A Renormalisation Decoder for Kitaev's Toric Code

Wouter Rozendaal and Gilles Zémor

QuDATA, LaBRI - 25 January 2024

Wouter Rozendaal and Gilles Zémor. *Analysis of the Error-Correcting Radius of a Renormalisation Decoder for Kitaev's Toric Code*. 2023. arXiv: 2309.12165 [quant-ph]

Constructing Kitaev's Toric Code

Kitaev's code:

- Qubits indexed by edges of a toric tiling

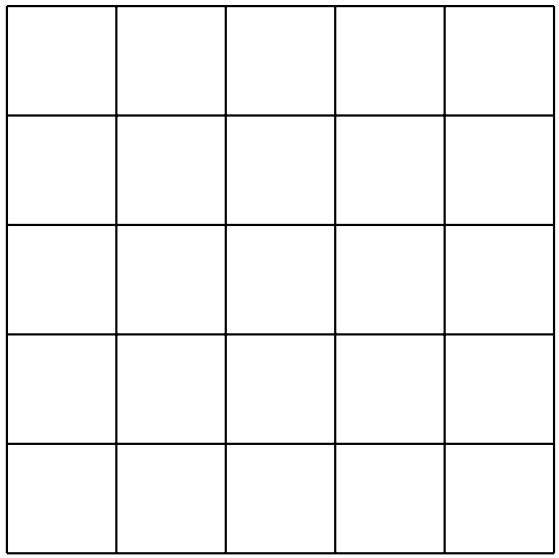


Figure: Tiling of a 2-dimensional torus

Constructing Kitaev's Toric Code

Kitaev's code:

- Qubits indexed by edges of a toric tiling
- CSS stabiliser code $(\mathbf{H}_X, \mathbf{H}_Z)$

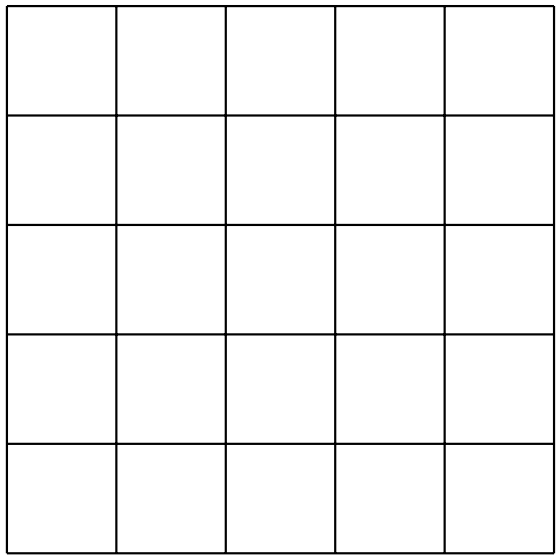


Figure: Tiling of a 2-dimensional torus

Constructing Kitaev's Toric Code

Kitaev's code:

- Qubits indexed by edges of a toric tiling
- CSS stabiliser code $(\mathbf{H}_X, \mathbf{H}_Z)$
  - Rows of $\mathbf{H}_X$

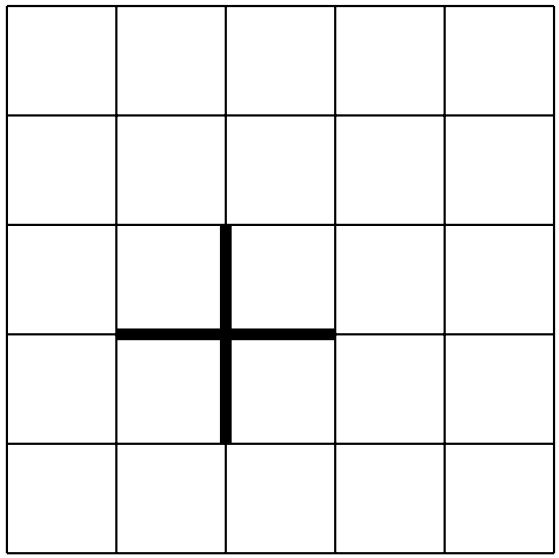


Figure: Elementary cocycle

Constructing Kitaev's Toric Code

Kitaev's code:

- Qubits indexed by edges of a toric tiling
- CSS stabiliser code $(\mathbf{H}_X, \mathbf{H}_Z)$
  - Rows of $\mathbf{H}_Z$

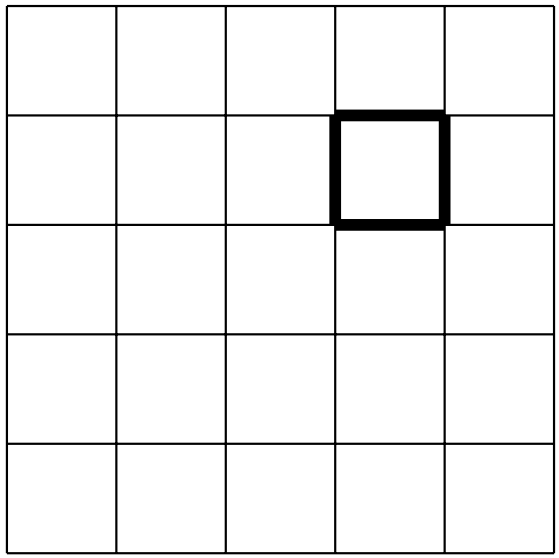


Figure: Elementary cycle

Constructing Kitaev's Toric Code

Kitaev's code:

- Qubits indexed by edges of a toric tiling
- CSS stabiliser code $(\mathbf{H}_X, \mathbf{H}_Z)$
  - Orthogonality condition: $\mathbf{H}_X \mathbf{H}_Z^\intercal = \mathbf{0}$

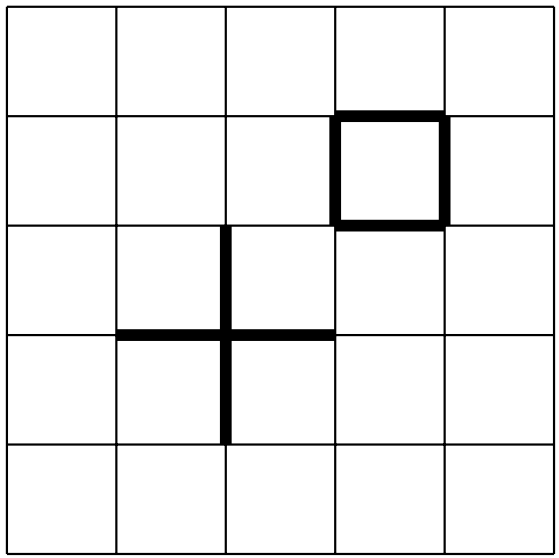


Figure: Elementary cocycles and cycles meet in an even number of edges

Parameters of Kitaev's Toric Code

- Toric tiling of size $m \times m \to m^2$ vertices, $2m^2$ edges

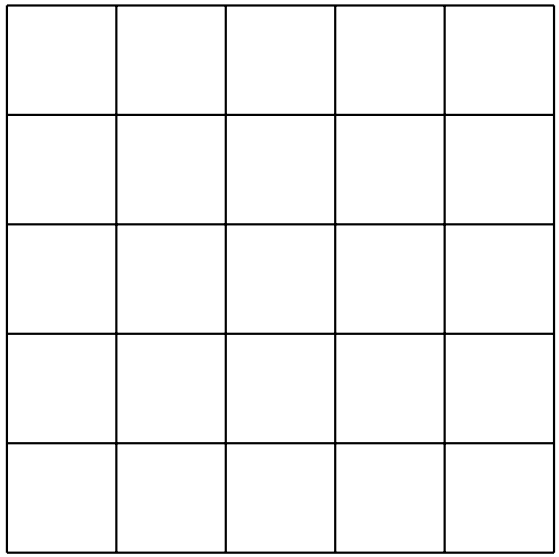


Figure: Tiling of a 2-dimensional torus

Parameters of Kitaev's Toric Code

- Toric tiling of size $m \times m \to m^2$ vertices, $2m^2$ edges
- Parameters of the toric code:

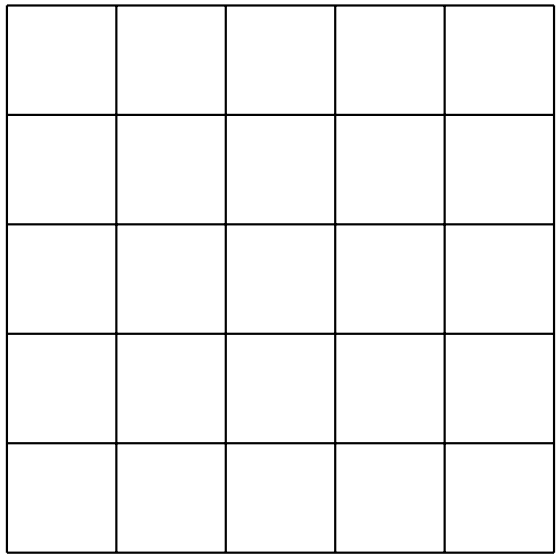


Figure: Tiling of a 2-dimensional torus

Parameters of Kitaev's Toric Code

- Toric tiling of size $m \times m \to m^2$ vertices, $2m^2$ edges
- Parameters of the toric code:
  - Length: $n = 2m^2$

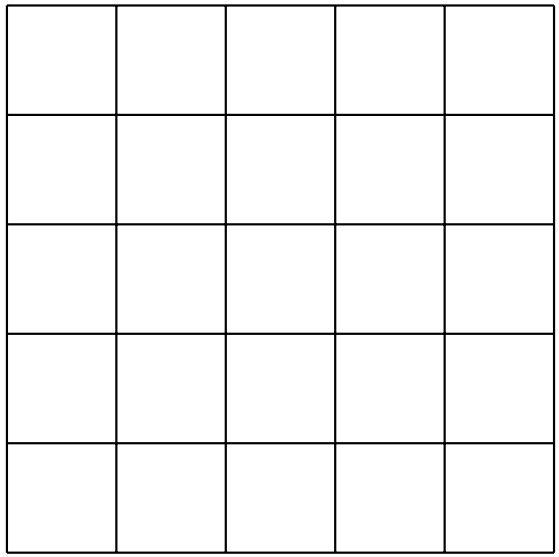


Figure: Tiling of a 2-dimensional torus

Parameters of Kitaev's Toric Code

- Toric tiling of size $m \times m \to m^2$ vertices, $2m^2$ edges
- Parameters of the toric code:
  - Length: $n = 2m^2$
  - Dimension: $n - \operatorname{rank} \mathbf{H}_X - \operatorname{rank} \mathbf{H}_Z = 2$

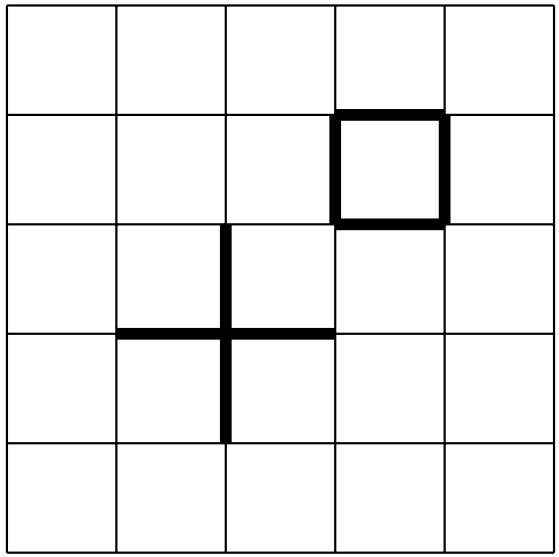


Figure: Generators of $\mathbf{H}_X$ and $\mathbf{H}_Z$

Parameters of Kitaev's Toric Code

- Toric tiling of size $m \times m \to m^2$ vertices, $2m^2$ edges
- Parameters of the toric code:
  - Length: $n = 2m^2$
  - Dimension: $n - \operatorname{rank} \mathbf{H}_X - \operatorname{rank} \mathbf{H}_Z = 2$
  - Minimum distance: $d = m = \sqrt{n/2}$

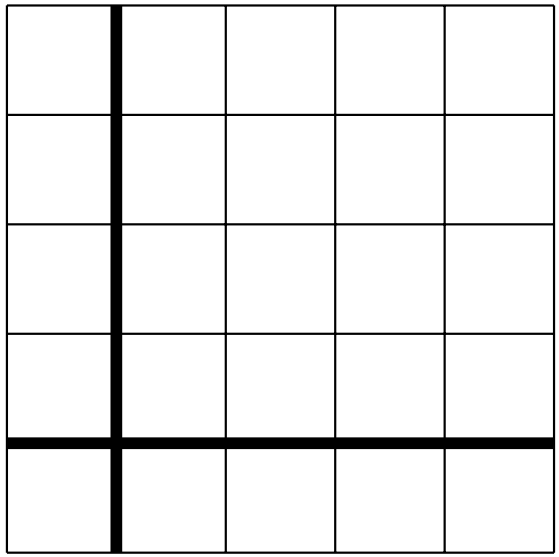


Figure: Non trivial cycles of smallest weight

Decoding Kitaev's Toric Code

Z-error pattern $\rightarrow$ error vector $\mathbf{e} \in \mathbb{F}_2^n$

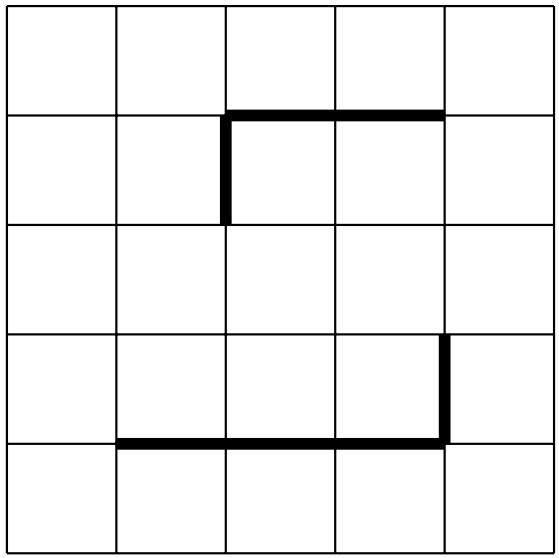


Figure: Error vector $\mathbf{e}$

Decoding Kitaev's Toric Code

Z-error pattern $\rightarrow$ error vector $\mathbf{e} \in \mathbb{F}_2^n$

- Input: syndrome measurement $\sigma(\mathbf{e}) := \mathbf{H}_X \mathbf{e}^\intercal$

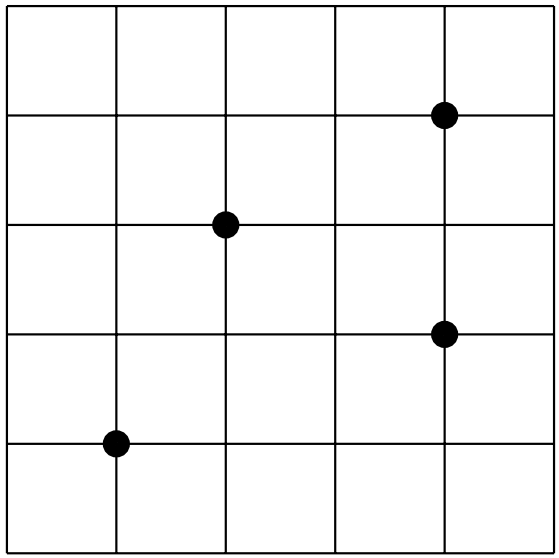


Figure: Syndrome $\sigma(\mathbf{e})$

Decoding Kitaev's Toric Code

Z-error pattern $\rightarrow$ error vector $\mathbf{e} \in \mathbb{F}_2^n$

- Input: syndrome measurement $\sigma(\mathbf{e}) := \mathbf{H}_X \mathbf{e}^\intercal$
- Output: $\hat{\mathbf{e}} \in \mathbb{F}_2^n$ such that $\sigma(\hat{\mathbf{e}}) = \sigma(\mathbf{e})$

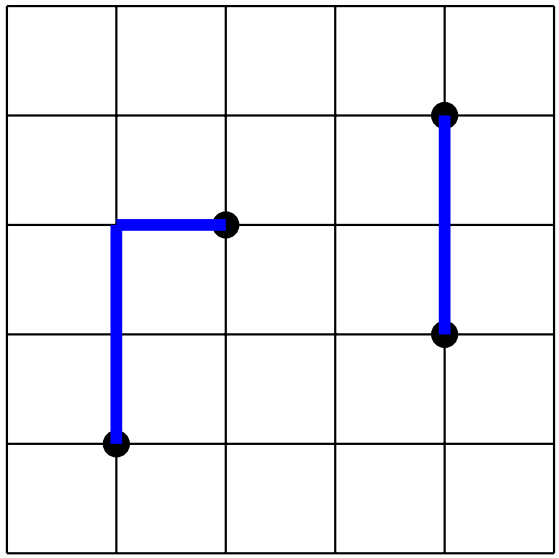


Figure: Output vector $\hat{\mathbf{e}}$ and its syndrome

Decoding Kitaev's Toric Code

Z-error pattern $\rightarrow$ error vector $\mathbf{e} \in \mathbb{F}_2^n$

- Input: syndrome measurement $\sigma(\mathbf{e}) := \mathbf{H}_X \mathbf{e}^\intercal$
- Output: $\hat{\mathbf{e}} \in \mathbb{F}_2^n$ such that $\sigma(\hat{\mathbf{e}}) = \sigma(\mathbf{e})$
- Successful decoding: $\mathbf{e} + \hat{\mathbf{e}}$ is in the row space of $\mathbf{H}_Z$

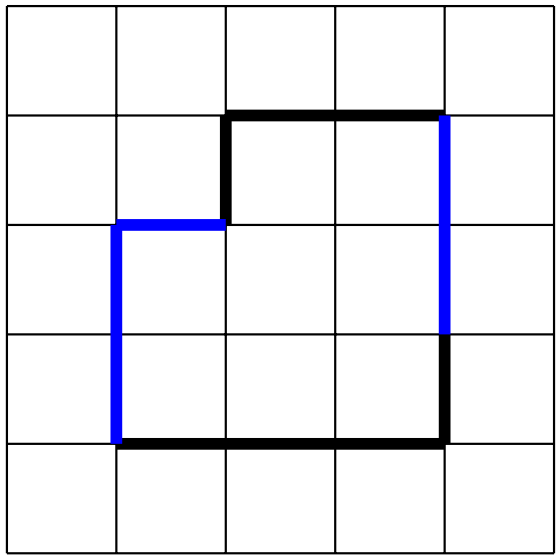


Figure: $\mathbf{e} + \hat{\mathbf{e}}$ is a trivial cycle

Decoding Kitaev's Toric Code

- Errors accumulate while a quantum algorithm is running
- Need for sub-linear decoding algorithms

Decoding Kitaev's Toric Code

- Errors accumulate while a quantum algorithm is running
- Need for sub-linear decoding algorithms

Renormalisation decoders!

Decoding Kitaev's Toric Code

- Errors accumulate while a quantum algorithm is running
- Need for sub-linear decoding algorithms

Renormalisation decoders!

- Renormalisation idea: Duclos-Cianci and Poulin (2010)
- Time-complexity in $O(n \log_2 n)$ and parallelisable to $O(\log_2 n)$
- High threshold values for bit-flip and depolarisation channels

Decoding Kitaev's Toric Code

- Errors accumulate while a quantum algorithm is running
- Need for sub-linear decoding algorithms

Renormalisation decoders!

- Renormalisation idea: Duclos-Cianci and Poulin (2010)
- Time-complexity in $O(n \log_2 n)$ and parallelisable to $O(\log_2 n)$
- High threshold values for bit-flip and depolarisation channels

Worst case behaviour?

Decoding Kitaev's Toric Code

- Errors accumulate while a quantum algorithm is running
- Need for sub-linear decoding algorithms

Renormalisation decoders!

- Renormalisation idea: Duclos-Cianci and Poulin (2010)
- Time-complexity in $O(n \log_2 n)$ and parallelisable to $O(\log_2 n)$
- High threshold values for bit-flip and depolarisation channels

Worst case behaviour?

What is the smallest weight of an error pattern for which decoding fails?

Example of Renormalisation Decoding

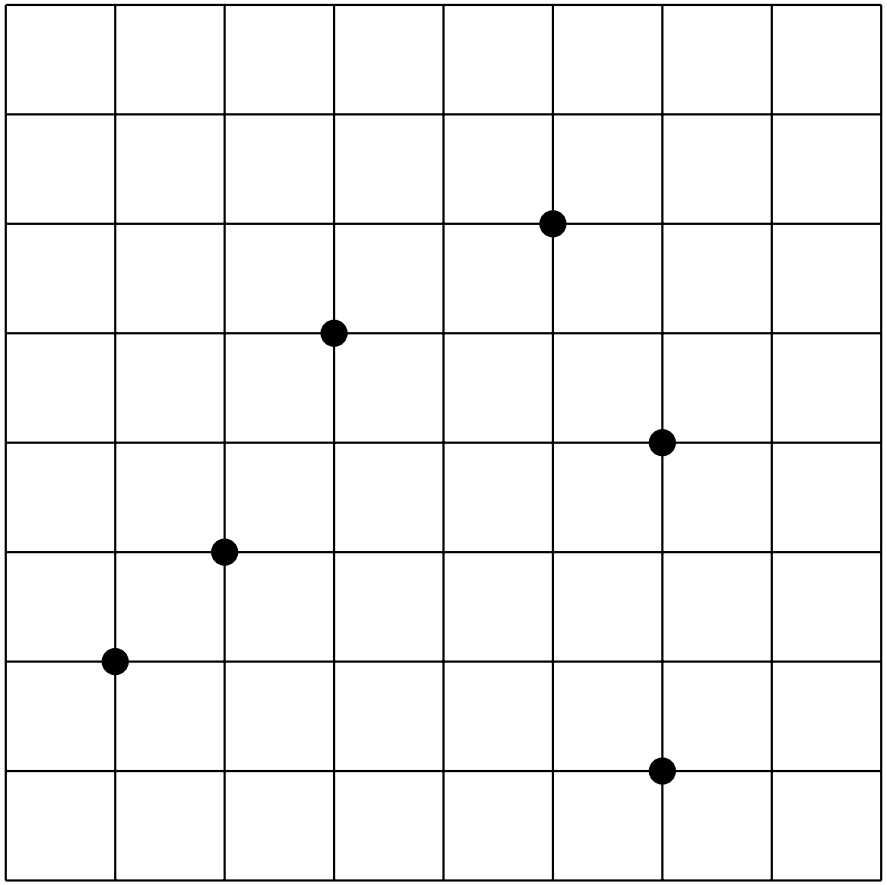


Figure: Decoding problem on a toric tiling.
Input: syndrome of an error vector $\mathbf{e}$

Example of Renormalisation Decoding

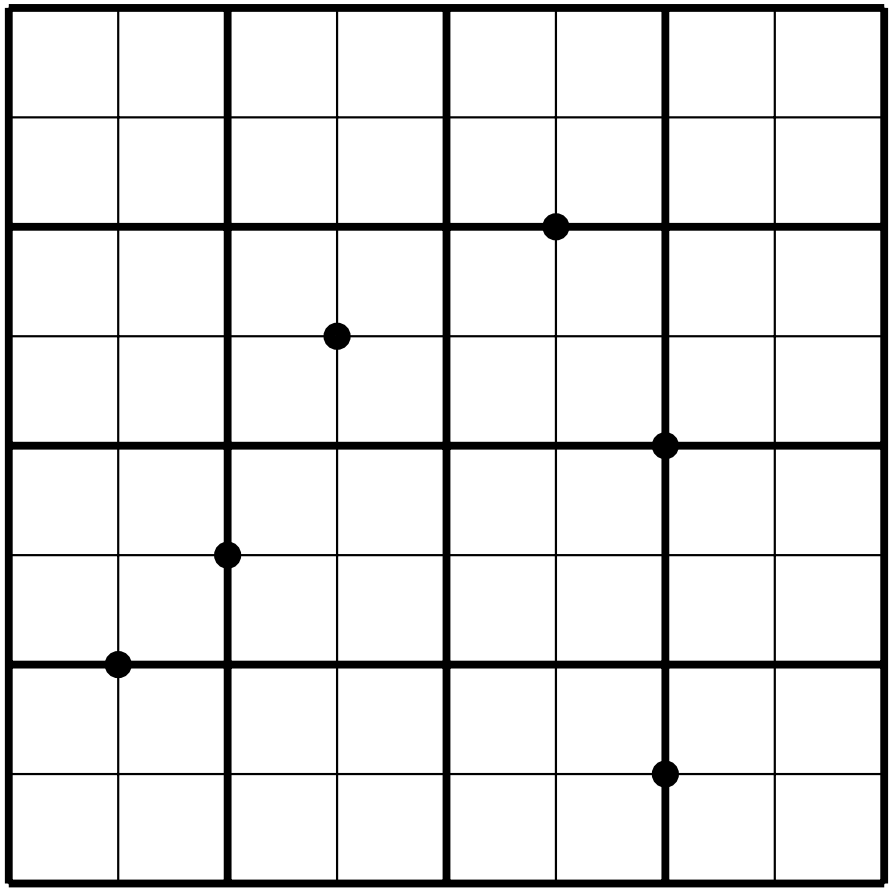


Figure: Reduction procedure:
Move the syndrome vertices to the next subtiling

Example of Renormalisation Decoding

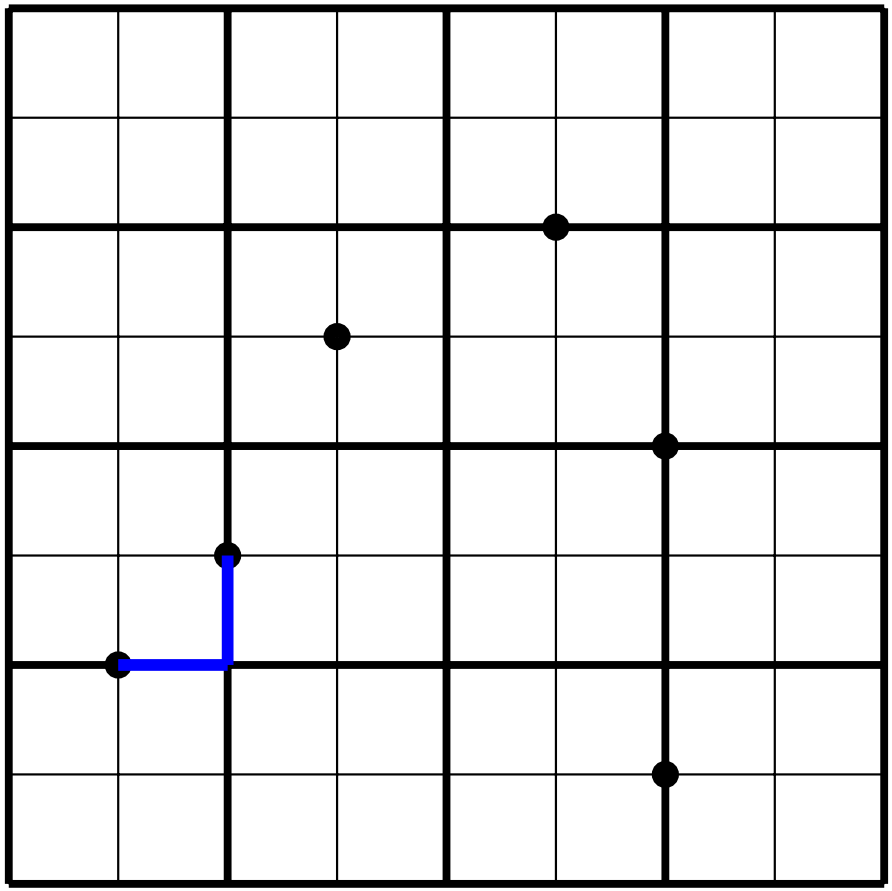


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

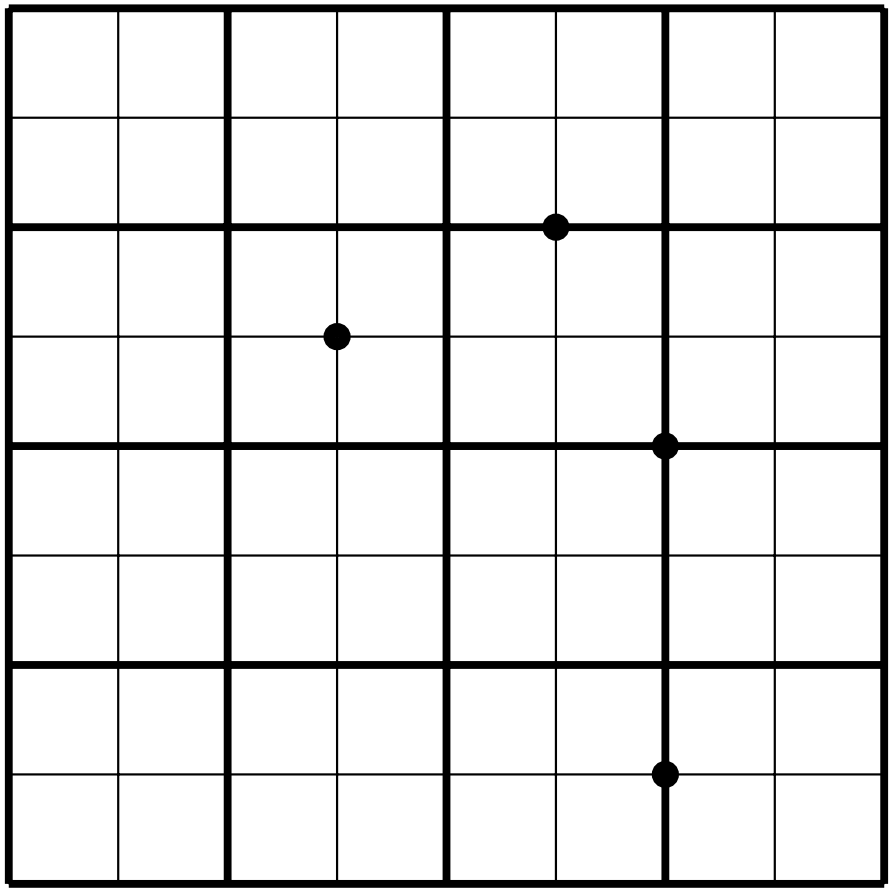


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

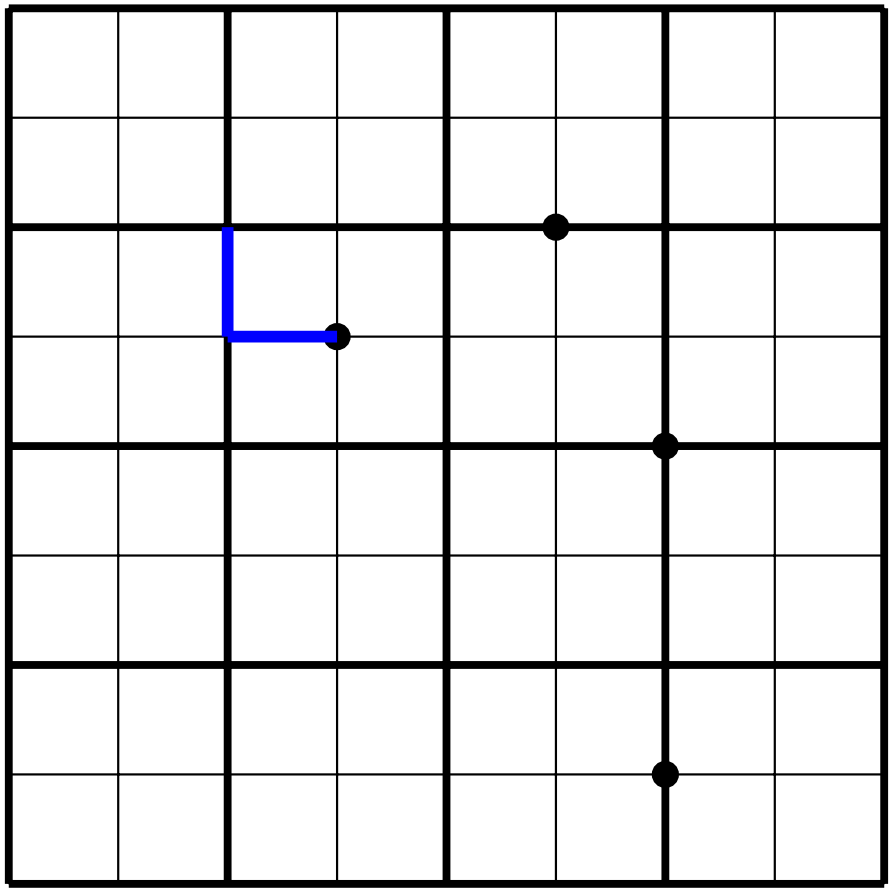


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

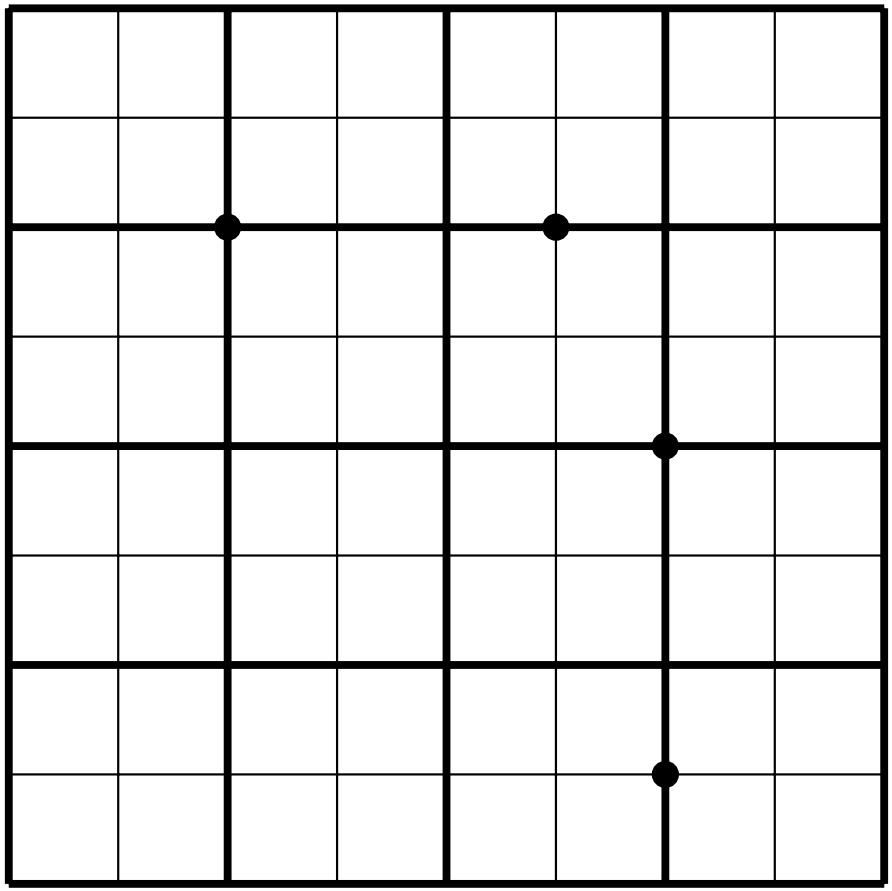


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

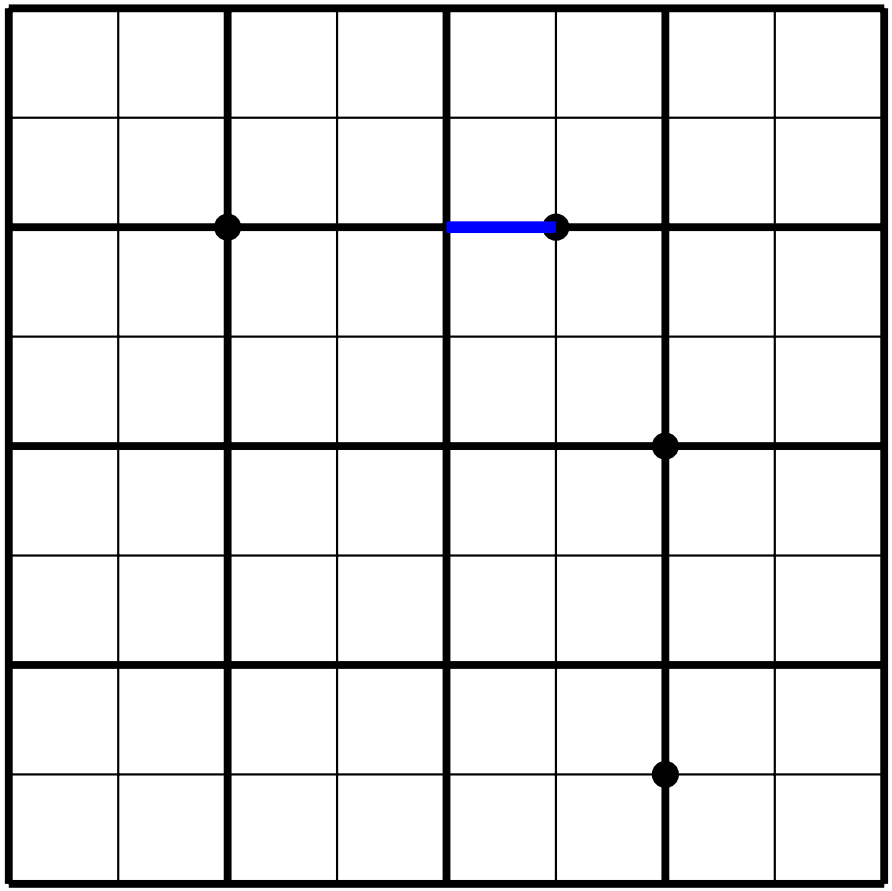


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

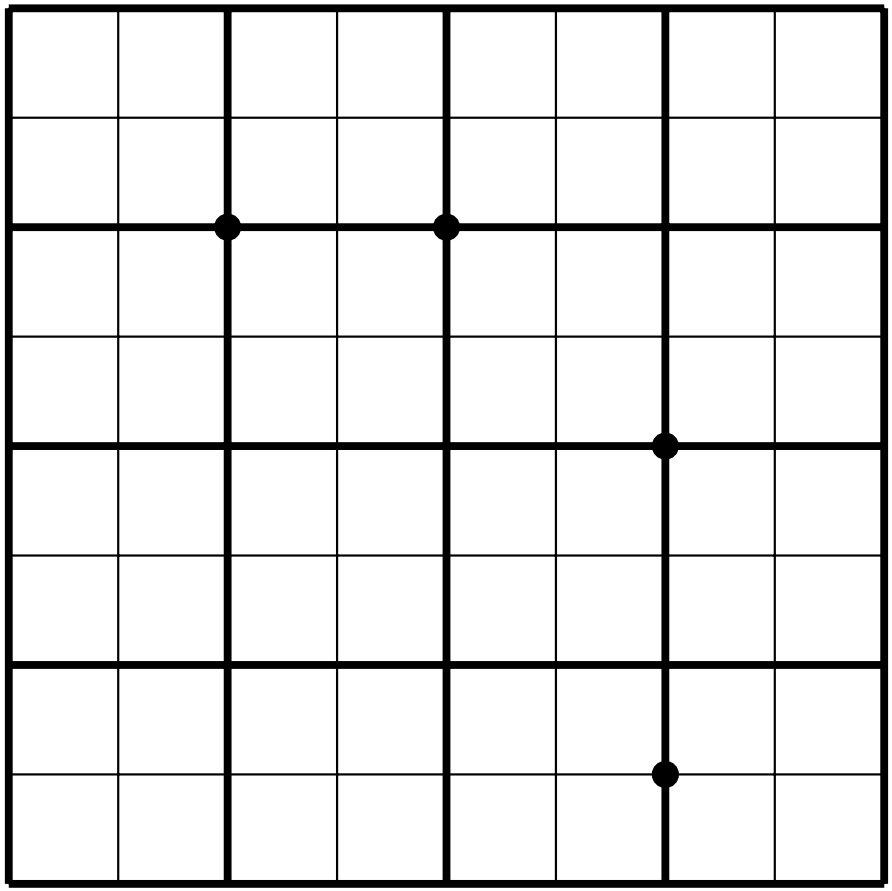


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

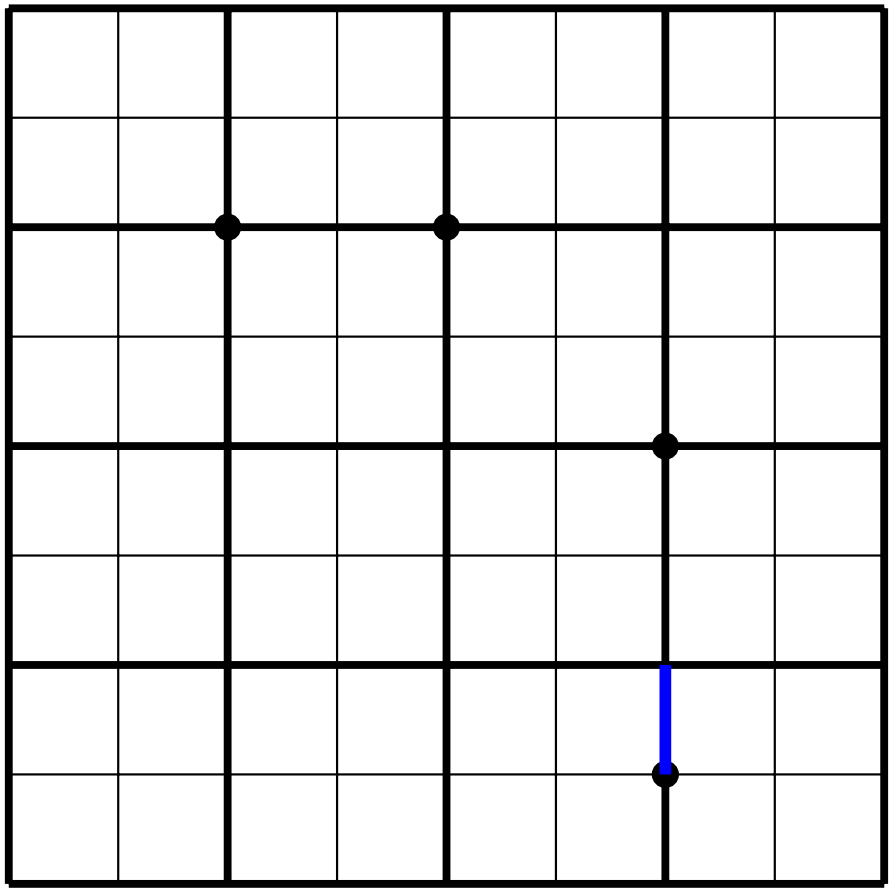


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

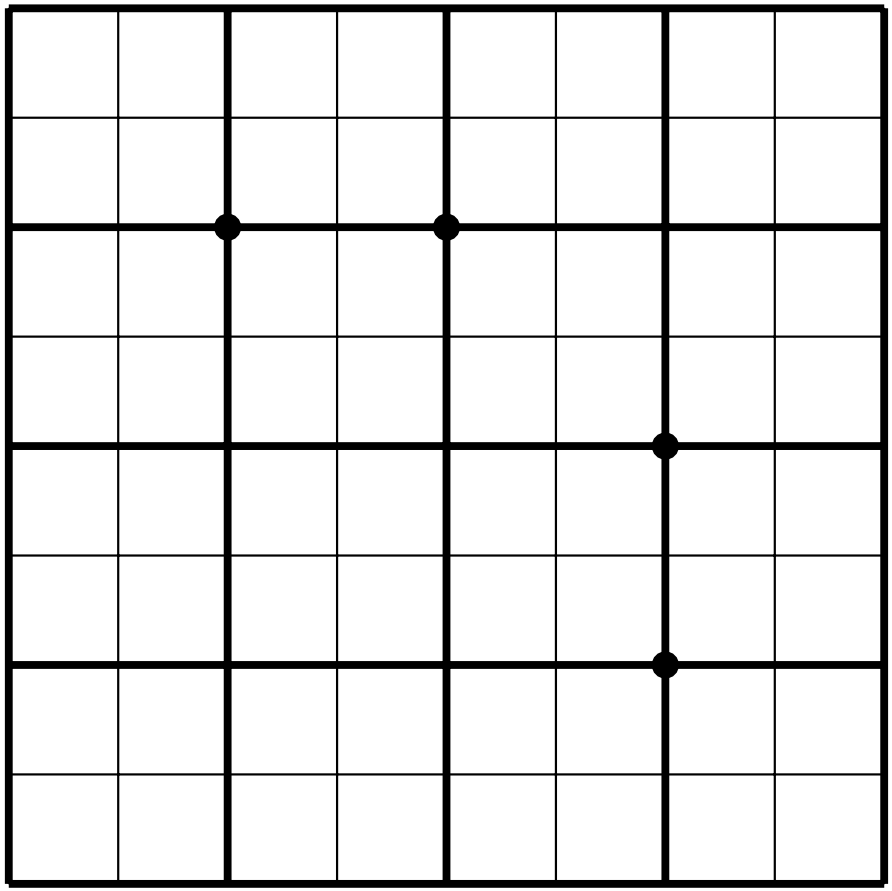


Figure: Locally shift remaining syndrome vertices

Example of Renormalisation Decoding

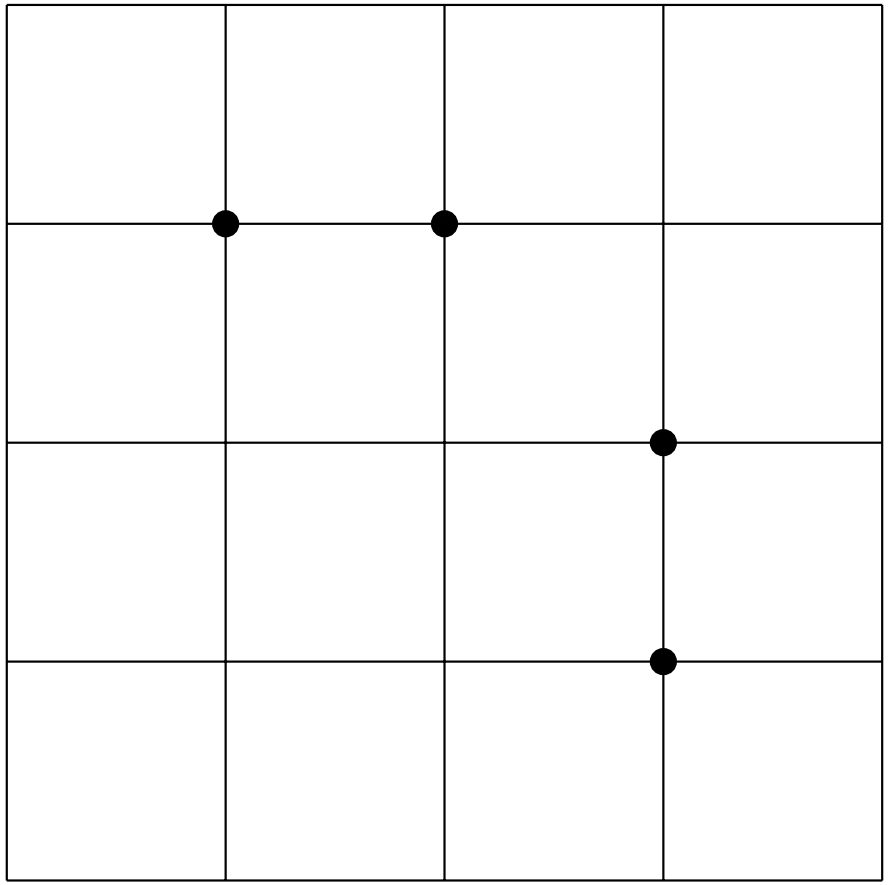


Figure: Decoding problem on the subtiling.
Input: syndrome of the vector $\mathbf{e} + \hat{\mathbf{e}}$

Example of Renormalisation Decoding

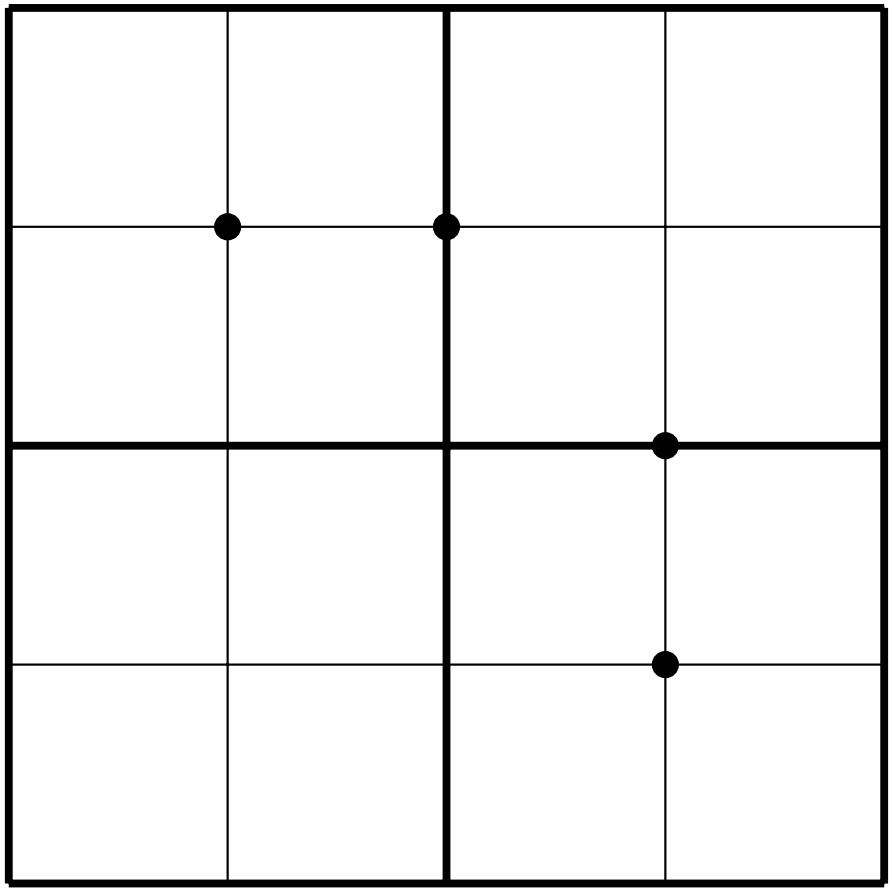


Figure: Reduction procedure:
Move the syndrome vertices to the next subtiling

Example of Renormalisation Decoding

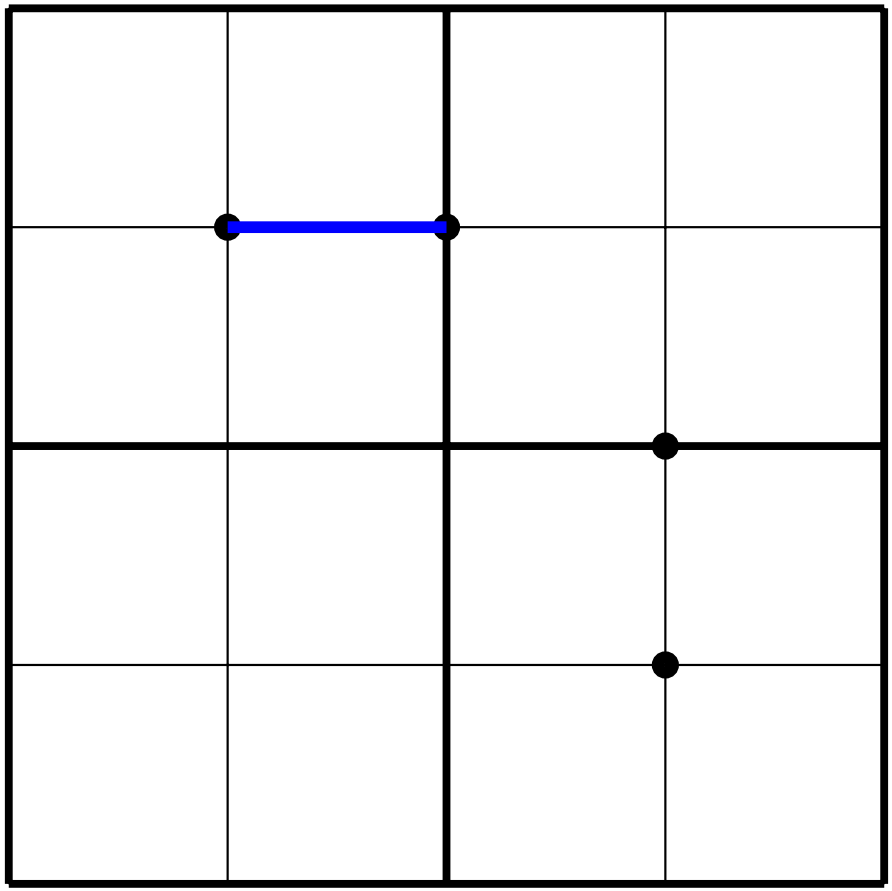


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

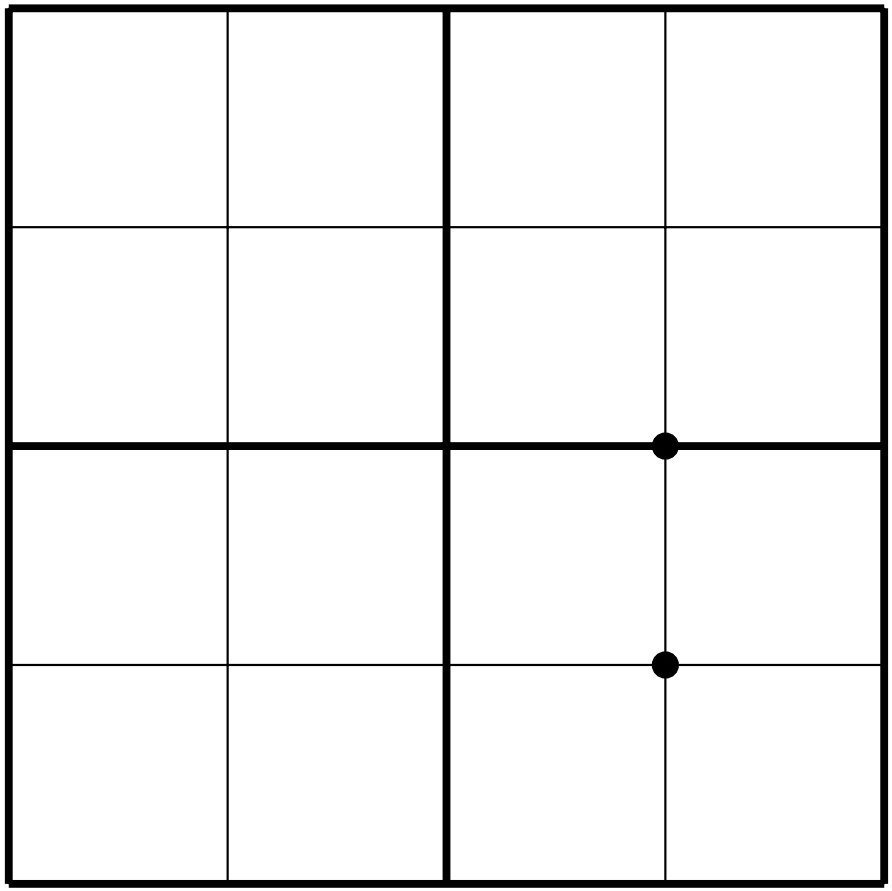


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

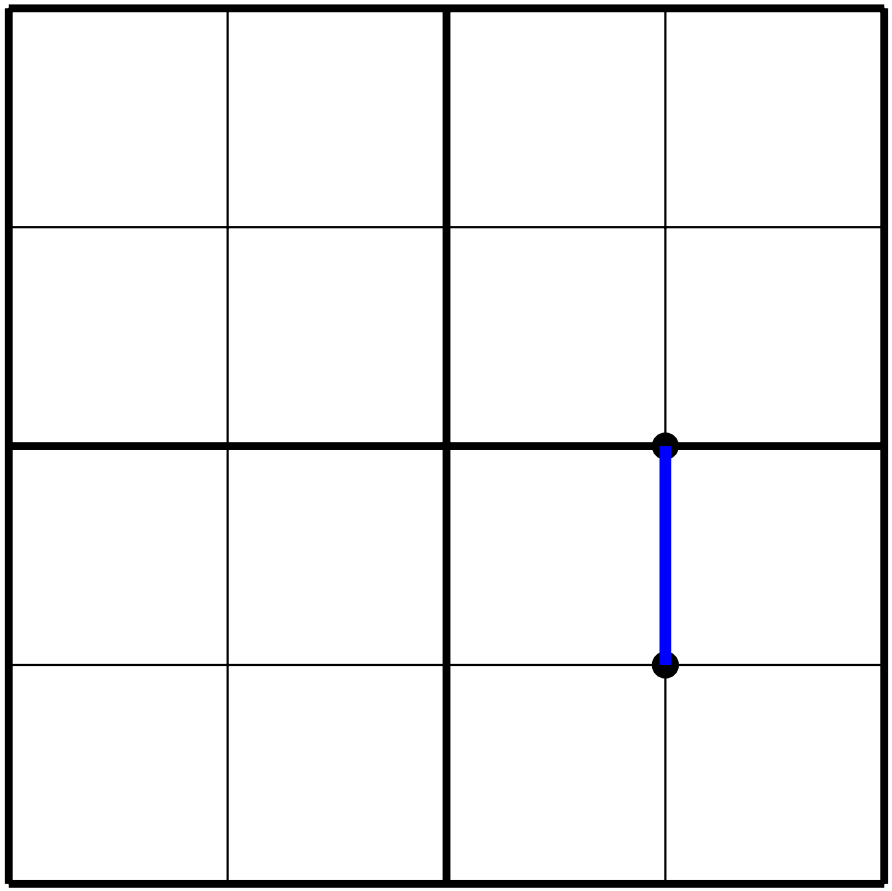


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

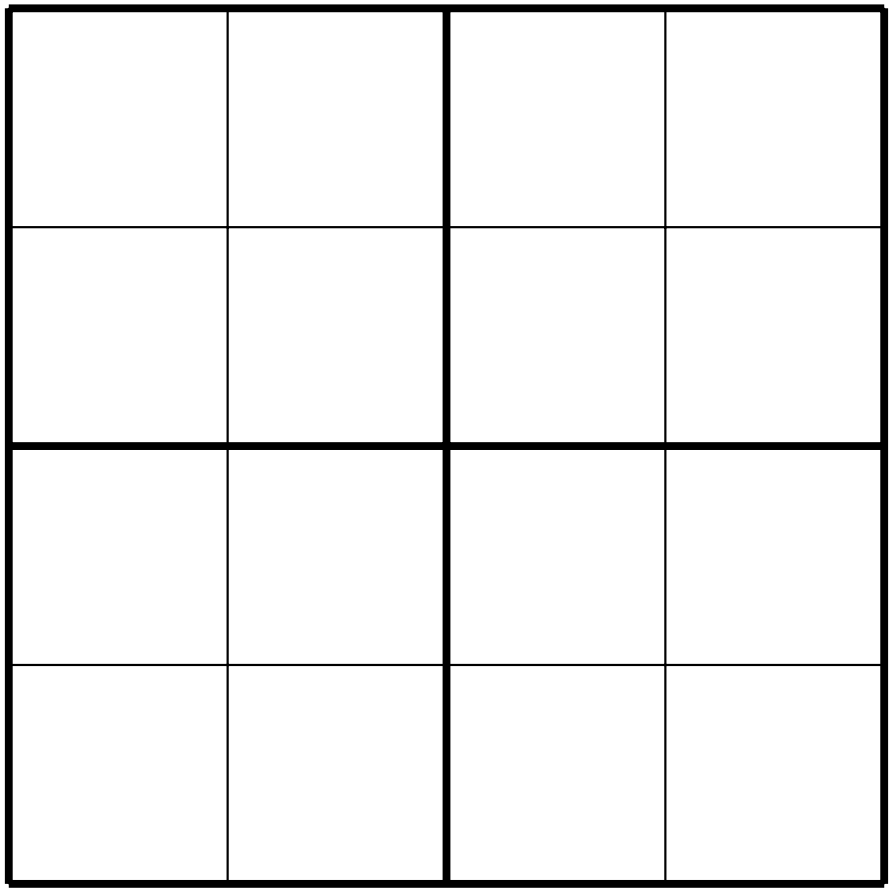


Figure: Locally pair-up syndrome vertices

Example of Renormalisation Decoding

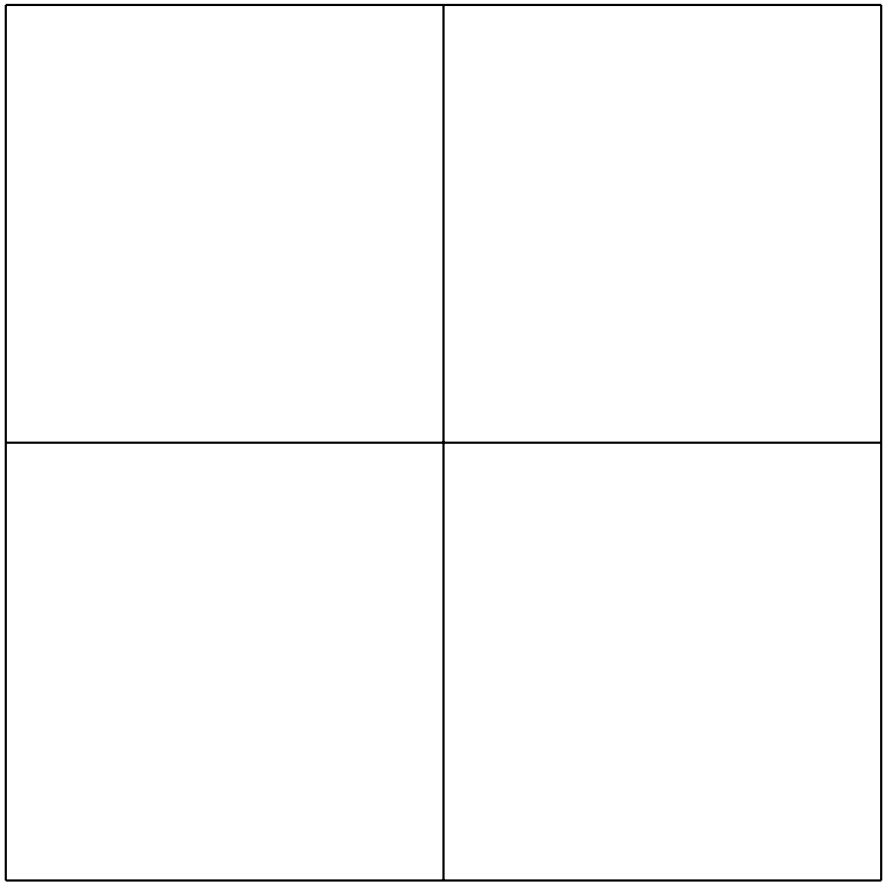


Figure: Decoding finishes.
All syndromes vertices have been paired-up

Example of Renormalisation Decoding

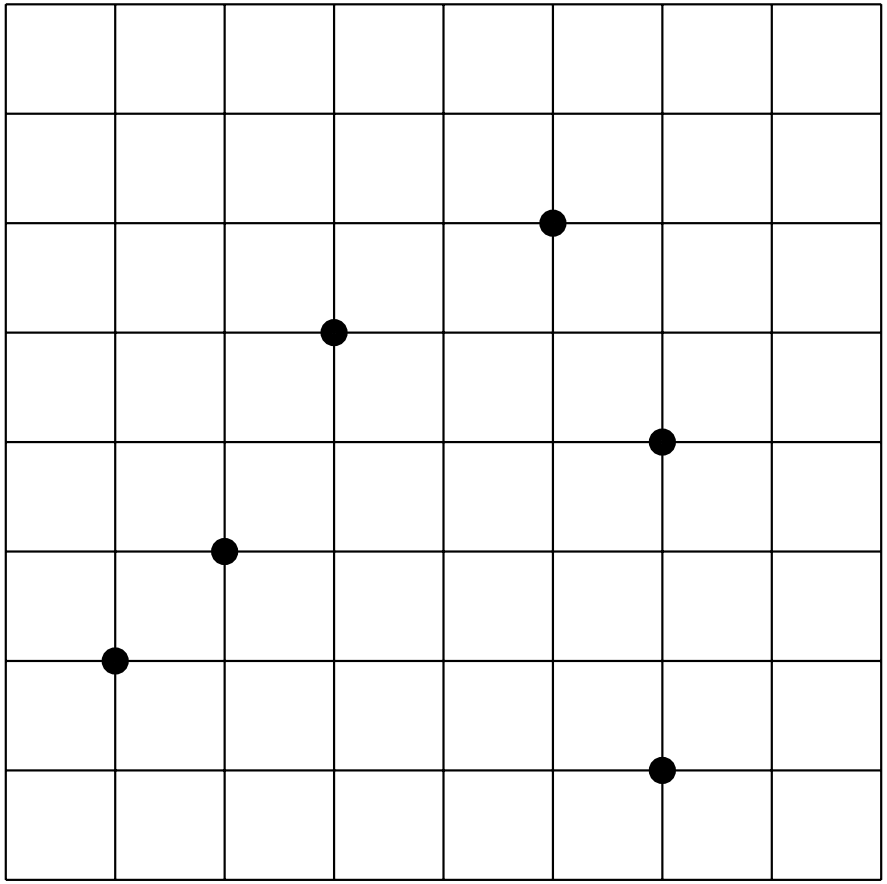


Figure: Syndrome of the error vector $\mathbf{e}$

Example of Renormalisation Decoding

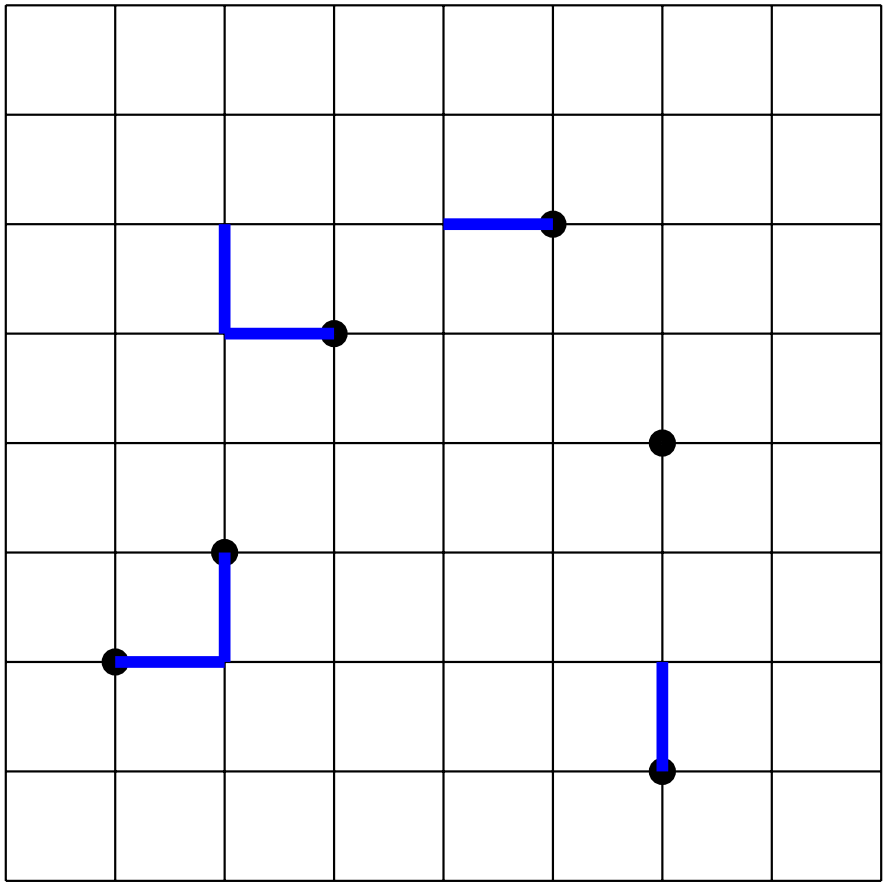


Figure: Output vector $\hat{\mathbf{e}}$ after the 1^{st} reduction step

Example of Renormalisation Decoding

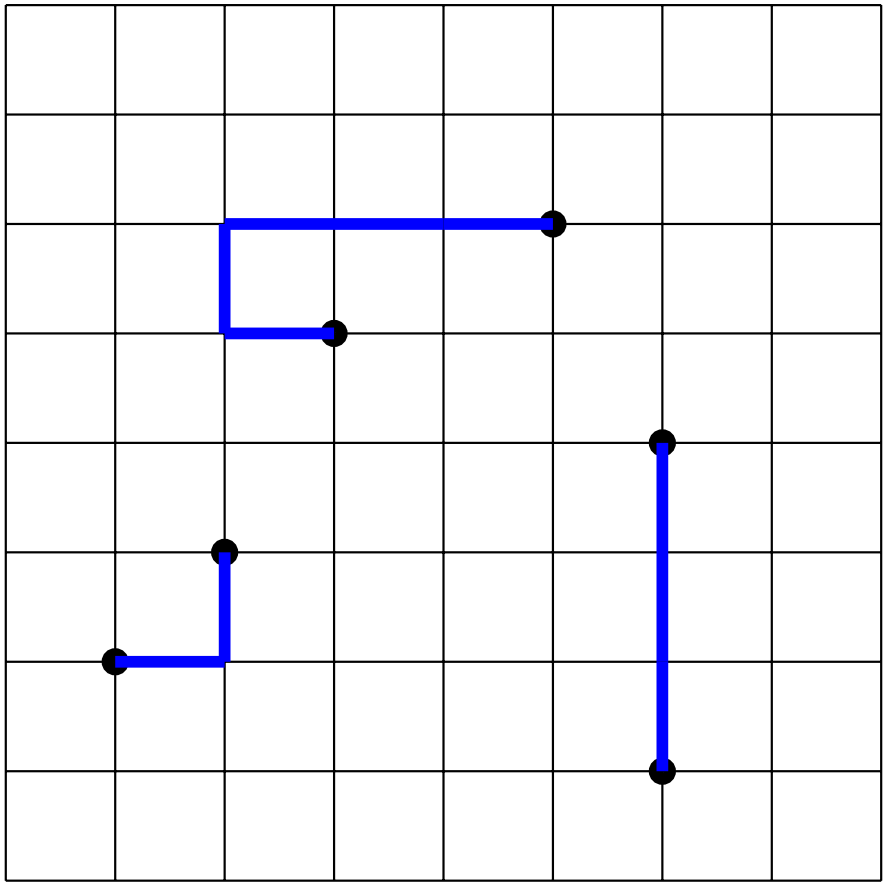


Figure: Output vector $\hat{\mathbf{e}}$ after the 2^{nd} reduction step

A Hard-Decision and Deterministic Reduction Procedure

Figure: 1: Locally pair up diagonally opposed syndrome vertices in D cells

A Hard-Decision and Deterministic Reduction Procedure

Figure: 1: Locally pair up diagonally opposed syndrome vertices in D cells

Figure: 2: Locally pair up neighbouring syndrome vertices in B and C cells

A Hard-Decision and Deterministic Reduction Procedure

Figure: 1: Locally pair up diagonally opposed syndrome vertices in D cells

Figure: 2: Locally pair up neighbouring syndrome vertices in B and C cells

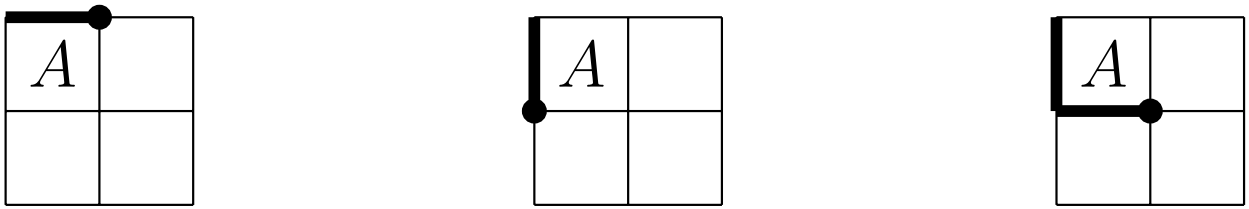
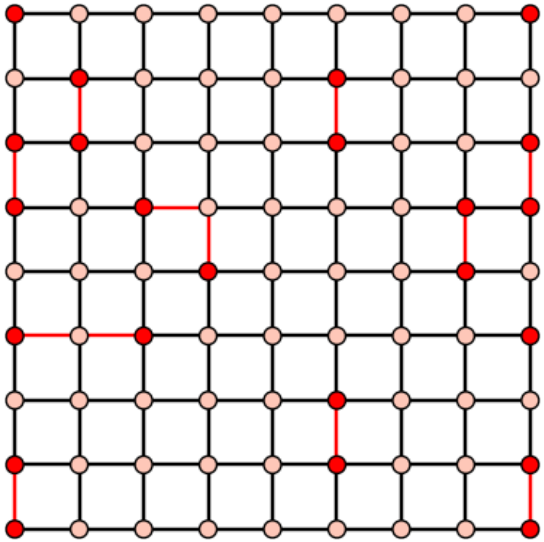
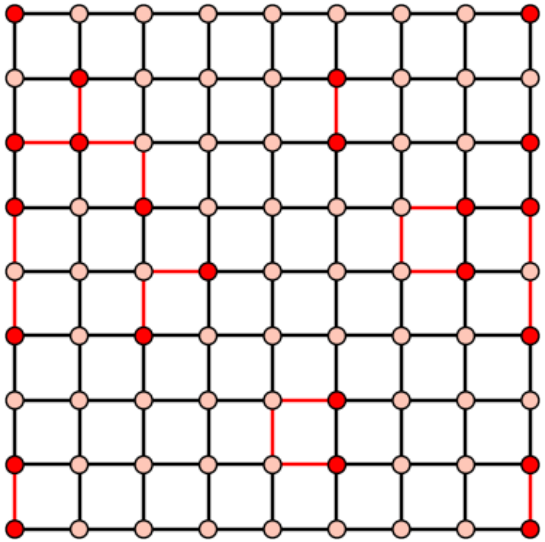


Figure: 3: Shift remaining syndrome vertices in A cells to their top-left corner

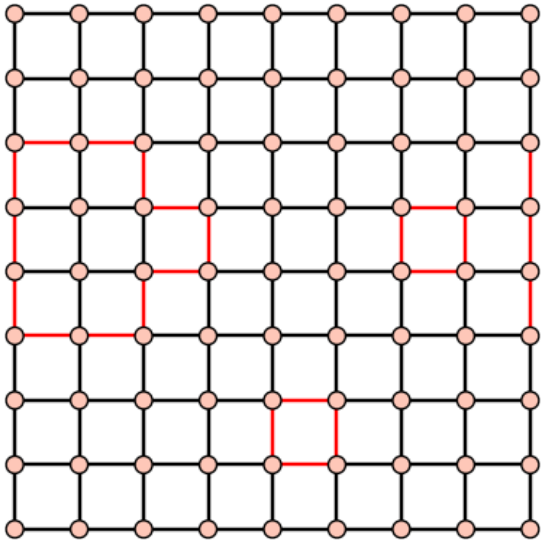
Analysis over the Bit-Flip Channel



(a) Randomly generated error pattern.



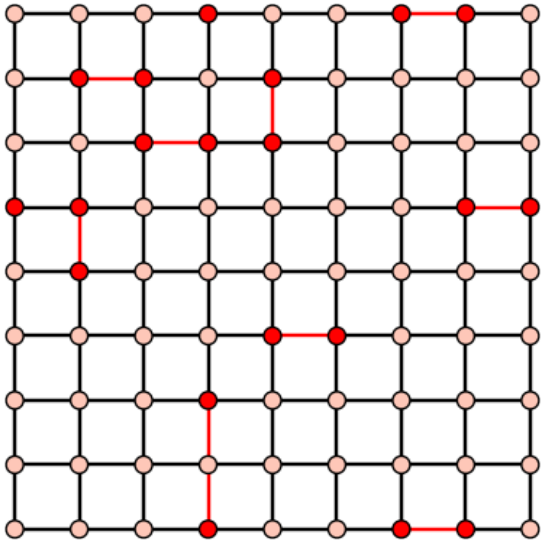
(b) Pattern proposed by the decoder.



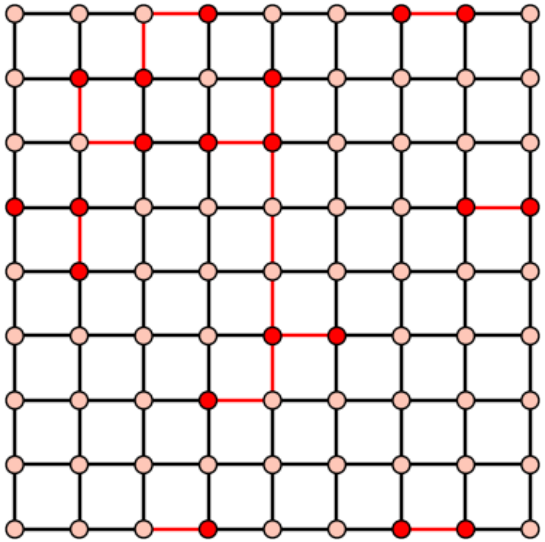
(c) Sum of the two patterns.

Figure: Successful decoding cycle.

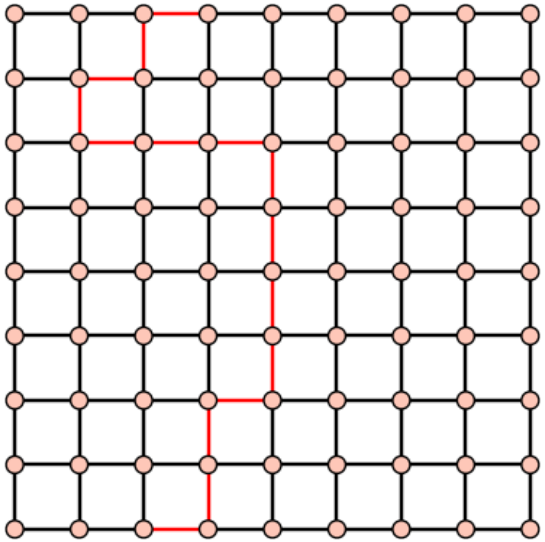
Analysis over the Bit-Flip Channel



(a) Randomly generated error pattern.



(b) Pattern proposed by the decoder.



(c) Sum of the two patterns.

Figure: Unsuccessful decoding cycle.

Analysis over the Bit-Flip Channel

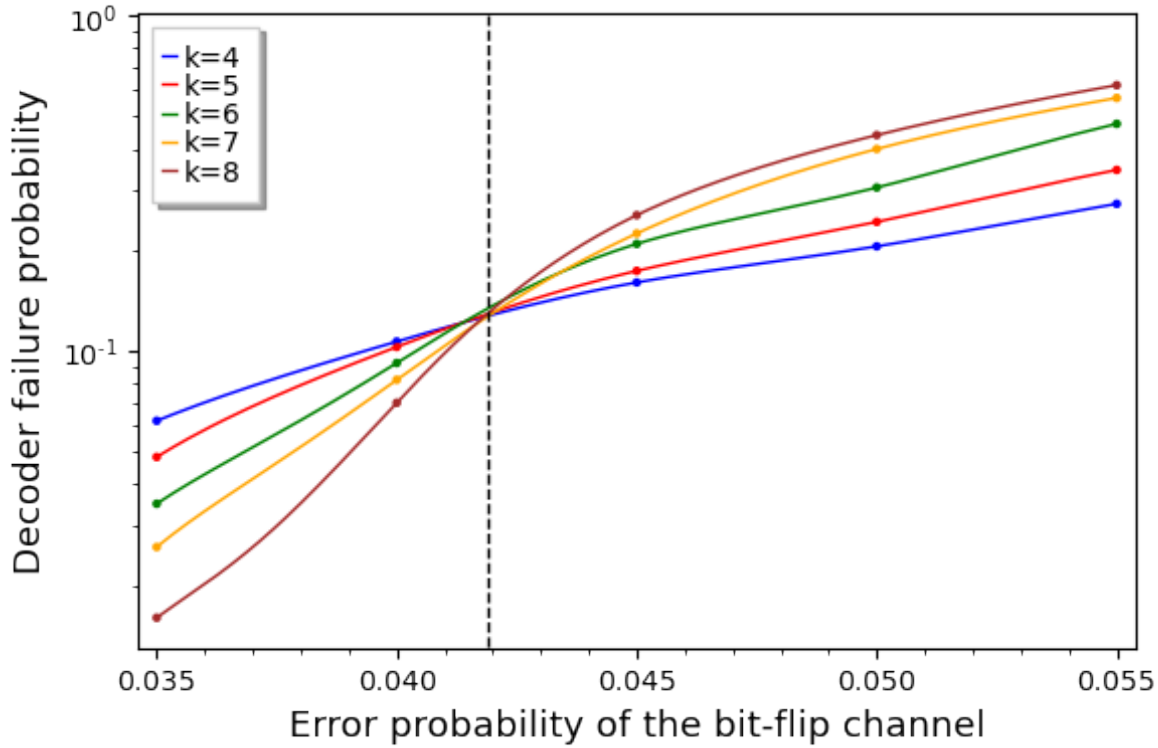


Figure: Results of the Monte Carlo simulations over the bit-flip channel.

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

$d = 2^0$

Figure: $\mathbf{e}_0$ is a minimal cycle of the toric tiling of size 1×1

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

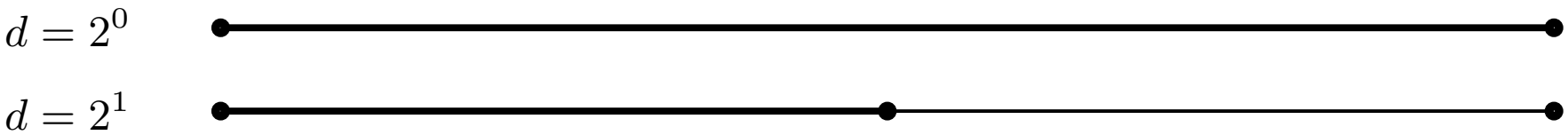


Figure: $\mathbf{e}_1$ is a half a cycle of the toric tiling of size 2×2

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

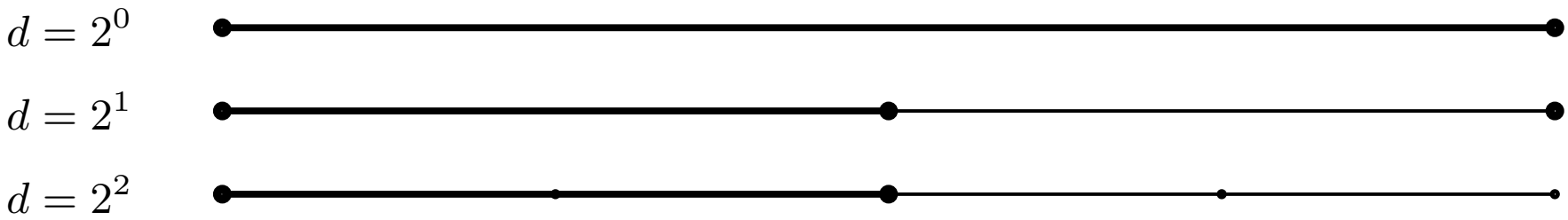


Figure: $\mathbf{e}_2$ is a preimage of $\mathbf{e}_1$ on the toric tiling of size 4×4

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

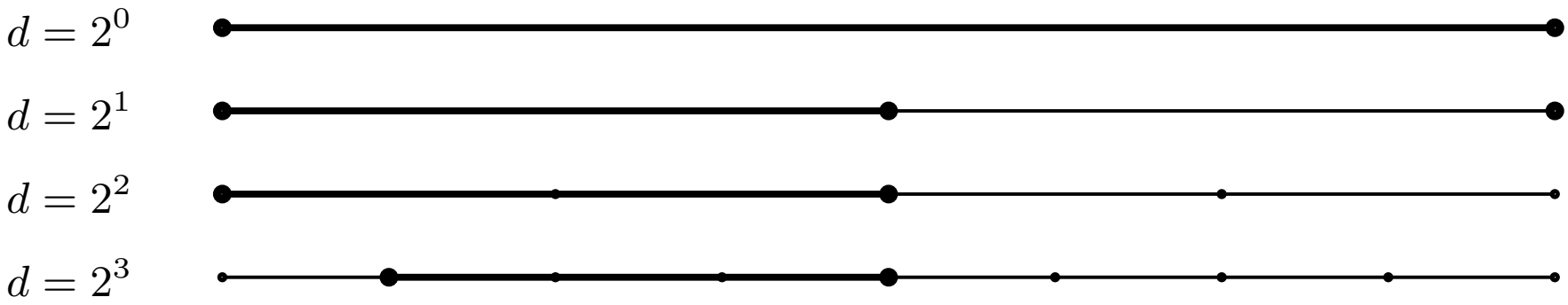


Figure: $\mathbf{e}_3$ is a preimage of $\mathbf{e}_2$ on the toric tiling of size 8×8

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

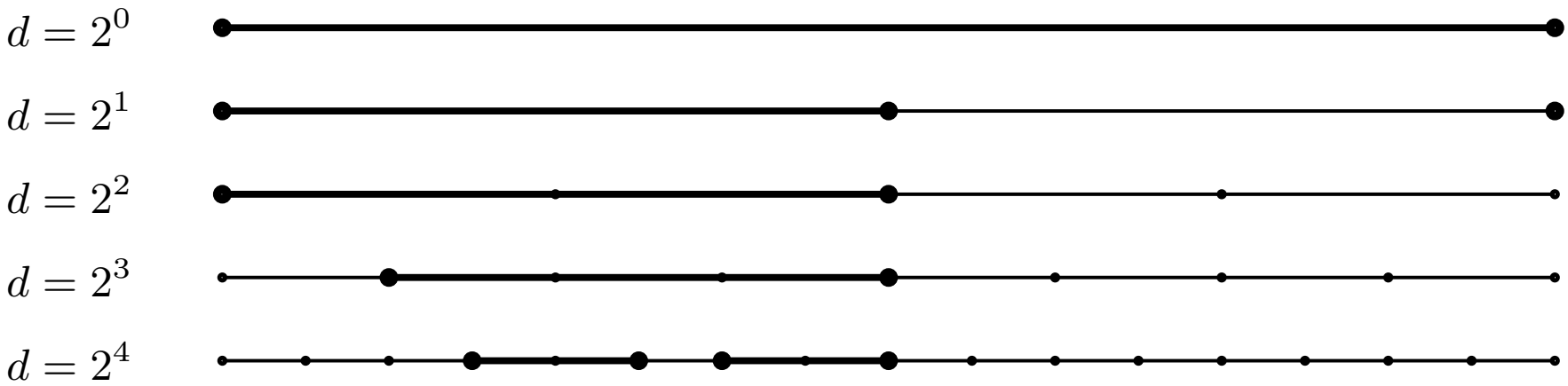


Figure: $\mathbf{e}_4$ is a preimage of $\mathbf{e}_3$ on the toric tiling of size 16×16

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

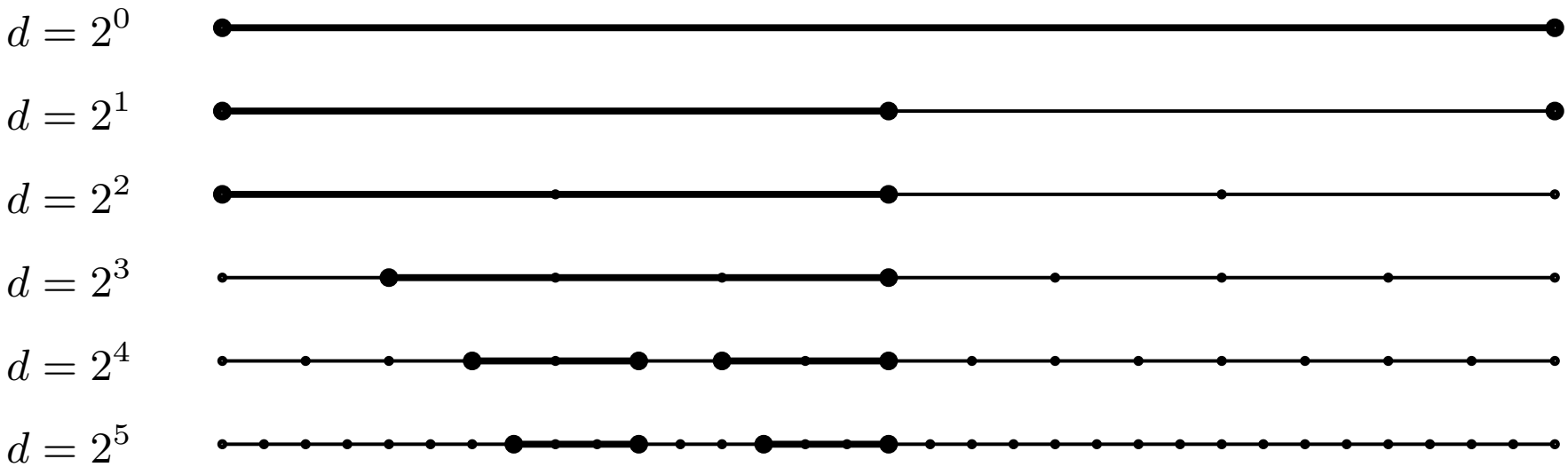


Figure: $\mathbf{e}_5$ is a preimage of $\mathbf{e}_4$ on the toric tiling of size 32×32

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

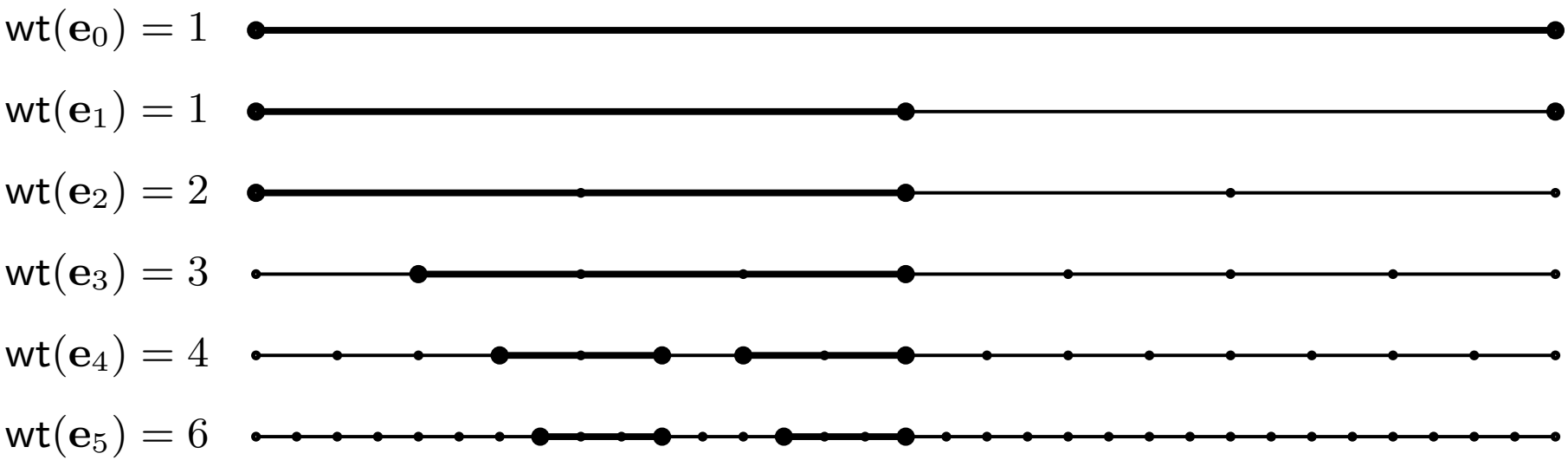


Figure: Weights of the error patterns

Analysis over the Adversarial Channel

Construction of 1-dimensional wrongly decoded error patterns:

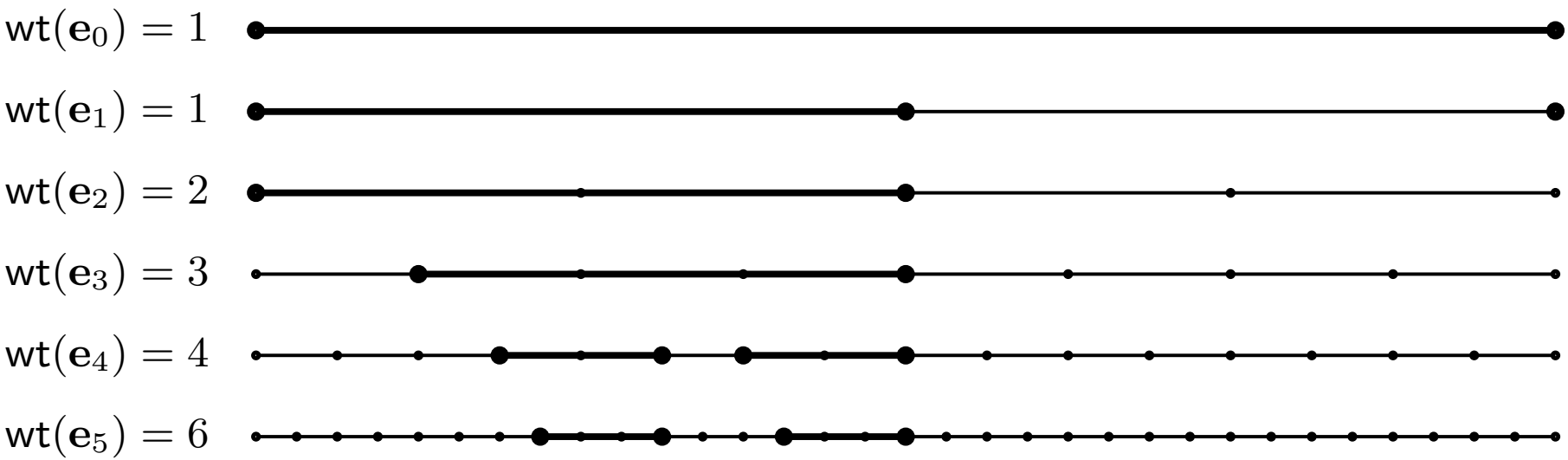


Figure: Weights of the error patterns

Upper Bound on the Error-Correcting Radius

There exist wrongly decoded error patterns whose weight scale like $d^{1/2}$.

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Idea of the proof:

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Idea of the proof:

- Evaluate the growth of $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i$ for increasing indexes
  - $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i)$: reduced weight of $\mathbf{e}_i$
  - P_i: number of paths of $\mathbf{e}_i$

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Idea of the proof:

- Evaluate the growth of $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i$ for increasing indexes
  - $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i)$: reduced weight of $\mathbf{e}_i$
  - P_i: number of paths of $\mathbf{e}_i$

Lemma

If $\mathbf{e}_{i+1}$ is a preimage of $\mathbf{e}_i$, then $\mathsf{wt}_\mathsf{r}(\mathbf{e}_{i+1}) + P_{i+1} \geq \frac{6}{5}(\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i)$

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Idea of the proof:

- Evaluate the growth of $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i$ for increasing indexes
  - $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i)$: reduced weight of $\mathbf{e}_i$
  - P_i: number of paths of $\mathbf{e}_i$

Lemma

If $\mathbf{e}_{i+1}$ is a preimage of $\mathbf{e}_i$, then $\mathsf{wt}_\mathsf{r}(\mathbf{e}_{i+1}) + P_{i+1} \geq \frac{6}{5}(\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i)$

- Apply the Lemma to a wrongly decoded error $\mathbf{e}_k$:
  $\mathsf{wt}_\mathsf{r}(\mathbf{e}_k) + P_k \geq 2\left(\frac{6}{5}\right)^{k-1}$

Analysis over the Adversarial Channel

Lower Bound on the Error-Correcting Radius

The renormalisation decoder corrects all errors of weight less than $\frac{5}{6}d^{\log_2(6/5)}$.

Idea of the proof:

- Evaluate the growth of $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i$ for increasing indexes
  - $\mathsf{wt}_\mathsf{r}(\mathbf{e}_i)$: reduced weight of $\mathbf{e}_i$
  - P_i: number of paths of $\mathbf{e}_i$

Lemma

If $\mathbf{e}_{i+1}$ is a preimage of $\mathbf{e}_i$, then $\mathsf{wt}_\mathsf{r}(\mathbf{e}_{i+1}) + P_{i+1} \geq \frac{6}{5}(\mathsf{wt}_\mathsf{r}(\mathbf{e}_i) + P_i)$

- Apply the Lemma to a wrongly decoded error $\mathbf{e}_k$:
  $\mathsf{wt}_\mathsf{r}(\mathbf{e}_k) + P_k \geq 2\left(\frac{6}{5}\right)^{k-1}$
- Regular Hamming weight of $\mathbf{e}_k$ is larger than:
  - reduced weight: $\mathsf{wt}(\mathbf{e}_k) \geq \mathsf{wt}_\mathsf{r}(\mathbf{e}_k)$
  - number of paths: $\mathsf{wt}(\mathbf{e}_k) \geq P_k$

Take Away

Renormalisation decoders:

- Recursively reduce the decoding problem to smaller codes
- Good time-complexity and can be parallelised
- High threshold values over bit-flip and depolarisation channels

Take Away

Renormalisation decoders:

- Recursively reduce the decoding problem to smaller codes
- Good time-complexity and can be parallelised
- High threshold values over bit-flip and depolarisation channels

Our deterministic renormalisation decoder:

- $d^{0.26} \lesssim$ error-correcting radius $\lesssim d^{0.5}$
- Threshold value over the bit-flip channel: $4.2\,\%$

Take Away

Renormalisation decoders:

- Recursively reduce the decoding problem to smaller codes
- Good time-complexity and can be parallelised
- High threshold values over bit-flip and depolarisation channels

Our deterministic renormalisation decoder:

- $d^{0.26} \lesssim$ error-correcting radius $\lesssim d^{0.5}$
- Threshold value over the bit-flip channel: $4.2\,\%$

Any deterministic renormalisation decoder:

- Existence of fractal-like wrongly decoded errors of weight d^{α}
- Improvements by increasing block size and using message-passing

Take Away

Renormalisation decoders:

- Recursively reduce the decoding problem to smaller codes
- Good time-complexity and can be parallelised
- High threshold values over bit-flip and depolarisation channels

Our deterministic renormalisation decoder:

- $d^{0.26} \lesssim$ error-correcting radius $\lesssim d^{0.5}$
- Threshold value over the bit-flip channel: $4.2\,\%$

Any deterministic renormalisation decoder:

- Existence of fractal-like wrongly decoded errors of weight d^{α}
- Improvements by increasing block size and using message-passing

Thank you for your attention! Any questions?