Finding the disjointness of stabilizer codes is NP-complete

John Bostanci[1] and Aleksander Kubica[2, 1, 3, 4]

[1]*Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada*
[2]*Perimeter Institute for Theoretical Physics, Waterloo, ON N2L 2Y5, Canada*
[3]*AWS Center for Quantum Computing, Pasadena, CA 91125, USA*
[4]*California Institute of Technology, Pasadena, CA 91125, USA*

The disjointness of a stabilizer code is a quantity used to constrain the level of the logical Clifford hierarchy attainable by transversal gates and constant-depth quantum circuits. We show that for any positive integer constant c, the problems of calculating the c-disjointness and approximating it to within a constant multiplicative factor are NP-complete. We provide bounds on the disjointness for various code families, including the CSS codes, concatenated codes and hypergraph product codes. We also describe numerical methods of finding the disjointness, which can be readily used to rule out the existence of any transversal gate implementing some non-Clifford logical operation in small stabilizer codes. Our results indicate that finding fault-tolerant logical gates for generic quantum error-correcting codes is a computationally challenging task.

Designing fault-tolerant schemes is an essential step toward scalable universal quantum computation [1–4]. To protect quantum information from the detrimental effects of noise one typically encodes it into an quantum error-correcting code. In addition to reliably storing quantum information, one also seeks to perform fault-tolerant logical operations on the encoded information.

One of the simplest ways to realize fault-tolerant logical operations is via transversal gates, which act independently on individual physical qubits and thus do not spread errors in an uncontrollable way. Recently, many works have been devoted to transversal gates implementing non-Clifford logical operations in topological quantum codes [5–11] and the consequent universal quantum computation schemes [12–17]. Transversal gates also prove useful for magic state distillation [18–20], as they form the backbone of many distillation protocols; see Ref. [21] and the references therein.

Logical operations implemented via transversal gates are somewhat limited. Namely, the computational universality of transversal gates is incommensurate with the capability of the underlying code to correct errors, as exemplified by the Eastin-Knill theorem [22, 23], and its approximate versions [24–26]. More generally, bounded-spread logical operators, which propagate errors in a benign way and include constant-depth quantum circuits and locality-preserving operators, are also computationally limited [27–31].

Although systematic approaches to finding transversal gates for generic quantum error-correcting codes are not known, for stabilizer codes [32] we can rule out the possibility of implementing certain logical operations via transversal gates. Namely, following Ref. [30] we can find the following upper bound

$$M \leq \lfloor \log_\Delta (d_\uparrow / d_\downarrow) \rfloor + 2 \tag{1}$$

on the M^{th} level of the logical Clifford hierarchy [33] attainable by transversal gates if we know the min-distance $d_\downarrow$, max-distance $d_\uparrow$ and disjointness Δ of the stabilizer code. The disjointness, roughly speaking, captures the maximal number of non-overlapping representatives of any given non-trivial logical Pauli operator. Until now, however, the problem of finding the disjointness as well as its computational hardness have not been explored.

In our work we focus on the problem of finding the disjointness of stabilizer codes, which serves as a proxy to understanding what are the admissible fault-tolerant logical gates. First, in Sec. II we show that for any positive integer c it is NP-hard to calculate the c-disjointness, as well as to approximate it to within a constant multiplicative factor. Our result thus indicates that finding fault-tolerant logical gates that can be implemented with generic quantum error-correcting codes is a computationally challenging task. Then, in Sec. III we discuss numerical methods of finding the disjointness, which we illustrate with the example of the $[\![14, 3, 3]\!]$ stabilizer code [34]. We also provide a strengthening of the bound in Eq. (1), which subsequently rules out the existence of any transversal gate implementing some non-Clifford logical operation in the aforementioned $[\![14, 3, 3]\!]$ stabilizer code. Lastly, in Sec. IV we provide bounds on the disjointness for various code families, including the CSS codes [35, 36], concatenated codes [37] and hypergraph product codes [38].

I. PRELIMINARIES

In this section, we briefly discuss basic constructions of stabilizer codes, as well as the notions of code distance and disjointness. We also comment on certain graph-theory problems and their computational complexity.

A. Distance and disjointness

Let $\overline{L} \in \mathcal{L}$ be any non-trivial logical Pauli operator for the stabilizer code $\mathcal{S}$. Following Ref. [30], we define the distance $d(\overline{L})$ to be the size of the support of the smallest representative of $\overline{L}$, i.e.,

$$d(\overline{L}) = \min_{L \in \overline{L}} |\operatorname{supp} L|, \tag{2}$$

and introduce the notions of the min-distance and max-distance as follows

$$d_{\downarrow} = \min_{\overline{L}\in\mathcal{L}} d(\overline{L}), \quad d_{\uparrow} = \max_{\overline{L}\in\mathcal{L}} d(\overline{L}). \tag{3}$$

Note that the min-distance is the same as the standard stabilizer code distance.

Let $\mathcal{A} \subseteq \overline{L}$ be a subset of representatives of $\overline{L}$ or, more generally, a multiset that allows for multiple instances for each of the representatives of $\overline{L}$. We say that $\mathcal{A}$ is a c-disjoint collection of representatives of $\overline{L}$, where c is a positive integer, iff for every qubit there are at most c elements of $\mathcal{A}$ that are supported on that qubit. We then define the c-disjointness $\Delta_c(\overline{L})$ to be the size of the largest c-disjoint collection for $\overline{L}$, divided by c, i.e.,

$$\Delta_c(\overline{L}) = \frac{1}{c}\max_{\mathcal{A}\subseteq\overline{L}}\{|\mathcal{A}| : \text{ at most } c \text{ elements } L\in\mathcal{A} \tag{4}$$

$$\text{are supported on any qubit}\}. \tag{5}$$

Subsequently, the disjointness $\Delta(\mathcal{S})$ of the stabilizer code $\mathcal{S}$ is defined as follows

$$\Delta(\mathcal{S}) = \sup_{c\geq 1}\min_{\overline{L}\in\mathcal{L}} \Delta_c(\overline{L}). \tag{6}$$

Note that the disjointness defined here, which allows representatives of $\overline{L}$ to be picked multiple times, is greater or equal to the disjointness defined in Ref. [30]. Thus, by using our definition in Eq. (1) we may obtain a tighter bound on the level of the logical Clifford hierarchy attainable by transversal gates.

In Sec. III we will show that supremum over c can be achieved by a finite c, although this fact is not obvious from the definition of $\Delta(\mathcal{S})$. For convenience, we will define $\Delta_c(\mathcal{S}) = \min_{\overline{L}\in\mathcal{L}} \Delta_c(\overline{L})$, and $\Delta(\overline{L}) = \sup_{c\geq 1} \Delta_c(\overline{L})$.

Finally, for any positive integer c we introduce the following decision problem based on the c-disjointness.

> **c-DISJOINTNESS**
> **Input**: A $(n-k)\times 2n$ binary matrix specifying an $[\![n,k,d]\!]$ stabilizer code $\mathcal{S}$, a string of $2n$ bits representing a logical Pauli operator $\overline{L}$ and a positive integer a.
> **Question**: Is the size of the largest c-disjoint collection of representatives of $\overline{L}$ greater or equal to a, i.e., $c\Delta_c(\overline{L}) \geq a$?

B. Stabilizer code constructions

Stabilizer codes are an important class of quantum error-correcting codes. A stabilizer code is defined by its stabilizer group $\mathcal{S}$, i.e., an Abelian subgroup of the Pauli group that does not contain $-I$. In what follows we identify the stabilizer code with its stabilizer group. The code space of the stabilizer code $\mathcal{S}$ is the simultaneous $(+1)$-eigenspace of all of the stabilizer operators. We denote by $[\![n,k,d]\!]$ a stabilizer code that encodes k logical qubits into n physical qubits and has code distance d. To specify the $[\![n,k,d]\!]$ stabilizer code, we can provide a binary matrix of size $(n-k)\times 2n$, whose rows correspond to independent stabilizer generators of $\mathcal{S}$. For concreteness, we identify a bit string $(b_1,\ldots,b_{2n}) \in \{0,1\}^{2n}$ with the following Pauli operator $\bigotimes_{i=1}^n X_i^{b_i} Z_i^{b_{i+n}}$, where P_i denotes a Pauli $P\in\{X,Z\}$ operator acting on qubit i.

For any stabilizer code, logical Pauli operators, which are the elements of the the normalizer of the stabilizer group in the Pauli group, can always be implemented as tensor products of single-qubit Pauli operators. We write $\overline{L}$ to represent a logical Pauli operator itself, as well as the set of its equivalent representatives. Also, we write $\mathcal{L}$ to denote the set of all non-trivial logical Pauli operators.

CSS codes.—A stabilizer code is a CSS code if there exists a choice of stabilizer generators such that every generator is either a Pauli X- or Z-type operator, i.e., a tensor product of single-qubit Pauli X or Z operators. We remark that for CSS codes there always exists a choice of logical Pauli $\overline{X}$ and $\overline{Z}$ operators, such that their representatives are some Pauli X- and Z-type operators, respectively.

Concatenated stabilizer codes.—Given two stabilizer codes $\mathcal{S}_1$ and $\mathcal{S}_2$ with parameters $[\![n_1,k_1,d_1]\!]$ and $[\![n_2,1,d_2]\!]$, respectively, we can concatenate them to obtain a new stabilizer code, which we denote by $\mathcal{S}_1\rhd\mathcal{S}_2$. To construct the concatenated code $\mathcal{S}_1\rhd\mathcal{S}_2$, we first encode the logical information into the stabilizer code $\mathcal{S}_1$. Then, we further encode each and every qubit of stabilizer code $\mathcal{S}_1$ into the stabilizer code $\mathcal{S}_2$, where every qubit of $\mathcal{S}_1$ is identified with a block of n_2 qubits of $\mathcal{S}_1\rhd\mathcal{S}_2$. The concatenated code $\mathcal{S}_1\rhd\mathcal{S}_2$ is a $[\![n_1n_2,k_1,d]\!]$ code, where $d\geq d_1d_2$.

A set of logical Pauli operators $\mathcal{C}$ is called a choice of logical Pauli operators for $\mathcal{S}_2$, if $\overline{P}\neq\overline{O}$ for all $P,O\in\mathcal{C}$ and $|\mathcal{C}|=4$ (or more generally $|\mathcal{P}|=4^k$ for a code with k logical qubits). For a n_1 qubit operator O, we define $O\rhd\mathcal{C}$ to be the result of replacing each single qubit Pauli operator of O with the corresponding choice of logical Pauli operator from $\mathcal{C}$ on the block of n_2 qubits identified with the qubit. We say that a collection of choices of logical Pauli operators for $\mathcal{S}_2$ is c-disjoint if the representatives of the same logical Pauli in each choice forms a c-disjoint collection for all choices logical Pauli operators of $\mathcal{S}_2$, and the choice of logical Identity is always $\mathbb{I}^{n_2}$.

One choice for the stabilizer group of $\mathcal{S}_1\rhd\mathcal{S}_2$ is

$$\mathcal{S}_1\rhd\mathcal{S}_2 = \Big\langle \mathbb{I}^{\otimes(j-1)n_1}\otimes S_2\otimes\mathbb{I}^{\otimes(n-j)n_1}, \\ S_1\rhd\mathcal{C} \;\Big|\; S_1\in\mathcal{S}_1, S_2\in\mathcal{S}_2, 1\leq j\leq n\Big\rangle, \tag{7}$$

for a choice of logical Pauli operator of $\mathcal{S}_2$, $\mathcal{C}$. Furthermore every logical Pauli operator of $\mathcal{S}_1\rhd\mathcal{S}_2$ is equivalent, up to a stabilizer element, to $L\rhd\mathcal{C}$ for some logical Pauli L of $\mathcal{S}_1$ and some choice of logical Pauli operator $\mathcal{C}$ for $\mathcal{S}_2$.

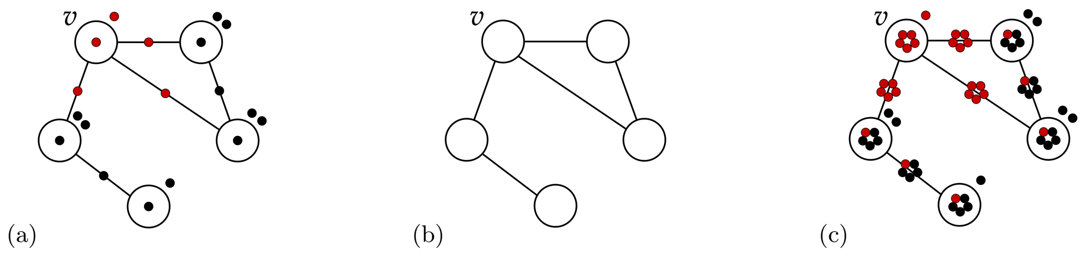

FIG. 1. (a) A graph $G = (V, E)$ can be used to define a CSS stabilizer code $\mathcal{S}_c^G$. In (b) and (c), we illustrate the construction for $c = 1$ and $c = 2$, respectively. Qubits (black and red dots) are placed at vertices and on edges of the graph G. We depict in red the support of the representative $X(v)$ of the logical Pauli operator $\overline{L}_c^G$ that is associated with the vertex $v \in V$.

Hypergraph product codes.—Given two binary matrices H_1 and H_2 of size $m_1 \times n_1$ and $m_2 \times n_2$, respectively, the corresponding hypergraph product code is specified by the following binary matrix

$$\left(\begin{array}{c|c} H_1 \otimes I_{m_2},\ I_{m_1} \otimes H_2 & 0_{m_1 m_2 \times (n_1 m_2 + n_2 m_1)} \\ \hline 0_{n_1 n_2 \times (n_1 m_2 + n_2 m_1)} & I_{n_1} \otimes H_2^{\mathrm{T}},\ H_1^{\mathrm{T}} \otimes I_{n_2} \end{array}\right), \quad (8)$$

where H^{T} denotes the transpose of H, $0_{a\times b}$ and I_a are the zero matrix and the identity matrix of size $a \times b$ and $a \times a$, respectively. Note that hypergraph product codes are CSS codes.

C. Computational complexity and graph theory

NP is the complexity class of problems that can be solved in polynomial time using a non-deterministic Turing machine, which can perform multiple operations at the same time in parallel at every time-step, and accepts if any one of the parallel operations leads to an accepting state. A problem is NP-hard if every problem in NP can be reduced to it in polynomial time, and a problem is NP-complete if it is both NP-hard and in NP.

Let $G = (V, E)$ be a graph with the sets of vertices V and edges E. We say that a subset of vertices $V' \subseteq V$ is an independent set for the graph G iff no two vertices in V' are joined by an edge in E. Moreover, we say that a collection $\mathcal{A} = \{A | A \subseteq V\}$ comprising subsets of the vertices of G is an independent collection for G iff for every pair of different $A, A' \in \mathcal{A}$, no two vertices belonging to different subsets are joined by an edge in E. Note that an independent set is a special case of an independent collection. We denote the size of a maximum independent set for G by $\alpha(G)$ and refer to it as the independence number of G. Then, the following decision problem is NP-complete [39].

INDEPENDENT SET
Input: A graph $G = (V, E)$ and a positive integer a.
Question: Is the independence number of G greater or equal to a, i.e., $\alpha(G) \geq a$?

Furthermore, the problem of approximating $\alpha(G)$ up to a multiplicative factor of $|V|^{1-\epsilon}$ is NP-hard [40, 41]. We use this result in our reduction in the following section.

II. HARDNESS OF FINDING AND APPROXIMATING THE c-DISJOINTNESS

We start this section by constructing a CSS stabilizer code $\mathcal{S}_c^G$ and logical Pauli operator $\overline{L}_c^G$ for any finite connected graph G and positive integer c. Then, we show that the c-disjointness $\Delta_c\left(\overline{L}_c^G\right)$ can be related to the independence number $\alpha(G)$ of the graph G. This, in turn, allows us prove our main theorem by reducing the the problem of finding the maximum independent set of G to the problem of finding the c-disjointness $\Delta_c\left(\overline{L}_c^G\right)$.

A. Constructing the CSS stabilizer code

Let $G = (V, E)$ be a graph and c be a positive integer constant. To define a CSS stabilizer code $\mathcal{S}_c^G$ associated with the graph G, we first place $\binom{|V|}{c-1}$ qubits on every vertex $v \in V$ and on every edge $e \in E$; see Fig. 1 for an illustrative example. We label each qubit by a pair (v, ν) or (e, ν), where $\nu \subseteq [|V|]$ is a subset of $c-1$ integers from $[|V|] = \{1, \ldots, |V|\}$. Additionally, we place one or two additional qubits at each vertex v depending on whether $\binom{|V|}{c-1}(\deg v + 1)$ is odd or even respectively, where $\deg v$ denotes the degree of v. We label those additional qubits by a pair (v, i), where $i \in \{1, 2\}$. We denote by P_q a Pauli operator $P = X, Z$ supported on qubit with label q. We denote by $Q(v)$ and $Q(e)$ all the qubits placed at the vertex v and on the edge e, i.e., all the qubits with labels $(v, \cdot)$ and $(e, \cdot)$, respectively. We also denote by Q the set of all the qubits, i.e.,

$$Q = \bigcup_{v\in V} Q(v) \cup \bigcup_{e\in E} Q(e). \quad (9)$$

The stabilizer group $\mathcal{S}_c^G$ associated with G and c is

$$\mathcal{S}_c^G = \left\langle X(u)X(v), \prod_{q\in Q} Z_q \,\middle|\, u, v \in V \right\rangle, \quad (10)$$

where $X(v)$ denotes a Pauli X operator associated with a vertex $v \in V$, which we define as follows

$$X(v) = \prod_{q\in Q(v)} X_q \prod_{e \ni v} \Bigg(\prod_{q \in Q(e)} X_q \Bigg). \tag{11}$$

Note that for any $v \in V$ the Pauli X operator $X(v)$ is supported on the even number of qubits. Thus, $X(v)$ and $\prod_{q\in Q} Z_q$ commute and $\mathcal{S}_c^G$ is an Abelian subgroup of the Pauli group satisfying $-I \notin \mathcal{S}_c^G$. Moreover, for any $u, v \in V$ the Pauli X operators $X(u)$ and $X(v)$ are the representatives of the same non-trivial logical Pauli operator, which we denote by $\overline{L}_c^G$. The last statement follows from the fact that $X(u)$ and $X(v)$ commute with the all stabilizer operators, are not contained in the stabilizer group $\mathcal{S}_c^G$, and their product forms a stabilizer operator $X(u)X(v)$. For convenience, for any subset of vertices $A \subseteq V$ we define

$$X(A) = \prod_{v\in A} X(v). \tag{12}$$

Note that if $|A| \equiv 0 \mod 2$, then $X(A)$ is a stabilizer operator; otherwise, $X(A)$ is a representative of the logical Pauli operator $\overline{L}_c^G$.

We remark that if the graph G has at least $c+2$ vertices, i.e., $|V| \geq c+2$, then the stabilizer code $\mathcal{S}_c^G$ associated with G and c is error-detecting, i.e., its min-distance satisfies $d_\downarrow > 1$. This, in turn, implies that the disjointness of $\mathcal{S}_c^G$ is greater than one, i.e., $\Delta\left(\mathcal{S}_c^G\right) > 1$; see Lemma 2(ii) in Ref. [30]. To establish the claim that $d_\downarrow > 1$, it suffices to check that every single-qubit Pauli operator anticommutes with some stabilizer operator from $\mathcal{S}_c^G$. Since the stabilizer operator $\prod_{q\in Q} Z_q$ is supported on every qubit, any single-qubit Pauli X or Y operator anticommutes with it. For any vertex $v \in V$ a Pauli Z operator on qubit (v,i) or (v,ν) anticommutes with a stabilizer operator $X(u)X(v)$ for any vertex $u \in V \setminus \{v\}$ or $u \in V \setminus (\{v\} \cup \nu)$, respectively. Lastly, for any edge $e \in E$ a Pauli Z operator on qubit (e, ν) anticommutes with a stabilizer operator $X(u)X(v)$ for any vertex $u \in V \setminus (\{v,w\} \cup \nu)$, where $v, w \in V$ are two vertices incident to e. Note that we use $|V| \geq c+2$ to guarantee that the set $V \setminus (\{v,w\}\cup\nu)$ is non-empty.

B. Relating the c-disjointness to the independence number

Let $\mathcal{S}_c^G$ and $\overline{L}_c^G$ be a stabilizer code and a logical Pauli operator, which are associated with the graph $G=(V,E)$ and the positive integer c. We abuse terminology and say that a collection $\mathcal{A} = \{A | A \subseteq V\}$ comprising subsets of the vertices of G is a c-disjoint collection for $\overline{L}_c^G$ iff $\{\prod_{v\in A} X(v) | A \in \mathcal{A}\}$ is a c-disjoint collection for $\overline{L}_c^G$, and that a subset $A \subseteq V$ of vertices acts on a qubit if $\prod_{v\in A} X(v)$ acts on the qubit. We remark that $\mathcal{A}$ is allowed to be a multiset.

Now, we establish two technical lemmas.

Lemma 1. *Let $G=(V,E)$ be a graph and c be a positive integer. For any independent set $V' \subseteq V$, the following collection*

$$\mathcal{A} = \begin{cases} \{\{v\} | v \in V'\} & \text{if } |V'|(c+1) \equiv 0 \mod 2, \\ \{\{v\}|v \in V'\} \cup \{V'\} & \text{otherwise,} \end{cases} \tag{13}$$

is a c-disjoint collection for the logical operator $\overline{L}_c^G$ of the stabilizer code $\mathcal{S}_c^G$.

Proof. We first show that $\{\{v\}|v \in V'\}$ is a c-disjoint collection. Every qubit labeled (v,i) will be supported at most 1 operator (if $v \in V'$), every qubit labeled (v,ν) will have one overlapping operator for every vertex in $V' \cap (\{v\}\cup\nu)$, which has size at most c because ν is taken to be size $c-1$. Similarly, any qubit labeled $(\{u,v\},\nu)$ will have one overlapping operator for every vertex in $V' \cap (\{u,v\} \cup \nu)$. Because V' is independent, it contains at most 1 of u or v so the size of this set is also at most c.

It remains to be seen that when $|V'|$ is odd and c is even (the second case in Eq. (13)), we can add $\{V'\}$ and still form a c-disjoint collection. Because $|V'|$ is odd, the set represents a valid representative of the logical Pauli. Any qubit that had strictly fewer than c operators from $\{\{v\}|v \in V'\}$ acting on it will have at most c operators from $\{\{v\}|v \in V'\} \cup \{V'\}$ acting on it. Furthermore, because c must be an even positive integer, $c \geq 2$, so all qubits labeled (v,i) have fewer than c operators from $\{\{v\}|v\in V'\}\cup\{V'\}$ acting on it.

Thus, we restrict our attention to the qubits that have c many operators from $\{\{v\}|v \in V'\}$ acting on them. These qubits are exactly those labeled (v,ν) where $|V' \cap (\{v\}\cup\nu)| = c$ and $(\{u,v\},\nu)$ where $|V'\cap(\{u,v\}\cup\nu)| = c$. Since V' is an independent set, at most one of u, v can be in V' for every edge $\{u,v\}$.

Consider any qubit $q = (v,\nu)$ such that $V'\cap(\{v\}\cup\nu) = (\{v\}\cup\nu)$, which has size c. The operator corresponding to V' is $\prod_{u\in V'} X(u)$, where the action of each $X(u)$ on q is X_q. Because c is even, the action of $\prod_{u\in V'} X(u)$ on q is $X_q^c = I_q$. So, at most c operators from $\{\{v\}|v \in V'\} \cup \{V'\}$ act on q.

Similarly, for any qubit labeled $(\{u,v\},\nu)$, such that (without loss of generality) $V' \cap (\{u,v\}\cup\nu) = (\{v\}\cup\nu)$, the action of V' on the qubit is I since c is even. Thus, at most c operators from $\{\{v\}|v\in V'\}\cup\{V'\}$ act on the qubit. Therefore, $\mathcal{A}$ defined in Eq. (13) is a c-disjoint collection. □

Lemma 2. *Let c be a positive integer and $G=(V,E)$ be a connected graph, such that $\alpha(G) \geq c^2(c-1)/2$. Then, the c-disjointness $\Delta_c\Big(\overline{L}_c^G\Big)$ for the logical operator $\overline{L}_c^G$ of the stabilizer code $\mathcal{S}_c^G$ is given by*

$$\Delta_c\Big(\overline{L}_c^G\Big) = (\alpha(G)+b)/c, \tag{14}$$

where $b = \alpha(G)(c+1) \mod 2$.

Proof. The previous lemma gave a lower bound on the disjointness in the form of an explicit c-disjoint collection, so to prove this lemma we need to find an upper bound on the disjointness. Let $\mathcal{A}$ be a maximal c-disjoint collection of representatives of $\overline{L}_c^G$. We will show that there exists an independent collection of representatives of size $|\mathcal{A}|-b$, but first we will show that there exists an independent sub-collection of $\mathcal{A}$ with size greater than $c+1$. Consider the following sequence of sub-collections of $\mathcal{A}$, starting with $A_1 = \mathcal{A}$.

1. Select $L_i = \arg\min_{L\in A_i} |L|$
2. Pick any $v_i \in L_i \setminus \bigcup_{j=1}^{i-1} L_j$
3. $A_{i+1} = \{L \in A_i | L\setminus \bigcup_{j=i}^{i} L_j \neq \emptyset \text{ and } v_i \notin L\}$

Because $\mathcal{A}$ is a c-disjoint collection, each vertex v_i appears in at most c many L_i, and there can be at most c many operators completely contained in $\bigcup_{j=i}^{i} L_j$. Thus, $|A_{i+1}| \geq A_i - ic$. Because $\alpha(G)$ is taken to be greater than $c^2(c-1)/2+3$, this implies that every A_i will be non-empty. Finally, the collection $\{L \in \mathcal{A} \mid \forall i \in [c-1].\, v_i \notin L\}$ will have size at least $\alpha(G) - c(c-1) \geq c+1$, and will be an independent collection.

Now, let $\mathcal{A}'$ be the maximal independent sub-collection from $\mathcal{A}$. We know that $|\mathcal{A}'| \geq c+1$ because we just found an independent sub-collection of $|\mathcal{A}|$ of that size, so we can select $c-1$ many $v_i \in L_i \in \mathcal{A}'$. Because $\mathcal{A}'$ is an independent collection, each of the L_i contains v_i and no other v_j that were selected. We note that we have the freedom to pick 2 more vertices and representatives, which we will specify later. We will first show that all of the representatives in $\mathcal{A}\setminus\mathcal{A}'$ have a specific set of properties, namely their support must contain the support of every representative in $\mathcal{A}'$.

We first show that for every $L \in \mathcal{A}\setminus\mathcal{A}'$, $L \cap \left(\bigcup_{A\in\mathcal{A}'} A\right) \neq \emptyset$. Assume for the sake of contradiction that $L \cap \left(\bigcup_{A\in\mathcal{A}'} A\right) = \emptyset$. Because $\mathcal{A}'$ is the maximal independent collection, L can not be independent of every subset of vertices in $\mathcal{A}'$, otherwise we would add L to $\mathcal{A}'$ to get a bigger independent collection. So there must exist some edge (u, v) such that $u \in L$ and $v \in \bigcup_{A\in\mathcal{A}'} A$. Without loss of generality, assume that v was one not one of the $c-1$ vertices selected, and that we can choose a L_c that contains v but none of $\{v_i \mid i \in [c-1]\}$. Then the qubit $(e, \{v_i \mid i \in [c-1]\})$ supports $c+1$ many operators: $X(L)$ and $X(L_i)$ for all $i \in [c]$. $X(L)$ is supported on the qubit because it contains u, and does not contain v, or any of $\{v_i \mid i \in [c-1]\}$, because each v_i is in $\mathcal{A}'$ and L has no overlap with $\mathcal{A}'$. Each of the $X(L_i)$ is supported on the qubit because none of them contain u, and each of them contains exactly 1 of $\{v_i \mid i \in [c]\}$. However, L and all operators of $\mathcal{A}'$ are taken from $\mathcal{A}$, a c-disjoint collection. We conclude that $L \cap \left(\bigcup_{A\in\mathcal{A}'} A\right) \neq \emptyset$.

Now we show that that $\bigcup_{A\in\mathcal{A}'} A \subseteq L$ for every $L \in \mathcal{A}\setminus\mathcal{A}'$. Assume for the sake of contradiction that $\bigcup_{A\in\mathcal{A}'} A \nsubseteq L$. Then there exists a vertex $v \in \left(\bigcup_{A\in\mathcal{A}'} A\right) \setminus L$. We also know that $L \cap \left(\bigcup_{A\in\mathcal{A}'} A\right) \neq \emptyset$, so we can select a $u \in L \cap \left(\bigcup_{A\in\mathcal{A}'} A\right)$. Without loss of generality, assume that u and v are not in any of $\{L_i \mid i \in [c-1]\}$, and choose L_u and L_v that contain u and v respectively, but none of $\{v_i \mid i \in [c-1]\}$. It may be the case that $L_u = L_v$, but we only need that they can be chosen to not contain any of the other $c-1$ vertices. One of the qubits $(u, \{v_i \mid i \in [c-1]\})$ or $(v, \{v_i \mid i \in [c-1]\})$ is guaranteed to support $X(L)$, on top of other c different logical operators, either $X(L_1), \ldots, X(L_{c-1}), X(L_u)$ or $X(L_1), \ldots, X(L_{c-1}), X(L_v)$, respectively. L contains an even number of vertices from $\{v_i \mid i \in [c-1]\}$, then it will be supported on $(u, \{v_i \mid i \in [c-1]\})$, because it also contains u. If, on the other hand, L contains an odd number of vertices from $\{v_i \mid i \in [c-1]\}$, because it doesn't contain v it will be supported on $(v, \{v_i \mid i \in [c-1]\})$. The other c operators $X(L_1), \ldots, X(L_{c-1}), X(L_u)$ or $X(L_1), \ldots, X(L_{c-1}), X(L_v)$ are supported on the qubits because they contain exactly 1 of u, v, or the v_i. Once again this is in a contradiction with $\mathcal{A}$ being a c-disjoint collection, so we conclude that $L \supseteq \left(\bigcup_{A\in\mathcal{A}'} A\right)$.

Let us first show that $|\mathcal{A}\setminus\mathcal{A}'| \leq 1$ if $c \equiv 0 \mod 2$. Otherwise we could pick $L, K \in \mathcal{A}\setminus\mathcal{A}'$, and the qubit $(v_1, \{v_i\}_{i\in[c-1]})$ (note the repetition of v_1) will support $c+1$ logical operators from $X(L), X(K)$ and $X(L_i)$ for all $i \in [c-1]$, because each of those operators contains an odd number of representatives from $\{v_i \mid i \in [c-1]\}$, as each $X(L_i)$ contains only v_i, and $X(L)$ and $X(K)$ contain all $c-1$ of them.

If $c \equiv 1 \mod 2$, it must be that $\mathcal{A} = \mathcal{A}'$. Otherwise we pick an $L \in \mathcal{A}\setminus\mathcal{A}'$, and select an additional v_c and L_c from $\mathcal{A}'$. Then the qubit $(v_c, \{v_i\}_{i\in[c-1]})$ would support $c+1$ operators from $X(L)$ and $X(L_i)$ for all $i \in [c]$, because L contains every vertex from $\{v_i \mid i \in [c]\}$, an odd sized set.

At this point, we know that $|\mathcal{A}'| \geq |\mathcal{A}| - 1$ if $c \equiv 0 \mod 2$, and otherwise $|\mathcal{A}'| \geq |\mathcal{A}|$. It remains to be shown that if $c \equiv 0 \mod 2$ and $\alpha(G) \equiv 0 \mod 2$, there exists an independent collection of size $\geq |\mathcal{A}|$. Assume that there exists one $L \in \mathcal{A}\setminus\mathcal{A}'$, $c \equiv 0 \mod 2$, and $\alpha(G) \equiv 0 \mod 2$.

If $L \neq \bigcup_{A\in\mathcal{A}'} A$, then we show that $\mathcal{A}' \cup \{L\setminus \bigcup \mathcal{A}'\}$ is an independent collection. Assume for the sake of contradiction that it is not an independent collection. Because G is a connected graph, there exists an edge $e \in E$ incident to vertices $u \in L \setminus \bigcup \mathcal{A}'$ and $v \in \bigcup \mathcal{A}'$, as these two collections can not share a vertex. Let $v_c = v$ so that the qubit $(e, \{v_i\}_{i\in[c-1]})$ supports $c+1$ different logical operators $X(L), X(L_1), \ldots, X(L_c)$, leading to a contradiction. $\mathcal{A}' \cup \{L\setminus \bigcup \mathcal{A}'\}$ is an independent collection of size $|\mathcal{A}|$.

Otherwise, it must be the case that $L = \bigcup_{A\in\mathcal{A}'} A$. Then $|L| \equiv |\bigcup_{A\in\mathcal{A}'} A| \equiv \sum_{A\in\mathcal{A}'} |A| \equiv |\mathcal{A}'| \mod 2$, so $X(L)$ is a representative of $\overline{L}_c^G$ if and only if $|\mathcal{A}'| \equiv 1 \mod 2$. Thus, if $|\mathcal{A}'| \equiv 0 \mod 2$, L can not be a valid representative of $\overline{L}_c^G$, so $|\mathcal{A}'| = |\mathcal{A}|$. If $|\mathcal{A}'| \equiv 1 \mod 2$ $|\mathcal{A}'|$, we know that $\alpha(G) \equiv 0 \mod 2$, so $|\mathcal{A}'|$ must be less than $\alpha(G)-1$. Thus $|\mathcal{A}| \leq |\mathcal{A}'|+1 \leq \alpha(G)$, which implies there exists an independent collection of size at least $|\mathcal{A}|$.

Now we have shown that if $\alpha(G)(c+1) \not\equiv 1 \mod 2$, there exists an independent collection of size $|\mathcal{A}|$, and otherwise there exists an independent collection of size $|\mathcal{A}|-1$, where $\mathcal{A}$ is a c-disjoint collection of $\overline{L}_c^G$ of maximal size, i.e. $|\mathcal{A}| = c\Delta_c\left(\overline{L}_c^G\right)$. Thus, we have shown that $|\mathcal{A}| \leq \alpha(G)+b$, since $\alpha(G)$ is greater than the size of the independent collection.

If $b=1$ if and only if we are in the second case in Eq. (13). Thus Lemma 1 implies that there exists a c-disjoint collection for $\overline{L}_c^G$ of size at least $\alpha(G)+b$, i.e. $|\mathcal{A}| \geq \alpha(G)+b$. These two bounds tell us that

$$|\mathcal{A}| = \alpha(G)+b. \tag{15}$$

Substituting $|\mathcal{A}|$ into $\Delta_c\left(\overline{L}_c^G\right) = |\mathcal{A}|/c$ yields Eq. (14). □

C. Putting things together

Now, we are ready to establish the main result of our work, which asserts that it is NP-complete to calculate (or even approximate) the c-disjointness for a fixed positive integer constant c.

Theorem 1. For any positive integer constant c the decision problem c-DISJOINTNESS is NP-complete. Furthermore, for any $\epsilon > 0$ it is NP-hard to approximate the c-disjointness to within a multiplicative factor of $n^{(1-\epsilon)/(c+1)}$, where n is the number of physical qubits of the stabilizer code.

Proof. First note that INDEPENDENT SET is NP-complete, even if we restrict our attention to the connected graphs with the independence number at least $c^2(c-1)/2$, which we require in our proof. We can show that c-DISJOINTNESS is NP-hard by reducing INDEPENDENT SET to it.

Let $G=(V,E)$ be a graph satisfying $\alpha(G) \geq c^2(c-1)/2$. We construct a new graph $G'=(V',E')$ as follows: for every vertex $v \in V$ we introduce two vertices $v_1, v_2 \in V'$, and for every edge $(u,v) \in E$ we introduce four edges $(u_1,v_1),(u_1,v_2),(u_2,v_1),(u_2,v_2) \in E'$. One can show that

$$\alpha(G') = 2\alpha(G). \tag{16}$$

Following Sec. II A, we construct the stabilizer code $\mathcal{S}_c^{G'}$ and the logical operator $\overline{L}_c^{G'}$. Since $\alpha(G') \equiv 0 \mod 2$, then from Lemma 2 we obtain

$$c\Delta_c\left(\overline{L}_c^{G'}\right) = \alpha(G') = 2\alpha(G) \tag{17}$$

and, subsequently, $\alpha(G) \geq a$ iff $c\Delta_c\left(\overline{L}_c^{G'}\right) \geq 2a$. We conclude that the output of the c-DISJOINTNESS problem for the instance $\mathcal{S}_c^{G'}$, $\overline{L}_c^{G'}$ and $2a$ is the output of INDEPENDENT SET for G and a, which, in turn, establishes that c-DISJOINTNESS is NP-hard.

Now we turn to approximation algorithms. Given the graph $G=(V,E)$, the number of physical qubits in the stabilizer code $\mathcal{S}_c^G$ is

$$n \leq |V|\left(\binom{|V|}{c-1}+2\right) + |E|\binom{|V|}{c-1} < |V|^{c+1}, \tag{18}$$

as $|E| \leq \binom{|V|}{2}$. Depending on the parity of $\alpha(G)(c+1)$, the size of the largest c-disjoint collection $c\Delta_c\left(\overline{L}_c^G\right)$ for the logical operator $\overline{L}_c^G$ is equal to either $\alpha(G)$ or $\alpha(G)+1$. Thus, for any $\epsilon > 0$ approximating the c-disjointness $\Delta_c\left(\overline{L}_c^G\right)$ to within a multiplicative factor of $n^{(1-\epsilon)/(c+1)}$ implies that we can approximate $\alpha(G)$ to within a multiplicative factor of $n^{(1-\epsilon)/(c+1)} < |V|^{(1-\epsilon)}$. Because approximating $\alpha(G)$ to a multiplicative factor of $|V|^{(1-\epsilon)}$ is known to be NP-hard, approximating the c-disjointness to a factor of $n^{(1-\epsilon)/(c+1)}$ is also NP-Hard.

Finally, we show that c-DISJOINTNESS is in NP, which will complete the proof that it is NP-complete. Let us consider a polynomial time verifier that takes as its witness a collection of representatives of $\overline{L}$, $\mathcal{A}$. The verifier needs to perform check the following: (i) every qubit supports at most c operators from $\mathcal{A}$, (ii) every element of $\mathcal{A}$ is a representative of $\overline{L}$ and (iii) the size of $\mathcal{A}$ is greater than ca. (i) and (iii) can be checked in polynomial time because every c-disjoint collection is smaller than size cn. Condition (ii) can be stated equivalently as: $PL \in \mathcal{S}$ for every $P \in \mathcal{A}$, where L is the representative in the problem instance. In order to check this property, the verifier takes the binary matrix of size $(n-k) \times 2n$ that describes $\mathcal{S}$, M, and computes its rank. For every $P \in \mathcal{A}$, the verifier appends a row corresponding to a string of $2n$ bits representing the operator PL to M and takes this new matrix's rank. If the ranks are the same, then P is a representative of $\overline{L}$, and otherwise it is not. Since finding the rank of a binary matrix can be done in polynomial time, the witness can be verified in polynomial time, and c-DISJOINTNESS is in NP, and thus NP-complete. □

III. DISJOINTNESS IN PRACTICE

In this section we express the disjointness as a linear program with exponentially many variables, allowing for us to find the disjointness in exponential time. We use the linear program to find the disjointness of the $[\![14,3,3]\!]$ stabilizer code, and ultimately rule out the possibility that transversal gates of the $[\![14,3,3]\!]$ code existin the third level of the Clifford Hierarchy.

A. A linear program for disjointness

Let $\mathcal{S}$ be a stabilizer code, and $\overline{L}$ be a logical Pauli of the code. We aim to describe an optimization problem whose optimal value will be the disjointness of $\overline{L}$. For every representative L of $\overline{L}$ we can define a variable $x_L \in \mathbb{Z}_{\geq 0}$ representing selecting L to be in a c-disjoint collection x_L many times. The constraint of being c-disjoint is equivalent to: for every qubit q, the sum of x_L for which L is supported on q is less than c.

This exponentially large integer linear program solves for the c-disjointness. However, by relaxing the integer constraints, we will arrive at a linear program whose optimal value is equal to the disjointness of $\overline{L}$, shown below.

$$\text{maximize} \qquad \Delta = \sum_{L\in\overline{L}} x_L, \tag{19}$$

$$\text{subject to} \qquad \forall q \in Q : \sum_{L\in\overline{L}: q\in\operatorname{supp} L} x_L \leq 1, \tag{20}$$

$$\forall L \in \overline{L} : x_l \geq 0. \tag{21}$$

Let Δ^* to be the optimal value achieved by Eq. (19), subject to the constraints in Eqs. (21) and (20).

Theorem 2. The optimal value of the linear program, $\Delta*$, is $\Delta(\overline{L})$. Furthermore it is achieved by a rational solution expressed with $\operatorname{poly}(2^{n-k})$ bits.

Proof. Given a linear program of the form $\max\{c^{\mathrm{T}}x : Ax = b, x \geq 0\}$ where A, b and c have rational entries, if there exists a finite optimal solution, there exists a rational optimal solution whose bit size is polynomially bounded in terms of the bit sizes of A, b, and c [42]. Every constraint in the linear equation defines a hyperplane, and the intersection of any set of these hyperplanes can be described by a number of bits polynomial in the size of A, b, and c. Because there exists an optimal solution, there must exist an optimal solution that is a vertex of the convex hull, and can be described by an intersection of some of these hyperplanes. Thus, there exists an optimal solution whose bit size is polynomially bounded in the sizes of A, b, and c.

We can express the constraints in Eq. (20) in this standard form, where A is a binary matrix of size $2^{n-k} \times n$, and b and c are vectors of all 1. This fact implies that there exists a choice of x_L that satisfies Eqs. (20) and (21) and achieves value Δ^* in Eq. (19) that can be expressed with $\operatorname{poly}(2^{n-k})$ bits.

For any $c \in \mathbb{Z}_{>0}$, consider the optimal c-disjoint collection, which has k^*_L copies of the representative L. If we set $x_L = k^*_L/c$, by the definition of c-disjointness,

$$\sum_{\operatorname{supp} L\ni q} x_L = \sum_{\operatorname{supp} L\ni q} k^*_L/c \leq 1. \quad \forall q \tag{22}$$

Furthermore, the value of Δ in Eq. (19) is $\Delta_c(()\overline{L})$, which implies that $\Delta^* \geq \Delta_c(\overline{L})$. Since this holds for all c,

$$\Delta^* \geq \sup_{c>0}(\Delta_c(\overline{L})) = \Delta(\overline{L}). \tag{23}$$

Every x_L appears in at least one constraint, so every feasible solution to Eqs. (20) and (21) is bounded by setting $x_L = 1$ for all L. In other words, there exists a finite optimal solution to the linear program. Thus, there exists some an optimal rational solution $\{x^*_L\}$ that can be expressed with $\operatorname{poly}(2^{n-k})$ bits.

So, there exists some finite c^* such that $x^*_L c^*$ is an integer for all L, and thus defines a c^*-disjoint collection when taking each representative L $x^*_L c^*$ many times. Thus,

$$\Delta^* = \Delta_{c^*}(\overline{L}) \leq \Delta(\overline{L}). \tag{24}$$

So we have that $\Delta^* = \Delta(\overline{L})$, and is achieved by a rational solution that can be expressed in $\operatorname{poly}(2^{n-k})$ bits. □

Corollary 1. *For every stabilizer code $\mathcal{S}$ and logical Pauli operator $\overline{L}$ of $\mathcal{S}$, $\Delta(\overline{L}) = \Delta_{c^*}(\overline{L})$ for some finite $c^* = 2^{\operatorname{poly}(2^n)}$.*

Proof. In the proof of Thm. 2, we used the fact that there exists an optimal rational solution $\{x^*_L\}$ and finite c^* such that every $x^*_L c^*$ is an integer and the collection formed by including every L x^*_L times is c^*-disjoint and has size $c^*\Delta(\overline{L})$. To prove this corollary, we simply need to bound c^*.

Thm. 2 also tells us that every x^*_L can be written in $\operatorname{poly}(2^{n-k})$ bits, which means that, as a fraction, the denominator of x^*_L is bounded by $2^{\operatorname{poly}(2^{n-k})}$. We will c^* to be the least common denominator of the denominators of x^*_L, when written as a fraction, which clearly implies $x^*_L c^*$ is an integer for all L. Our choice of c^* is bounded by the product of the denominators, which is, in turn, bounded as follows:

$$c^* \leq 2^{\operatorname{poly}(2^{n-k})2^{n-k}} = 2^{\operatorname{poly}(2^{n-k})}. \tag{25}$$

For this choice of c^*, we know there exists a c^*-disjoint collection of size $c^*\Delta(\overline{L})$, which proves the corollary. □

This result immediately implies the following remark.

Remark. *The disjointness can be equivalently expressed as follows*

$$\Delta(\mathcal{S}) = \min_{\overline{L}} \max_{c\geq 1} \Delta_c(\overline{L}). \tag{26}$$

Proof. From Cor. 1, for every $\overline{L}$, there exists a $c^*_{\overline{L}}$ such that $\Delta(\overline{L}) = \Delta_{c^*_{\overline{L}}}(\overline{L})$. By the definition of $\Delta(\overline{L})$, $\Delta_{kc^*_{\overline{L}}}(\overline{L}) = \Delta_{c^*_{\overline{L}}}(\overline{L})$ for every $k \geq 1$. Let $c^*_{\mathcal{S}} = \prod c^*_{\overline{L}}$, then for all $\overline{L}$, $\Delta_{c^*_{\mathcal{S}}}(\overline{L}) = \Delta(\overline{L})$. Thus,

$$\max_{c\geq 1}\min_{\overline{L}} \Delta_c(\overline{L}) = \min_{\overline{L}} \Delta_{c^*_{\mathcal{S}}}(\overline{L}) = \min_{\overline{L}} \max_{c\geq 1} \Delta_c(\overline{L}). \tag{27}$$

Which is exactly Eq. (26). □

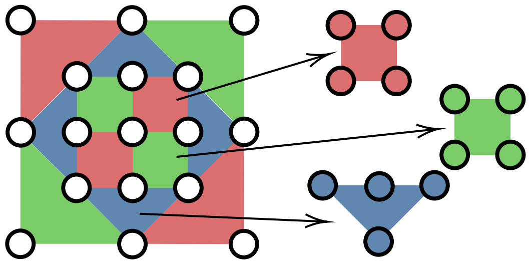

FIG. 2. The $[\![14,3,3]\!]$ stabilizer code is defined by placing qubits (white dots) on the vertices of a rhombic dodecahedron and introducing X-, Y- and Z-type stabilizer generators for every red, green and blue face, respectively. We depict "the azimuthal projection", where the four corner qubits are identified. Dots colored in red, green and blue correspond to single-qubit Pauli X, Y and Z operators, respectively.

The linear program in Eqs. (19),(20) and (21), with 2^{n-k} variables and n constraints, can be solved in time $O(2^{2.5(n-k)})$, and solving for $\Delta(\mathcal{S})$ requires solving this linear program for all $2^{2k}-1$ logical Pauli operators. Thus, $\Delta(\mathcal{S})$ can be found in time $O(2^{2.5n-0.5k})$ for a stabilizer code $\mathcal{S}$ with n physical qubits and k logical qubits. An implementation of this algorithm can be found on github [43], and apply it to an interesting stabilizer code in Sec. III B.

B. An illustrative example

Now, we focus on an illustrative example of the recently introduced $[\![14,3,3]\!]$ stabilizer code [34]; see Fig. 2. We would like to understand whether any non-Clifford logical operators for this code can be implemented via transversal gates. If that was the case, then due to its small size this code could prove useful in, for instance, magic state distillation protocols. Since the $[\![14,3,3]\!]$ stabilizer code is a non-CSS code, there are no off-the-shelf techniques to find transversal gates; however, we use the disjointness to rule out the possibility of certain logical operations.

A brute force approach to computing the c-disjointness, even for $c=1$ and small stabilizer codes, is computationally infeasible. In the case of the $[\![14,3,3]\!]$ stabilizer code, there are $4^3-1=63$ different non-trivial logical Pauli operators, and each of them has $2^{11}=2048$ representatives. Thus, for each logical Pauli operator there are $2^{2048}\approx 2\times 10^{618}$ possible subsets of its representatives, and we would need to check each and every subset for the qubit overlap condition (assuming that the 1-disjointness can be achieved with a set rather than a multiset).

We can, however, use the linear program specified in Eqs. (19)-(21), which has 2048 variables and 14 constraints (positivity constraints are not normally counted). We numerically find that the disjointness of the $[\![14,3,3]\!]$ stabilizer code is 2; see the source code [43]. We also find that the max-distance of the $[\![14,3,3]\!]$ stabilizer code is $d_\uparrow=6$. Thus, using the bound from Eq. (1) we conclude that any transversal gate can only implement logical operations within the third level of the logical Clifford hierarchy.

We now obtain an improvement of the bound in Eq. (1), which we subsequently use to rule out the possibility of any transversal gate that implements a non-Clifford logical operation for the $[\![14,3,3]\!]$ stabilizer code. Recall that the main proof idea in Ref. [30] is to evaluate the (nested) group commutator of the transversal gate and M logical Pauli operators for the stabilizer code $\mathcal{S}$. If for every M-tuple of logical Pauli operators the resulting operator is a trivial logical operator, then the transversal gate is guaranteed to implement an operator from the M^{th} level of the logical Clifford hierarchy. To recast this condition, it is useful to introduce the following quantity

$$\Omega_M(\mathcal{S}) = \max_{\{\overline{L}_i\}_{i\in[M]}} \min_{L_i\in\overline{L}_i} \left| \bigcap_{i\in[M]} \operatorname{supp} L_i \right|, \tag{28}$$

where for each M-tuple of logical Pauli operators $\{\overline{L}_i\}_{i\in[M]}$ we seek an M-tuple of corresponding representatives $L_i\in\overline{L}_i$, whose intersection $\bigcap_{i\in[M]}\operatorname{supp} L_i$ is the smallest. Then, we can formulate the following strengthening of Theorem 5 from Ref. [30].

Theorem 3. Consider a stabilizer code $\mathcal{S}$ with min-distance $d_\downarrow$. If M is a positive integer satisfying

$$\Omega_M(\mathcal{S}) < d_\downarrow, \tag{29}$$

then any transversal gate implements an operator from the M^{th} level of the logical Clifford hierarchy.

We will verify that $\Omega_2(\mathcal{S})<3$ for the $[\![14,3,3]\!]$ stabilizer code. Consider any pair of logical Pauli operators $\{\overline{L}_1,\overline{L}_2\}$ for which, without loss of generality, $\Delta(\overline{L}_1)>2$. Because the max distance of $\mathcal{S}$ is 6, we can find a representative of $\overline{L}_2$ of weight 6, and by the Lemma 4 from Ref. [30] (scrubbing lemma), there exists a representative of $\overline{L}_1$ such the overlap in support is at most size 2.

Thus, we restrict our attention to pairs of logical Pauli operators for which both operators achieve $\Delta(\overline{L}_i)=2$. There are exactly 4 of these operators, depicted in Fig. 3. By brute force, using the functions available in [43], we verify that for all 6 pairs of these 4 operators, there exists a choice of representatives that overlap on strictly less than 2 qubits. We conclude that all of the transversal gates of the $[\![14,3,3]\!]$ stabilizer code lie in the second level of the Clifford hierarchy.

IV. BOUNDS ON THE DISJOINTNESS

A. CSS codes

In this section we show that for CSS codes, the disjointness can be approximated without mixing operators with different typed Pauli operators.

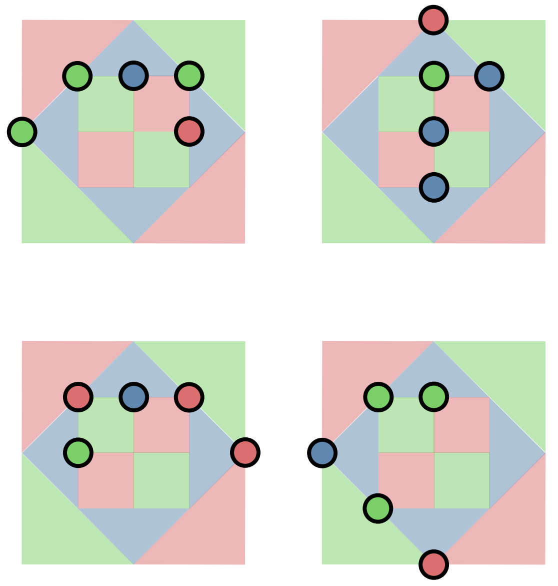

FIG. 3. For the $[\![14,3,3]\!]$ stabilizer code there are four logical Pauli operators with disjointness 2. We depict their smallest-weight representatives, where dots colored in red, green and blue correspond to single-qubit Pauli X, Y and Z operators, respectively.

Lemma 3. *Given a CSS code $\mathcal{S}$ and logical Pauli $\overline{P} \in \{\overline{X}, \overline{Z}\}$, for all c there exists a c-disjoint collection for the logical operator $\overline{P}$ of maximal size for which each representative is implemented using only single qubit P type Pauli operators.*

Proof. Recall that for $\overline{P}$, there exists a representative of $\overline{P}$ that is implemented using only single qubit P-type operators. We can also choose stabilizer generators of $\mathcal{S}$ such that some generators are implemented using only single qubit X-type operators and other generators are implemented using only single qubit Z-type operators.

Consider a c-disjoint collection for $\overline{P}$ of maximal size, $\mathcal{A}$. Every representative $L \in \mathcal{A}$ is equal to LS_XS_Z for some stabilizer generators $S_X, S_Z \in \mathcal{S}$, where S_X is the product of X type stabilizer generators, and S_Z is the product of Z type stabilizer generators.

Take $\overline{P} = \overline{X}$, then we can replace all instances of PS_XS_Z with PS_X in $\mathcal{A}$. The support of this new operator is contained in the support of the original, and thus this new collection is also a c-disjoint collection with the same size as $\mathcal{A}$. Performing this for every representative in $\mathcal{A}$ yields a c-disjoint collection such that every representative in the collection is implemented using only single qubit X type operators.

If $\overline{P} = \overline{Z}$, replacing PS_XS_Z with PS_Z similarly yields a c-disjoint collection with the same size as $\mathcal{A}$. Thus, there exists a maximal c-disjoint collection using only single qubit logical P type operators. □

One implication of this remark is that in for CSS codes, when $\overline{L}$ contains a representative implemented using only X- or Z-type single qubit operators, the variables x_L in Eqs. (19), (20), and (21) can be taken to only be over representatives that are implemented using X- or Z-type operators respectively.

Theorem 4. Given a CSS code, let $\overline{L} = \overline{X}\,\overline{Z}$ be a logical Pauli operator that can be decomposed into non-trivial logical X- and Z-type operators. Then, the following inequalities hold

$$\frac{1}{2}\min\left(\Delta\left(\overline{X}\right), \Delta\left(\overline{Z}\right)\right) \le \Delta\left(\overline{L}\right) \le \min\left(\Delta\left(\overline{X}\right), \Delta\left(\overline{Z}\right)\right). \tag{30}$$

Proof. Fix some c, and let $\mathcal{A}$ be a c-disjoint collection of maximal size for $\overline{L}$. Every representative in $\mathcal{A}$ can be expressed as a representative of $\overline{X}$ times a representative of $\overline{Z}$. Removing the representative of $\overline{Z}$ for every operator of $\mathcal{A}$ yields a c-disjoint collection of $\overline{X}$, and a similar c-disjoint collection exists for $\overline{Z}$ when removing the representatives of $\overline{X}$. Thus, $\Delta_c(\overline{L}) \le \min(\Delta_c(\overline{X}), \Delta_c(\overline{Z}))$. Taking the max over c yields the right hand inequality in Eqn. 30.

Now fix some c and consider the c-disjoint collections $\mathcal{A}_X$ and $\mathcal{A}_Z$ for $\overline{X}$ and $\overline{Z}$ of maximal size, and let $N = \min(|\mathcal{A}_X|, |\mathcal{A}_Z|)$. Index each element of the multi-set $\mathcal{A}_X$ so that L_X^i is the i^{th} operator in $\mathcal{A}_X$ and L_Z^i is the i^{th} operator in $\mathcal{A}_Z$. The precise ordering of elements does not matter, so long as N operators from $\mathcal{A}_X$ and $\mathcal{A}_Z$ are assigned an index from $[N]$. Then we can construct a $2c$-disjoint collection of size N by taking $\{L_X^i L_Z^i : i \in [N]\}$.

For the i^{th} operator in this new collection, denoted L^i, $\operatorname{supp} L^i \subseteq \operatorname{supp} L_x^i \bigcup \operatorname{supp} L_Z^i$. Thus, for every qubit q

$$|\{L^i : q \in \operatorname{supp} L^i\}| \le |\{L_X^i : q \in \operatorname{supp} L_X^i\}| + |\{L_Z^i : q \in \operatorname{supp} L_Z^i\}| \tag{31}$$

In this new collection, each qubit is acted on by at most $2c$ representatives, which means that $2c\Delta_{2c}(\overline{L}) \ge N$. This result holds for all choices of c, so

$$\frac{1}{2}\min\left(\Delta\left(\overline{X}\right), \Delta\left(\overline{Z}\right)\right) \le \Delta\left(\overline{L}\right). \tag{32}$$

With the upper bound from Eq. (31), we have Eq. (30). □

This implies that if, in the definition of disjointness, the min was only taken over X and Z type logical Pauli operators, the result would be a 2 approximation to the disjointness. However, doing this still requires exponential time to the best of our knowledge.

B. Concatenated stabilizer codes

In this section, we prove that the disjointness of concatenated codes is at least the product of the disjointness of it's constituents. The authors note that [31] stated and proved this theorem independently and concurrently to this paper, however we still find it interesting to state the result here, as our conclusions and use-cases will be different than theirs.

Theorem 5. For any two stabilizer codes $\mathcal{S}_1$ and $\mathcal{S}_2$ the following equality holds:

$$\Delta(\mathcal{S}_1 \rhd \mathcal{S}_2) \geq \Delta(\mathcal{S}_1)\,\Delta(\mathcal{S}_2)\,, \tag{33}$$

and given M_1 and M_2 satisfying Eq. (1) for codes $\mathcal{S}_1$ and $\mathcal{S}_2$, the transversal gates of $\mathcal{S}_1 \rhd \mathcal{S}_2$ lie in the $M = \max(M_1, M_2)^{th}$ level of the Clifford hierarchy.

Proof. For any logical Pauli of $\mathcal{S}_1 \rhd \mathcal{S}_2$, $\overline{P} = \overline{L \rhd \mathcal{C}}$, and c_1 and c_2, we will show that $\Delta_{c_1 c_2}(\overline{P}) \geq \Delta_{c_1}(\mathcal{S}_1)\,\Delta_{c_2}(\mathcal{S}_2)$. This inequality will immediately imply Eq. (33).

By the definition of disjointness, there exists a c_1-disjoint collection $\mathcal{A}_L$ of representatives of L of size $c_1\Delta_{c_1}(\overline{L})$. Similarly, there exists a c_2-disjoint collection $\mathcal{A}_\mathcal{C}$ of choices of logical Pauli operators of $\mathcal{S}_2$ of size $c_2\Delta_{c_2}(\mathcal{S}_2)$. Recall that in every choice in $\mathcal{A}_\mathcal{C}$, we have that the identity is represented by the all identity operator.

Consider the collection

$$\mathcal{A}_{\overline{P}} = \{L \rhd C \mid L \in \mathcal{A}_{\overline{L}}, C \in \mathcal{A}_\mathcal{C}\}. \tag{34}$$

For every $L \in \mathcal{A}_{\overline{L}}$, the collection $\{L \rhd C | C \in \mathcal{A}_\mathcal{C}\}$ overlaps on at most c_2 times on each qubit in blocks where L is non-trivial. Furthermore, at most c_1 many of such collections act on any individual block of qubits. Thus, $\mathcal{A}_{\overline{P}}$ is a c_1c_2-disjoint collection. The size of this collection is $c_1c_2\Delta_{c_1}(\overline{L})\,\Delta_{c_2}(\mathcal{S}_2)$. So

$$\Delta_{c_1c_2}(\overline{P}) \geq \Delta_{c_1}(\overline{L})\,\Delta_{c_2}(\mathcal{S}_2) \geq \Delta_{c_1}(\mathcal{S}_1)\,\Delta_{c_2}(\mathcal{S}_2)\,, \tag{35}$$

and taking the max over both c_1 and c_2 yields Eq. (33).

We use this to bound the level of the Clifford hierarchy that transversal gates of the concatenated code can lie in. The min- and max- distances of a concatenated code obey the following inequalities:

$$d_\downarrow(\mathcal{S}_1 \rhd \mathcal{S}_2) \geq d_\downarrow(\mathcal{S}_1) d_\downarrow(\mathcal{S}_2) \tag{36}$$
$$d_\uparrow(\mathcal{S}_1 \rhd \mathcal{S}_2) \leq d_\uparrow(\mathcal{S}_1) d_\uparrow(\mathcal{S}_2). \tag{37}$$

Let $M = \max(M_1, M_2)$, then we have

$$\begin{aligned} d_\uparrow(\mathcal{S}_1 \rhd \mathcal{S}_2) &\leq d_\uparrow(\mathcal{S}_1) d_\uparrow(\mathcal{S}_2) && (38)\\ &< d_\downarrow(\mathcal{S}_1)\Delta(\mathcal{S}_1)^{M_1} d_\downarrow(\mathcal{S}_2)\Delta(\mathcal{S}_2)^{M_2} && (39)\\ &\leq d_\downarrow(\mathcal{S}_1)\Delta(\mathcal{S}_1)^{M} d_\downarrow(\mathcal{S}_2)\Delta(\mathcal{S}_2)^{M} && (40)\\ &\leq d_\downarrow(\mathcal{S}_1 \rhd \mathcal{S}_2)\Delta(\mathcal{S}_1 \rhd \mathcal{S}_2)^{M}. && (41) \end{aligned}$$

From Ref. [30], this implies that transversal gates of $\mathcal{S}_1 \rhd \mathcal{S}_2$ lie in the M^{th} level of the Clifford hierarchy. □

We note that the level of Clifford hierarchy allowed by disjointness might be worse than the tightest bound, as we saw in Sec. III B. This means it is still possible that a concatanted code could have transversal gates in higher levels of the Clifford hierarchy than any of its constutients if, for example, there are other ways to bound the level of the Clifford hierarchy that transversal gates of the constituent codes can lie in, that don't apply to the concatenated code.

C. Hypergraph product codes

We first define the analog of disjointness for binary matrices, which can be associated with a classical linear code. In order to do this definition, we will make use of the Lemma 3.

Given a binary matrix H, we can form a CSS code H^X by taking the stabilizer where each generator is a row of H, with 1 replaced by X and 0 replaced by I. For this stabilizer, the distance is clearly 1, because we can find a single qubit X operator that isn't in the span of H^X. We also note that $\Delta(H)$ is 1, because there must exist a Z type logical Pauli such that every representative of that Z type logical Pauli overlaps on a single qubit [30].

However this does not apply to X type logical Pauli operators of this code, so restricting our attention to these logical Paulis can yield interesting values of disjointness. So, given a CSS stabilizer code $\mathcal{S}$, define $\mathcal{L}_X$ to be the set of X type logical Pauli operators, and define a new quantity $\Delta^X(\mathcal{S})$ as follows:

$$\Delta^X(\mathcal{S}) = \max_{c\geq 1} \min_{\overline{L}\in\mathcal{L}_X} \Delta_c(\overline{L}) \tag{42}$$

We take the max over c because it is guaranteed to be achieved by a finite value due to Thm. 2. This definition makes reasonable sense because, as we showed in Lemma 3, a logical X type Pauli has c-disjoint collection of representatives that can be implemented only using X type single qubit Pauli operators, so the disjointness of these logical Pauli's is unaffected by the lack of Z type stabilizers.

The main goal of this section is to show that the following relationship holds between the disjointness of the hypergraph product and it's constituents.

Theorem 6. Given binary matrices H_1 and H_2 the following relation holds

$$\Delta(\mathrm{HP}(H_1, H_2)) \leq \min(\Delta^X(H_1^X), \Delta^X(H_2^X)). \tag{43}$$

Proof. In order to prove this theorem, it will suffice to show that for every X type logical Pauli of H_1^X, there exists a logical Pauli of $\mathrm{HP}(H_1, H_2)$ that has a smaller disjointness. Since the disjointness is the min over representatives, the disjointness of the hypergraph product code can't be larger than the disjointness of that subset.

Given a logical Pauli L_1 of H_1^X, represented by an n bit string b_1 (because L_1 uses only X type operators), take any stabilizer element of H_2^X, S_2, with n bit representation b_2. Consider the following operator:

$$L = (b_1 \otimes I_{m_2}, I_{m_1} \otimes b_2)^X. \tag{44}$$

Here the superscript X represents replacing 1 with a single qubit X operator on the corresponding qubit, and 0 with I. Because this operator interacts on a separate space than the Z type stabilizer elements of the hypergraph product code, it must commute with all of the stabilizer elements of $\mathrm{HP}(H_1, H_2)$. Since L_1 was a logical

Pauli operator of H_1^X, L is also not a stabilizer element of the hypergraph product code either. Thus, it is a logical Pauli operator of $\mathrm{HP}(H_1, H_2)$.

By Lem. 3, in order to find an optimal c-disjoint set for this logical Pauli, we only need to consider multiplying by X type stabilizer elements, which are all of the form $(b \otimes I_{m_2}, ...)^X$ for some n bit representation b for some stabilizer element $S \in H_1^X$. Thus, any c-disjoint collection of representatives of L immediately implies yields a c-disjoint collection for the logical Pauli that L_1, so $\Delta_c(\overline{L}) \leq \Delta_c(\overline{L}_1)$ for all c.

This implies that $\Delta(\mathrm{HP}(H_1, H_2)) \leq \Delta^X(H_1^X)$, and a symmetric strategy works for representatives of H_2^X, which proves the theorem. $\square$

Unlike other bounds in this section, this is an upper bound on the disjointness of a quantum code. That means that it can not be used to bound the level of the Clifford hierarchy that transversal gates lie in directly, as the largest value of M possible in Eq. (1) is inversely proportional to Δ. However, [31] show that code families whose disjointness is unbounded as the code size increases can not have universal fault tolerant gate sets, so we hope that this result be useful for finding universal fault tolerant gate sets for hypergraph product codes. In particular, taking the hypergraph product of a large binary matrix with a code with small classical disjointness will always yield a quantum code with small disjointness. So, there is hope that with the right pair of classical codes, one can construct a code with good distance that still has transversal gates in high levels of the Clifford hierarchy. This task is made more difficult by the fact that the distance of hypergraph product codes is (roughly) the distance of the minimum of the two classical codes used in the hypergraph product, meaning finding these codes will require finding families of classical codes with high distance and low classical disjointness.

V. DISCUSSION

The main result of our work, which is Theorem 1, established that for any positive integer constant c the problem of calculating the c-disjointness (or even approximating it up to within a multiplicative factor) is NP-complete. Although we have not shown that calculating the disjointness of stabilizer codes is hard, Theorem 1 seems to point in this direction. In general, our results indicate that finding fault-tolerant logical gates for generic quantum error-correcting codes is a computationally challenging task. Other results presented in our work included: (i) formulating a linear program to calculate the disjointness, (ii) strengthening the main result of Ref. [30], and (iii) providing bounds on the disjointness for various stabilizer code families.

We hope that our work initiates and motivates a thorough search of methods of finding fault-tolerant logical gates. Moreover, we expect that there exist efficient algorithms to address the problem of calculating the disjointness for certain code families, such as topological quantum codes. We emphasize that clever usage of the underlying code symmetries might further simplify this problem. For example, for codes that are invariant under some derangement of qubits (permutations with no fixed points), the number of variables in the linear programming formulation of disjointness can be reduced from 2^n to $2^n/n$. Also, it would be very interesting to find the connection between the disjointness and other code quantities, such as the price [44].

Finally, we remark that the problem of calculating the c-disjointness is reminiscent of the knapsack problem, one of the well studied NP-complete problems [45]. One might consider extending the disjointness in a similar fashion to the multiple knapsack problem [45, 46]. In particular, we can introduce the notion of a regional disjointness, where different regions of the code have different disjointness constraints applied to them. As explained in Appendix VI, computing the regional 0-disjointness is #P-complete for general stabilizer codes and logical Pauli operators. This notion of regional disjointness is reminiscent of certain physical quantum computing systems, where physical location of qubits may affect the number of gates that can be applied before decoherence occurs. While the property does not have an implication on the transversal gates of a code, we hope that it, and other generalizations of the disjointness, may prove to be useful properties of stabilizer codes.

ACKNOWLEDGMENTS

A.K. thanks David Gosset, Andrew Landahl, Pooya Ronagh and Jamie Sikora for helpful discussions. A.K. acknowledges funding provided by the Simons Foundation through the "It from Qubit" Collaboration. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. This work was completed prior to A.K. joining AWS Center for Quantum Computing.

VI. APPENDIX

A. Regional Disjointness

Here we present a generalization of the disjointness known as the regional disjointness and provide a brief proof that the problem is #P-Complete. It is still not known whether or not this property is more useful then the disjointness as a tool for studying the transversal gates of Quantum Error Correcting Codes.

Given a stabilizer $\mathcal{S}$, logical Pauli operator $\overline{L}$, and a subset of qubits R, the regional c-disjointness (for a integer constant $c \geq 0$), $\Delta_c(\overline{L}, R)$, is the size of the largest collection of representatives of $\overline{L}$ such that the set of operators don't overlap more than c times on any qubit in R. This problem might be akin to having certain weak qubits that might decohere quicker than other qubits, and trying to find minimum number of operations that can be performed with those weaknesses.

We emphasize one important distinction between subset disjointness and disjointness. The disjointness is defined to be the size of a collection divided by c, where as the regional disjointness is simply the size of the collection. This is because for regional disjointness, $c = 0$ is non-trivial, as the region R need not contain every qubit.

#P is the class of problems that can be defined as counting the number of accepting paths of a polynomial time non-deterministic Turing machine. The problem of regional disjointness extends the problem of disjointness that was discussed in this paper, so it is at least NP-hard to determine if it is greater than some number k, but we will show that it is in fact #P-complete to calculate the regional disjointness when $c = 0$. The complexity of the problem for arbitrary c is not known.

Theorem 7. The problem of computing the regional 0-disjointness for general stabilizer codes and any region is #P-Complete.

Proof. We first show that regional disjointness for $c = 0$ is in #P. Construct a verifier that guesses a random representative of $\overline{L}$ by choosing $n-k$ bits and multiplying $\overline{L}$ by the corresponding stabilizer generators. The verifier accepts if none of the representatives of $\overline{L}$ act non-trivially on a qubit in R. The number of successful paths for this polynomial time verifier is exactly the subset disjointness in this instance. Thus, subset disjointness is in #P.

To show that the problem is #P-Hard, we reduce to #SAT. Given a boolean formula, the #SAT problem is to count the number of input assignments that satisfy the formula. In order to perform the reduction, we will make gate gadgets, which transform NOT and AND gates into sections of a stabilizer code and logical Pauli that use a constant number of qubits.

In the following diagrams, entries above the horizontal line are stabilizer generators, and the entry below the horizontal line is the logical Pauli. We use ellipses to hide parts of the stabilizer, but the operator below the line can be made to be a logical Pauli of the stabilizer by replacing the ellipses with a choice of single qubit Pauli operators on $O(n)$ qubits. Qubits of the stabilizer code that are highlighted red are in the subset R.

Consider the following stabilizer generators and logical Pauli, meant to represent a NOT gate.

$$\begin{array}{rl} \text{Input:} & Z \ \dots \\ \neg\ \text{Input:} & Z \ \dots \\ \hline \overline{L}: & Z \ \dots \end{array} \tag{45}$$

In this example, only the first qubit is in R, meaning that on all other qubits of the code, any number of representatives are allowed to act non-trivially on the qubit. There are exactly 2 representatives of this logical Pauli that satisfy the regional disjointness condition: $\overline{L}(\text{Input})$ and $\overline{L}(\neg\ \text{Input})$. If Input is chosen as the representative of $\overline{L}$, then the second qubit has the identity applied to it, and otherwise the second qubit has a single qubit Z operator applied to it.

Thus, the second qubit in this code has a single qubit Z operator applied to it if and only if the choice of representative of $\overline{L}$ corresponds to NOT picking the Input.

Now consider the following stabilizer generators and logical Pauli operator meant to represent an AND gate.

$$\begin{array}{rccccc} \text{Input 1:} & I & X & I & I & \dots \\ \text{Input 2:} & I & Z & I & I & \dots \\ & Z & Y & Z & I & \dots \\ & X & X & Z & I & \dots \\ & X & Z & Z & X & \dots \\ & X & I & I & I & \dots \\ & X & I & I & X & \dots \\ & Z & I & I & X & \dots \\ \text{Input 1} \wedge \text{Input 2:} & Z & I & I & I & \dots \\ \hline \overline{L}: & I & I & Z & I & \dots \end{array} \tag{46}$$

We claim that the only representatives of $\overline{L}$ that satisfy the subset disjointness condition on the red qubits are those that pick the last stabilizer generator if and onlyt if both input 1 and input 2 are also chosen.

If both the stabilizer generators corresponding to Input 1 and Input 2 are chosen, then the subset disjointness constraint implies that the third stabilizer generator must be picked too, as any other choice leaves the last 2 qubits with a non-trivial operator applied to them. Thus, the subset disjointness on the third qubit can only satisfy the subset disjointness constraint if the last stabilizer generator is picked too.

If either Input 1 or Input 2 is chosen, then we see that the only way to make the last 2 qubits have trivial operators is to select one of the fourth or fifth stabilizer generators, depending on whether Input 1 or 2 is chosen. We note that any stabilizer that has only has 2 of the third, fourth, and fifth stabilizer generators will violate the subset disjointness constraint on the last highlighted qubit, so no other combination of stabilizer generators can satisfy the subset disjointness constraint. If one of the fourth or fifth stabilizer generators is chosen, the sixth or seventh must also be chosen in order to satisfy subset disjointness on the third qubit and final qubit. Thus, the only representatives of $\overline{L}$ that have one of Input 1 or 2 chosen are those that also don't have the final stabilizer generator.

If neither Inputs 1 or 2 are chosen, the only representatives that satisfy subset disjointness are those that include all of the third, fourth, and fifth stabilizer generators. These representatives must select the eighth stabilizer generator too, because the single qubit X on the final qubit and Z on the third qubit can't be cancelled any other way. Thus, in this case the final stabilizer generator can not be chosen as part of the representative of $\overline{L}$.

Thus, the only representatives of $\overline{L}$ that satisfy subset disjointness have the final stabilizer generator if and only if both Input 1 and Input 2 are also in the representative of the logical Pauli operator.

Given a logic circuit composed of NOT and AND gates, we add one stabilizer generator for each gate in the circuit, including input gates. For each NOT gate, append the qubits in Table 45 with the same operators, where Input is replaced with the stabilizer generator corresponding to the input gate in the circuit, and $\neg$ Input is the generator corresponding to this NOT gate.

For each AND gate, append the qubits, stabilizer generators and single qubit operators in Table 46, where Input 1 and Input 2 are the stabilizer generators corresponding to the input gates to the AND gate, and the final stabilizer generator is the stabilizer corresponding to this AND gate. For the output of the circuit, add the following qubits and single qubit operators.

$$\begin{array}{r|cc} \text{Circuit Output:} & Z & \ldots \\ \hline \overline{L}: & \color{red}{Z} & \ldots \end{array} \tag{47}$$

This way, the only representatives of the logical Pauli that satisfy regional disjointness are those that include the stabilizer generator corresponding to the circuit output.

Finally, append qubits and single qubit operators to ensure that the code is a valid stabilizer code and $\overline{L}$ is a logical Pauli of the stabilizer. These additional qubits will be outside of R, and give the degrees of freedom required to have a non-trivial regional disjointness.

Given this choice of stabilizer and logical Pauli, the only representatives of the logical Pauli that satisfy the regional disjointness constraint pick stabilizer generators that follow the circuit logic, and who select the generator corresponding to the output of the circuit. The regional disjointness of this code and logical Pauli is exactly the number of satisfying assignments to the circuit, and the stabilizer and logical Pauli were constructed in polynomial time from the circuit, so regional disjointness if #P-Complete.

□

[1] P. Shor, in *Proceedings of 37th Conference on Foundations of Computer Science* (IEEE Comput. Soc. Press, 1996) pp. 56–65.
[2] A. M. Steane, Physical Review Letters **78**, 2252 (1997).
[3] E. Knill, Nature **434**, 39 (2005).
[4] E. Knill, Physical Review A **71**, 042322 (2005).
[5] H. Bombín, New Journal of Physics **17**, 083002 (2015).
[6] A. Kubica and M. E. Beverland, Physical Review A **91**, 032330 (2015).
[7] F. H. Watson, E. T. Campbell, H. Anwar, and D. E. Browne, Physical Review A **92**, 022312 (2015).
[8] A. Kubica, B. Yoshida, and F. Pastawski, New Journal of Physics **17**, 083026 (2015).
[9] H. Bombín, arXiv:1810.09575 (2018).
[10] T. Jochym-O'Connor and T. J. Yoder, Physical Review Research **3**, 13118 (2021).
[11] M. Vasmer and A. Kubica, in preparation (2021).
[12] H. Bombín, New Journal of Physics **18**, 043038 (2016).
[13] H. Bombín, arXiv:1810.09571 (2018).
[14] A. Kubica, *The ABCs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter*, Ph.D. thesis (2018).
[15] M. Vasmer and D. E. Browne, Physical Review A **100**, 012312 (2019).
[16] B. J. Brown, Science Advances **6** (2020), 10.1126/sciadv.aay4929.
[17] J. Iverson and A. Kubica, in preparation (2021).
[18] E. Knill, arXiv:quant-ph/0404104 (2004).
[19] E. Knill, arXiv:quant-ph/0404104 (2004).
[20] S. Bravyi and A. Kitaev, Physical Review A **71**, 022316 (2005).
[21] M. E. Beverland, A. Kubica, and K. M. Svore, PRX Quantum **2**, 020341 (2021).
[22] B. Eastin and E. Knill, Physical Review Letters **102**, 110502 (2009).
[23] B. Zeng, A. Cross, and I. L. Chuang, IEEE Transactions on Information Theory **57**, 6272 (2011).
[24] P. Faist, S. Nezami, V. V. Albert, G. Salton, F. Pastawski, P. Hayden, and J. Preskill, Physical Review X **10**, 041018 (2020).
[25] M. P. Woods and Á. M. Alhambra, Quantum **4**, 245 (2020).
[26] A. Kubica and R. Demkowicz-Dobrzański, Physical Review Letters **126**, 150503 (2021).
[27] S. Bravyi and R. König, Physical Review Letters **110**, 170503 (2013).
[28] F. Pastawski and B. Yoshida, Physical Review A **91**, 13 (2015).
[29] M. E. Beverland, O. Buerschaper, R. Koenig, F. Pastawski, J. Preskill, and S. Sijher, Journal of Mathematical Physics **57**, 44 (2016).
[30] T. Jochym-O'Connor, A. Kubica, and T. J. Yoder, Physical Review X **8**, 021047 (2018).
[31] P. Webster, M. Vasmer, T. R. Scruby, and S. D. Bartlett, arXiv:2012.05260 (2020).
[32] D. Gottesman, Physical Review A **54**, 1862 (1996).
[33] D. Gottesman and I. L. Chuang, **402**, 390 (1999).
[34] A. J. Landahl, arXiv:2010.06628 (2020).
[35] A. Calderbank and P. Shor, Physical Review A **54**, 1098 (1996).
[36] A. Steane, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **452**, 2551

(1996).
[37] E. Knill and R. Laflamme, arXiv:quant-ph/9608012 (1996).
[38] J. P. Tillich and G. Zemor, IEEE Transactions on Information Theory **60**, 1193 (2014).
[39] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman and Company, 1979).
[40] J. Hastad, *Acta Mathematica*, , 627 (1996).
[41] D. Zuckerman, Theory of Computing **3**, 103 (2007).
[42] G. B. Dantzig, Activity analysis of production and allocation **13**, 339 (1951).
[43] J. Bostanci, "Disjointness," `https://github.com/electsigon/disjointness` (2020).
[44] F. Pastawski and J. Preskill, Physical Review X **7** (2017), 10.1103/physrevx.7.021022.
[45] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations* (John Wiley & Sons, Inc., USA, 1990).
[46] C. Chekuri and S. Khanna, in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00 (Society for Industrial and Applied Mathematics, USA, 2000) p. 213–222.