n & -1 & -\tilde{p}_1 & \ldots & -\tilde{p}_1^n \\ (1-\Delta)v_2 & (1-\Delta)v_2\tilde{p}_2 & \ldots & (1-\Delta)v_2\tilde{p}_2^n & -1 & -\tilde{p}_2 & \ldots & -\tilde{p}_2^n \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ (1-\Delta)v_{2n+1} & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1} & \ldots & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1}^n & -1 & -\tilde{p}_{2n+1} & \ldots & -\tilde{p}_{2n+1}^n \\ 1 & 1 & \ldots & 1 & 1 & 1 & \ldots & 1 \end{bmatrix} \cdot \begin{pmatrix} \mathrm{WE}_0(\mathcal{C}) \\ \mathrm{WE}_n(\mathcal{C}) \\ \vdots \\ B_0 \\ \vdots \\ B_n \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 2^n \end{pmatrix}$$

Can we assume them to be equalities ? Yes ! (Lemma. 6.2) Iff $\Delta \leq 1/\mathrm{polylog}(n)$

Are these constraints all linearly independent ? Yes ! (Lemma. 6.3)

The last step – solving the constraints

$2n+2$ constraints can be constructed in polynomial time:

$$\begin{bmatrix} (1-\Delta)v_1 & (1-\Delta)v_1\tilde{p}_1 & \dots & (1-\Delta)v_1\tilde{p}_1^n & -1 & -\tilde{p}_1 & \dots & -\tilde{p}_1^n \\ (1-\Delta)v_2 & (1-\Delta)v_2\tilde{p}_2 & \dots & (1-\Delta)v_2\tilde{p}_2^n & -1 & -\tilde{p}_2 & \dots & -\tilde{p}_2^n \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ (1-\Delta)v_{2n+1} & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1} & \dots & (1-\Delta)v_{2n+1}\tilde{p}_{2n+1}^n & -1 & -\tilde{p}_{2n+1} & \dots & -\tilde{p}_{2n+1}^n \\ 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \cdot \begin{pmatrix} \mathsf{WE}_0(\mathcal{C}) \\ \mathsf{WE}_n(\mathcal{C}) \\ \vdots \\ B_0 \\ \vdots \\ B_n \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 2^n \end{pmatrix}$$

Can we assume them to be equalities ? Yes ! (Lemma. 6.2) Iff $\Delta \leq 1/\text{polylog}(n)$

Are these constraints all linearly independent ? Yes ! (Lemma. 6.3)