Tensor-Network Codes*

Oza Harshkumar Ajaykumar (20929)
Quantum Technology, IISc Bengaluru

Manish Kumar (21044)
Quantum Technology, IISc Bengaluru

(Dated: April 22, 2023)

Tensor network codes assist in designing larger stabilizer codes by employing smaller stabilizer codes. This paper starts with preliminary ideas for the tensor network and its algebra. Then we keep building up the crucial ideas of representing stabilizer codes as tensors. We demonstrate how to build larger stabilizer codes using valid tensor contraction rules. An efficient tensor contraction method allows for an efficient code distance estimation and maximum likelihood decoding. We explicitly demonstrate these ideas for a 20-qubit holographic code. Several topological codes can be constructed via tensor networks. We demonstrate such construction for rotated surface codes. Finally, we briefly touch on the idea of tensor network doping for rotated surface code that helps in reducing the number of minimum-weight logical operators. It is supplemented with a comparative evaluation of the error-correcting capability of doped and un-doped tensor codes.

Original paper: Local tensor-network codes by Terry Farrelly et.al 2022 New J. Phys. 24 043015

I. INTRODUCTION TO TENSOR NETWORKS

One of the reasons that tensor networks are so helpful is the straightforward notation used to describe them. Tensor network notation (TNN) can be considered a generalization of Einstein summation notation. We will first introduce tensors in TNN notation, followed by some operations that can be performed on them.

A. Tensors

Tensors are a generalization of vectors and matrices. A d-dimensional vector is an element of $\mathbb{C}^d$. An $m \times n$-dimensional matrix is an element of $\mathbb{C}^{m\times n}$. Similarly, a rank-r tensor of dimensions $d_1 \times d_2 \times \cdots \times d_r$ is an element of $\mathbb{C}^{d_1\times d_2\times\cdots\times d_r}$. Mathematically, a rank-r tensor T is represented as $T_{i_1 i_2 \cdots i_r}$. The number of values an index can take is called the *dimension (d)* of that index. Clearly, scalars, vectors, and matrices are rank-0, 1, and 2 tensors, respectively.

In TNN, a single tensor is simply represented by a geometric shape with legs sticking out of it. Each leg corresponds to an index analogous to the indices of Einstein notation. Fig. 1(a) shows a representation of a matrix in TNN, and Fig. 1(b) shows a representation of a rank-r tensor.

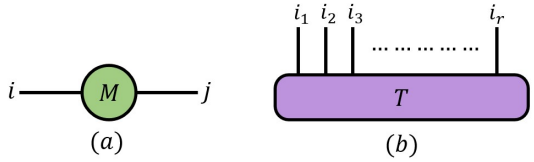


FIG. 1. (a) A matrix in TNN. (b) A rank-r tensor in TNN.

In some contexts, the shape used and the direction of the legs can indicate certain properties of the tensor or index, but for a general network, they do not carry any special significance. For example, when representing quantum states, it is often convenient to use the direction of legs to denote whether the vectors are representing a 'ket' or its dual 'bra.'

B. Tensor operations

The main advantage of TNN comes in representing tensors made up of several other tensors. The operations we will explain are those of the *tensor product*, *trace*, and *contraction*.

1. Tensor product

The tensor product is a generalization of the outer product of vectors. On a given set of indices, the value of the tensor product is the element-wise product of the values of each constituent tensor. The binary tensor product mathematically in index notation is written as:

$$[A \otimes B]_{i_1,\cdots,i_r,j_1,\cdots,j_s} := A_{i_1,\cdots,i_r} \cdot B_{j_1,\cdots,j_s} \qquad (1)$$

Diagrammatically, this binary tensor product is simply represented by placing two tensors next to each other, as shown in Fig. 2. The value of a network containing disjoint tensors is simply the product of the constituent values.

* The present manuscript is a review article scripted as part of a term paper at IISc Bengaluru

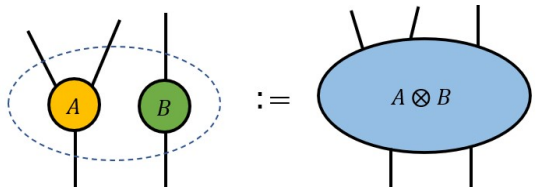


FIG. 2. Tensor product of two rank-3 tensors in TNN.

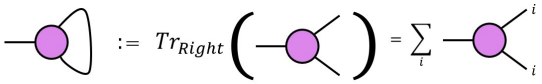


FIG. 3. Trace operation on the right indices of a rank-3 tensor in TNN.

2. *Trace*

Consider a tensor A, for which the x_{th} and y_{th} indices have the same dimensions (i.e., $d_x = d_y$). The partial trace over these two dimensions is simply a joint summation over that index. Mathematically:

$$\begin{aligned}&[Tr_{x,y}A]_{i_1,\cdots,i_{x-1},i_{x+1},\cdots,i_{y-1},i_{y+1},\cdots,i_r}\\ &:=\sum_{\alpha=1}^{d_x} A_{i_1,\cdots,i_{x-1},\alpha,i_{x+1},\cdots,i_{y-1},\alpha,i_{y+1},\cdots,i_r}\end{aligned} \tag{2}$$

Similar to Einstein's notation, this summation is implicit in TNN, indicated by the corresponding legs being joined. An advantage is that these summer-over indices need not be named, making the notation neat for large networks. Fig. 3 shows this operation over the two indices of a rank-3 tensor in TNN.

3. *Contraction*

The most common tensor operation is *contraction*. This operation corresponds to a tensor product followed by a trace between indices of the two tensors. Fig. 4 shows this operation between two rank-3 tensors.

The indices being contracted are called *internal indices*; free legs correspond to *external indices*. Some familiar examples of contraction in TNN are shown in Fig. 5.

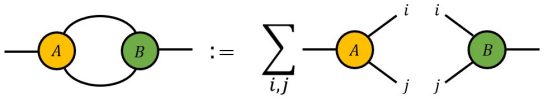


FIG. 4. Contraction operation between two rank-3 tensors in TNN.

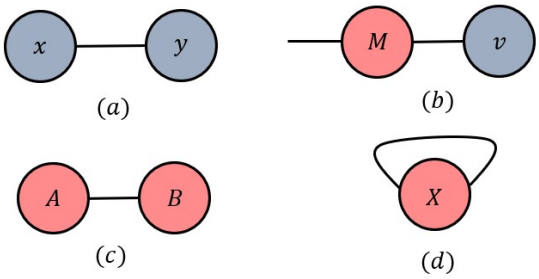


FIG. 5. Some common contraction operations in TNN. (a) vector inner product. (b) Matrix-vector product. (c) Matrix-matrix product. (d) Trace of a matrix.

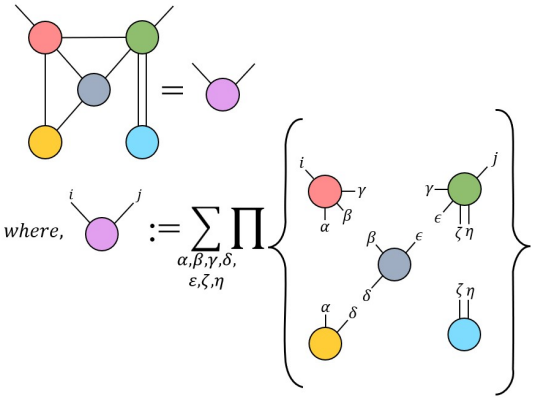


FIG. 6. A generic tensor network.

C. Tensor networks

A tensor network is a diagram that tells us how to combine several tensors into a single composite tensor. The rank of this equivalent tensor is equal to the number of unmatched/free legs. For a given configuration of external indices, the value of the tensor network (hence of the equivalent tensor) is given by the product of the values of the constituent tensors, summed over all internal index labeling consistent with the contractions. Consider a tensor network shown in Fig. 6. The external indices are labeled by i and j, whereas the internal indices are labeled by $\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta$.

Fig. 7(a) shows a network build from four tensors A, B, C, and D. The values of constituent tensors are shown in Fig. 7(b). All the indices are three-dimensional, starting from 0. In this network, there are no free legs, hence no external indices. Hence, all the indices correspond to internal indices. The value of the tensor network (and of equivalent tensor T) is calculated as follows:

$$T = \sum_{\alpha,\beta,\gamma,\epsilon,\delta=0}^{2} A_{\alpha\beta}B_{\alpha\gamma\delta}C_{\delta\epsilon}D_{\beta\gamma\epsilon} = 1080 \tag{3}$$

Let us first review the basics of stabilizer codes prior to

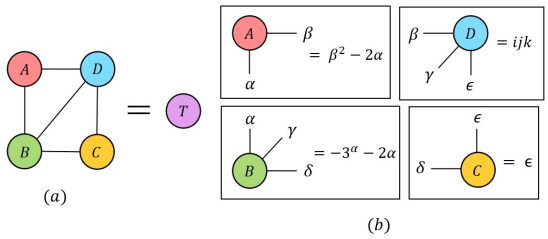


FIG. 7. An example of tensor network. (a) Tensor network. (b) Constituent tensors.

representing them utilizing tensors and TNN.

II. STABILIZER CODES

Consider the n-qubit Pauli group $\mathcal{P}_n$. The group contains elements of the form $i^l\sigma^{i_1}\otimes\cdots\otimes\sigma^{i_n}$, with $l\in\{0,1,2,3\}$. We are denoting the identity and Pauli operators using $\sigma^0=I,\sigma^1=X,\sigma^2=Y,\sigma^3=Z$. A stabilizer group $\mathbb{S}$ is an abelian subgroup of $\mathcal{P}_n$, i.e., $\mathbb{S}\leq\mathcal{P}_n$. The vector space (a subspace of Hilbert space), which is used to encode logical information (also called codespace), is fixed by all the elements of the stabilizer group. Every state $|\psi\rangle$ in the codespace satisfies $S|\psi\rangle=|\psi\rangle\ \forall S\in\mathbb{S}$. Let r be the number of independent generators of $\mathbb{S}$ (denoted by S_i), then the dimension of the codespace is 2^{n-k}, which corresponds to the number of encoded logical qubits $k=n-r$.

The logical operators of the code also generate a non-abelian group $\mathbb{L}\leq\mathcal{P}_n$. This group has a set of k $\bar{Z}$ and k $\bar{X}$ generators, and they all commute with all the stabilizers $S\in\mathbb{S}$. We denote them by $\bar{Z}_i$ and $\bar{X}_i$, with $i\in\{1,\ldots,k\}$. These generators anti-commute pairwise (forming hyperbolic pairs), i.e., $\bar{X}_i\bar{Z}_j=(-1)^{\delta_{ij}}\bar{Z}_j\bar{X}_i$.

It will be helpful (during ML decoding) to consider the abelian group of operators called pure error (also known as destabilizers) $\varepsilon\leq\mathcal{P}_n$. This group has $n-k$ generators E_i which are chosen to satisfy $E_iS_j=(-1)^{\delta_{ij}}S_jE_i$ with $E_i^2=I$. Proof of the existence of such generators is mentioned in [4]. The full Pauli group on n physical qubits is generated by products of $E_i,S_i,\bar{X}_j$, and $\bar{Z}_j$.

To detect errors, we measure each of the stabilizer generators S_i, giving eigenvalues ±1. This gives us an error syndrome which is a length-r vector $\vec{s}=(s_1,\ldots,s_r)$ of the measurement outcomes. $s_i=0$ $(i\in\{1,\ldots,r\})$ if the error commutes with the corresponding generator S_i (i.e., the measurement outcome is 1) and $s_i=1$ if the error anti-commutes with the corresponding generator (i.e., the measurement outcome is -1). We denote the pure error corresponding to syndrome $\vec{s}$ by $E(\vec{s})=\prod_i E_i^{s_i}$. But $E(\vec{s})$ is not the only syndrome with syndrome $\vec{s}$. Any error $E^{'}=E(\vec{s})SL$ for any $L\in\mathbb{L}$ and any $S\in\mathbb{S}$, will give the same syndrome. Finding the optimal correction operator given the syndrome is called decoding, which is extremely difficult in general.

TABLE I.
Stabilizer generators and logical operators for the [[5,1,3]] code.

Qubit	1	2	3	4	5
S_1	X	Z	Z	X	I
S_2	I	X	Z	Z	X
S_3	X	I	X	Z	Z
S_4	Z	X	I	X	Z
$\bar{X}_1$	X	X	X	X	X
$\bar{Z}_1$	Z	Z	Z	Z	Z

III. STABILIZER CODES AS TENSORS

To represent arbitrary stabilizer codes using tensors, first, we need to represent Pauli operators using a string of integers. Using a similar convention using in section II, we can represent, for example, the operator $XZIYYZI$ as $(1,3,0,2,2,3,0)$.

For a stabilizer code with n physical qubits and k logical qubits, we define the rank-n tensors as:

$$T(L)_{i_1,\ldots,i_n}=\left\{\begin{array}{l}1,\ if\ \sigma^{i_1}\otimes\ldots\otimes\sigma^{i_n}\in SL\\ 0,\ otherwise\end{array}\right\}\qquad(4)$$

for each logical operator $L\in\mathbb{L}$. Here indices $i_k\in\{0,1,2,3\}$, and $\mathbb{S}L$ is the set of all operators SL with $S\in\mathbb{S}$. This means $\mathbb{S}L$ is the coset of $\mathbb{S}$ with respect to logical operator L. Hence, $T(L)_{i_1,\ldots,i_n}$ can also be thought of as the indicator function for all operators in the class L. If $L=I$ (i.e., $\mathbb{S}L=\mathbb{S}$), then $T(I)_{i_1,\ldots,i_n}$ is the tensor indicating the stabilizer group. Signs of the stabilizers are not included in this representation, though once these are fixed for the generators, they are determined for the whole group.

Table I shows stabilizer generators and logical operators of [[5,1,3]] code, with 5 physical qubits and 1 logical qubit. The tensors for this code $T(L)_{i_1,\ldots,i_6}$ have 16 non-zero values out of $4^5=1024$ values, for each $L\in\{I_1,\bar{X}_1,\bar{Y}_1,\bar{Z}_1\}$, e.g., $T(\bar{X}_1)_{11111}=1$.

IV. GENERATION OF NEW STABILIZER CODES

After representing stabilizer codes using tensors (we will call them code tensors), we can perform the contraction operation to combine several of them to generate new code tensors (also new stabilizer codes represented by them, provided *Theorem 1* is satisfied). It also follows that if the tensor network (formed by the combination of individual code tensors) is efficiently con-

tractible, the newly formed code can also be efficiently decoded. In general, the contraction of tensor networks is an NP-complete problem. A technique thus could be to start with a tensor network geometry that is efficiently contractible and then vary the constituent code tensors within this geometry to search for good codes. For example, for holographic codes, the decoder is efficient without any approximations.

Now, the tensor describing a tensor product of two stabilizer codes is the product of the code tensors. Mathematically:

$$\begin{aligned} &T''(L\otimes L')_{i_1,\ldots,i_n,j_1,\ldots,j_{n'}} \\ &= T(L)_{i_1,\ldots,i_n} T(L')_{j_1,\ldots,j_{n'}} \end{aligned} \tag{5}$$

Now, some of these indices (now total indices are $n+n'$) can be contracted to generate new codes. For the newly formed tensor to be representing a valid stabilizer code, the following theorem should be satisfied.

Theorem 1. — Consider two code tensors $T(L)_{i_1,\ldots,i_n}$ and $T'(L')_{j_1,\ldots,j_{n'}}$ which have n and n' physical qubits and k and k' logical qubits, respectively. We get new tensors describing a new stabilizer code by contracting indices (for simplicity, choose qubits 1 to l for both codes), i.e.,

$$\begin{aligned} &T_{new}(L\otimes L')_{i_{l+1},\ldots,i_n,j_{l+1},\ldots,j_{n'}} = \\ &\sum_{k_1,\ldots,k_l\in\{0,1,2,3\}} T(L)_{k_1,\ldots,k_l,i_{l+1},\ldots,i_n} T(L')_{k_1,\ldots,k_l,j_{l+1},\ldots,j_{n'}} \end{aligned} \tag{6}$$

provided either one of these codes can distinguish any Pauli error on qubits 1 to l. If the contraction is valid, the T_{new} describes a stabilizer code with $n+n'-2l$ physical qubits and $k+k'$ logical qubits. Proof of this theorem is given in Appendix B.

The above theorem allows us to iteratively build large codes with consistency. The tensor network can have any geometry, and the constituent tensors can have any number of logical qubits. Before contracting certain indices, we just need to check if the newly formed code will be a valid code. The stabilizer generators and logical operators for the new stabilizer code can be found in $O(n+n')$ time.

Let us look at an example to understand the method. Fig. 8 shows generation of [[9,3,3]] code from three [[5,1,3]] codes. The stabilizer generators and logical operators for the [[5,1,3]] code are shown in Table I. The newly formed code has $3\times 15-2\times 3=9$ physical qubits and $3\times 1=3$ logical qubits. In [5], authors have written a Julia package [7] to do the underlying calculations. The code used for the above example is given in Appendix A.

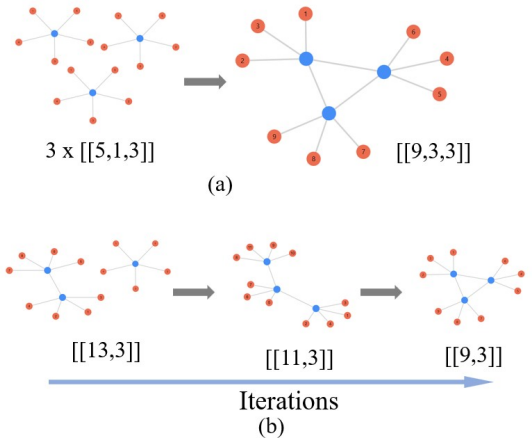


FIG. 8. An example showing the interactive method to build a [[9,3,3]] code from three [[5,1,3]] codes. (a) Tensor product of three [[5,1,3]] code tensors after 3 contraction operations generate a [[9,3,3]] code. (b) Iterations (or intermediate steps) and corresponding codes.

V. CODE DISTANCE VIA TENSOR NETWORK

In this section, we will see a mathematical foundation (and a concrete example) to generate a histogram for weights of stabilizers and logical operators via tensor contraction. Then we use the fact that the minimum weight of logical operators correspond to the code distance. To realize this, first, the histogram-generating tensor function is created. Then we look for any efficient tensor contraction methods. We will see next how an efficient contraction of a tensor network would imply an efficient estimation of weight for logical operators.

We start with defining tensor indicator functions. It requires a weight-tensor that indicates if an operator has a specific weight or not, as below:

$$W_w^{i_1,\ldots i_n} = \left\{ \begin{array}{l} 1,\ if\ weight(\sigma^{i_1}\otimes\ldots\otimes\sigma^{i_n}) = w \\ 0,\ otherwise \end{array} \right\} \tag{7}$$

The class-L indicator function can be equivalently written as:

$$T(L)_{i_1,\ldots i_n} = T^{l_1,\ldots,1_n}_{i_1,\ldots,i_n} \tag{8}$$

The above two indicator functions are combined as below to get another tensor function that primarily counts how many operators exist with weight $'w'$.

$$C_w^{l_1,\ldots,l_n} = W_w^{i_1,\ \ldots\ ,i_n}\ T^{l_1,\ldots,l_n}_{i_1,\ldots,i_n} \tag{9}$$

Our prime goal is to find the minimum possible weight of logical operators. But above weight counter counts something more than that. Basically, its output is the number of all the operators in the centralizer that has a certain weight. Hence, it includes the counts for both stabilizer and logical operators. This stems from the very nature of how the T(L) tensor is defined. Notice the case

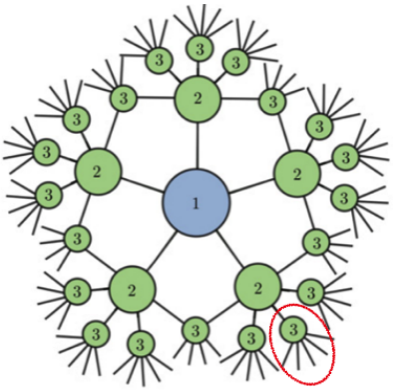


FIG. 9. An example of holographic code. The portion in red circle is the seed code.

of L $= I_1 \otimes ... \otimes I_n$ trivially implies a set of stabilizers itself. Equivalently saying, $C_w^{0,...,0}$ corresponds to number of stabilizer operators with weight $'w'$. This can be fixed by subtracting the contribution of the stabilizer count from the total count as formulated below:

$$D_w = \sum_{l_1,...,l_n} C_w^{l_1,...,l_n} - C_w^{0,...,0} \tag{10}$$

But, exhaustively taking all possible values of $'w'$ and $l_1, ..., l_4$ will turn out to be inefficient. Now from here we need to find an efficient strategy to simplify this calculation by looking into the nature of tensors involved in it. Since we have seen these tensors are quite special one, i.e, they can be contracted if the original code tensor is contractable. This is achieved via strategically splitting the weight indicator tensor $W_w^{i_1,...,i_n}$ into smaller tensors that aids in efficient contraction of counting tensor $C_w^{l_1,...,l_n}$. This partitioning goes as below:

$$W_{w_1,\ w_2}^{i_1,...i_n} = \left\{ \begin{array}{l} 1,\ weight(\sigma^{i_1} \otimes ... \otimes \sigma^{i_n}) = w_1 - w_2 \\ 0,\ otherwise \end{array} \right\} \tag{11}$$

A closer look at $W_{w_1,\ w_2}^{i_1,...i_n}$ reveals it is a generalization of $W_w^{i_1,...i_n}$. It follows directly from its definition as putting $w_2 = 0$ implies $W_{w_1,\ 0}^{i_1,...i_n} = W_{w_1}^{i_1,...i_n}$. We can now chain these tensors as per the contraction requirement as below:

$$W_{w_3,w_1}^{i_1,...,i_n} = \sum_{i_1,...,i_n} W_{w_3,w_2}^{i_1,...,i_n} W_{w_2,w_1}^{j_1,...,j_n} \tag{12}$$

Now we will see an illustrative example where an efficient tensor contraction leads to an efficient calculation of operator weight distribution (histogram). We start with the holographic code (Fig.1). It has 20 free legs corresponding to physical qubits.

The weight tensor for this holographic code is shown in Fig.2(a). Now we have to strategically chain weight tensor W as per equation (6). Basically we want to efficiently contracts all free legs with the help of 'chained

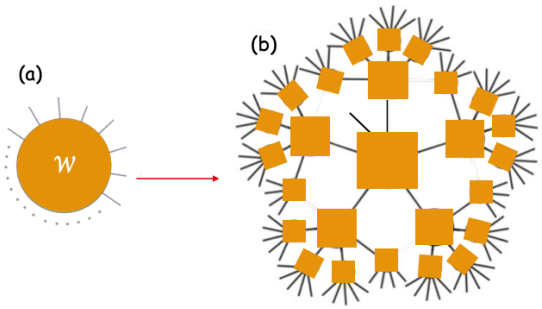


FIG. 10. Chaining of Weight tensor in order to have an efficient contraction with the code tensor

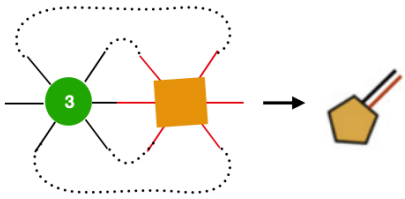


FIG. 11. Contraction of tensor indices between seed tensor and the weight tensor. Kindly notice that shapes and colors are only used to distinguish tensors. The number of free legs is the defining criteria for a tensor.

version' of weight tensor. This is possible if chaining is done while keeping structure of the tensor network of the code (Fig. 1) in mind. Fig.2 shows a way to realize this. Now if combine free legs of each tensors as shown in Fig.3, the new tensor emerges as shown in Fig.4. The result is a tensor with one free legs (i.e, vector). This vector corresponds to D_w, which is number of logical operator with w weight.

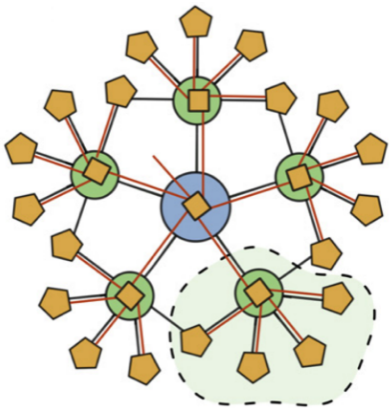


FIG. 12. Contraction of code tensor and weight tensor resulting into a vector D_w. Notice one (red) free leg at the center of the tensor structure implying it is a vector.

VI. DECODING VIA TENSOR NETWORK

In this section we will see how maximum likelihood decoding (MLD) can be estimated efficiently if we consider the contraction of the given tensor network code. The core strategy is somewhat similar to what we discussed in the last section. First we cast the problem into tensor formula. Then we look for any possible simplification of the tensor formula via contraction of the underlying tensors. In MLD problem, we measure syndrome $\vec{s}$. This corresponds to error $E(\vec{s})SL$, where symbols have their usual meaning. Now the task is to find the $'L'$ which maximizes the below quantity:

$$\chi(L,\vec{s}) = \sum_{S} prob(E(\vec{s})SL) \tag{13}$$

Now we formulate $prob(E(\vec{s})SL)$ in tensor form. We need to express SL and L in terms of pauli matrices as $SL = \sigma^{i_1} \otimes ... \otimes \sigma^{i_n}$ *and* $E(\vec{s}) = \sigma^{e_1} \otimes ... \otimes \sigma^{e_n}$. This naturally implies:

$$prob[E(\vec{s})SL] = prob[\sigma^{e_1}\sigma^{i_1} \otimes ... \otimes \sigma^{e_n}\sigma^{i_n}] \tag{14}$$

Now we can consistently define:

$$\mathcal{E}(\vec{s})^{i_1,\ \cdots\ ,i_n} =: prob(\sigma^{e_1}\sigma^{i_1} \otimes ... \otimes \sigma^{e_n}\sigma^{i_n}) \tag{15}$$

Finally we have tensor version of MLD estimation as below:

$$\chi(L,\vec{s}) = \sum_{S} prob[E(\vec{s}) \cdot SL] = T(L)_{i_1,...,i_n}\ \mathcal{E}(\vec{s})^{i_1,\ \cdots\ ,i_n} \tag{16}$$

Now there is a need to assess how efficiently we can simplify the above tensor estimation by utilising tensor contraction of the given code. The Quantum channel information can significantly impact the complexity of the problem. The best case is yield for uncorrelated quantum channel model. This assumption simplifies the term $prob(\sigma^{e_1}\sigma^{i_1} \otimes ... \otimes \sigma^{e_n}\sigma^{i_n}) = prob(\sigma^{e_1}\sigma^{i_1}) \cdot prob(\sigma^{e_2}\sigma^{i_2})....$ This converts the n-rank tensor(Fig.5) into n vectors:

$$\mathcal{E}(\vec{s})^{i_1,\ \cdots\ ,i_n} = prob(\sigma^{e_1}\sigma^{i_1}) \cdot prob(\sigma^{e_2}\sigma^{i_2})... \tag{17}$$

The decomposed n vectors (Fig.6) acts on the T(L) tensor to give a scalar quantity which correspond to $\chi(L,\vec{s})$. This technique has been found to be effective in an efficient MLD estimation for Holographic codes, (planar) surface codes and MERA tensor network. One can extend this idea for concatenated (or convolutional) codes, but they already have other exact decoding schemes. Table.1 summarizes the effectiveness of tensor network for the case of planar surface code and Holographic code.

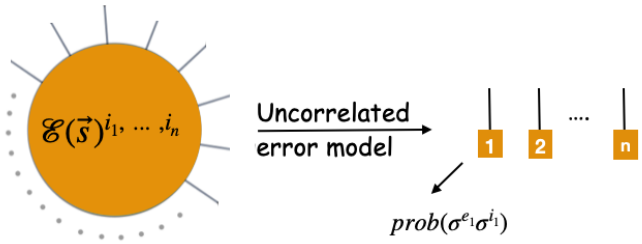


FIG. 13. Uncorrelated error assumption breaks $'n'$ rank tensor into bunch of n vectors.

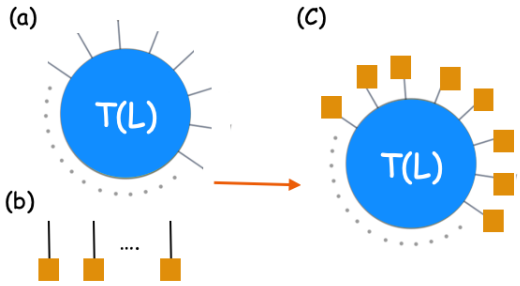


FIG. 14. (a) A model tensor code with n legs. (b) $\mathcal{E}(\vec{s})$ is decomposed as n vectors due to uncorrelated noise model. (c) Both of them combine to form a tensor with zero legs (a scalar corresponding to $\chi(L,\vec{s})$)

VII. TOPOLOGICAL CODES AS TENSOR NETWORK

In this section we will construct tensor network for rotated surface codes. This idea on surface code will ultimately be extended to doping tensor networks in the next section.

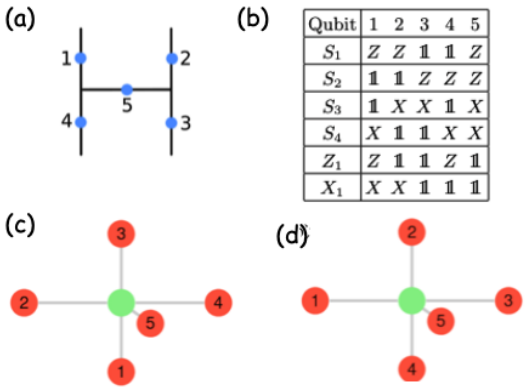
In rotated surface code, X and Z plaquette operators are arranged alternatively similar to black and white square on a chess board. We start with a fragment of a surface code (Fig.7(a)). It has four stabilizer generator (S_1, S_2, S_3, S_4) and a pair of logical operators(X_1, Z_1). Realizing a tensor out of this stabilizer is very similar to the strategy discussed in section II. But we briefly recap it here. We build two different but related tensors. They aid in desired contraction for building larger surface codes from these seed tensors. The first tensor (say, type-1) is inspired out of commuting operator set $(S_1, S_2, S_3, S_4, X_1)$ as shown in Fig.7(c). The second tensor (say, type-2) comes out of commuting operator set $(S_1, S_2, S_3, S_4, Z_1)$ as shown in Fig.7(d). Their unique orientation is to make sure the contracted tensor indeed lead us to a valid and larger surface code.

Now we use the above framework to design X-plaquette operator. It involves writing stabilizer generators in type-I and II form as per the requirement. Fig.8 illustrate how we get the two seed tensor code for the case of S_4. We repeat this procedure for S_2 to get two more seed codes.

TABLE II. Complexity of Maximum Likelihood Decoding via Tensor Network contraction [a]

Codes	MLD (Exhaustive computation)	MLD (Tensor Contraction)	Channel assumptions
Planar Codes [b]	$\sim \mathcal{O}(2^{n^2})$	$\sim \mathcal{O}(n^2)$	Uncorrelated error
Holographic Codes[c]	$\sim \mathcal{O}(2^{bn^2})$	$\sim \mathcal{O}(n^{max[\frac{2.37}{c},\ 1]})$	Uncorrelated error

[a] Source of Surface code data is S. Bravyi et.al (2014) Phys. Rev. A, 90:032326, 2014. Holographic code: Terry Farrelly et.al (2021) Phys. Rev. Lett. 127, 040507
[b] Here 'n' implies the number of physical qubits.
[c] Parameter b and c depends on actual code under study



Qubit	1	2	3	4	5
S_1	Z	Z	𝟙	𝟙	Z
S_2	𝟙	𝟙	Z	Z	Z
S_3	𝟙	X	X	𝟙	X
S_4	X	𝟙	𝟙	X	X
Z_1	Z	𝟙	𝟙	Z	𝟙
X_1	X	X	𝟙	𝟙	𝟙

FIG. 15. (a) A fragment of surface code [[5,1,2]]. (b) Generators for stabilizers and logical operators for the fragment surface code. (c) [Type-1] Tensor representing operator the commuting set (S_1, S_2, S_3, S_4 *and* X_1) (d) [Type-2] Tensor representing the commuting operator set (S_1, S_2, S_3, S_4 *and* Z_1). The numbered red balls represents physical qubits and its position. Take them as free legs of the tensor represented by green ball.

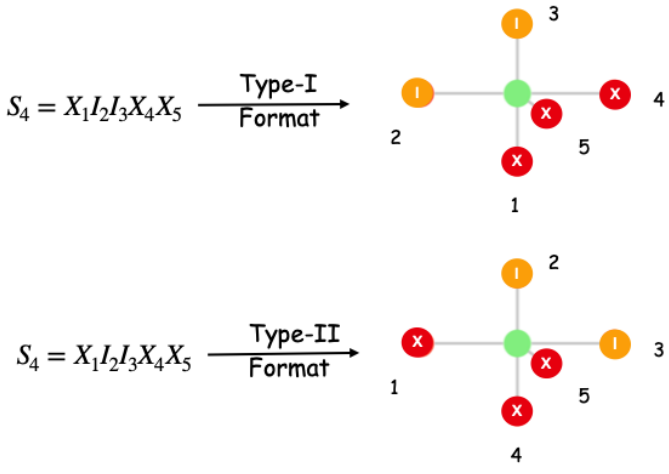


FIG. 16. (a) Writing stabilizer S_4 in terms of two types of tensor defined in Fig.7(c,d). Same operation is also done for S_2 for achieving final contraction mentioned in Fig.9

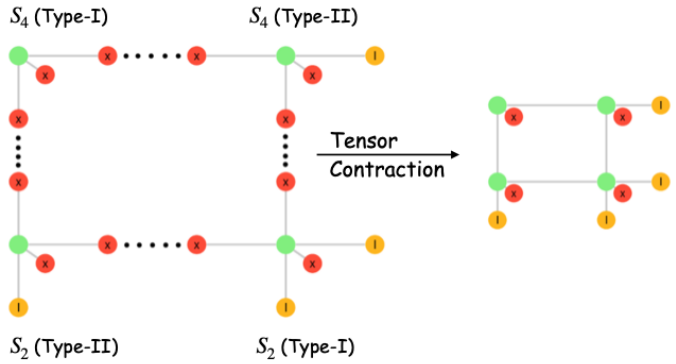


FIG. 17. We gather all four seed tensor to form X-plaquette operator. Red balls represent pauli-X while yellow goes for identity. Omitted legs correspond to legs with identity operator.

Now all four seed tensors are combined in suitable way to get X-plaquette operator (Fig.9).

To get Z-plaquette operator, we make use of generator S_1 and S_3 and repeat the procedure accordingly. This method can be extended to create a large stabilizer surface codes in similar fashion.

VIII. DOPING TENSOR NETWORKS

Now we will discuss the idea of tensor network doping and its potential benefit in error correction. Fig.18 is a tensor network for [[25,1,5]] rotated planar surface code. But one of the green ball (a tensor) has been replaced with blue ball (another tensor). The doped code is still a stabilizer code due to specific choice made while selecting the blue tensor. We choose the blue tensor to be a permutation of the legs of the original green tensor with single-Clifford unitaries applied to it. While such doping has no impact on code size and code distance but, it significantly alters the number of lowest-weight logical operators. Authors has compared the impact of such doping on successful error correction probability in Fig.19. He illustrate this effect for doped and un-doped versions of [[25,1,5]] and [[49,1,7]] codes. The key observation from the graph is the increase in the success rate for doped codes when compared to their un-doped counterpart. The maximum difference of 1.6% has been

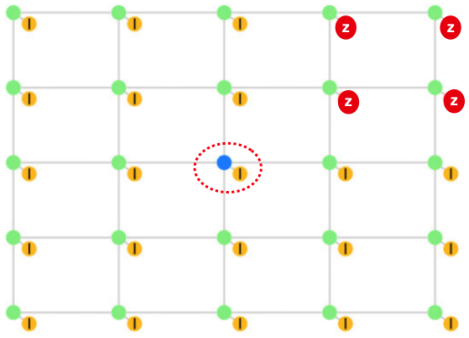


FIG. 18. [[25,1,5]] rotated planar surface code as a tensor network code. Each green (ball) tensor is five-qubit surface code as illustrated in Fig.15. We replaced one of the green tensor with blue tensor as a part of the doping (see red circle). This doped code has some enhancement in error correction capability than un-doped counterpart (as illustrated in Fig.19).

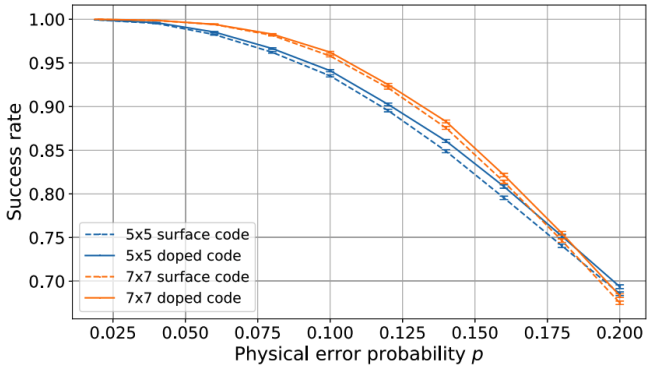


FIG. 19. [Performance of doped (dash-lined) and un-doped (Solid line) under depolarizing noise]. success rate implies to successful error correction probability while p means single-qubit error probability.

seen for [[49,1,7]] code at $p = 0.16$. He consider such enhancement for the doped codes could be due to decrease in number of lowest-weight logical operators. But this enhancement comes at a prize. Some of the stabilizers of the doped code now has higher weight than before. Such change in the weight of stabilizers might not be desired in certain cases like fault-tolerant setting, where higher-weight stabilizer is likely to cause more error due to imperfect measurement.

CONTRIBUTIONS OF AUTHORS

In the present manuscript, Harshkumar scripted sections I, II, III, IV, Appendix A, and Appendix B. Manish Kumar scripted V, VI, VII, and VIII. We have a common understanding of the whole manuscript through extensive discussion and literature reviews.

REFERENCES

[1] Victor V. Albert and Philippe Faist. Pastawski-yoshida-harlow-preskill (happy) code, 2022.

[2] Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Phys. Rev. A*, 90:032326, Sep 2014.

[3] Jacob C Bridgeman and Christopher T Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22):223001, may 2017.

[4] Terry Farrelly, Robert J. Harris, Nathan A. McMahon, and Thomas M. Stace. Tensor-network codes. *Phys. Rev. Lett.*, 127:040507, Jul 2021.

[5] Terry Farrelly, David K Tuckett, and Thomas M Stace. Local tensor-network codes. *New Journal of Physics*, 24(4):043015, apr 2022.

[6] Andrew J. Ferris and David Poulin. Tensor networks and quantum error correction. *Phys. Rev. Lett.*, 113:030501, Jul 2014.

[7] qecsim. The julia package implementation, 2022.

APPENDIX A

Fig. 8 shows an example of the generation of new [[9,3,3]] stabilizer code using three [[5,1,3]] codes. Authors in [5] have written a Julia package to do the underlying calculations and print tensor networks. The following code was used to generate the tensor networks during the iterations and to calculate code distance.

Julia code to generate the new code

```
using TensorNetworkCodes
using TensorNetworkCodes.TNDistance

# Start with the five-qubit code
tn_five_qubit = TensorNetworkCode(
    five_qubit_code())

# Contract with another five-qubit code (leg 5
# of the first code with leg 1 of the second)
new_code = contract(tn_five_qubit, tn_five_qubit
    , [[5,1]]);

# Prints left image of Fig. 8(b).
plot_code(new_code;use_coords=false)


# Contract with another five-qubit code
new_code = contract(new_code, tn_five_qubit,
    [[8,1]]);

# Prints center image of Fig. 8(b).
plot_code(new_code;use_coords=false)

# Contract two existing legs
new_code = fusion(new_code,[1,11])

# Prints the right (final code) image of Fig. 8(
    b).
plot_code(new_code;use_coords=false)
```

Code distance

```
dist = tn_distance(new_code)
```

```
n = num_qubits(new_code)
k = Int64(length(new_code.logicals)/2)

println("This is a [[$n,$k,$dist]] code!")
```

```
Output: This is a [[9,3,3]] code!
```

APPENDIX B

Suppose we have the two stabilizer-code tensors $T(L)_{i_1,\ldots,i_n}$ and $T'(L')_{j_1,\ldots,j_{n'}}$ which have n and n' physical qubits and k and k' logical qubits, respectively. Our goal is to show that the tensors

$$T_{new}(L\otimes L')_{i_{l+1},\ldots,i_n,j_{l+1},\ldots,j_{n'}} = \sum_{k_1,\ldots,k_l\in\{0,1,2,3\}} T(L)_{k_1,\ldots,k_l,i_{l+1},\ldots,i_n} T(L')_{k_1,\ldots,k_l,j_{l+1},\ldots,j_{n'}} \tag{18}$$

describe a new stabilizer code provided that at least one of the codes $T(L)$ or $T'(L')$ can distinguish any Pauli error on the set of qubits corresponding to the contracted legs.

For simplicity, we will first show that $T_{new}(L\otimes L')$ describes a valid stabilizer code if only one index is being contracted (i.e., $l=1$), then it will be generalized to any arbitrary l. Without loss of generality, we can assume that tensor $T(L)$ can distinguish arbitrary Pauli errors on the qubits corresponding to the contracted legs.

For $T_{new}(L\otimes L')$ to describe a valid stabilizer code, we need to prove the following three properties:

(i) The elements of $T_{new}(L\otimes L')$ must be either zero or one as in equation (4).

(ii) $T_{new}(I\otimes I')$ must describe a stabilizer group of $n+n'-2$ physical qubits with $k+k'$ logical qubits.

(iii) $T_{new}(L\otimes L')$ must describe the logical coset of $L\otimes L'$ with the correct commutation relations between all representatives of logical operators. E.g., all stabilizers should commute, meaning all operators described by $T_{new}(I\otimes I')$ should commute.

Proof of (i): As the entries of $T(L)$ and $T(L')$ are either zero or one, from (18), it is clear that $T_{new}(L\otimes L')$ can take only positive integer values for any set of indices. Suppose an entry of $T_{new}(L\otimes L')$ is greater than one. This means, for some values of indices $i_1,\ldots,i_n$ and some L, the tensor $T(L)_{k,i_2,\ldots,i_n}=1$ for more than one value of the index k. Then both $\sigma^k\otimes\sigma^{i_2}\otimes\ldots\otimes\sigma^{i_n}$ and $\sigma^{k'}\otimes\sigma^{i_2}\otimes\ldots\otimes\sigma^{i_n}$ with $k\neq k'$ must be stabilizers of logical operators. Taking the product of these two operators, we find that $\sigma^{k''}\otimes I\otimes\ldots\otimes I$ is also a stabilizer or logical operator for some $k''\neq 0$. But any code having this property cannot detect the single-qubit error $\sigma^{k''}$ on qubit 1 since every stabilizer will commute with $\sigma^{k''}\otimes I\otimes\ldots\otimes I$, so we have a contradiction. This means the entries of $T_{new}(L\otimes L')$ can only be zero or one.

Proof of (ii): It is useful to think in terms of stabilizers of $T(L)$ and $T'(L')$ before contracting the tensor indices. We need to find which stabilizers survive the contraction: exactly those that match on qubits 1 to $T(L)$ and 1' to $T'(L')$. As $T(L)$ can distinguish an arbitrary error on qubit 1, we can choose its stabilizer generators to have a useful form. We choose exactly one stabilizer generator S_1 to have a Pauli X on qubit 1 and exactly one stabilizer generator S_2 to have a Pauli Z on qubit 1, with the rest having identities on qubit 1. For any stabilizer generator in $T'(L')$, $S_j^{'}$, which acts like σ^k on its qubit 1', we can choose $\mu_1,\mu_2\in\{0,1\}$ such that $P_j=(S_1)^{\mu_1}(S_2)^{\mu_2}$ acts like σ^k on qubit 1 of $T(L)$. The only stabilizer generators of the tensor product of both codes $T(L)$ and $T'(L')$ that survive the index contraction are then $P_j\otimes S_j^{'}$ for all j and $S_i\otimes I'$ for $i\in\{3,\ldots,n-k\}$. This means there are two fewer stabilizer generators after contraction. Before contraction, the stabilizer group of the two codes had $n-k+n'-k'$ generators, and after contraction, the stabilizer group of the new code has $n-k+n'-k'-2$ generators. In the new code, there are $n+n'-2$ physical qubits, so the number of logical qubits must be $k+k'$.

Proof of (iii): Consider any representatives of two operators $L_1\otimes L_1'$ and $L_2\otimes L_2'$ on the new code, which can be written as $\sigma^{\underline{a}}\otimes\sigma^{\underline{b}}$ and $\sigma^{\underline{c}}\otimes\sigma^{\underline{d}}$, where $\sigma^{\underline{a}}$ is shorthand for $\sigma^{a_2}\otimes\ldots\otimes\sigma^{a_n}$, which acts on first $n-1$ qubits and $\sigma^{\underline{b}}$ is shorthand for $\sigma^{b_n}\otimes\ldots\otimes\sigma^{b_{n+n'-1}}$, which acts on first qubits n to $n+n'-2$. These follow the algebraic relation,

$$\begin{aligned}(L_1\otimes L_1')(L_2\otimes L_2') &= (\sigma^{\underline{a}}\otimes\sigma^{\underline{b}})(\sigma^{\underline{c}}\otimes\sigma^{\underline{d}})\\ &= z(\sigma^{\underline{c}}\otimes\sigma^{\underline{d}})(\sigma^{\underline{a}}\otimes\sigma^{\underline{b}}) = z(L_2\otimes L_2')(L_1\otimes L_1')\end{aligned} \tag{19}$$

for some $z\in\{1,-1\}$. Before contraction, the corresponding operators are $(\sigma^l\otimes\sigma^{\underline{a}})\otimes(\sigma^l\otimes\sigma^{\underline{b}})$ representing $L_1\otimes L_1'$ and $(\sigma^k\otimes\sigma^{\underline{c}})\otimes(\sigma^k\otimes\sigma^{\underline{d}})$ representing $L_2\otimes L_2'$ for some $l,k\in\{0,1,2,3\}$. These obey the same algebraic relation,

$$\begin{aligned}&[(\sigma^l\otimes\sigma^{\underline{a}})\otimes(\sigma^l\otimes\sigma^{\underline{b}})][(\sigma^k\otimes\sigma^{\underline{c}})\otimes(\sigma^k\otimes\sigma^{\underline{d}})]\\ &= z[(\sigma^k\otimes\sigma^{\underline{c}})\otimes(\sigma^k\otimes\sigma^{\underline{d}})][(\sigma^l\otimes\sigma^{\underline{a}})\otimes(\sigma^l\otimes\sigma^{\underline{b}})]\end{aligned} \tag{20}$$

because σ^l and σ^k both appear twice, so whether they commute or anti-commute does not matter. So, this verifies that the stabilizer group, described by $T_{new}(I\otimes I')$, is abelian. It also implies that any representation of the logical group of operators has the correct algebraic structure, e.g., if each code $T(L)$ and $T'(L')$ had a single logical qubit, then, for example, $X\otimes Z'$ would anti-commute with $Y\otimes Z'$.

Generalization to many qubits: If multiple indices are being contracted, all the same arguments apply except that we need to choose the stabilizer generators to have the following form on the first l qubits of the code $T(L)$. Stabilizer generators are chosen such that,

$$\begin{aligned} S_1 &= \sigma^1 \otimes I \otimes \ldots \otimes I \otimes A_1 \\ S_2 &= \sigma^3 \otimes I \otimes \ldots \otimes I \otimes A_2 \\ S_3 &= I \otimes \sigma^1 \otimes I \otimes \ldots \otimes I \otimes A_3 \\ S_4 &= I \otimes \sigma^3 \otimes I \otimes \ldots \otimes I \otimes A_4 \\ &\vdots \\ S_{2l-1} &= I \otimes \ldots \otimes I \otimes \sigma^1 \otimes A_{2l-1} \\ S_{2l} &= I \otimes \ldots \otimes I \otimes \sigma^3 \otimes A_{2l}, \end{aligned} \tag{21}$$

where are A_i are tensor products of Paulis on qubits $l+1$ to n. This means that for any combination of Pauli operators on the first $2l$ stabilizers, there is a unique product of the first $2l$ stabilizers that has that form on those qubits.

The stabilizer generators can always be chosen to have this form if the code $T(L)$ can distinguish all Pauli errors on qubits $1tol$. To see this, consider the errors,

$$\begin{aligned} E_1 &= \sigma^3 \otimes I \ldots \\ E_2 &= \sigma^1 \otimes I \ldots \\ &\vdots \\ E_{2l-1} &= I \otimes \ldots \otimes I \otimes \sigma^3 \otimes I \ldots \\ E_{2l} &= I \otimes \ldots \otimes I \otimes \sigma^1 \otimes I \ldots \end{aligned} \tag{22}$$

Given any set of stabilizer generators S_i, at least one will anti-commute with E_1 (otherwise, they could not detect that error). Relabel the indices of the stabilizer generators such that S_1 anti-commutes with E_1, and for any other generator S_i that also anti-commutes with E_1 make the replacement $S_i \to S_i S_1$, so that the only stabilizer that anti-commutes with E_1 is S_1. Repeat this procedure for E_2, so that the only stabilizer anti-commuting with E_2 is S_2. Repeat this for each E_i up to $i = 2l$. Finally, because, e.g., S_1 commutes with all E_i except E_1, it must have the form $\sigma^1 \otimes I \otimes \ldots \otimes I \otimes A_1$, where A_1 is some tensor product of Paulis on qubits $l+1$ to n, and similarly for the other stabilizers.