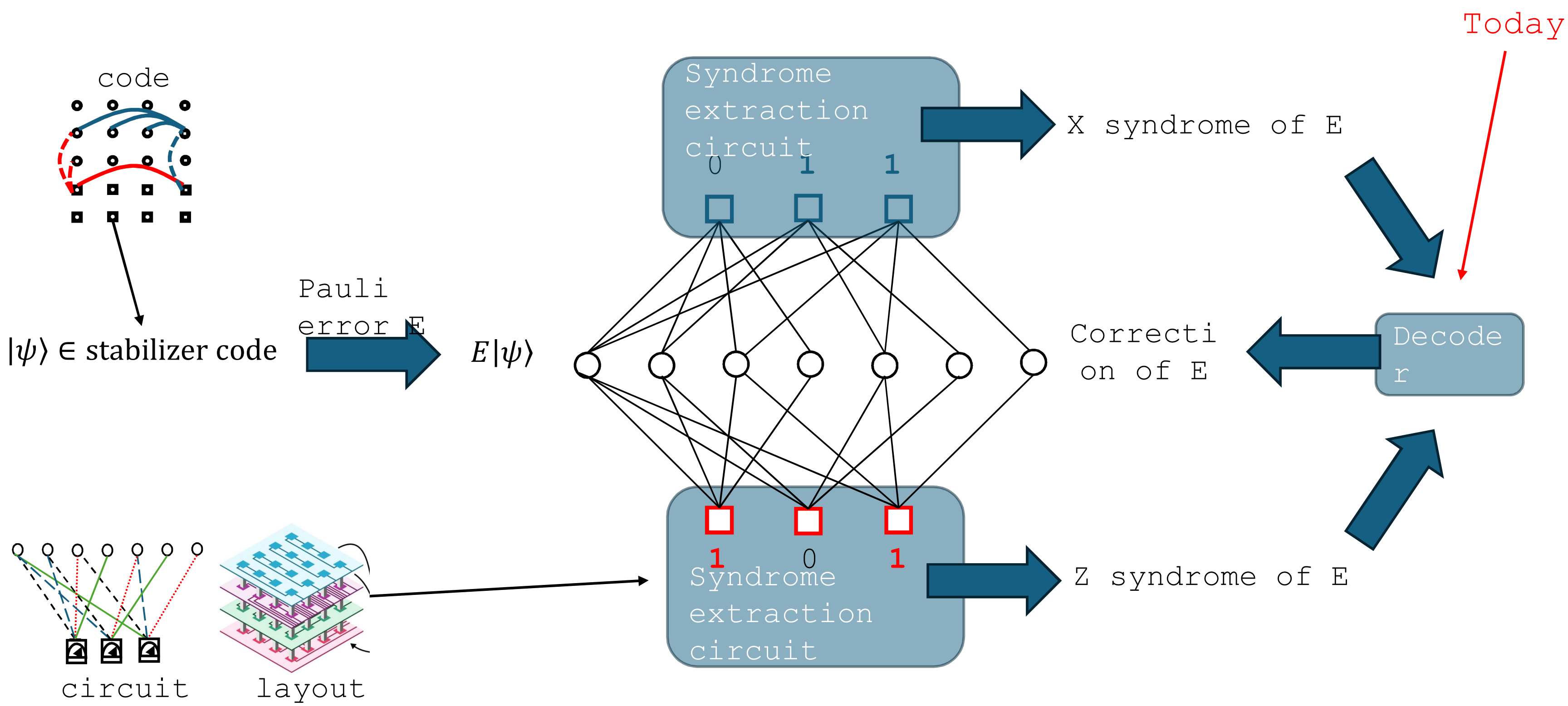
Quantum LDPC codes
Lecture 3

Nicolas Delfosse
Microsoft

PCMI Summer School 2023
July 27th 2023

Overview of the whole scheme



The decoding problem

MLE, MLC and MW decoders

MLE = Most Likely Error
MLC = Most Likely Coset
MW = Min Weight

- Model = Perfect measurement
- Qubit noise = Probability distribution $\Pr(E)$ over $\{I, X, Y, Z\}^n$

Def. A decoder is a map $D: \{0,1\}^r \to \{I, X, Y, Z\}^n$.

- It is a MW decoder if $D(\sigma)$ is a min weight Pauli error with syndrome σ.
- It is a MLE decoder if $D(\sigma)$ is a Pauli error that maximizes $\Pr(E|\sigma)$.
- It is a MLC decoder if $D(\sigma)$ is a Pauli error that maximizes $\Pr(E.S|\sigma)$.

Comparison

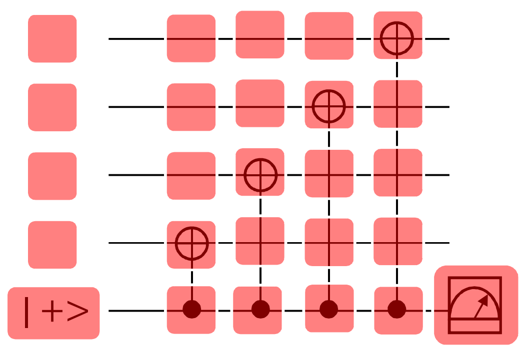
- In general MLC > MLE > MW

- If low noise rate + low correlation probability then

  MLE ≈ MLC ≈ MW

- A MLC decoder may achieve a higher threshold. For surface codes:
  - MLE threshold ≈ 16%
  - MLC threshold ≈ 19%

Standard Pauli noise models



Perfect measurement model:
- Noise on data qubits

Phenomenological model:
- Noise on data qubits
- Noise on measurements

Circuit noise:
- Noise on data qubits
- Noise on measurements
- Noise on ancilla qubits
- Noise on gates
- Noise on waiting qubits

From perfect measurements to circuit model

The syndrome extraction circuit is noisy.

=> We need to correct circuit faults

But Circuit faults ≈ Pauli errors for a larger code.

=> We focus on Pauli errors because:

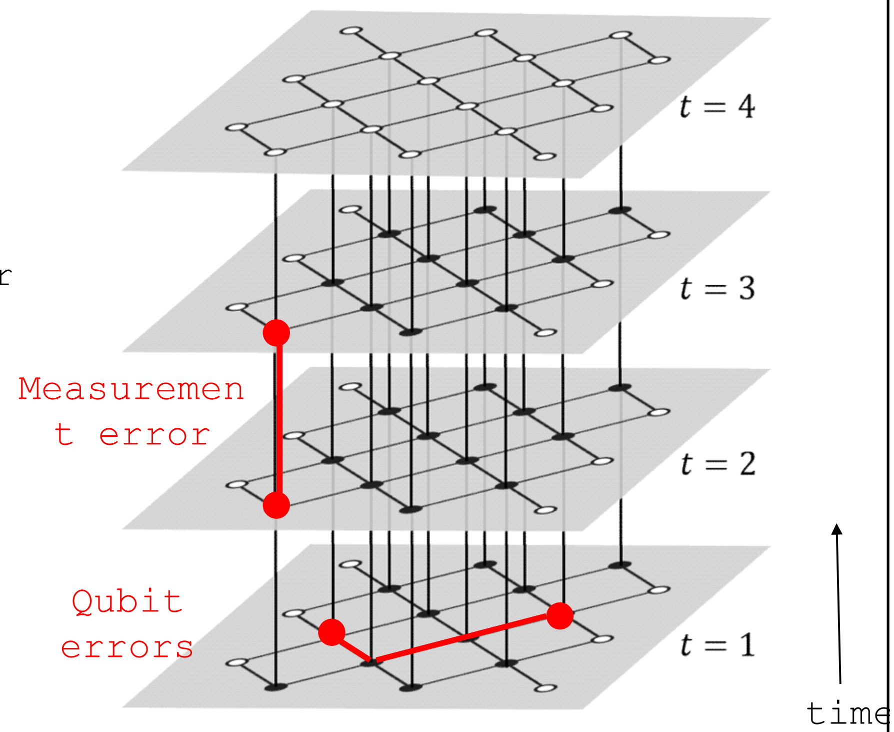
Ex.

- Circuit faults in 2D surface codes

≈ Pauli errors in 3D surface code[1].

- Circuit fault in a Clifford circuit

≈ Pauli errors in the spacetime code[2].



Three rounds of

1. Dennis, Kitaev, Landhal, Preskill - Topological quantum memory (2001)
2. Delfosse, Paetznick - Spacetime codes of Clifford circuits (2023)

Lookup table decoder

Computational complexity

Computational complexity:

- MLE decoding for stabilizer codes is NP-hard[1]
- MLC decoding for stabilizer codes is #P-hard[2]

In practice:

- Use highly structure codes (Hamming, Reed Muller, BCH, Reed Solomon, Turbo, Polar, LDPC, Spatially-Coupled)
- Exploit this structure to design an efficient decoder.
- The decoder must be adapted to the resource available: memory, compute, energy, space, time, technology, cost

1. Berlekamp, McEliece, Van Tilbor
2. Iyer, Poulin (2015)

Implementation of a MW decoder using a lookup table

Input: Code + bound M.

Output: A MW decoder for the correction of all errors with weight $\leq M$

1. Initialize $D(\sigma) = 0$ for all σ.
2. For $w = 1,2,\ldots,M$ do:
3. Loop over Pauli errors E with weight w and
4. Compute $\sigma(E)$
5. If $D(\sigma) = 0$, set $\mathrm{D}(\sigma) = E$
6. Return D

Question. What is the size of D?

$$\sum_{w=`1,\ldots,M} \binom{n}{w} 3^w$$

Checks	Correctio n
100	Flip 4
010	Flip 6
110	Flip 2
001	Flip 7
101	Flip 3
011	Flip 5
111	Flip 1

LUT decoder for Hamming cod

Example

Is there a LUT decoder in my laptop?

Claim:

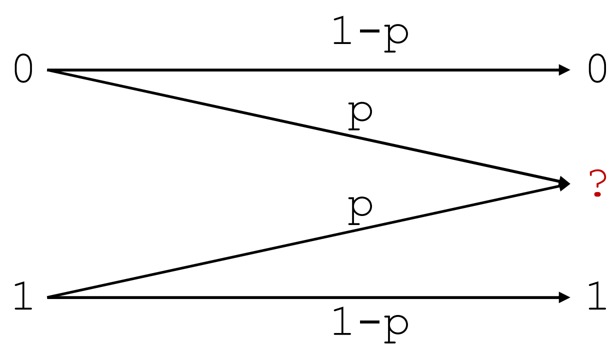
- The flash memory uses LDPC codes with length $n \approx 8{,}000$.
- The code has distance $d \approx 30$

Is that feasible?

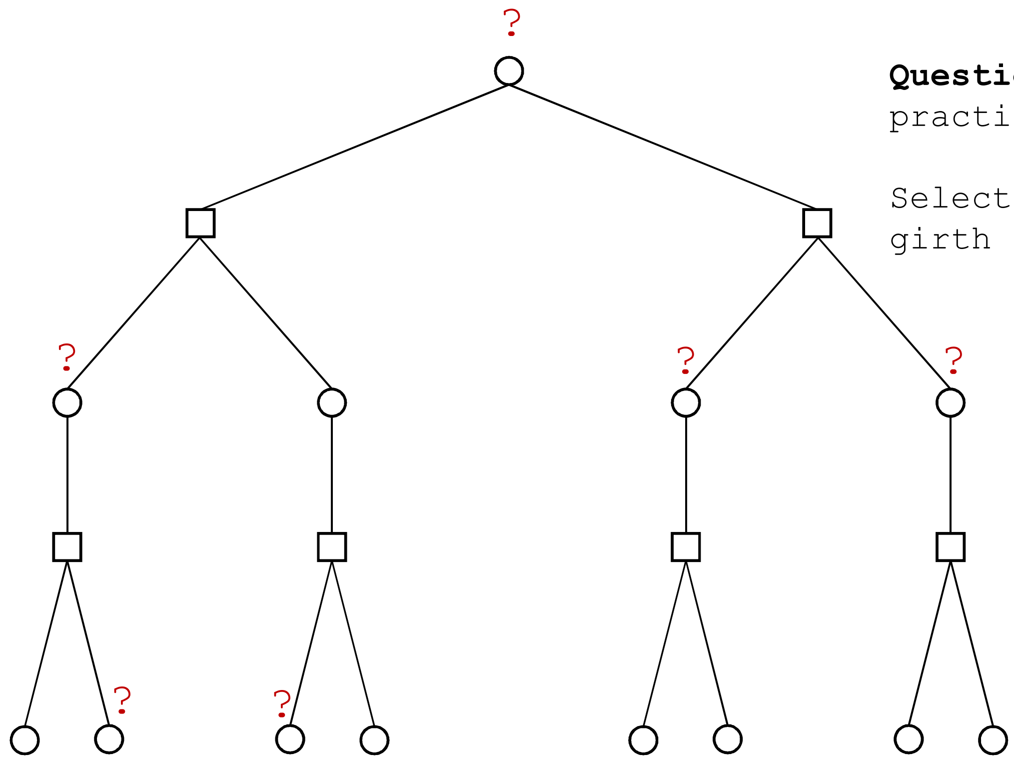
- We need to store all corrections with weight up to $M = 15$.
- Cost: $\geq \binom{8{,}000}{15} \approx 2.10^{46}$ bits =
- 20 trillions quetabits

Belief Propagation decoder for classical codes

Erasure channel



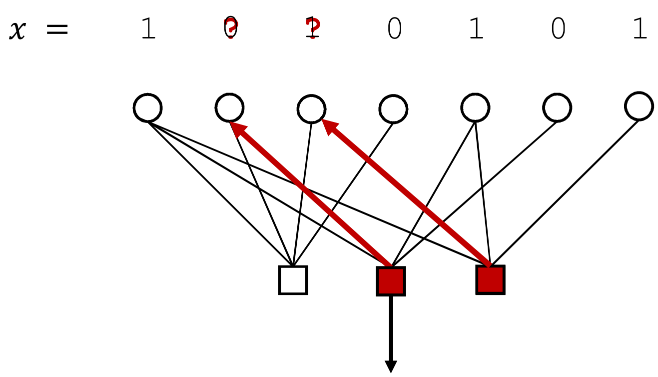
Correction of an erased bit with a (2,3)-code



Question: Does it work in practice?

Select graph with large girth (no short cycle).

Classical peeling decoder



$$x_1 + x_2 + x_5 + x_6 = 0$$
$$\Rightarrow 1 + x_2 + 1 + 0 = 0$$
$$\Rightarrow x_2 = 0$$

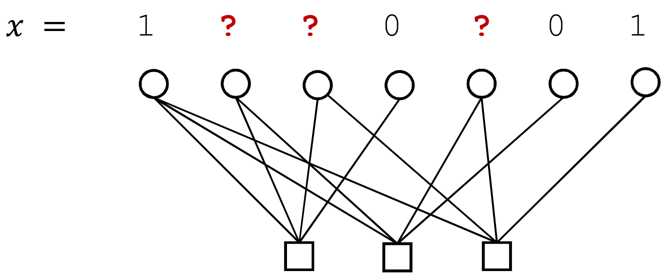
Def. A *dangling check* is a check connected to a single erased bit.

Peeling decoder:

1. While there exists a dangling check do:
2. Select a dangling check and use it to correct the incident bit.

Zyablov, Pinsker – 1974

Stopping sets



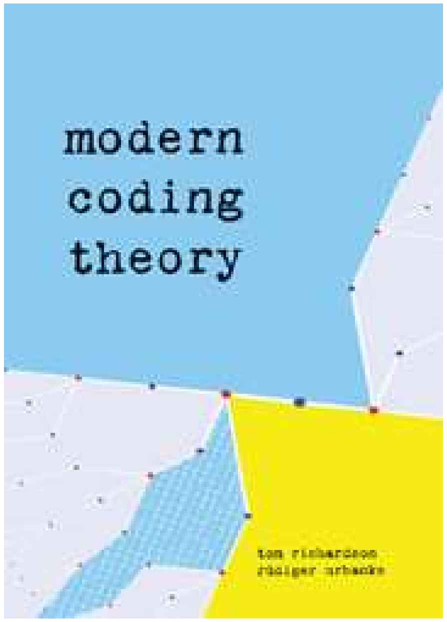
Def. A *stopping set* is a set of erased bits with no dangling check.

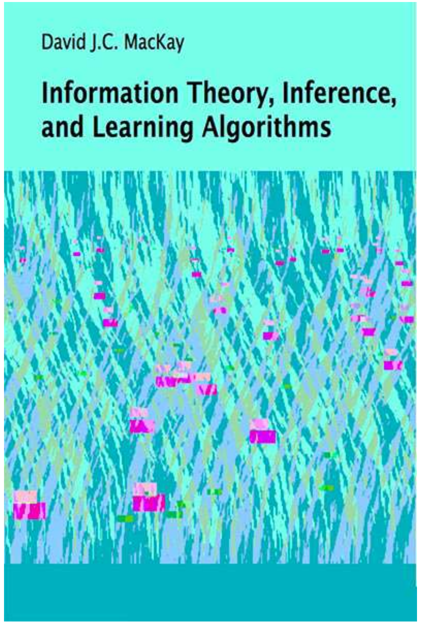
Prop. [Zyablov, Pinsker - 1974]. The peeling decoder fails iff the erasure contains a stopping set.

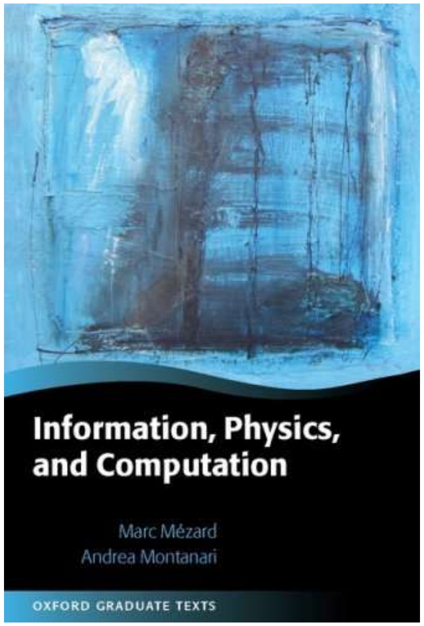
Theorem. [Richardson, Urbanke - 2001]. For carefully designed classical LDPC codes, the probability of an erased stopping sets vanishes.

Basic idea: Design graphs with no short cycle.

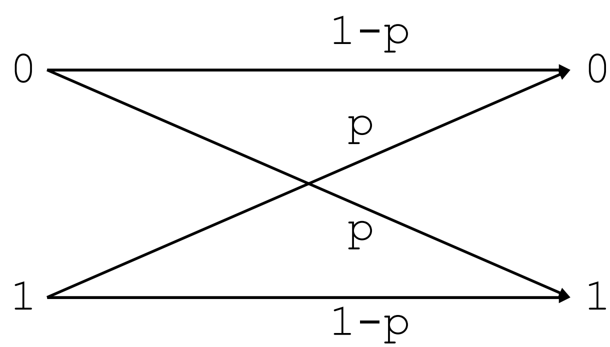
References


modern
coding
theory
tom richardson
rüdiger urbanke


David J.C. MacKay
Information Theory, Inference,
and Learning Algorithms


Information, Physics,
and Computation
Marc Mézard
Andrea Montanari
OXFORD GRADUATE TEXTS

Binary symmetric channel



Marginal bit-flip probability

$$x_1 x_2 \dots x_n \xrightarrow{\text{channel}} y_1 y_2 \dots y_n$$

Goal: Compute $P(x_i = 0 \mid y)$ and $P(x_i = 1 \mid y)$ for all i.

We can use this value to correct each bit.

How?

Marginal bit-flip probability

We want to evaluate

$$P(x_1 = 0 \mid y) = \sum_{\substack{x_1=0 \\ x\in C \\ x_2, x_3, \dots, x_n}} P(y \mid x) \mathbf{1}_{x\in C}$$

$$= \sum_{\substack{x_1=0 \\ x_2, x_3, \dots, x_n}} \prod_{i=1,..n} P(y_i \mid x_i)\, \mathbf{1}_{x\in C}$$

What is $\mathbf{1}_{x\in C}$ for the distance-3 repetition code?

- $x \in C$ iff $x_1 + x_2 = 0$ and $x_2 + x_3 = 0$
- Therefore $\mathbf{1}_{x\in C} = (1 + x_1 + x_2)(1 + x_2 + x_3)$

Marginal bit-flip probability

We want to evaluate

$$P(x_1 = 0 \mid y) = \sum_{\substack{x_1=0 \\ x_2, x_3, \dots, x_n}} \prod_{i=1,..n} P(y_i \mid x_i) \, \mathbf{1}_{x \in C}$$

It a function of the form

$$\sum_{x_2, x_3, \dots, x_n} f_1(x) \dots f_m(x)$$

For LDPC codes each f_i depends only on a small number of variables x_i.

How many multiplications are needed?

- $O(2^k n)$ multiplications.

Example

Compute the sum

$$\sum_{x_1, x_2, x_3, x_4} f(x_1, x_2, x_3) g(x_4)$$

$$\left(\sum_{x_1, x_2, x_3} f(x_1, x_2, x_3) \right) \left(\sum_{x_4} g(x_4) \right)$$

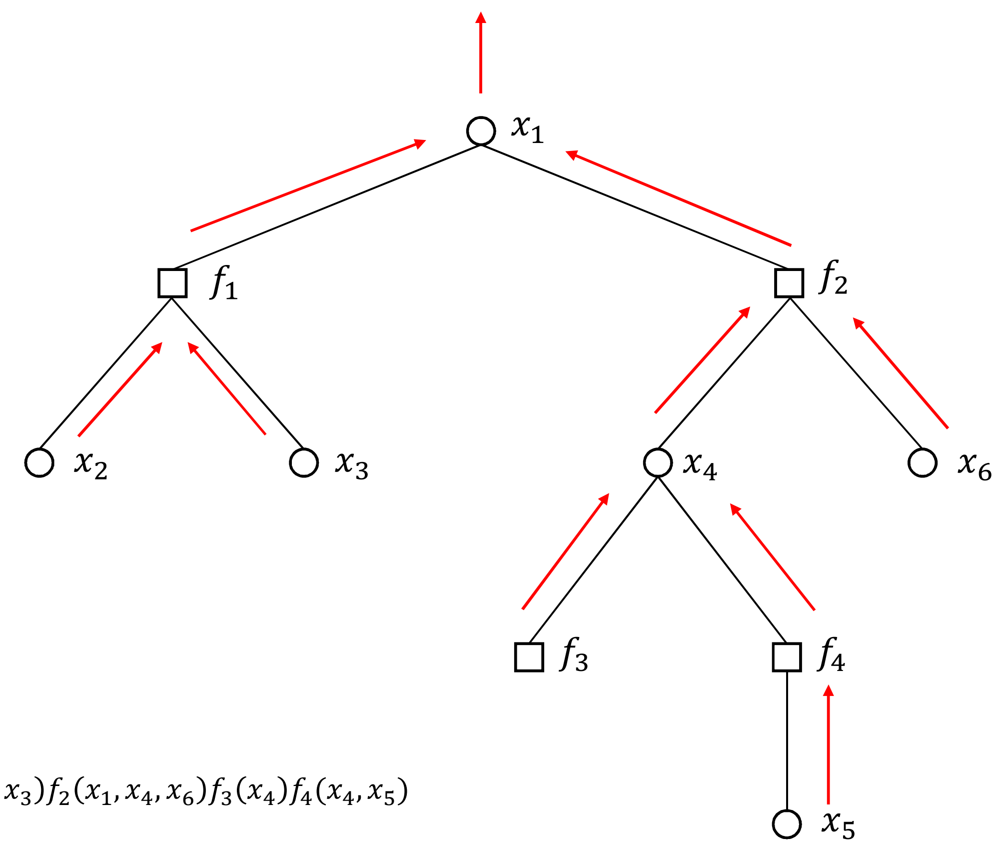
Marginal computation by Belief Propagation

Example: Compute the sum

$$\sum_{x_2, x_3, x_4, x_5, x_6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

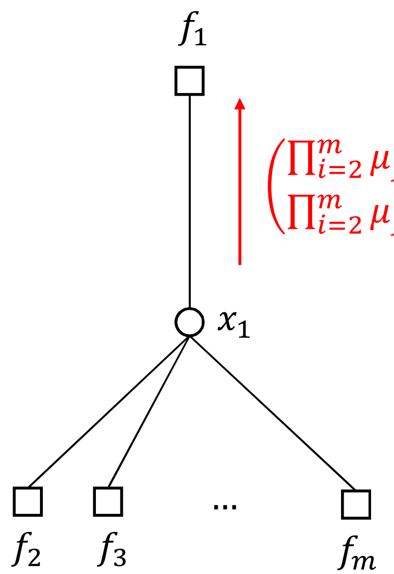
Strategy:

- Represent the f_i and their variables as a graph (called factor graph).
- Use the graph topology to optimize the computation of partial sums.



$$\sum_{x_2, x_3, x_4, x_5, x_6} f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)$$

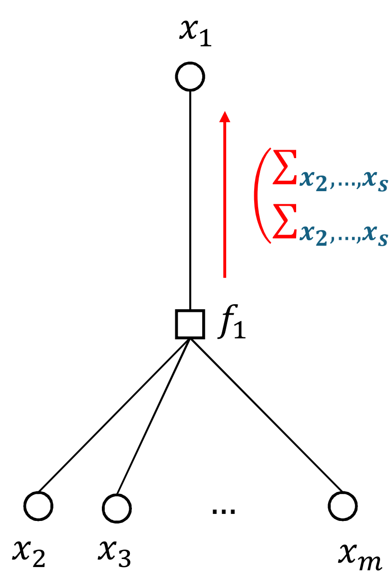
Belief Propagation messages



$$\begin{pmatrix} \prod_{i=2}^{m} \mu_{f_i \to x_1}(\mathbf{0}) \\ \prod_{i=2}^{m} \mu_{f_i \to x_1}(\mathbf{1}) \end{pmatrix}$$ = product of incoming messages

From variables to checks

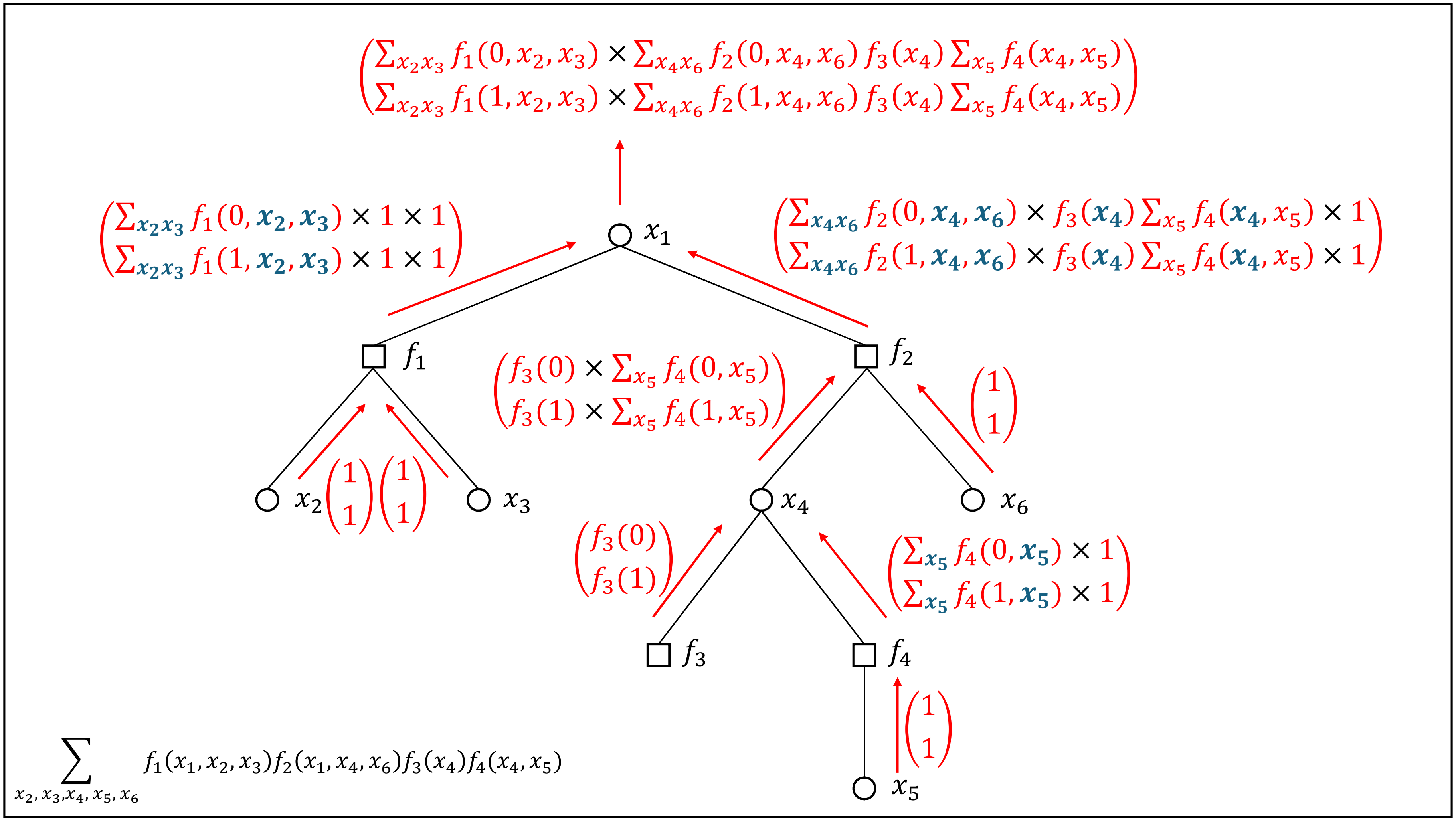
Belief Propagation messages



$$\begin{pmatrix} \sum_{x_2,\ldots,x_s} f_1(0, x_2, \ldots, x_m) \ \prod_{i=2}^{m} \mu_{x_i \to f_1}(x_i) \\ \sum_{x_2,\ldots,x_s} f_1(1, x_2, \ldots, x_m) \ \prod_{i=2}^{m} \mu_{x_i \to f_1}(x_i) \end{pmatrix}$$

= sum over all values of incoming va

From checks to variable



$\begin{pmatrix} \sum_{x_2x_3} f_1(0,x_2,x_3) \times \sum_{x_4x_6} f_2(0,x_4,x_6)\, f_3(x_4) \sum_{x_5} f_4(x_4,x_5) \\ \sum_{x_2x_3} f_1(1,x_2,x_3) \times \sum_{x_4x_6} f_2(1,x_4,x_6)\, f_3(x_4) \sum_{x_5} f_4(x_4,x_5) \end{pmatrix}$
$\begin{pmatrix} \sum_{x_2x_3} f_1(0,x_2,x_3) \times 1 \times 1 \\ \sum_{x_2x_3} f_1(1,x_2,x_3) \times 1 \times 1 \end{pmatrix}$
x_1
$\begin{pmatrix} \sum_{x_4x_6} f_2(0,x_4,x_6) \times f_3(x_4) \sum_{x_5} f_4(x_4,x_5) \times 1 \\ \sum_{x_4x_6} f_2(1,x_4,x_6) \times f_3(x_4) \sum_{x_5} f_4(x_4,x_5) \times 1 \end{pmatrix}$
f_1
$\begin{pmatrix} f_3(0) \times \sum_{x_5} f_4(0,x_5) \\ f_3(1) \times \sum_{x_5} f_4(1,x_5) \end{pmatrix}$
f_2
$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
x_2 $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
x_3
x_4
x_6
$\begin{pmatrix} f_3(0) \\ f_3(1) \end{pmatrix}$
$\begin{pmatrix} \sum_{x_5} f_4(0,x_5) \times 1 \\ \sum_{x_5} f_4(1,x_5) \times 1 \end{pmatrix}$
f_3
f_4
$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$\sum_{x_2,x_3,x_4,x_5,x_6} f_1(x_1,x_2,x_3) f_2(x_1,x_4,x_6) f_3(x_4) f_4(x_4,x_5)$
x_5

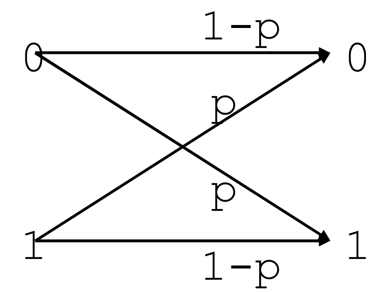
Belief Propagation – Final comments

Applications:

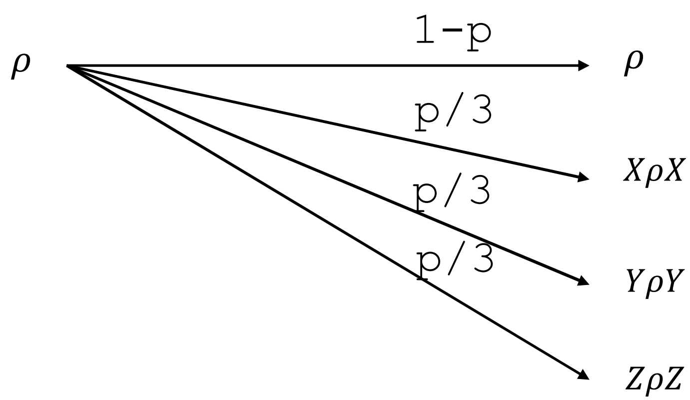
- In a tree: BP compute *exactly* the bit-flip probabilities.
- In a graph: BP compute an *approximation* of the bit-flip probabilities.
- For LDPC codes: For LDPC code with large girth (no short cycle), BP compute a *good approximation* of the bit-flip probabilities.
- Progressive edge growth: Algorithm producing LDPC codes with large girth.

The quantum decoding problem

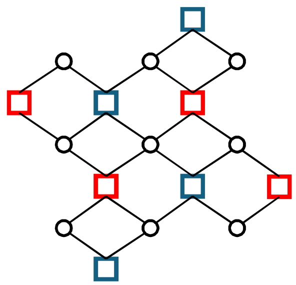
Depolarizing channel



Classical binary symmetric



First decoding attempt: BP



Classical case:

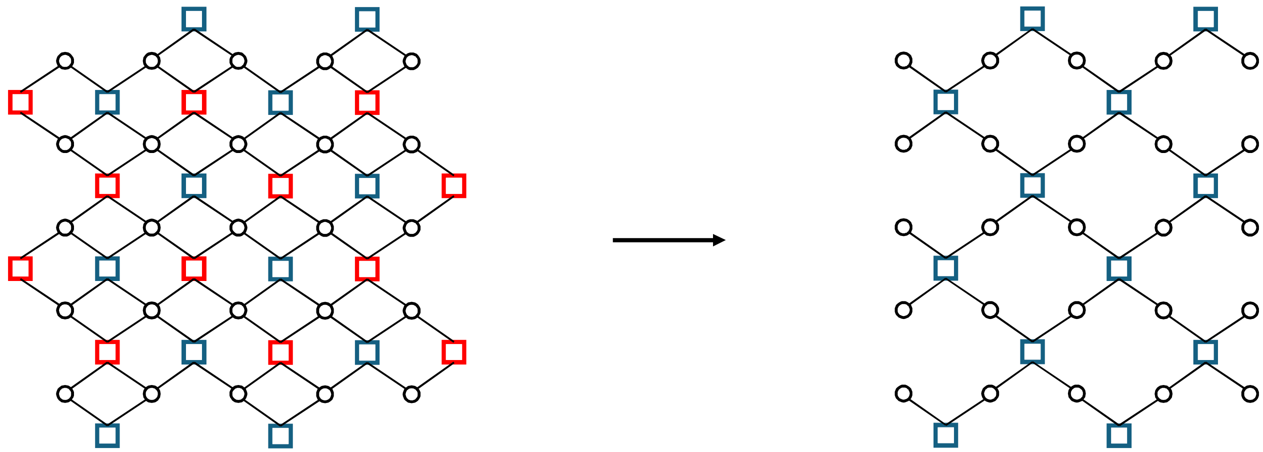
- Use BP to compute $P(x_i = 0 \mid y)$ and $P(x_i = 1 \mid y)$.
- Correct by selecting the most likely value $x_i = 0$ or 1.

Quantum case:

- Use BP to compute $P(E_i = I \mid s), P(E_i = X \mid s), P(E_i = Y \mid s)$ and $P(E_i = Z \mid s)$.
- BP does not perform well for two reasons:
  - The quantum Tanner graph contains many 4-cycles because of the commutation relations.
  - The event $E_i = I$ is not well defined up to a stabilizer.

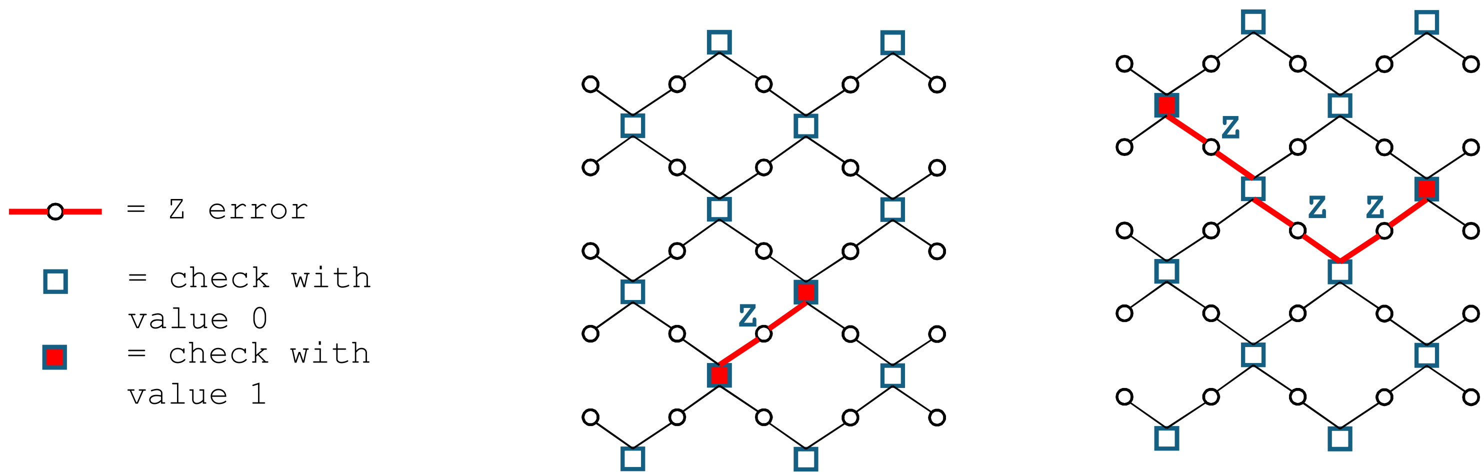
UF decoder for LDPC codes

Separation of X and Z errors



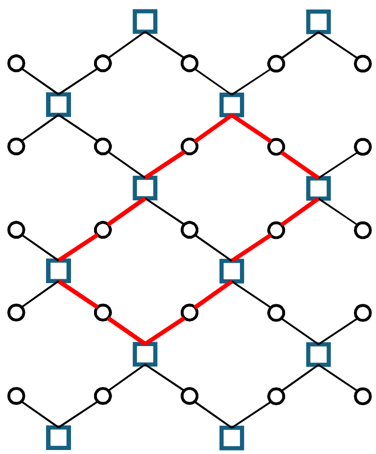
In what follows we focus on Z errors.

Error detection

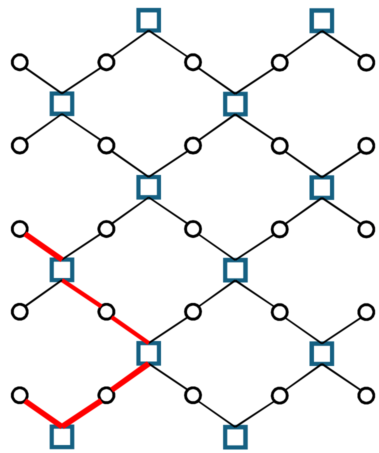


A chain of errors is detected at its endpoints

Trivial errors and logical errors

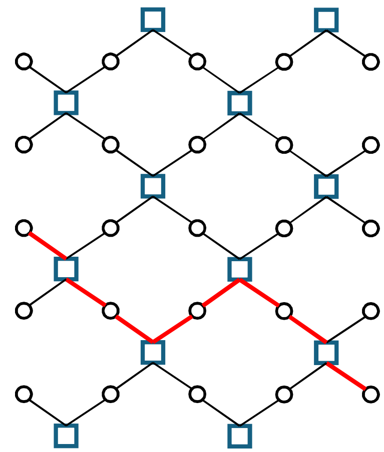


Loops are trivial errors.



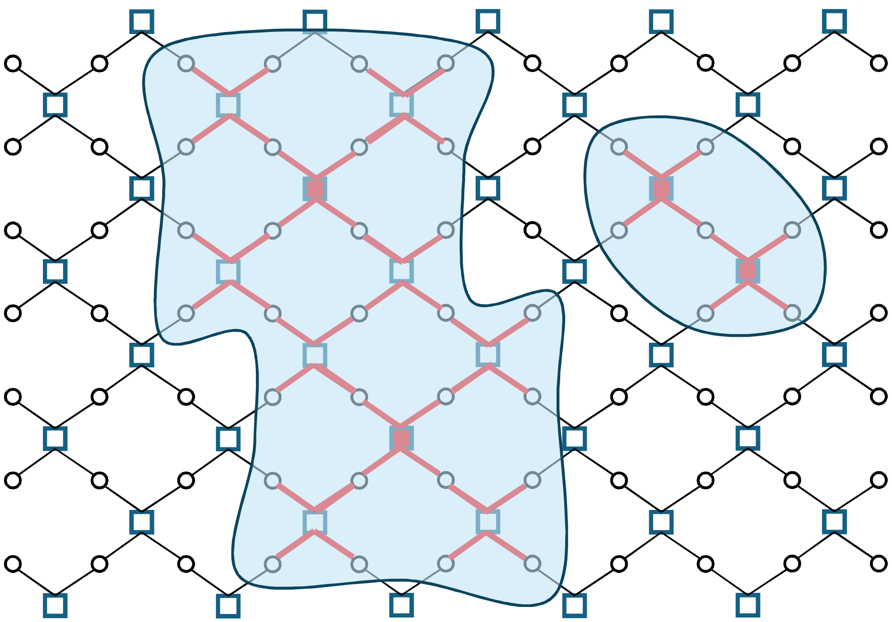
Paths connecting the same side are trivial errors.

No effect



Paths connecting the two opposite sides are logical errors.

Union-Find decoder



Delfosse, Nickerson (2017) [arxiv1709.06218]

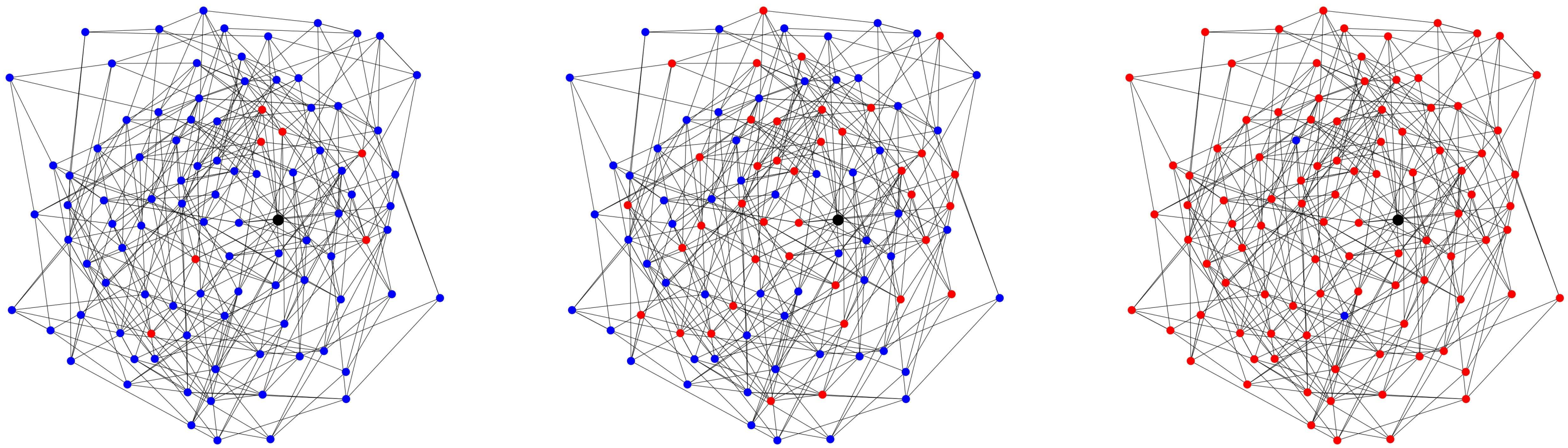
Union-Find decoder:

1. Grow clusters around check with value 1.
2. Stop growing a cluster when it becomes correctable.
3. Correct each cluster independently.

Remark:

We track the growing cluster using a Union-Find data structure which leads to an almost-linear complexity

The problem with LDPC codes



Clusters grow
too fast!

Union-Find decoder for QLDPC codes

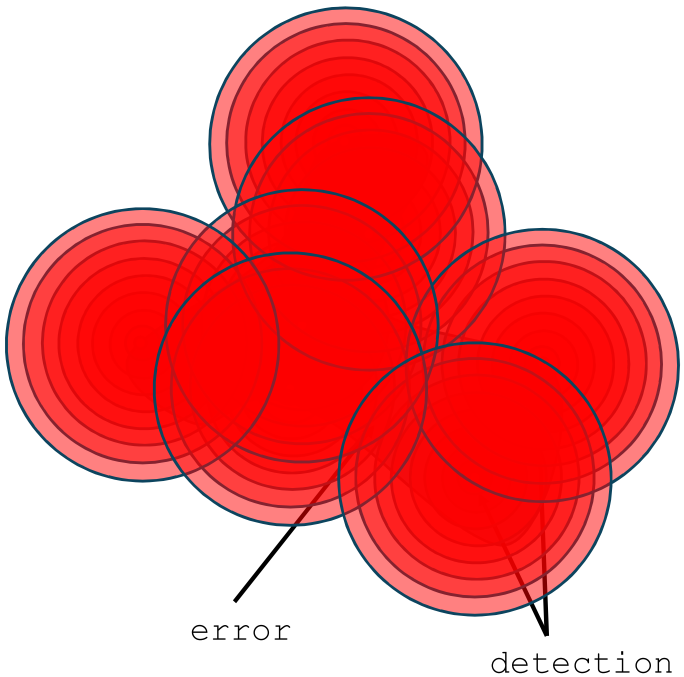
Input: A syndrome value $s_c = 0$ or 1 for each Z check.

Output: A correction for X errors.

1. Initialize $\mathcal{E}$ = set of Z checks with syndrome 1. ($\mathcal{E}$ = growing clusters)
2. While there exist an uncorrectable cluster in $\mathcal{E}$:
3. Grow all the clusters in $\mathcal{E}$.
4. Check if the clusters are correctable.
5. Find a correction inside each cluster of $\mathcal{E}$.
6. Return the product of the corrections of the clusters.

Delfosse, Londe, Beverland (2021)

The covering radius



Def. The *covering radius of a syndrome* s is the min radius such that the red balls cover an error with syndrome s.

Theorem (informal). If the covering radius of the syndrome is small, then the Union-Find decoder succeeds.

Applications: The UF decoders corrects n^a errors for:

- Quantum expander codes
- Hyperbolic codes[1] in dimension $D \geq 3$ [Guth, Lubotzky, (2013)]

Delfosse, Londe, Beverland (2021)

Union-Find decoder - Complexity

Complexity:

- For surface codes and color codes: $O(n\alpha(n))$.

  $\alpha(n)$ is the inverse of Ackermann's function.

- For LDPC codes: $O(n^4)$.

Research question: Improve the complexity of the UF decoder for LDPC codes.

Research question: Design a UF decoder that corrects (d-1)/2 errors for toric codes in all dim $D \geq 3$.
See section 7 of arxiv:2103.08049.

BP-OSD

BP+OSD$_0$ decoder

error = (0 0 0 1 0 0 0)

H = $\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$ => Syndrome s = $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

(.12 .17 .05 .31 .32 .06 .01)

H′ = $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

=> x′ = (0 1 0)

=> x = (0 0 0 1 0 0 0)

Goal: *Find a likely error x with H x = s*

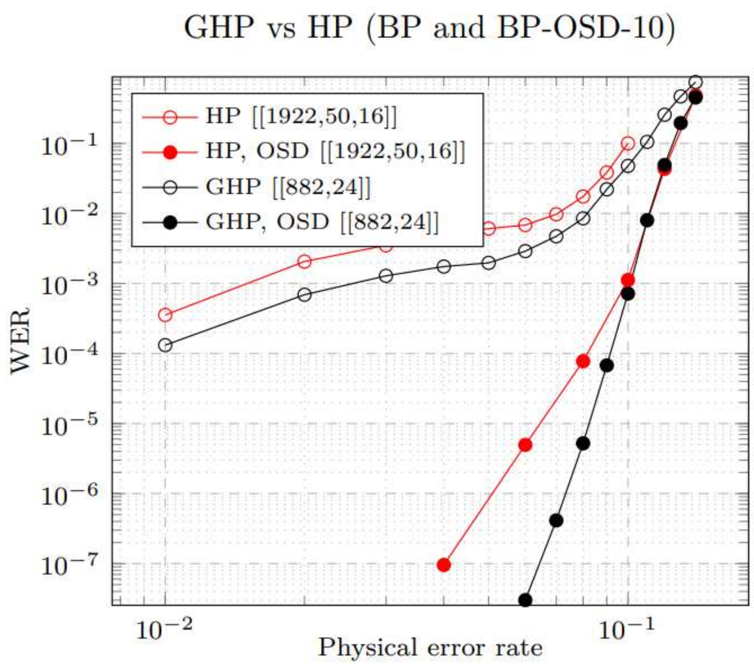
1. Estimate all bit flip probabilities using BP.
2. Select a basis H′ of columns made with highest probability columns.
3. Solve H′ x′ = s.
4. Construct x from x′.

Panteleev, Kalachev - arXiv:1904.02703.

BP+OSD$_0$ decoder - Complexity

Complexity:

- with OSD$_0$: $O(n^3)$.
- with OSD$_w$: $O(2^w n^3)$.



Panteleev, Kalachev - arXiv:1904.02703

Conclusion: Which decoder should we use?

- PB: Does not work well with quantum LDPC because of short cycles
- UF for QLDPC: corrects a poly number of errors but may reduce the distance
- BP-OSD: Heuristic but seems to behave well in simulation.