Topological Concatenation of 2D Color Codes

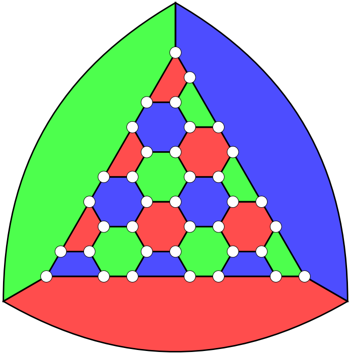
Alexandre Guernut Christophe Vuillot

June 19, 2021

2D triangular color codes

A topological stabilizer code

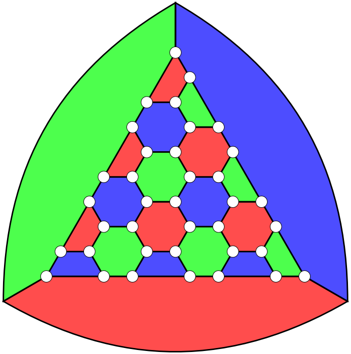
- Stabilizers are associated to the faces of a tiling of a sphere



(a) Sphere tiling

A topological stabilizer code

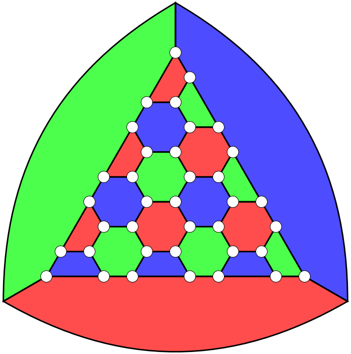
- Stabilizers are associated to the faces of a tiling of a sphere
- Qubits are at the vertices of the tiling



(a) Sphere tiling

A topological stabilizer code

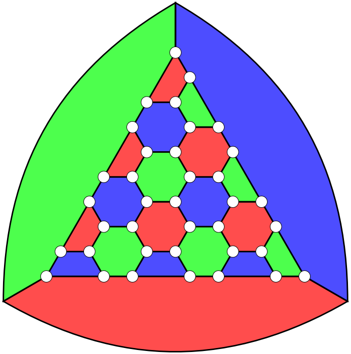
- Stabilizers are associated to the faces of a tiling of a sphere
- Qubits are at the vertices of the tiling
- Each face is associated to an X and a Z stabilizer acting on the vertices



(a) Sphere tiling

A topological stabilizer code

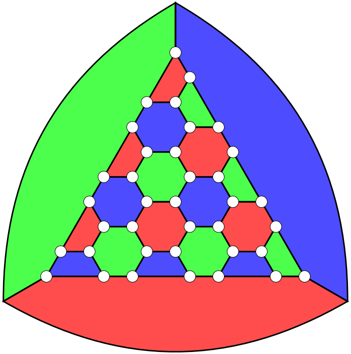
- Stabilizers are associated to the faces of a tiling of a sphere
- Qubits are at the vertices of the tiling
- Each face is associated to an X and a Z stabilizer acting on the vertices
- One vertex and three faces are removed to obtain one logical qubit



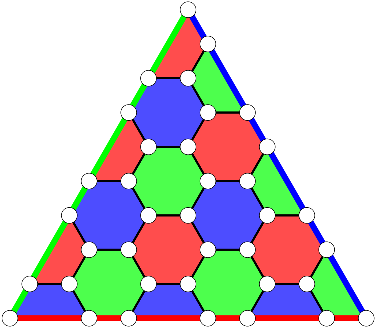
(a) Sphere tiling

A topological stabilizer code

- Stabilizers are associated to the faces of a tiling of a sphere
- Qubits are at the vertices of the tiling
- Each face is associated to an X and a Z stabilizer acting on the vertices
- One vertex and three faces are removed to obtain one logical qubit



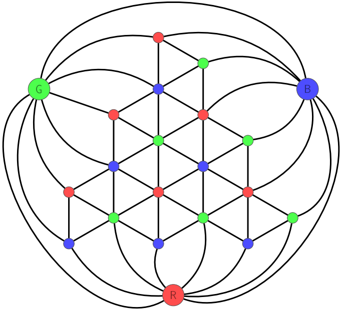
(a) Sphere tiling



(b) 2D Triangular color code

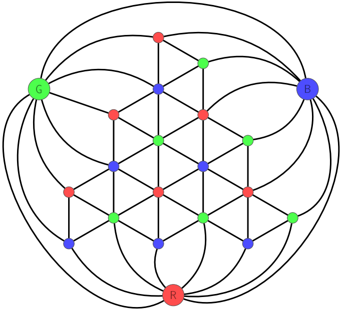
Dual representation

- The dual graph is often more useful



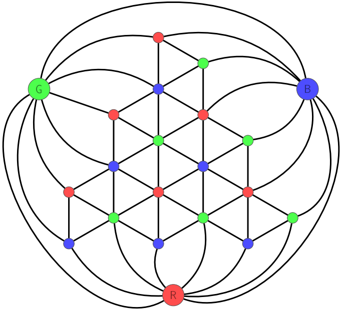
Dual representation

- The dual graph is often more useful
- Stabilizers are now vertices that act on triangular faces (the qubits)



Dual representation

- The dual graph is often more useful
- Stabilizers are now vertices that act on triangular faces (the qubits)
- Colored boundaries are now boundary nodes



Study cases

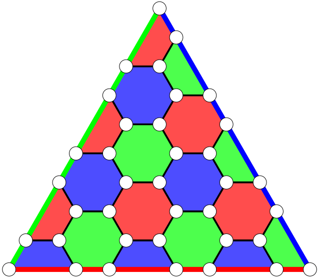
- Any 3-colorable, 3-valent tiling of a sphere could be used

Study cases

- Any 3-colorable, 3-valent tiling of a sphere could be used
- We limit our interest to two particular regular tilings

Study cases

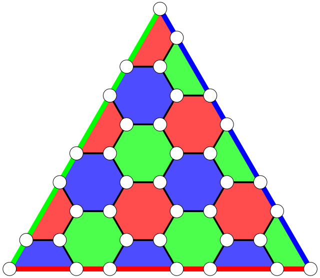
- Any 3-colorable, 3-valent tiling of a sphere could be used
- We limit our interest to two particular regular tilings



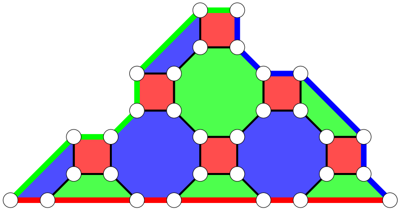
(a) $6-6-6$: $[\![n=\frac{3}{4}d^2+\frac{1}{4},1,d]\!]$

Study cases

- Any 3-colorable, 3-valent tiling of a sphere could be used
- We limit our interest to two particular regular tilings



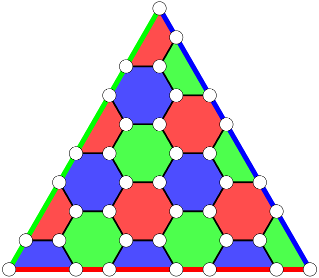
(a) $6-6-6 : [\![n = \frac{3}{4}d^2 + \frac{1}{4}, 1, d]\!]$



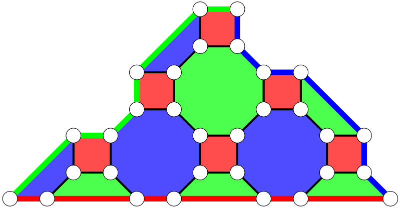
(b) $4-8-8 : [\![n = \frac{1}{2}d^2 + d - \frac{1}{2}, 1, d]\!]$

Study cases

- Any 3-colorable, 3-valent tiling of a sphere could be used
- We limit our interest to two particular regular tilings
- They are the only ones whose leading coefficient in the number of qubits representation as a function of the distance is less than 1



(a) $6-6-6 : [\![n = \frac{3}{4}d^2 + \frac{1}{4}, 1, d]\!]$



(b) $4-8-8 : [\![n = \frac{1}{2}d^2 + d - \frac{1}{2}, 1, d]\!]$

Y operators

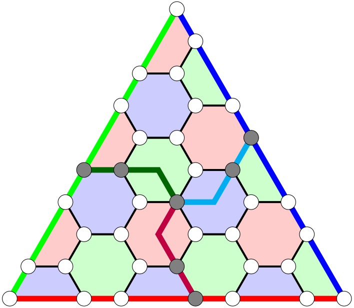
- A colored string is a sequence of edges of the same color

Y operators

- A colored string is a sequence of edges of the same color
- Three strings crossing at the same point form a non-trivial logical operator if they start on the boundaries

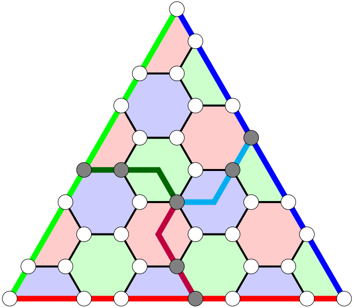
Y operators

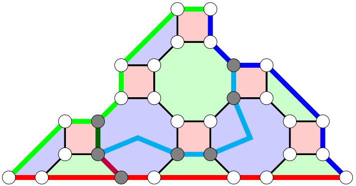
- A colored string is a sequence of edges of the same color
- Three strings crossing at the same point form a non-trivial logical operator if they start on the boundaries



Y operators

- A colored string is a sequence of edges of the same color
- Three strings crossing at the same point form a non-trivial logical operator if they start on the boundaries





Topological concatenation of color codes

Improving logical error rate

- Once one has both a quantum code and a decoding procedure whose threshold is above the physical error rate

[1] Daniel Gottesman, A Theory of Fault-Tolerant Quantum Computation (1997)

Improving logical error rate

- Once one has both a quantum code and a decoding procedure whose threshold is above the physical error rate
- There are two ways of improving the logical error rate

[1] Daniel Gottesman, A Theory of Fault-Tolerant Quantum Computation (1997)

Improving logical error rate

- Once one has both a quantum code and a decoding procedure whose threshold is above the physical error rate
- There are two ways of improving the logical error rate
- Doing usual concatenation[1], which can make non-planar stabilizers appear

[1] Daniel Gottesman, A Theory of Fault-Tolerant Quantum Computation (1997)

Improving logical error rate

- Once one has both a quantum code and a decoding procedure whose threshold is above the physical error rate
- There are two ways of improving the logical error rate
- Doing usual concatenation[1], which can make non-planar stabilizers appear
- In the case of topological codes, we can increase the distance, with a quadratic cost in the number of qubits used (BPT bound)

[1] Daniel Gottesman, A Theory of Fault-Tolerant Quantum Computation (1997)

Improving logical error rate

- Once one has both a quantum code and a decoding procedure whose threshold is above the physical error rate
- There are two ways of improving the logical error rate
- Doing usual concatenation[1], which can make non-planar stabilizers appear
- In the case of topological codes, we can increase the distance, with a quadratic cost in the number of qubits used (BPT bound)
- Topological concatenation is a hybrid of these two methods

[1] Daniel Gottesman, A Theory of Fault-Tolerant Quantum Computation (1997)

Plain surgery

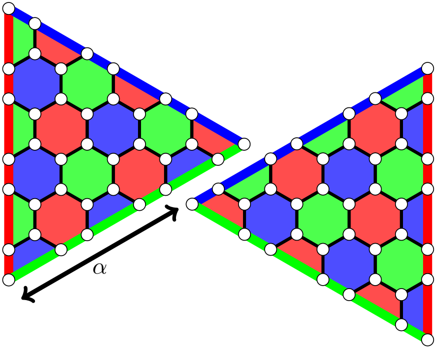
- Plain surgery merges topological codes while keeping encoded qubits

Plain surgery

- Plain surgery merges topological codes while keeping encoded qubits
- The merge process can be tuned to trade between distance and ease of measurement

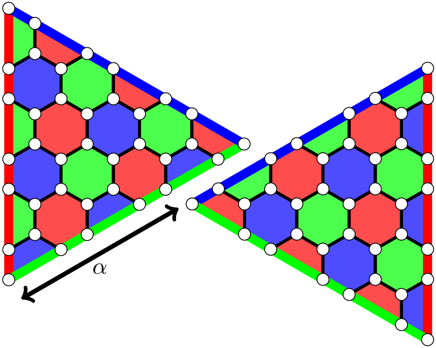
Plain surgery

- Plain surgery merges topological codes while keeping encoded qubits
- The merge process can be tuned to trade between distance and ease of measurement



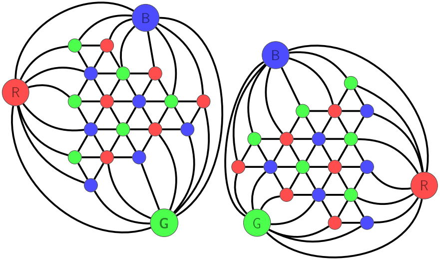
Merging two color codes

- It is not obvious which form the stabilizers should have while concatenating two triangular color codes



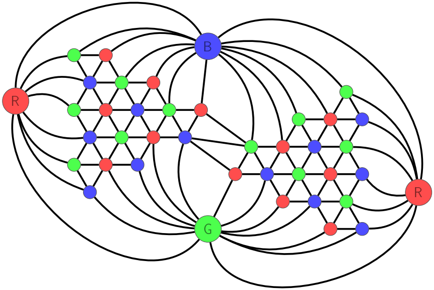
Merging two color codes

- It is not obvious which form the stabilizers should have while concatenating two triangular color codes
- The dual view makes it clearer, as stabilizers are vertices and qubits triangles



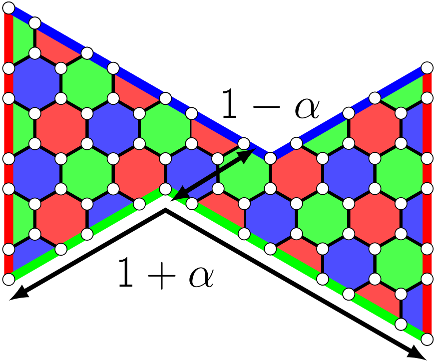
Merging two color codes

- It is not obvious which form the stabilizers should have while concatenating two triangular color codes
- The dual view makes it clearer, as stabilizers are vertices and qubits triangles
- Boundary nodes are merged and stabilizers in the overlap regions are linked



Merging two color codes

- It is not obvious which form the stabilizers should have while concatenating two triangular color codes
- The dual view makes it clearer, as stabilizers are vertices and qubits triangles
- Boundary nodes are merged and stabilizers in the overlap regions are linked

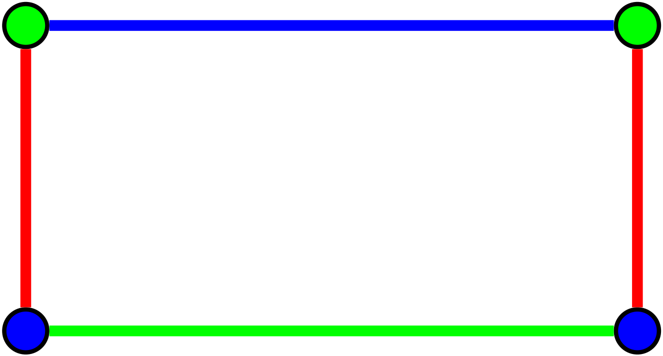


Non-trivial measurements

- Y operators on several qubits can be combined to measure product operators

Non-trivial measurements

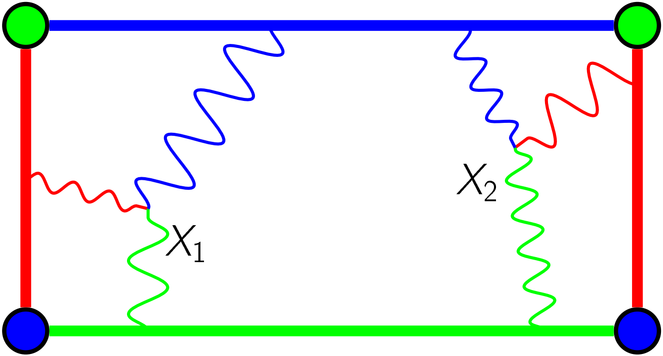
- Y operators on several qubits can be combined to measure product operators



Schematic of logical operators

Non-trivial measurements

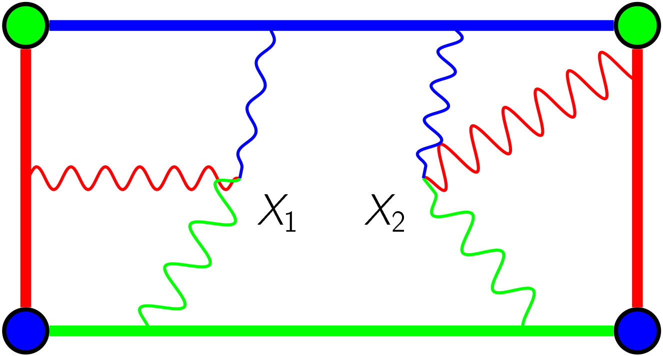
- Y operators on several qubits can be combined to measure product operators



Schematic of logical operators

Non-trivial measurements

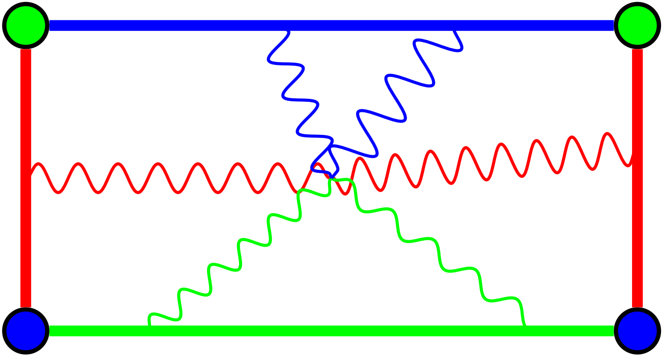
- Y operators on several qubits can be combined to measure product operators



Schematic of logical operators

Non-trivial measurements

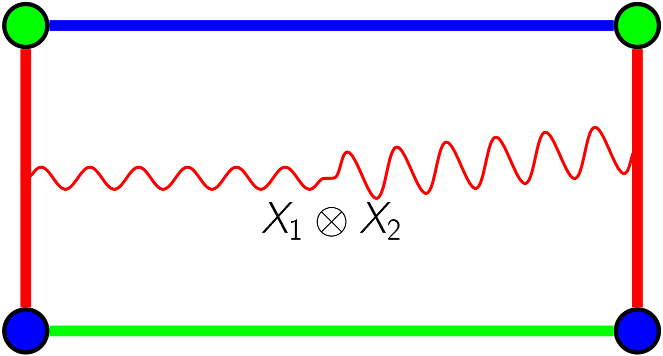
- Y operators on several qubits can be combined to measure product operators



Schematic of logical operators

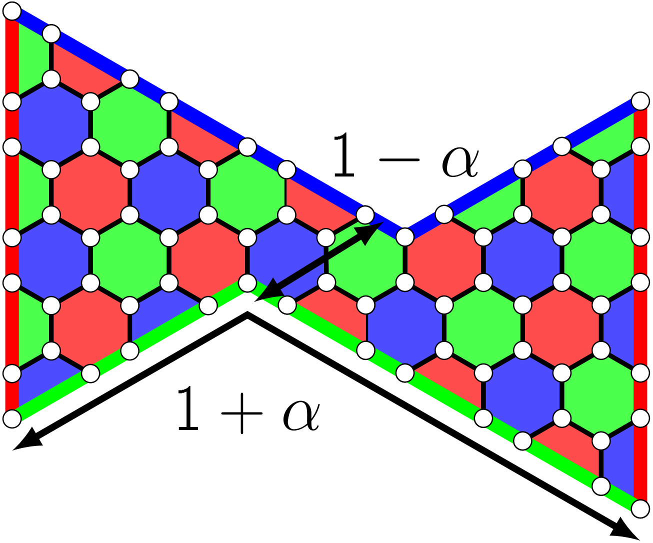
Non-trivial measurements

- Y operators on several qubits can be combined to measure product operators
- The product of the two logical operators can be represented by a red-string from left to right:



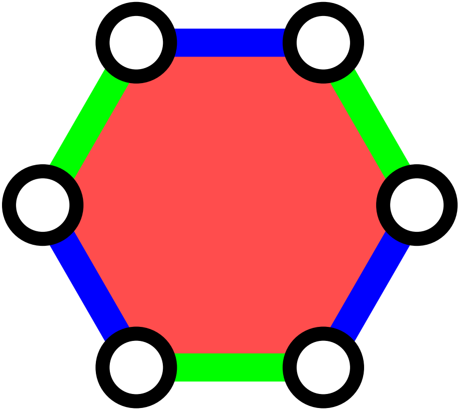
Schematic of logical operators

Non-trivial measurements



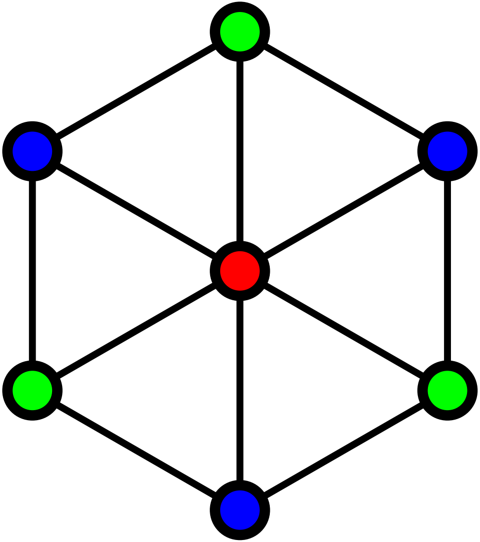
Upper level stabilizers

Stabilizers are product operators on some qubits:



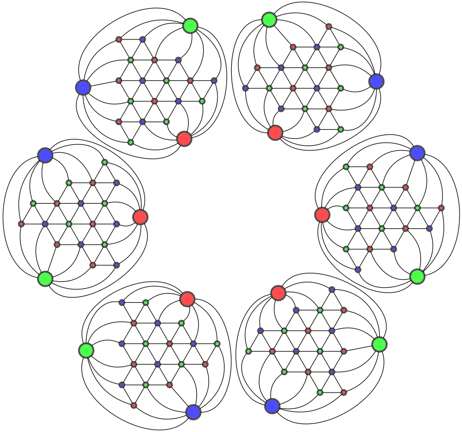
Upper level stabilizers

We can have a look at the dual view:



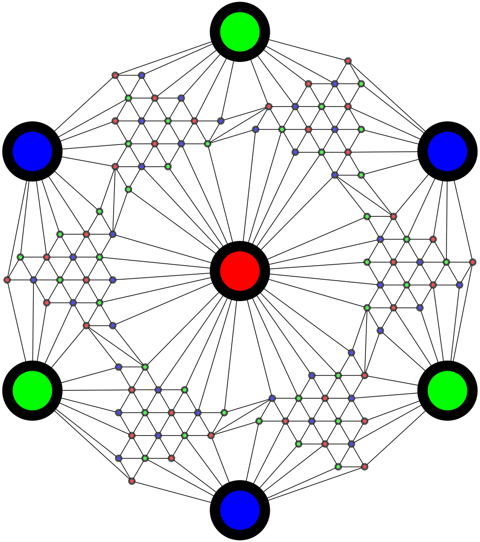
Upper level stabilizers

We can then replace the physical qubits by logical ones:



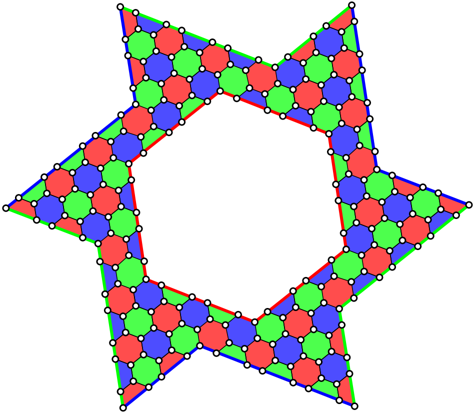
Upper level stabilizers

We can apply the merge procedure to neighboring qubits:



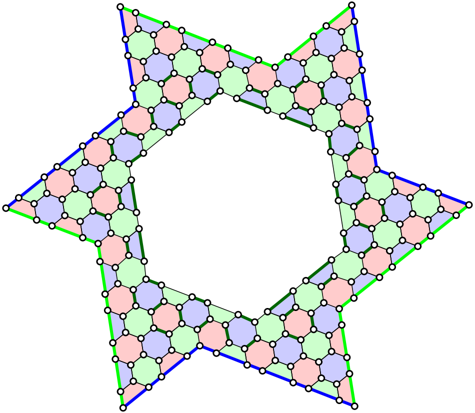
Upper level stabilizers

In the primal view:



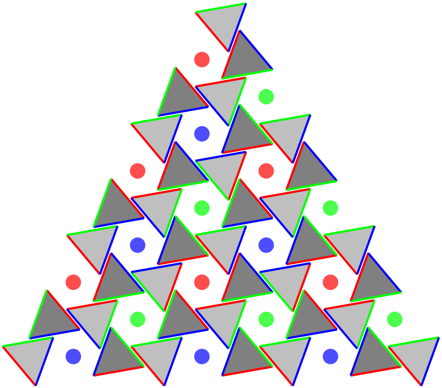
Upper level stabilizers

The large stabilizer can me measured by measuring colored edges in its surroundings:



Topologically concatenated color codes

Repeating the process around all the stabilizers with all qubits considered:



Characteristics of concatenated codes

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

$$[\![\qquad\qquad\qquad]\!]$$

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

$$[\![n_1 n_0 \qquad\qquad\qquad\qquad\qquad\qquad]\!]$$

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

$$[\![n_1 n_0, {\color{red}1} \qquad\qquad\qquad\qquad]\!]$$

Theoretical expectations

- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

$$[\![n_1 n_0, 1, ? \qquad\qquad\qquad\qquad]\!]$$

Theoretical expectations

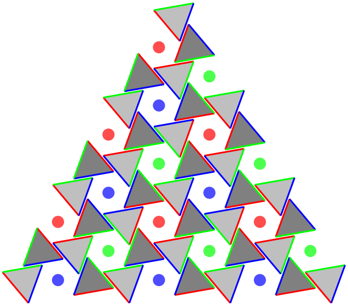
- Any kind of 2D triangular color code can be used at any level of encoding
- Suppose we use a $[\![n_1, 1, d_1]\!]$ code as an upper-level template and $[\![n_0, 1, d_0]\!]$ codes to encode physical qubits
- Suppose we use α to parametrize the concatenation
- What do we expect to get for the concatenated code ?

$$[\![n_1 n_0, 1, f(\alpha, d_0, d_1)]\!]$$

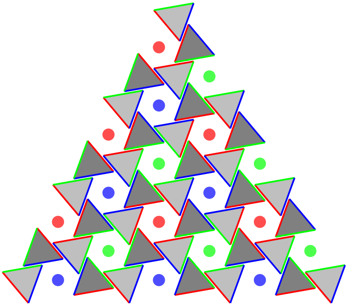
- Determining the distance of a code is hard as we need to find the smallest-weighted non-trivial logical operator

- Determining the distance of a code is hard as we need to find the smallest-weighted non-trivial logical operator
- The minimal weight can be upper bounded by the weight of the edge operator, which is easy to compute

- Determining the distance of a code is hard as we need to find the smallest-weighted non-trivial logical operator
- The minimal weight can be upper bounded by the weight of the edge operator, which is easy to compute



- Determining the distance of a code is hard as we need to find the smallest-weighted non-trivial logical operator
- The minimal weight can be upper bounded by the weight of the edge operator, which is easy to compute



$$d_0 \left((d_1 - 1) \frac{\alpha + 1}{2} + 1 \right)$$

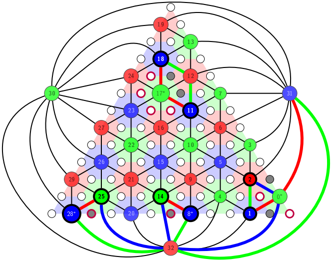
Distance evaluation

Lattice type	$d_0 = d_1$	α	Concatenated distance	n	Qubit gain
6-6-6	11	$\frac{3}{11}$	81	8281	+68%
6-6-6	11	$\frac{7}{11}$	101	8281	+8%
6-6-6	11	$\frac{9}{11}$	111	8281	-11%
6-6-6	111	$\frac{81}{111}$	10671	85.10^6	+0%
6-6-6	111	$\frac{91}{111}$	11221	85.10^6	-10%
4-8-8	11	$\frac{3}{11}$	81	5041	+50%
4-8-8	11	$\frac{7}{11}$	101	5041	-3%
4-8-8	11	$\frac{9}{11}$	111	5041	-20%
4-8-8	111	$\frac{49}{111}$	10671	40.10^6	-1%
4-8-8	111	$\frac{91}{111}$	11221	40.10^6	-37%

Decoding the concatenated color codes

Triangular color code decoder

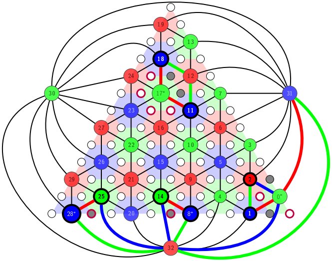
- A computationally efficient decoder for triangular color codes as been presented by [2]



[2] Chamberland, Triangular color codes on trivalent graphs with flag qubits (2020)
[3] https://github.com/networkx/networkx
[4] https://github.com/oscarhiggott/PyMatching

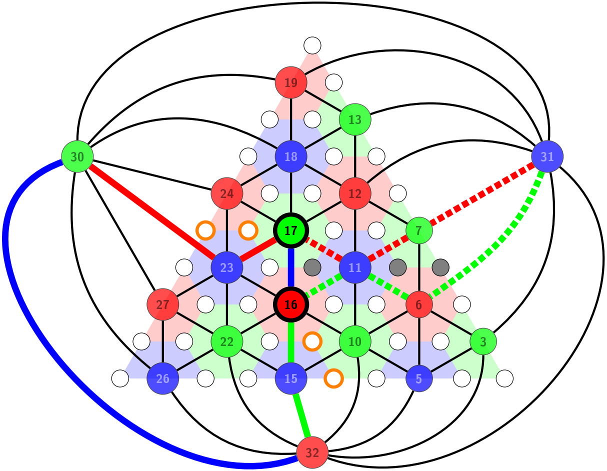
Triangular color code decoder

- A computationally efficient decoder for triangular color codes as been presented by [2]
- Our Python implementation uses NetworkX[3] for graphs and PyMatching[4] for syndrome pairings.

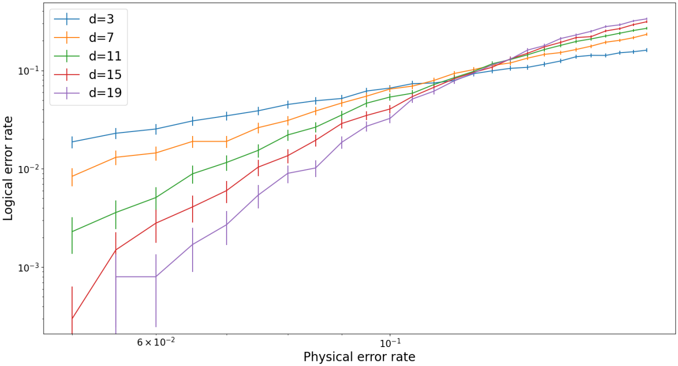


[2] Chamberland, Triangular color codes on trivalent graphs with flag qubits (2020)
[3] https://github.com/networkx/networkx
[4] https://github.com/oscarhiggott/PyMatching

Decoder limitations

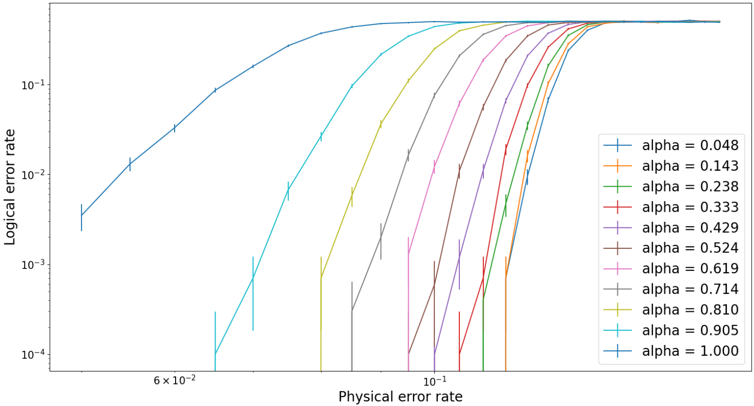


Logical error rate (non concatenated case)



Logical error rate (concatenated case)

Concatenation of a distance 21 code with itself (39601 qubits)



Further prospects

Further prospects

- We might want to try using a better decoder

Further prospects

- We might want to try using a better decoder
- We might want to try decoding recursively

Further prospects

- We might want to try using a better decoder
- We might want to try decoding recursively
- We might want to try choosing a different geometry for the upper level code (toric geometry)