

Hamiltonian Decoded Quantum Interferometry

Yihui Quek



quekpottheories



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Why don't we have a working quantum computer?

On the theory side, there are at least two gaps:

Why don't we have a working quantum computer?

On the theory side, there are at least two gaps:

Lack of useful quantum algorithms

Two large algorithmic paradigms:

- Hidden subgroup-type (Shor's etc.)
- Amplitude amplification-type (Hamiltonian simulation etc.)

We know very few applications, and almost none outside physics/chemistry!

Why don't we have a working quantum computer?

On the theory side, there are at least two gaps:

Lack of useful quantum algorithms

Two large algorithmic paradigms:

- Hidden subgroup-type (Shor's etc.)
- Amplitude amplification-type (Hamiltonian simulation etc.)

We know very few applications, and almost none outside physics/chemistry!

Lack of practical fault-tolerant schemes

- Which error-correcting codes to use
- How to decode efficiently
- How to handle realistic noise (non-Markovian/biased etc)
- Reducing enormous space/time overhead for fault-tolerance

Why don't we have a working quantum computer?

On the theory side, there are at least two gaps:

Lack of useful quantum algorithms

Two large algorithmic paradigms:

- Hidden subgroup-type (Shor's etc.)
- Amplitude amplification-type (Hamiltonian simulation etc.)

We know very few applications, and almost none outside physics/chemistry!

Today:

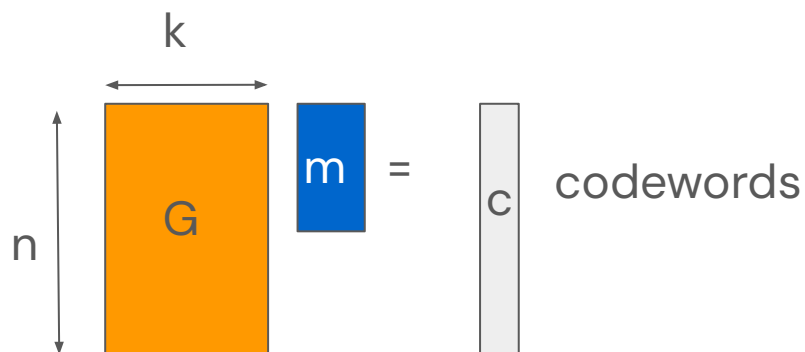


Lack of practical fault-tolerant schemes

- Which error-correcting codes to use
- **How to decode efficiently**
- How to handle realistic noise (non-Markovian/biased etc)
- Reducing enormous space/time overhead for fault-tolerance

Classical error correcting codes

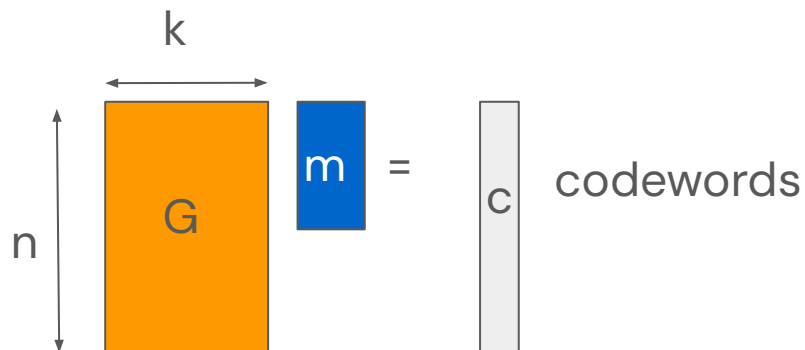
$$\mathcal{C} = \{Gm : m \in \mathbb{F}_2^k\}$$



G = Generator matrix, fixed for the code

Classical error correcting codes

$$\mathcal{C} = \{Gm : m \in \mathbb{F}_2^k\}$$

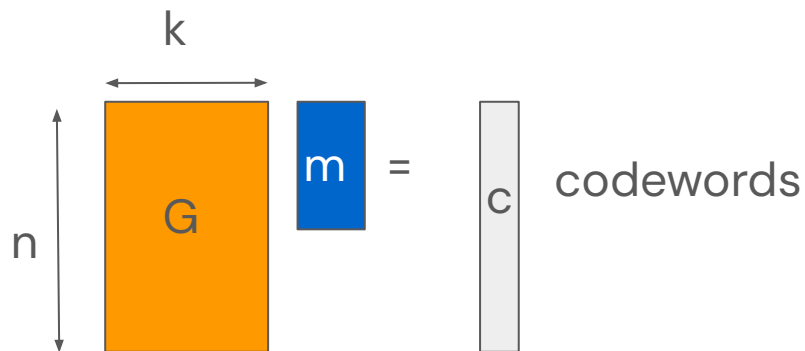


G = Generator matrix, fixed for the code

Given c , you could always recover m **efficiently**
→ overconstrained system of linear equations.

Classical error correcting codes

$$\mathcal{C} = \{Gm : m \in \mathbb{F}_2^k\}$$



G = Generator matrix, fixed for the code

Decoding:

Input: $Gm + e$

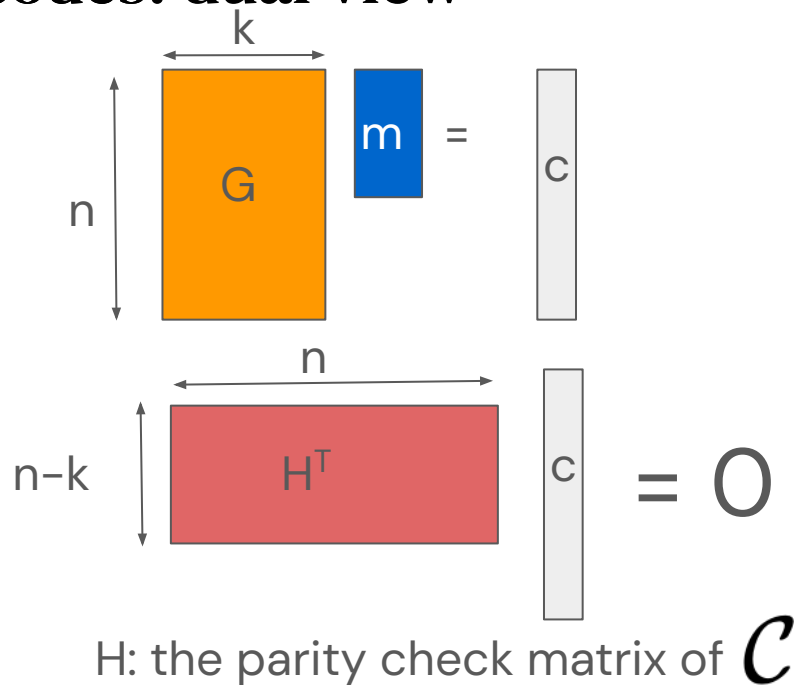
Output: m

Random error vector $e \in \mathbb{F}_2^n$
Typically assumed
low-weight (few 1's).

Challenging and
in general not
efficient!

Classical error correcting codes: dual view

$$\begin{aligned}\mathcal{C} &= \{Gm : m \in \mathbb{F}_2^k\} \\ &= \{c : H^T c = 0\}\end{aligned}$$

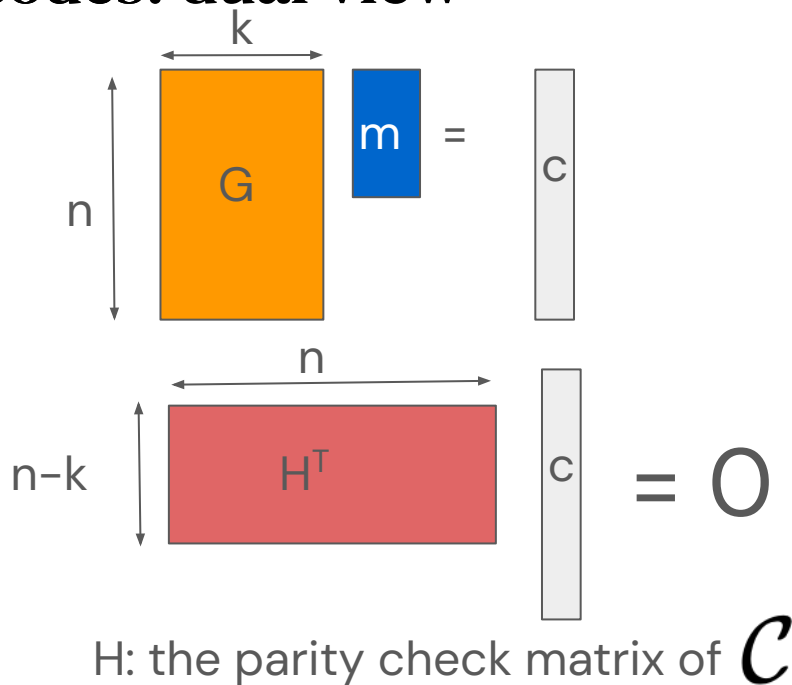


Classical error correcting codes: dual view

$$\begin{aligned}\mathcal{C} &= \{Gm : m \in \mathbb{F}_2^k\} \\ &= \{c : H^T c = 0\}\end{aligned}$$

This relation holds because

$$H^T G = 0$$



An equivalent view of decoding

Decoding:

Input: $Gm + e$

Output: m

(Previous definition)

An equivalent view of decoding

Decoding:

Input: $Gm + e$

Output: m

(Previous definition)

Given that $H^T G = 0 \Rightarrow H^T (Gm + e) = H^T e$

An equivalent view of decoding

Decoding:

Input: $Gm + e$

Output: m

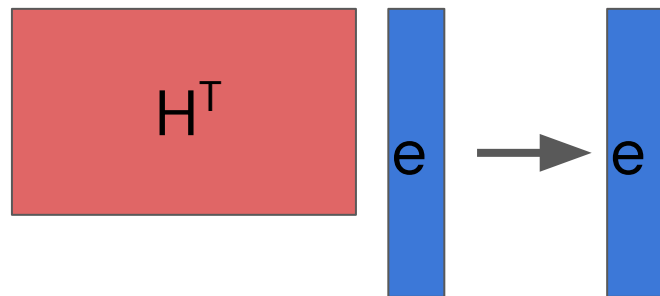
(Previous definition)

Given that $H^T G = 0 \Rightarrow H^T (Gm + e) = H^T e$

Syndrome decoding:

Input: $H^T e$

Output: e



We'll be using this view a lot!

$e \in \mathbb{F}_2^n$ assumed low-weight error.

Codes: a shared backbone of physics, CS and learning

In physics:

Article | [Open access](#) | Published: 09 December 2024

Quantum error correction below the surface code threshold

[Google Quantum AI and Collaborators](#)

[Nature](#) **638**, 920–926 (2025) | [Cite this article](#)

[nature](#) > [articles](#) > [article](#)

Article | [Open access](#) | Published: 26 February 2025

Hardware-efficient quantum error correction via concatenated bosonic qubits

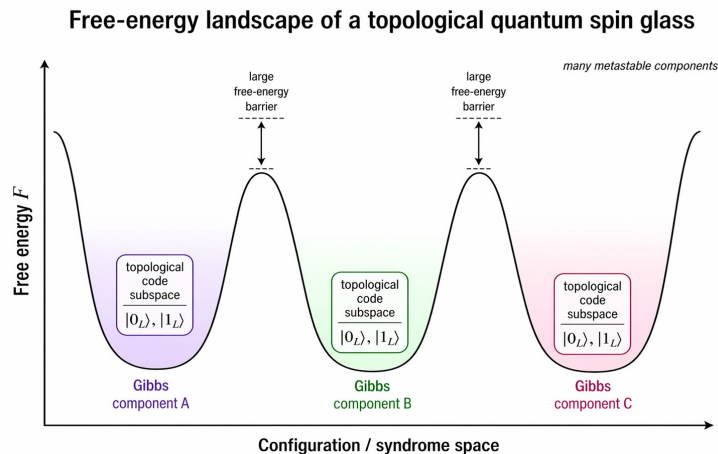
[Harald Putterman](#) , [Kyungjoo Noh](#), [Connor T. Hann](#), [Gregory S. MacCabe](#), [Shahriar Aghaeimeibodi](#), [Rishi N. Patel](#), [Menyoung Lee](#), [William M. Jones](#), [Hesam Moradinejad](#), [Roberto Rodriguez](#), [Neha Mahuli](#), [Jefferson Rose](#), [John Clai Owens](#), [Harry Levine](#), [Emma Rosenfeld](#), [Philip Reinhold](#), [Lorenzo Monceli](#), [Joshua Ari Alcid](#), [Nasser Alidoust](#), [Patricio Arrangoiz-Arriola](#), [James Barnett](#), [Przemyslaw Bienias](#), [Hugh A. Carson](#), [Cliff Chen](#), ... [Oskar Painter](#)  [+ Show authors](#)

[Nature](#) **638**, 927–934 (2025) | [Cite this article](#)

Codes: a shared backbone of physics, CS and learning

In physics:

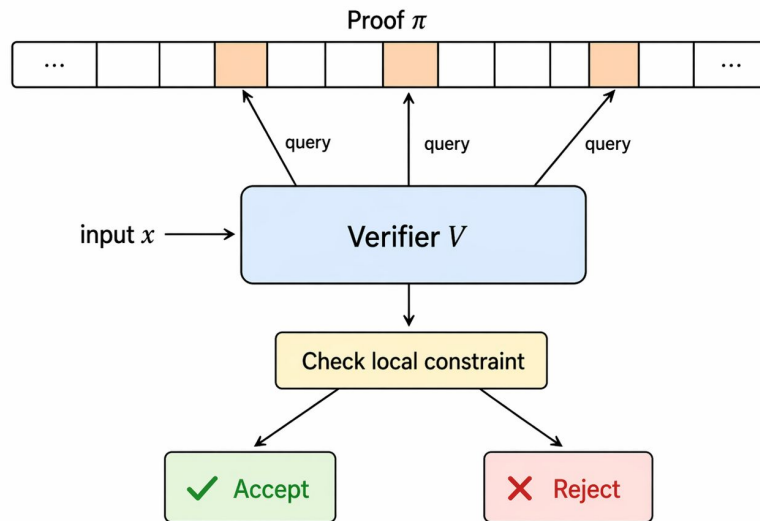
- Certain qLDPC expander code Hamiltonians exhibit topological quantum spin glass order at low temperature [PRBK24] → candidate for quantum memory.



Codes: a shared backbone of physics, CS and learning

In CS:

- Probabilistically checkable proofs: a proof of some NP problem, that can be verified only by looking at $O(1)$ bits of it.
- Can be constructed using ‘locally-testable codes’.

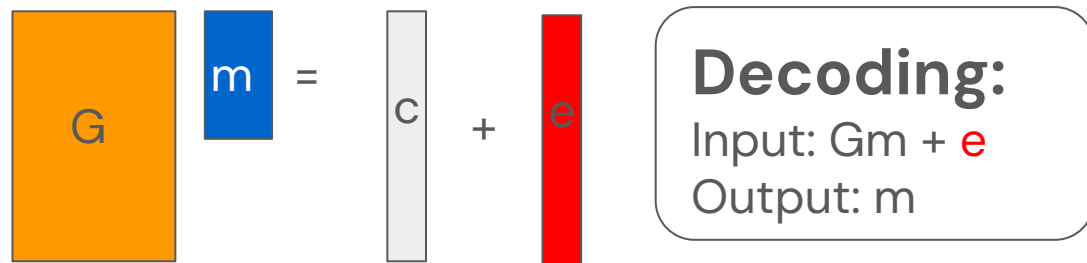


Everything on this slide is in $\mathbb{F}_2$ (binary)

Codes: a shared backbone of physics, CS and learning

In learning theory:

- Decoding a random linear code = a (very famous) learning problem.

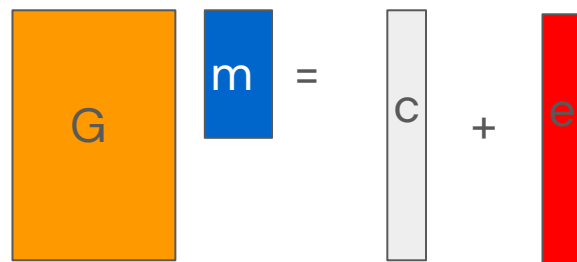


Everything on this slide is in $\mathbb{F}_2$ (binary)

Codes: a shared backbone of physics, CS and learning

In learning theory:

- Decoding a random linear code = a (very famous) learning problem.



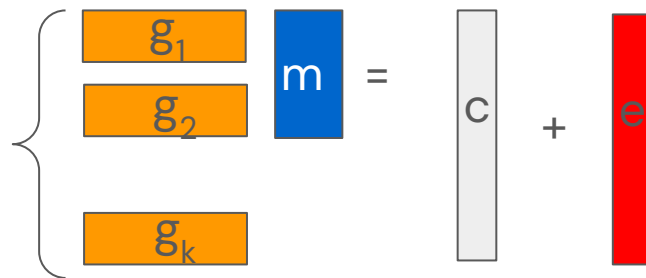
Decoding:

Input: $Gm + e$

Output: m

Equivalently:

k = sample complexity

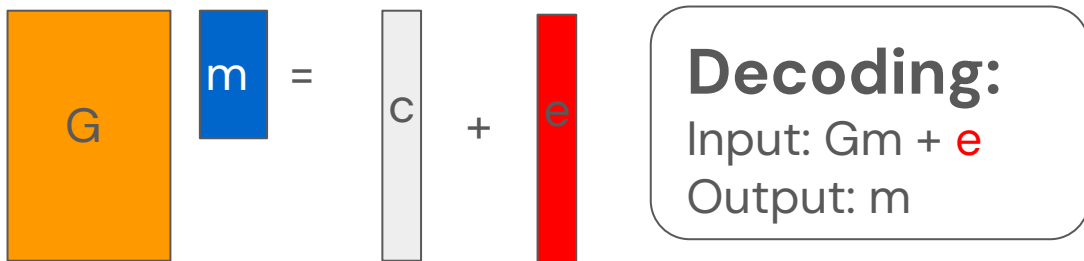


Learning Parities with Noise

Input: $\langle g_1, m \rangle + e_1 \dots \langle g_k, m \rangle + e_k$

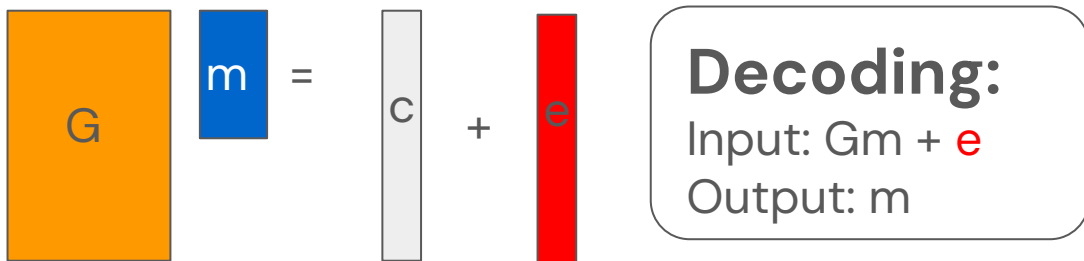
Output: m

Decoding is hard, unless there is structure.



- If G is a *random* linear code: this problem is called LPN, and is in NP.
- Hard enough that we have built cryptography out of its hardness.

Decoding is hard, unless there is structure.



If G is a *random* linear code: this problem is called LPN, and is in NP.

- Hard enough that we have built cryptography out of its hardness.

However, we typically work with structured G that have been specially built so that decoding is efficient.

- E.g. Reed-Solomon or LDPC codes.

Today: the decoders-to-algorithms paradigm

Decoders can be used to construct quantum algorithms, too!

Chapter 1: From cryptography to decoding

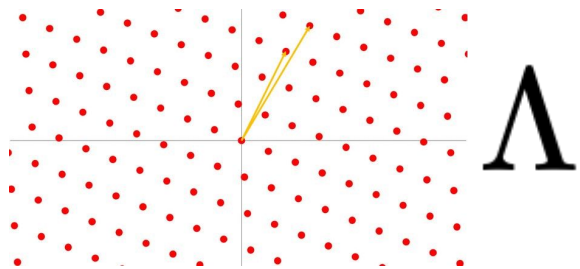
The birth of a beautiful idea

An intriguing observation by Regev (2009)

Problem 1: Short Codeword Problem

Given a code $C \subseteq \mathbb{F}_q^n$, find a codeword $x \in C$ such that

$$\|x\|_2 \leq \gamma\sqrt{n}$$

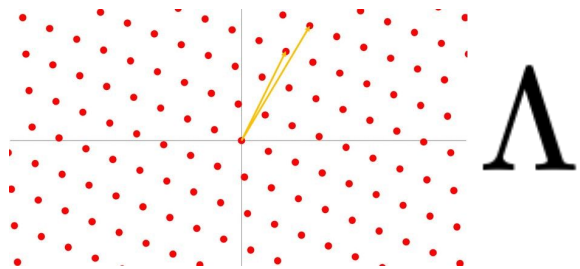


An intriguing observation by Regev (2009)

Problem 1: Short Codeword Problem

Given a code $C \subseteq \mathbb{F}_q^n$, find a codeword $x \in C$ such that

$$\|x\|_2 \leq \gamma\sqrt{n}$$



Problem 2: Decoding a random q-ary code

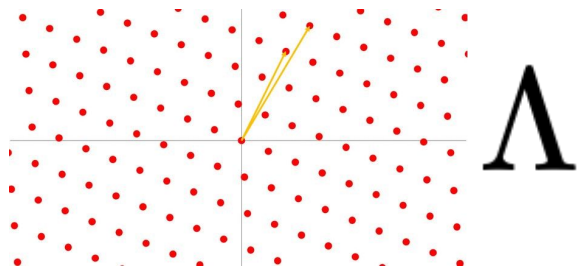
Given a random code $C \subseteq \mathbb{F}_q^n$ and a codeword from C corrupted by Gaussian noise, **decode**.

An intriguing observation by Regev (2009)

Problem 1: Short Codeword Problem

Given a code $C \subseteq \mathbb{F}_q^n$, find a codeword $x \in C$ such that

$$\|x\|_2 \leq \gamma\sqrt{n}$$



quantum-reduces to

Problem 2: Decoding a random q-ary code

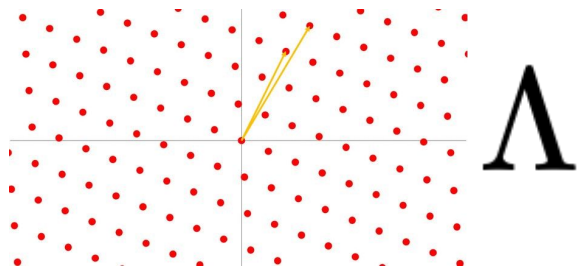
Given a random code $C \subseteq \mathbb{F}_q^n$ and a codeword from C corrupted by Gaussian noise, **decode**.

An intriguing observation by Regev (2009)

Problem 1: Short Codeword Problem

Given a code $C \subseteq \mathbb{F}_q^n$, find a codeword $x \in C$ such that

$$\|x\|_2 \leq \gamma\sqrt{n}$$



Key idea: this problem can be solved if we can prepare the target state

$$\sum_{x \in \Lambda} e^{-\|x\|_2^2 / \sigma^2} |x\rangle$$

Length of solution

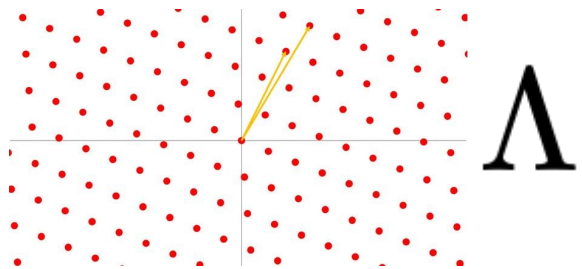
Why? Because if we measure this state, we sample a short solution w.h.p.

An intriguing observation by Regev (2009)

Problem 1: Short Codeword Problem

Given a code $C \subseteq \mathbb{F}_q^n$, find a codeword $x \in C$ such that

$$\|x\|_2 \leq \gamma\sqrt{n}$$



Key idea: prepare a state which
“multiplies quantum states”

x is a valid codeword

Short vectors have
higher weights

$$\begin{aligned} & \left(\sum_{x \in \Lambda} |x\rangle \right) \times \left(\sum_x e^{-\|x\|^2/\sigma^2} |x\rangle \right) \\ &= \sum_{x \in \Lambda} e^{-\|x\|^2/\sigma^2} |x\rangle \end{aligned}$$

An intriguing observation by Regev (2009)

Regev's big idea:
Such states can be prepared if we can run a **decoder for a random q-ary code** in quantum superposition.

Key idea: prepare a state which
“multiplies quantum states”

x is a valid codeword

Short vectors have
higher weights

$$\sum_{x \in \Lambda} |x\rangle \times \sum_x e^{-\|x\|^2/\sigma^2} |x\rangle = \sum_{x \in \Lambda} e^{-\|x\|^2/\sigma^2} |x\rangle$$

An intriguing observation by Regev (2009)

Regev's big idea:

Such states can be prepared if we can run a **decoder for a random q -ary code** in quantum superposition.

More famously known as:

Learning with Errors

- Believed to be hard, both classically and quantumly.
- One of the few post-quantum cryptographic assumptions we know!

An intriguing observation by Regev (2009)

Regev's big idea:

Such states can be prepared if we can run a **decoder for a random q-ary code** in quantum superposition.

More famously known as:

Learning with Errors

- Believed to be hard, both classically and quantumly.
- One of the few post-quantum cryptographic assumptions we know!

Cryptographic insight: SCP quantum-reduces to LWE.

A decade later, Regev's reduction would find its way outside of cryptography and into quantum algorithms, too.

Chapter 2: The revival of the idea

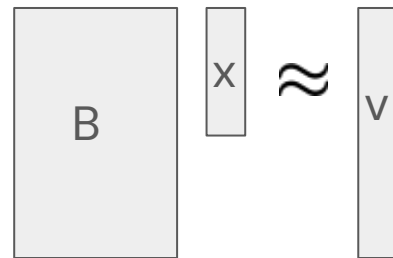
Decoded Quantum Interferometry, Jordan et al. Nature 2025

Problem setting: Combinatorial optimization

E.g. Max-XORSAT(B, v)

- Input: B, v . $B \in \mathbb{F}_2^{m \times n}$, $v \in \mathbb{F}_2^m$

We are given an **inconsistent** system of m linear equations: some of the bits in v may have been **flipped**.



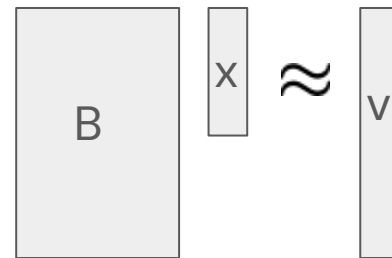
Problem setting: Combinatorial optimization

E.g. Max-XORSAT(B, v)

- Input: B, v . $B \in \mathbb{F}_2^{m \times n}$, $v \in \mathbb{F}_2^m$

We are given an **inconsistent** system of m linear equations: some of the bits in v may have been **flipped**.

- Output: find $x \in \mathbb{F}_2^n$ satisfying as many of the m linear equations as possible.



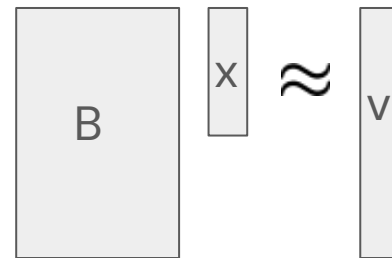
Problem setting: Combinatorial optimization

E.g. Max-XORSAT(B, v)

- Input: B, v . $B \in \mathbb{F}_2^{m \times n}, v \in \mathbb{F}_2^m$

We are given an **inconsistent** system of m linear equations: some of the bits in v may have been **flipped**.

- Output: find $x \in \mathbb{F}_2^n$ satisfying as many of the m linear equations as possible.



Equivalent: **find x maximizing the objective function f**

$$\max_{x \in \mathbb{F}_2^n} \text{SAT}(x) - \text{UNSAT}(x) \\ =: f(x)$$

To avoid clutter, I will write states without normalizing them properly henceforth.

Key Idea: prepare

$$|f\rangle = \sum_{x \in \mathbb{F}_2^n} f(x) |x\rangle$$

**f = objective function
for MaxXORSAT.**

Key Idea: prepare

$$|f\rangle = \sum_{x \in \mathbb{F}_2^n} f(x) |x\rangle \quad \mathbf{f = objective\ function\ for\ MaxXORSAT.}$$

Upon measurement: obtain solution x with probability

$$f(x)^2$$

→ optimal x is output with *highest* probability.

Key Idea (refined): prepare

$$|f^\ell\rangle = \sum_{x \in \mathbb{F}_2^n} f^\ell(x) |x\rangle \quad \mathbf{f = objective\ function\ for\ MaxXORSAT.}$$

Upon measurement: obtain solution x with probability

$$f^\ell(x)^2$$

→ optimal x is output with *highest* probability, **much higher than all other x .**

Refined approach: prepare state $|f^\ell\rangle$

To heighten the peak at optimal x , can prepare instead

$$|f^\ell\rangle = \sum_{x \in \mathbb{F}_2^n} f^\ell(x) |x\rangle$$

It suffices to prepare the Quantum Fourier Transform

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^T \mathbf{e}\rangle$$

Target state

then FT back.

Wait! We've seen this idea in a different guise.

Regev prepares:

Valid codewords

Short vectors have higher weights =
objective function values

$$\sum_{x \in \Lambda} |x\rangle \times \sum_x e^{-||x||^2/\sigma^2} |x\rangle = \sum_{x \in \Lambda} e^{-||x||^2/\sigma^2} |x\rangle$$

Wait! We've seen this idea in a different guise.

Regev prepares:

Valid codewords

Short vectors have higher weights =
objective function values

$$\sum_{x \in \Lambda} |x\rangle \times \sum_x e^{-\|x\|^2/\sigma^2} |x\rangle = \sum_{x \in \Lambda} e^{-\|x\|^2/\sigma^2} |x\rangle$$

For max-XORSAT, we now want to prepare

$$\sum_{y: y \in Bx} |Bx\rangle \times \sum_{y \in \mathbb{Z}^m} f_v(y) |y\rangle$$

Valid solutions

Solutions weighted by their **objective function values**

Goal: solve MaxXORSAT(B,v)

Preparing $|\hat{f}^\ell\rangle$

1. Prepare Dicke state
(easy)

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle$$

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^T \mathbf{e}\rangle \quad (\text{Target state})$$

Preparing $|\hat{f}^\ell\rangle$

1. Prepare Dicke state
(easy)

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle$$

2. Append a new register.
Matrix multiply with B into
second register.

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle |B^\top \mathbf{e}\rangle$$

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^\top \mathbf{e}\rangle \quad (\text{Target state})$$

Preparing $|\hat{f}^\ell\rangle$

1. Prepare Dicke state
(easy)

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle$$

2. Append a new register.
Matrix multiply with B into
second register.

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle |B^\top \mathbf{e}\rangle$$

*Almost there, except
for the pesky **first**
register!*

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^\top \mathbf{e}\rangle \quad (\text{Target state})$$

Preparing $|\hat{f}^\ell\rangle$

1. Prepare Dicke state
(easy)

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle$$

*Would like to set
 $|\mathbf{e}\rangle \rightarrow |0\rangle$ in order to
remove it. Can we
compute $\mathbf{e}$ from $B^T \mathbf{e}$?*

2. Append a new register.
Matrix multiply with B into
second register.

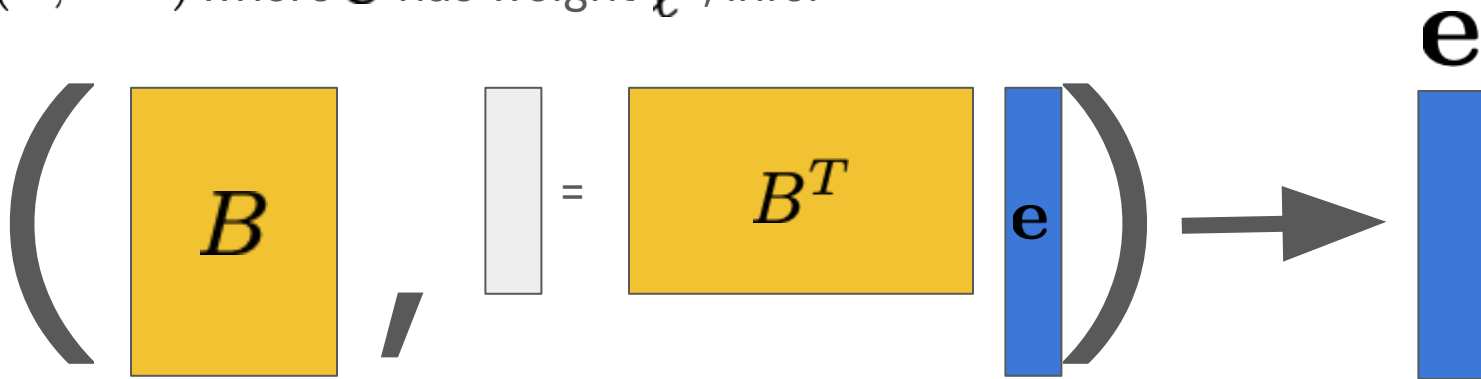
$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle |B^T \mathbf{e}\rangle$$

*Almost there, except
for the pesky **first**
register!*

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^T \mathbf{e}\rangle \quad (\text{Target state})$$

But this is a decoding problem!

Given $(B, B^T \mathbf{e})$ where $\mathbf{e}$ has weight ℓ , infer $\mathbf{e}$



But this is a decoding problem!

Given $(B, B^T \mathbf{e})$ where $\mathbf{e}$ has weight ℓ , infer $\mathbf{e}$

The diagram illustrates the problem of inferring a vector $\mathbf{e}$ from the pair $(B, B^T \mathbf{e})$. On the left, a large left parenthesis $($ is followed by a yellow square matrix labeled B , a comma $,$, a small gray vertical rectangle representing an unknown vector, an equals sign $=$, a yellow rectangular matrix labeled B^T , and a blue vertical rectangle labeled $\mathbf{e}$. A large right parenthesis $)$ follows. A thick gray arrow points from this expression to a single blue vertical rectangle on the right, which is labeled $\mathbf{e}$ at the top.

An underdetermined system of linear equations!

We can solve this by decoding the code whose parity check matrix is B^T .

Preparing $|\hat{f}^\ell\rangle$

1. Prepare Dicke state
(easy)

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle$$

2. Append a new register.
Matrix multiply with B into
second register.

$$\sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |\mathbf{e}\rangle |B^\top \mathbf{e}\rangle$$

Decode ℓ errors for
 $B^\top$

3. Set $|\mathbf{e}\rangle \rightarrow |\mathbf{0}\rangle$ by
running a decoder in
superposition. Then
remove it.

$$|\hat{f}^\ell\rangle = \sum_{\mathbf{e} \in \mathbb{F}_2^m, |\mathbf{e}|=\ell} (-1)^{\mathbf{v} \cdot \mathbf{e}} |B^\top \mathbf{e}\rangle \quad (\text{Target state})$$

What have we learned?

NP-hard Problem: Max XORSAT(B, v): Find x that **optimizes** $Bx \approx v$.

Reduces to decoding, in superposition, a code whose parity check matrix is B^T

What have we learned?

NP-hard Problem: Max XORSAT(B, v): Find x that **optimizes** $Bx \approx v$

Reduces to decoding, in superposition, a code whose parity check matrix is B^T

Why have we not proven that NP is in BQP:

Because decoding is not efficient for all codes; it is only so for structured codes!

What have we learned?

NP-hard Problem: Max XORSAT(B, v): Find x that **optimizes** $Bx \approx v$

Reduces to decoding, in superposition, a code whose parity check matrix is B^T

Why have we not proven that NP is in BQP:

Because decoding is not efficient for all codes; it is only so for structured codes!

Claimed **exponential** quantum speedup: 'Optimal Polynomial Intersection'.

Open question: break this speedup classically!

A question you may have:

**DQI reduces
classical optimization $\rightarrow$ classical
decoding.**

A question you may have:

DQI reduces
classical optimization $\rightarrow$ classical
decoding.

What about *quantum* optimization
problems?

Chapter 3: Solving quantum optimization problems via decoding



Hamiltonian Decoded Quantum Interferometry

Alexander Schmidhuber^{*1,2}, Jonathan Z. Lu^{*2}, Noah Shutt¹,
Stephen Jordan¹, Alexander Poremba^{†2,3}, and Yihui Quek^{†2,4}

¹*Google Quantum AI, Venice, CA*

²*Massachusetts Institute of Technology, Cambridge, MA*

³*Boston University, Boston, MA*

⁴*École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*

Quantum optimization problems are problems on Hamiltonians

Most Hamiltonians of interest are **local**: $H = \sum_j H_j$

Where each H_j acts non-trivially only on **constantly-many qubits**

Quantum optimization problems are problems on Hamiltonians

Most Hamiltonians of interest are **local**: $H = \sum_j H_j$ $H_j \sim j\text{-th row of } B$

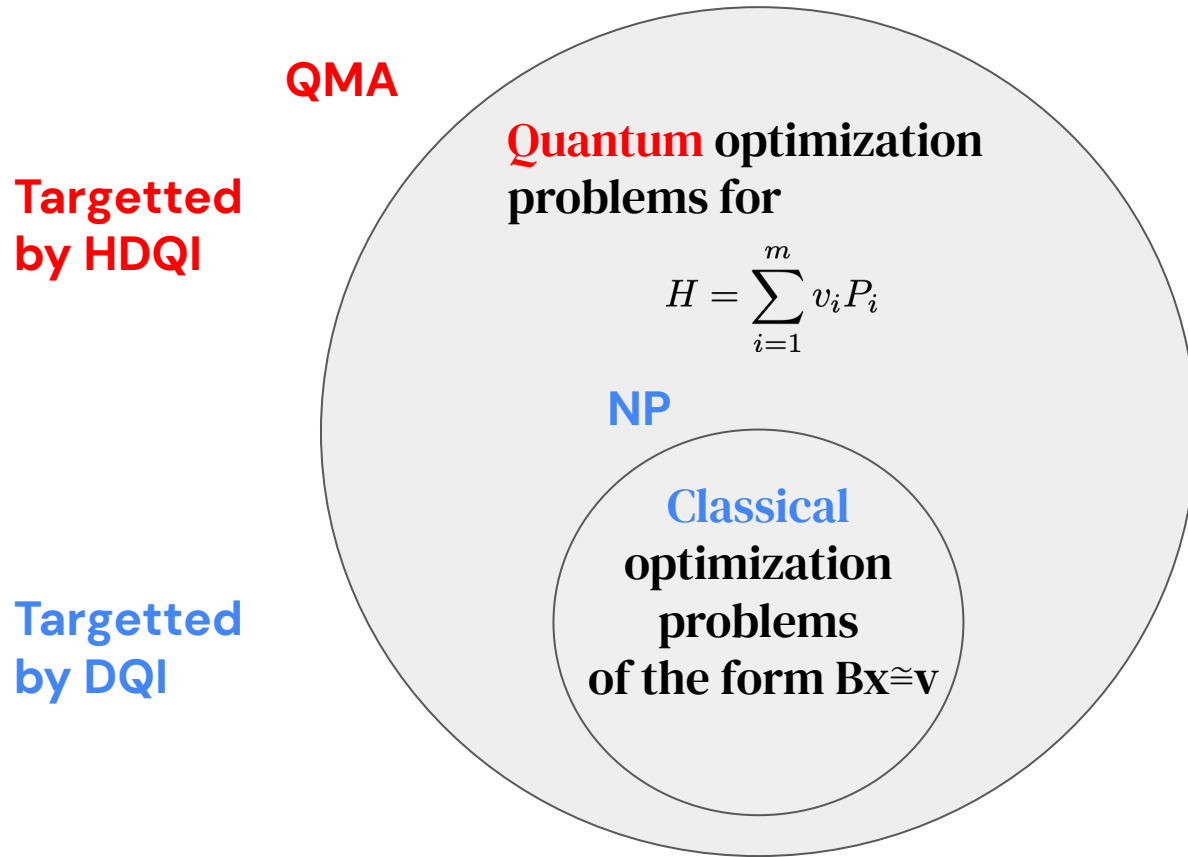
Where each H_j acts non-trivially only on **constantly-many qubits**

Quantumly:

Finding ground
state of H

Classically:

Finding optimal solution
to $Bx = v$



Strictly speaking, the labels 'QMA' and 'NP' refer to the **decision versions** of these problems (does there exist a good bitstring/quantum state)

QMA

Quantum optimization
problems for

$$H = \sum_{i=1}^m v_i P_i$$

NP

Classical
optimization
problems
of the form $Bx \cong v$

Corresponding **quantum**
optimization problem:
find the ground state of

$$H_{(B,v)} = \sum_{i=1}^m (-1)^{v_i} Z^{b_i}$$

Ground state of
 $H_{(B,v)}$ =
opt solution to
 $Bx \cong v$

QMA

**Quantum optimization
problems for**

$$H = \sum_{i=1}^m v_i P_i$$

NP

**Classical
optimization
problems
of the form $Bx \cong v$**

**DQI showed:
these problems
reduce to
classical
decoding**

QMA

Quantum optimization problems for

$$H = \sum_{i=1}^m v_i P_i$$

NP

Classical optimization problems of the form $Bx \cong v$

DQI showed: these problems reduce to classical decoding

HDQI shows: these problems also reduce to classical decoding (+ state preparation for noncommuting Hamiltonians)

What HDQI solves

Setting: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

Goal: find the ground state/ground state energy of H

What HDQI solves

Setting: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

~~**Goal: find the ground state/ground state energy of H**~~

We do much more than just ground states!

What HDQI solves

Setting: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

~~Goal: find the ground state/ground state energy of H~~

We do much more than just ground states!

Because we can prepare

$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle \langle \lambda_i^*| \quad \text{for} \quad H = \sum_i \lambda_i |\lambda_i\rangle \langle \lambda_i|$$

where $\mathcal{P}$ is an arbitrary degree- ℓ polynomial.

Why is this nontrivial?

State prepared by HDQI:

$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle |\lambda_i^*\rangle$$

- The eigenvalues and eigenvectors of H are not explicitly known, and diagonalizing H takes $O(2^{3n})$ time in general. We never have to do this.

Why is this nontrivial?

State prepared by HDQI:

$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle |\lambda_i^*\rangle$$

- The eigenvalues and eigenvectors of H are not explicitly known, and diagonalizing H takes $O(2^{3n})$ time in general. We never have to do this.

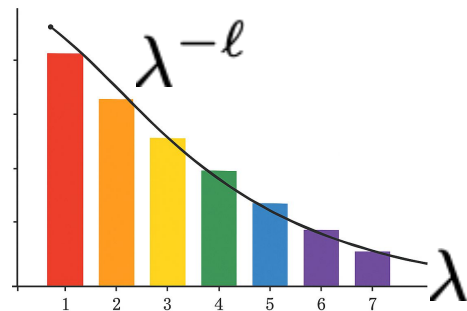
Why can't I just do a quantum singular value transform of block-encoded H ?

- Yes, but then you'd have to amplitude amplify $\|\mathcal{P}(H)\|$ times. We don't have to do so.

Choose $\mathcal{P}$ for your desired application

State prepared by HDQI:

$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle |\lambda_i^*\rangle$$



$\mathcal{P}(x) = x^{-\ell}$ (polynomial approximation thereof): ground state preparation

Choose $\mathcal{P}$ for your desired application

State prepared by HDQI:

$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle |\lambda_i^*\rangle$$

$\mathcal{P}(x) = x^{-\ell}$ (polynomial approximation thereof): ground state preparation

$\mathcal{P}(x) = \text{Threshold functions: spectral filters.}$

Choose $\mathcal{P}$ for your desired application

State prepared by HDQI: $\boxed{\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle \langle \lambda_i^*|} \Rightarrow \rho = \mathcal{P}(H)^2 / \text{Tr}(\mathcal{P}(H)^2)$

$\mathcal{P}(x) = x^{-\ell}$ (polynomial approximation thereof): ground state preparation

$\mathcal{P}(x) = \text{Threshold functions}$: spectral filters.

$\mathcal{P}(x) = \text{Polynomial approximation of } e^{-\beta x/2}$: Gibbs sampling.

The HDQI algorithm

Intuition

Let us assume for the time being that H is a commuting Pauli Hamiltonian.

HDQI tries to prepare the state:
$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle = \mathcal{P}(H) \sum_i |\lambda_i\rangle$$

Intuition

Let us assume for the time being that H is a commuting Pauli Hamiltonian.

HDQI tries to prepare the state:
$$\sum_i \mathcal{P}(\lambda_i) |\lambda_i\rangle = \mathcal{P}(H) \sum_i |\lambda_i\rangle$$

How could one prepare this state?

- Naive method: Prepare $\sum_i |\lambda_i\rangle$, then act on it with some unitary proportional to $\mathcal{P}(H)$.
- **Problem** with naive method: we don't know $\{|\lambda_i\rangle\}_i$

Enter: the maximally-entangled state

Extremely useful fact:

$$|\Gamma\rangle = \sum_{x \in \mathbb{R}_2^n} |x\rangle |x\rangle = \sum_{\lambda_i} |\lambda_i\rangle |\lambda_i^*\rangle$$

for *any* orthogonal basis $\{|\lambda_i\rangle\}_i$, including the eigenbasis of the Hamiltonian.

Enter: the maximally-entangled state

Extremely useful fact:

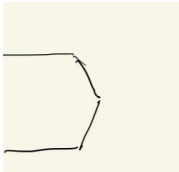
$$|\Gamma\rangle = \sum_{x \in \mathbb{R}_2^n} |x\rangle |x\rangle = \sum_{\lambda_i} |\lambda_i\rangle |\lambda_i^*\rangle$$

for *any* orthogonal basis $\{|\lambda_i\rangle\}_i$, including the eigenbasis of the Hamiltonian.

So we'll target to prepare the state

$$(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle = \sum_{\lambda} \mathcal{P}(\lambda_i) |\lambda_i\rangle |\lambda_i^*\rangle$$

(target state)

Notation: maximally-entangled state = $|\Gamma\rangle =$  (tensor network)

HDQI for commuting Hamiltonians

We also have an algorithm for noncommuting, but it's much more complicated.

Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

HDQI for commuting Hamiltonians: Overview

Three main steps:

1. State preparation (classical pre-processing)
2. State preparation (quantum part)
3. Uncomputation

HDQI for commuting Hamiltonians: Overview

Three main steps:

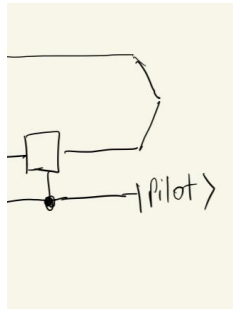
1. Pre-processing: compiling $\mathcal{P}(H)$ into sums of Paulis of equal Hamming weight
2. State preparation (quantum part)
3. Uncomputation

Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

HDQI for commuting Hamiltonians: Overview

Three main steps:

1. Pre-processing: compiling $\mathcal{P}(H)$ into sums of Paulis of equal Hamming weight
2. Append a “pilot state” encoding that sum, to $|\Gamma\rangle$. Act with Paulis computed in Step 1, controlled on $|\text{Pilot}\rangle$.
3. Uncomputation

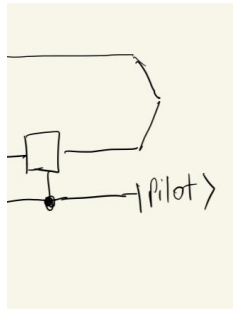


Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

HDQI for commuting Hamiltonians: Overview

Three main steps:

1. Pre-processing: compiling $\mathcal{P}(H)$ into sums of Paulis of equal Hamming weight
2. Append a “pilot state” encoding that sum, to $|\Gamma\rangle$. Act with Paulis computed in Step 1, controlled on $|\text{Pilot}\rangle$.
3. Uncomputation



NB: These two steps are always efficient for commuting Hamiltonians, may not be efficient for NONcommuting Hamiltonians.

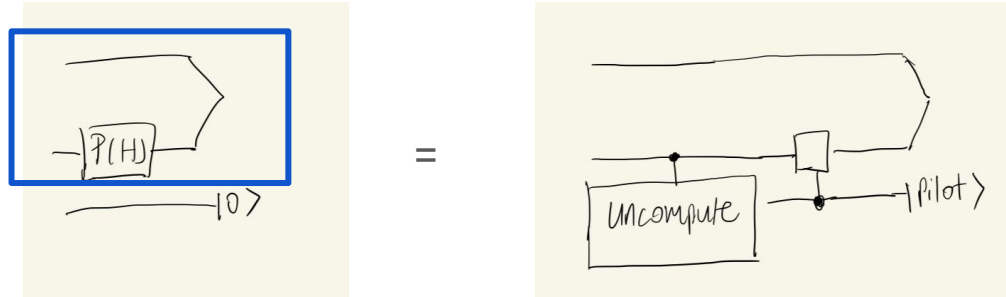
Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

HDQI for commuting Hamiltonians: Overview

Three main steps:

1. Pre-processing: compiling $\mathcal{P}(H)$ into sums of Paulis of equal Hamming weight
2. Append a “pilot state” encoding that sum, to $|\Gamma\rangle$. Act with Paulis computed in Step 1, controlled on $|\text{Pilot}\rangle$.
3. Uncomputation: set $|\text{Pilot}\rangle \rightarrow |0\rangle$

Target
state!



Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

HDQI for commuting Hamiltonians: Overview

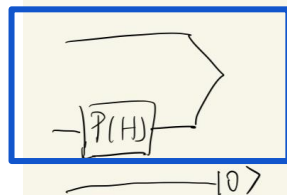
Three main steps:

1. Pre-processing: compiling $\mathcal{P}(H)$ into sums of Paulis of equal Hamming weight
2. Append a “pilot state” encoding that sum, to $|\Gamma\rangle$. Act with Paulis computed in Step 1, controlled on $|\text{Pilot}\rangle$.
3. Uncomputation: set $|\text{Pilot}\rangle \rightarrow |0\rangle$

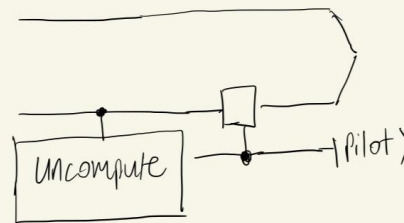
(Requires decoding)



Target
state!



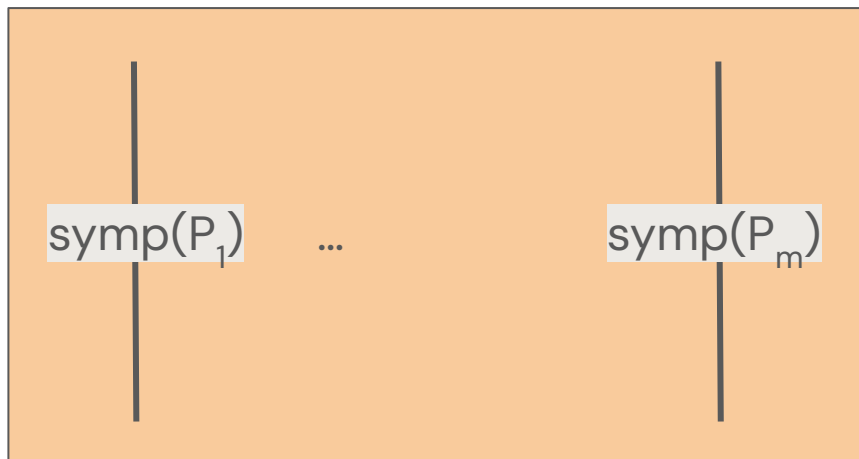
=



The decoding problem for HDQI:

Given: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

The corresponding **decoding** problem directly depends on H . Define the matrix:



$$=: B_H^T$$

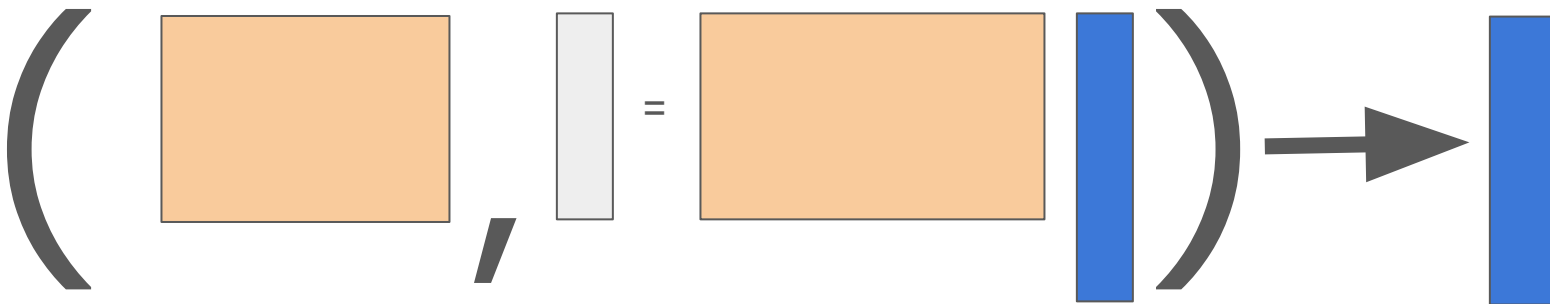
$\text{symp}(P) = 2n$ -bitstring
representation of an n -qubit
Pauli

The decoding problem for HDQI:

Given: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

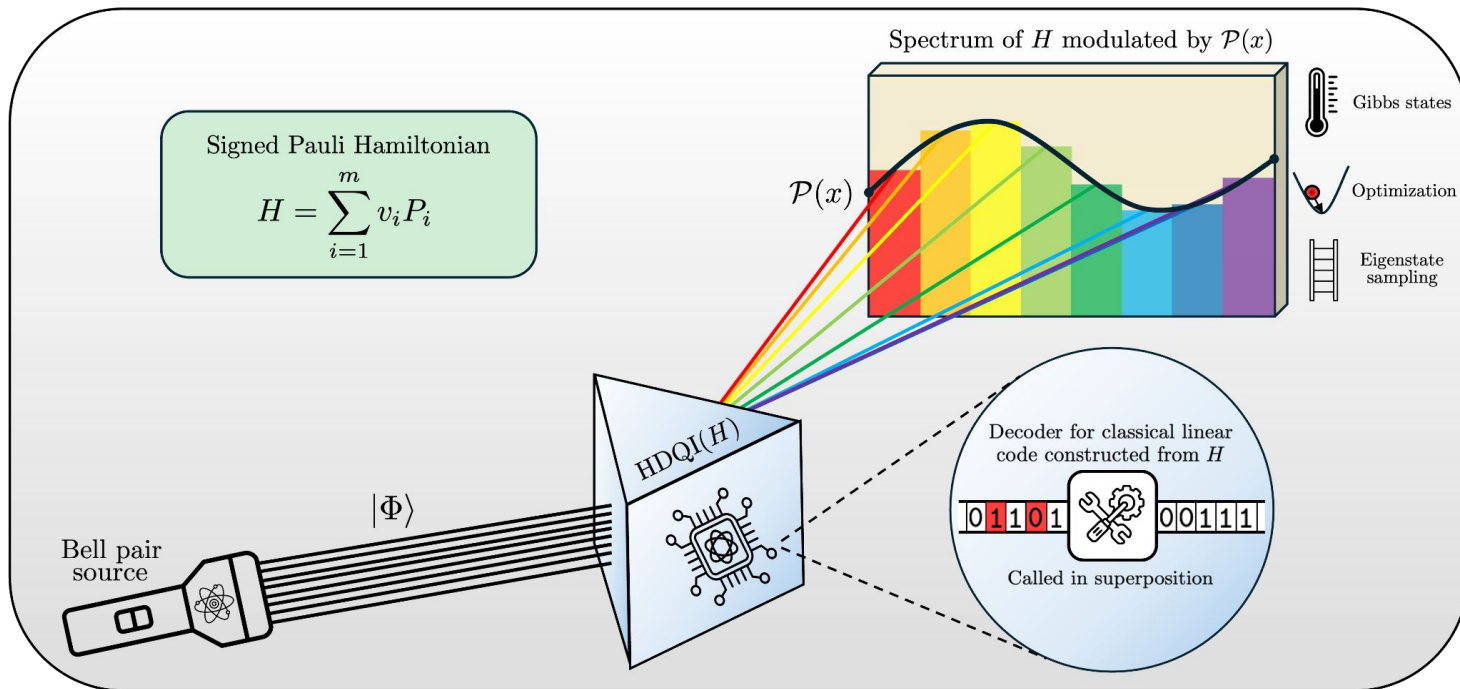
The corresponding **decoding** problem is:

Given $(B_H^T, B_H^T y)$, promised y of weight ℓ , recover y



Syndrome decoding for the code whose parity check matrix is B_H^T

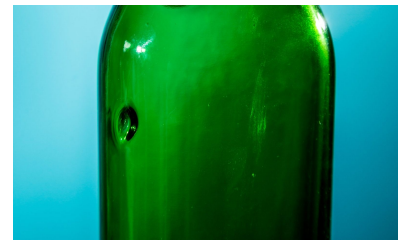
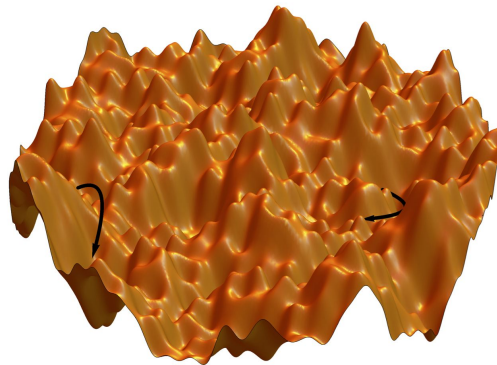
For commuting Hamiltonians, decoding is all you need.



HDQI is efficient on this family of spin glasses

Local classical spin glass (random Z-type terms)
defected into X-type terms with probability p

Spin glasses are
notoriously hard to
optimize because they
have extremely complex
energy landscapes.



Classical algorithms for stabilizer code Hamiltonians

- *Classical* algorithms for quantum Gibbs sampling some stabilizer code Hamiltonians.
 - Eigenbasis of stabilizer states (efficiently classically describable).
 - *Quantum* Gibbs samplers for some of these Hamiltonians were only developed last year: we can now sample classically.

Model	Geometry	$\dim \text{Symp}(H)$
Ising (ring)	1D, periodic	1
Surface/toric	2D, closed	2
Color code (stabilizer)	2D, closed	4
TI Pauli in 2D	2D, closed	$2K$
Haah's cubic code	3D, periodic	$O(1)$

Summary: QMA-hard Problem: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

Reduces to decoding, in superposition, a code whose parity check matrix is B_H^T

Summary: QMA-hard Problem: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

Reduces to decoding, in superposition, a code whose parity check matrix is B_H^T

Does not imply QMA in BQP because HDQI's efficiency depends on structure of the Hamiltonians.

Summary: QMA-hard Problem: $H = \sum_{i=1}^m v_i P_i$ where $v_i \in \{\pm 1\}$, $P_i = n$ -qubit Paulis

Reduces to decoding, in superposition, a code whose parity check matrix is B_H^T

Does not imply QMA in BQP because HDQI's efficiency depends on structure of the Hamiltonians.

Weirdly:

HDQI on Max-XORSAT $\neq$ DQI. We are the first 'Regev cousin' **not to require a Quantum Fourier transform.**

Vision 1: Make DQI/HDQI more useful!

- DQI/HDQI require the original problem to have “decodable structure” in order for the resulting algorithm to be efficient.

Can we get advantage with less structure?

- Can we extend HDQI to physics/chemistry/fermionic Hamiltonians?

Vision 2: Building up ‘decoders-to-algorithms’

A ‘third’ algorithmic paradigm (beyond HSP and amplitude amplification).



Vision 2: Building up ‘decoders-to-algorithms’

A ‘third’ algorithmic paradigm (beyond HSP and amplitude amplification).

**Coding
theory**

Better decoders lead
to better algorithmic
performance



Algorithms

New problems demand
new decoders



Bonus slides

Step 1: Classical preprocessing

Compile $\mathcal{P}(H)$ into a sum of elementary symmetric polynomials in the variables $\{v_i P_i\}_{i \in [m]}$.

$$\mathcal{P}(H) = \sum_{k=0}^{\ell} w_k \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}|=k}} \left(\prod_{i \in \mathbf{y}} v_i P_i \right)$$

Step 1: Classical preprocessing

Compile $\mathcal{P}(H)$ into a sum of elementary symmetric polynomials in the variables $\{v_i P_i\}_{i \in [m]}$.

$$\mathcal{P}(H) = \sum_{k=0}^{\ell} w_k \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}|=k}} \left(\prod_{i \in \mathbf{y}} v_i P_i \right) = \sum_{k=0}^{\ell} w_k \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}|=k}} v_{\mathbf{y}} P_{\mathbf{y}} .$$

$\mathbf{y}$ is a vector that picks out terms in H in the places that $\mathbf{y}$ has a 1

$$\prod_{i \in \mathbf{y}} v_i := v_{\mathbf{y}}; \quad \prod_{i \in \mathbf{y}} P_i := P_{\mathbf{y}}$$

Target: apply $(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle$

Step 1: Classical preprocessing

Compile $\mathcal{P}(H)$ into a sum of elementary symmetric polynomials in the variables $\{v_i P_i\}_{i \in [m]}$.

$$(\mathcal{P}(H) \otimes \mathbb{I})|\Gamma\rangle = \sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y \otimes \mathbb{I} |\Gamma\rangle$$

“All” we need to do now: apply a weighted sum of Paulis.

Step 2: Pilot state preparation

Target state: $\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$

Prepare the pilot state, which encodes the operations to apply to $|\Gamma\rangle$

$$|\text{Pilot}\rangle = \sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} |y\rangle$$

$|\text{pilot}\rangle$

Step 2: Pilot state preparation

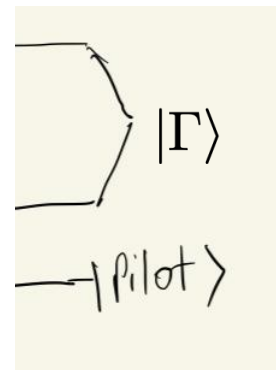
Target state: $\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$

Prepare the pilot state, which encodes the operations to apply to $|\Gamma\rangle$

$$|\text{Pilot}\rangle = \sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} |y\rangle$$

Append to $|\Gamma\rangle$, obtaining

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} |y\rangle \otimes |\Gamma\rangle$$



Step 2: Pilot state preparation

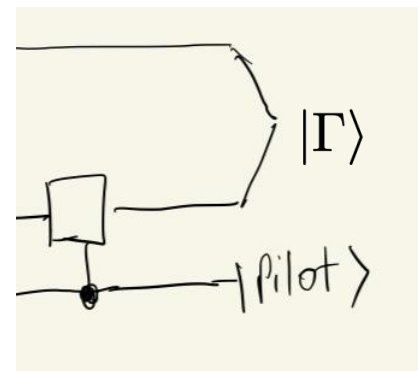
Target state: $\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$

Prepare the pilot state, which encodes the operations to apply to $|\Gamma\rangle$

$$|\text{Pilot}\rangle = \sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} |y\rangle$$

Append to $|\Gamma\rangle$, obtaining

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} |y\rangle \otimes |\Gamma\rangle$$



Conditioned on $\mathbf{y}$ in Pilot register, apply $\prod_{i \in y} v_i = v_y$ and P_y to $|\Gamma\rangle$.

Result of Step 2: Pilot state preparation

Target state: $\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$

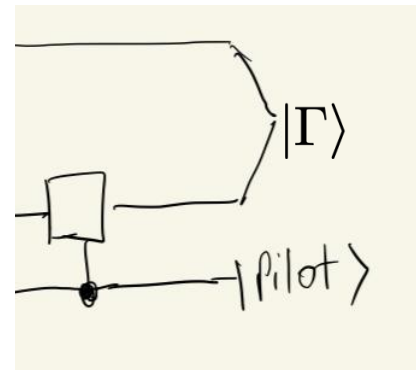
This obtains the state

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y |y\rangle \otimes P_y |\Gamma\rangle$$

Pilot register System register

Compare to target:

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$$



Result of Step 2: Pilot state preparation

Target state: $\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$

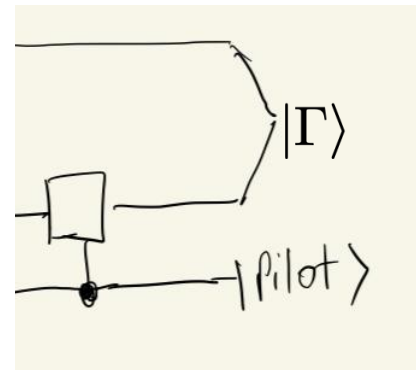
This obtains the state

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y \boxed{|y\rangle} \otimes P_y |\Gamma\rangle$$

Pilot register System register

Compare to target:

$$\sum_{k=0}^{\ell} w_k \sum_{\substack{y \in \mathbb{F}_2^m \\ |y|=k}} v_y P_y |\Gamma\rangle$$



We are almost there, **except for the pesky pilot register!**
Solution: uncompute it!

HDQI for commuting Hamiltonians: Step 2

Uncompute $|y\rangle$ given $P_y|\Gamma\rangle$, i.e. perform

$$U_{\text{Dec}} : |y\rangle \otimes P_y|\Gamma\rangle \rightarrow |0\rangle \otimes P_y|\Gamma\rangle$$

so we can remove the $|y\rangle$ register.

HDQI for commuting Hamiltonians: Step 2

Uncompute $|y\rangle$ given $P_y|\Gamma\rangle$, i.e. perform

$$U_{\text{Dec}} : |y\rangle \otimes P_y|\Gamma\rangle \rightarrow |0\rangle \otimes P_y|\Gamma\rangle$$

so we can remove the $|y\rangle$ register.

(Housekeeping: we can pretend that $P_y|\Gamma\rangle$ is simply $|P_y\rangle$, because it is a tensor product of Bell states, which are orthogonal.)

In fact, let's identify each $|P_y\rangle$ via its Pauli symplectic representation, i.e. the computational basis vector $|\text{symp}(P_y)\rangle$

HDQI for commuting Hamiltonians: Step 2

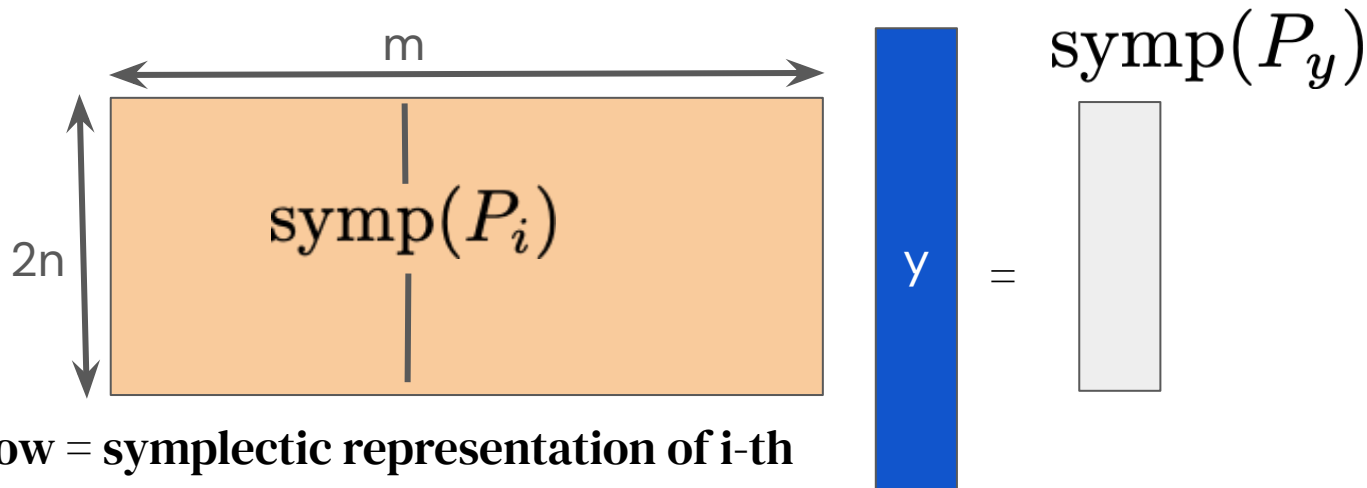
Uncomputation = from each $\text{symp}(P_{\mathbf{y}})$ where $P_{\mathbf{y}} = \left(\prod_{i \in \mathbf{y}} v_i P_i \right)$, learn $\mathbf{y}$
(promised $\mathbf{y}$ of weight ℓ)

Key observation: $\text{symp}(P_{\mathbf{y}})$ can be expressed as product of a special
`matrix of Paulis' with the vector $\mathbf{y}$.

→ uncomputation becomes syndrome decoding again

HDQI for commuting Hamiltonians: Step 2

Uncomputation = from each $\text{symp}(P_{\mathbf{y}})$ where $P_{\mathbf{y}} = \left(\prod_{i \in \mathbf{y}} v_i P_i \right)$, learn $\mathbf{y}$



i -th row = symplectic representation of i -th Pauli in H

Extensions of the technique

- Hamiltonians with arbitrary Pauli coefficients:

$$\sum_i \alpha_i P_i, \quad \alpha_i \in \mathbb{R}$$

HAMILTONIAN DECODED QUANTUM INTERFEROMETRY FOR GENERAL PAULI HAMILTONIANS

KAI FENG BU^{1,2}, WEICHEN GU¹, AND XIANG LI¹

ABSTRACT. In this work, we study the Hamiltonian Decoded Quantum Interferometry (HDQI) for the general Hamiltonians $H = \sum_i c_i P_i$ on an n -qubit system, where the coefficients $c_i \in \mathbb{R}$ and P_i are Pauli operators. We show that, given access to an appropriate decoding oracle, there exist efficient quantum algorithms for preparing the state $\rho_P(H) = \frac{\mathcal{P}^2(H)}{\text{Tr}(\mathcal{P}^2(H))}$, where $\mathcal{P}(H)$ denotes the matrix function induced by a univariate polynomial $\mathcal{P}(x)$. Such states can be used to approximate the Gibbs states of H for suitable choices of polynomials. We further demonstrate that the proposed algorithms are robust to imperfections in the decoding procedure. Our results substantially extend the scope of HDQI beyond stabilizer-like Hamiltonians, providing a method for Gibbs-state preparation and Hamiltonian optimization in a broad class of physically and computationally relevant quantum systems.

Extensions of the technique

- Hamiltonians with arbitrary Pauli coefficients:

$$\sum_i \alpha_i P_i, \quad \alpha_i \in \mathbb{R}$$

- Non-commuting Hamiltonians

Pre-processing step performed this decomposition

$$\mathcal{P}(H) = \sum_{k=0}^{\ell} w_k \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}|=k}} \left(\prod_{i \in \mathbf{y}} v_i P_i \right) = \sum_{k=0}^{\ell} w_k \sum_{\substack{\mathbf{y} \in \mathbb{F}_2^m \\ |\mathbf{y}|=k}} v_{\mathbf{y}} P_{\mathbf{y}}.$$

Assumed you can write Pauli product in any order → **not true when noncommuting**

HAMILTONIAN DECODED QUANTUM INTERFEROMETRY FOR GENERAL PAULI HAMILTONIANS

KAI FENG BU^{1,2}, WEICHEN GU¹, AND XIANG LI¹

ABSTRACT. In this work, we study the Hamiltonian Decoded Quantum Interferometry (HDQI) for the general Hamiltonians $H = \sum_i c_i P_i$ on an n -qubit system, where the coefficients $c_i \in \mathbb{R}$ and P_i are Pauli operators. We show that, given access to an appropriate decoding oracle, there exist efficient quantum algorithms for preparing the state $\rho_{\mathcal{P}(H)} = \frac{\mathcal{P}^2(H)}{\text{Tr}(\mathcal{P}^2(H))}$, where $\mathcal{P}(H)$ denotes the matrix function induced by a univariate polynomial $\mathcal{P}(x)$. Such states can be used to approximate the Gibbs states of H for suitable choices of polynomials. We further demonstrate that the proposed algorithms are robust to imperfections in the decoding procedure. Our results substantially extend the scope of HDQI beyond stabilizer-like Hamiltonians, providing a method for Gibbs-state preparation and Hamiltonian optimization in a broad class of physically and computationally relevant quantum systems.

Pilot state
begins to look
complicated

HDQI for non-commuting Hamiltonians

Nevertheless, we find that in some cases the Pilot state can be efficiently prepared.

Anti-commutation graph of H

$$H = \sum_{i=1}^m v_i P_i \text{ where } v_i \in \{\pm 1\}, P_i = n\text{-qubit Paulis}$$

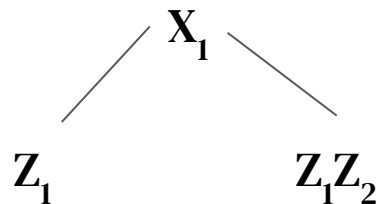
Vertices are all Pauli terms; put an edge between terms if they anticommute.

Anti-commutation graph of H

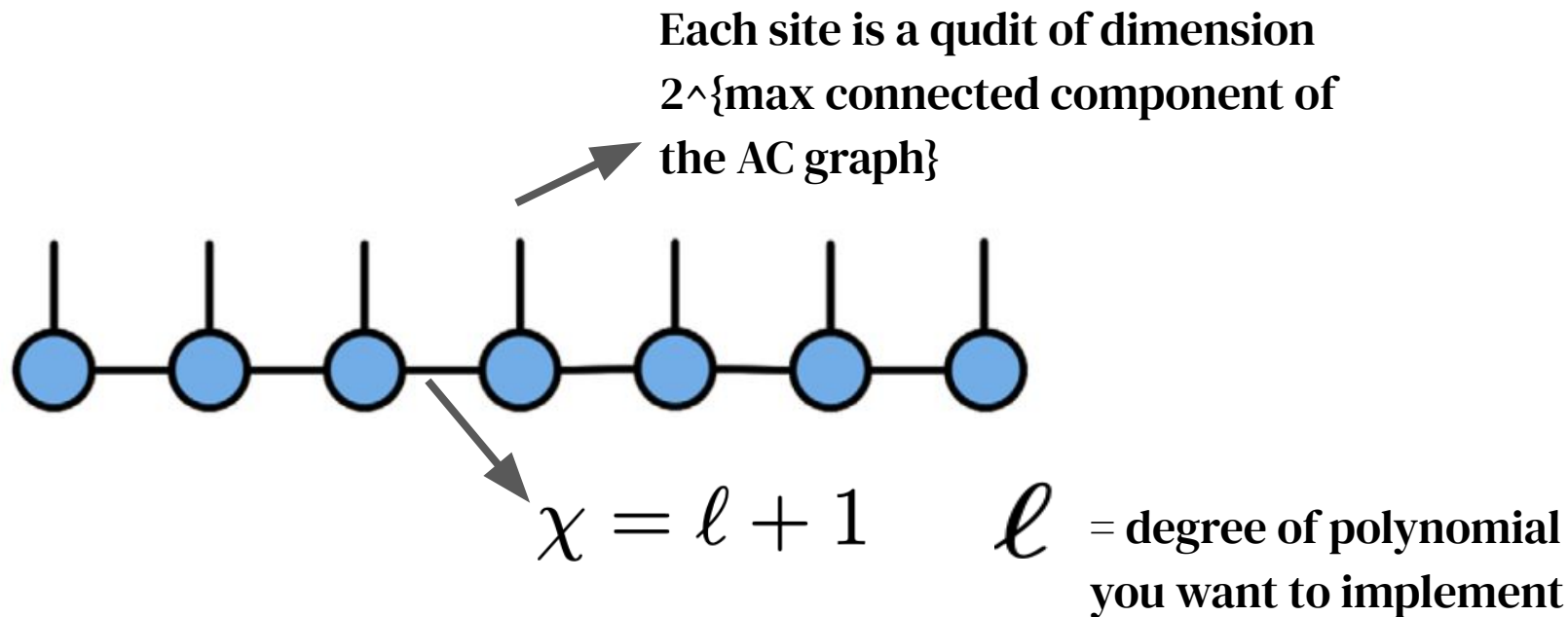
$$H = \sum_{i=1}^m v_i P_i \text{ where } v_i \in \{\pm 1\}, P_i = n\text{-qubit Paulis}$$

Vertices are all Pauli terms; put an edge between terms if they anticommute.

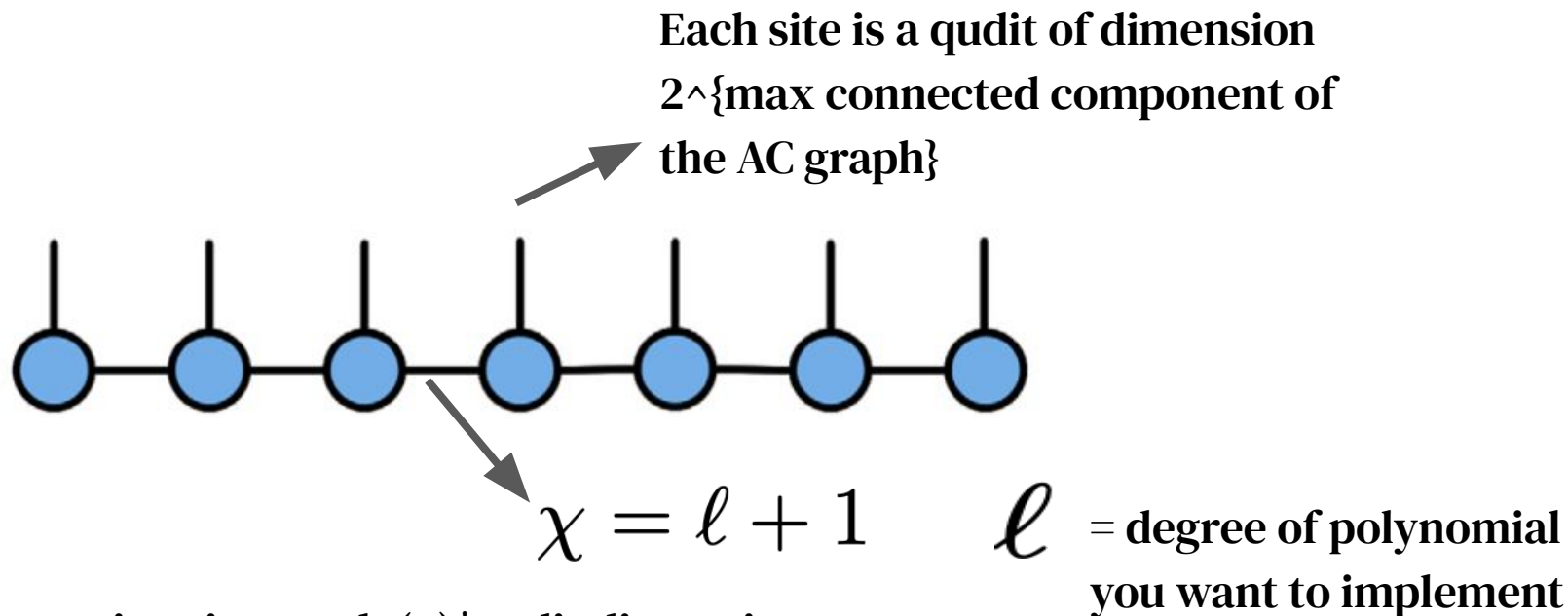
Example: for Hamiltonian $H = Z_1 + X_1 + Z_1 Z_2$:



Theorem: the pilot state is a Matrix Product State!



Theorem: the pilot state is a Matrix Product State!



Preparation time: $\text{poly}(n) \times \text{qudit dimension}$

Efficient if AC graph has **log-sized connected components**.