# Automated searching for quantum subsystem codes

Gregory M. Crosswhite
*Department of Physics, University of Washington, Seattle, Washington 98195, USA*

Dave Bacon
*Department of Computer Science & Engineering and Department of Physics, University of Washington, Seattle, Washington 98195, USA*

(Received 7 October 2010; published 10 February 2011)

Quantum error correction allows for faulty quantum systems to behave in an effectively error-free manner. One important class of techniques for quantum error correction is the class of *quantum subsystem codes*, which are relevant both to active quantum error-correcting schemes as well as to the design of self-correcting quantum memories. Previous approaches for investigating these codes have focused on applying theoretical analysis to look for interesting codes and to investigate their properties. In this paper we present an alternative approach that uses *computational* analysis to accomplish the same goals. Specifically, we present an algorithm that computes the optimal quantum subsystem code that can be implemented given an arbitrary set of measurement operators that are tensor products of Pauli operators. We then demonstrate the utility of this algorithm by performing a systematic investigation of the quantum subsystem codes that exist in the setting where the interactions are limited to two-body interactions between neighbors on lattices derived from the convex uniform tilings of the plane.

DOI: 10.1103/PhysRevA.83.022307 PACS number(s): 03.67.Pp, 02.70.−c

## I. INTRODUCTION

Quantum computers are a technological possibility because there exist methods for building these computers out of physical components that fail to operate in an error-free manner. The theory behind achieving this makes up the field of quantum error correction [1–6] and fault-tolerant quantum computing [7–12]. Of particular note is the threshold theorem for fault-tolerant quantum computing [8–10,12]. This theorem says that if a quantum system decoheres slowly enough, and sufficiently precise control is maintained over the system, then effectively arbitrary error-free quantum computations can be performed. The way that this is achieved is through the use of quantum information which is encoded across multiple quantum subsystems into a quantum error-correcting code.

Different quantum codes have different advantages and disadvantages for implementation in a fault-tolerant device [13]. In this paper we undertake a study of an important class of quantum codes, quantum stabilizer subsystem codes [14–17] generated by measurements that are tensor products of Pauli operators. Part of the significance of this class of codes is that they can be used to implement *passive* fault tolerance by turning the measurement operators into interaction terms forming a Hamiltonian that provides energetic protection against errors; the first example of such an approach was the toric code and related models due to Kitaev [18,19], and a plethora of related approaches have now been investigated [20–28].

Previous approaches for studying quantum subsystem codes have focused on using theoretical analysis to find and investigate new quantum subsystem codes. While powerful, theoretical analysis has some disadvantages: it is limited to the “cleverness” of the analyst, and it can be prohibitively expensive to perform systematic searches of large parameter spaces to pick out the gems in the dust. In this paper, we present an alternative approach that uses *computational* analysis to accomplish the same goals. The advantage of this approach is that one becomes limited by the power of the computer rather than the brain of the analyst.[1]

In this paper we present an algorithm that computes the optimal subsystem code for a given set of measurements consisting of tensor products of Pauli operators. In the process of doing this we also develop a formalism that allows us to prove that the algorithm is correct and that the code it computes is indeed the optimal code for the given measurements. We also prove bounds on the running time of the algorithm that show that the algorithm terminates (relatively) quickly when the optimal code is not very robust to errors. Because of this property, the algorithm can be applied to sift through a class of possible measurements to determine which (if any) result in a robust code.

To demonstrate the use of this algorithm, we focus on classes of measurement operators where each measurement is limited in action to two qubits—that is, to operators taking the form $P_i \cdot Q_j$, where $P_i$ and $Q_j$ are Pauli operators acting on respectively the $i$th and $j$th qubits of the system; examples of previous subsystem codes that have been constructed with this structure are the quantum compass model subsystem code [24] (including generalizations [29,30]) and topological subsystem codes [31]. In particular we focus on systems where the measurement operators only couple qubits that are neighboring on a periodic lattice arising from the convex uniform tilings of the plane. We perform a systematic study of the codes on lattices arising from nine of the eleven such tilings, and present the results of this search.

## II. BACKGROUND

We begin by a brief review of the notion of quantum error-correcting codes and in particular the subsystem stabilizer

[1] Of course, this is also the main *disadvantage* of this approach.

codes [14]. In quantum computation we seek to reliably store and manipulate quantum information. Unfortunately, real quantum systems are open systems that couple to their environment and quickly lose their coherence through the process of decoherence. Even more troubling, when one wishes to manipulate quantum information one can only do this with a fixed precision, which means that additional error is introduced at every step of the computation. While considerable progress has been made in finding systems with long coherence times, inevitably current quantum computers will fail before they achieve anything close to the amount of computation needed, for example, to break a public key cryptosystem [32]. However it turns out that one can generally repair damage to quantum information as long as one knows the form that the damage took. Furthermore one can build a "trap"—that is to say, a *quantum code*—that tricks nature into giving up the information about what damage has occurred to the quantum system.

The nature of codes is that they separate the space in which our computation exists from the space in which the physical information is stored; that is to say, although we design our quantum circuits to operate on some Hilbert space of qubits $\mathscr{C}$, each of these qubits does *not* directly correspond to a physical qubit, but rather there is some isomorphism that relates the entire Hilbert space $\mathscr{C}$ to the Hilbert space of physical qubits, $\mathscr{P}$. To distinguish between these two Hilbert spaces, we call the Hilbert space of qubits in whose terms the computation is expressed the *computational space* (or *logical space*), and the space of qubits which have physically been built the *physical space*.

Merely building an isomorphism between these two spaces is not enough to allow us to correct errors. For one thing, we need to add extra qubits to the computational space that contain a record of the damage that we can read out; thus, we shall say that the full computational space is $\mathscr{C} := \mathscr{R} \times \mathscr{Q}$, where the qubits that exist in $\mathscr{R}$ have the role of keeping a record of the errors that have been introduced by the environment, and the qubits that exist in $\mathscr{Q}$ are the qubits in whose terms our quantum algorithm is expressed.

We have to pick a strategy for reading out the information in $\mathscr{R}$ about the errors that have occurred on our system. One natural choice is to perform a single-qubit Pauli $Z$ operator measurement on each qubit on $\mathscr{R}$. In order to build the trap element into our system, we need to ensure that whenever nature strikes at the physical space $\mathscr{P}$ and produces errors in a form that we intend to correct, this action must be isomorphic to a strike on the computational space that leaves a *measurable* record in $\mathscr{R}$. For our choice of measuring Pauli $Z$ errors, these are errors that are isomorphic to any operator that *anticommutes* with the $Z$ operator of at least one of the qubits in $\mathscr{R}$. Note that although we speak of measuring the qubits in $\mathscr{R}$, the measurement operator of interest in $\mathscr{R}$ is mapped to an operator in the physical space $\mathscr{P}$; this isomorphic operator is referred to as a *stabilizer*, and the full set of operators on $\mathscr{P}$ which are isomorphic to our chosen measurement operators on $\mathscr{R}$ are referred to as the *stabilizers* of the code.

Up to this point, the formalism we have described is known as *stabilizer codes* [6,33–35] and its essential characteristic is that in determining the syndrome of the physical error, one makes a measurement of all of the qubits in $\mathscr{R}$. What if, however, we relaxed this constraint and only measured some of the qubits in $\mathscr{R}$? That is to say, what if we split the qubits in $\mathscr{R}$ into two categories: *stabilizer qubits* whose states we care about and which we measure to obtain an error syndrome, and *gauge qubits* whose states we do not care about. (The latter get their name from the fact that they provide a "gauge" degree of freedom, i.e., a degree of freedom that is irrelevant to us.) Then we would have that $\mathscr{R} = \mathscr{S} \times \mathscr{G}$, where $\mathscr{S}$ is the space in which the stabilizer qubits exist, and $\mathscr{G}$ is the space in which the so-called gauge qubits exist; such a scheme is known as a *stabilizer subsystem code* [14]. In this case, we shall use the term *stabilizers* to denote the set of operators in $\mathscr{P}$ which are isomorphic to our chosen measurement operators of interest in $\mathscr{S}$.

At first there might not seem to be an advantage to this approach, since it essentially means adding qubits to our code that are "wasted"; however, in practice subsystem codes have many advantages. The first advantage is that since we do not care about what happens to the gauge qubits, some quantum errors on the system will neither result in detectable errors nor destroy the information in the logical qubits [14–17,24,36]. A second advantage is that we no longer need our error-correcting measurements on the physical system to commute with each other, as long as they all commute with the stabilizers and logical qubit operators, since then the fact that they do not commute affects only the gauge qubits, which we do not care about [37]. This sometimes allows one to effectively measure a stabilizer which is a nontrivial $k$-qubit measurement by using a series of two qubit measurements [37]. The individual measurements in this series do not commute (so they cannot be simultaneously measured); however, the stabilizer syndrome can nonetheless be reconstructed from these measurements. A third advantage arises from the fact that subsystem codes often require *fewer* measurements to diagnose errors than similar nonsubsystem codes, which results in improved performance [13,37]; counterintuitively, turning stabilizer codes into subsystem stabilizer codes often results in higher thresholds for fault-tolerant quantum computing. Finally, subsystem codes can often be implemented in a more local manner than nonsubsystem codes as exemplified by the quantum compass model code [24,37].

There are now many examples of stabilizer subsystem codes in the literature. One of the first nontrivial subsystem codes to be described is a code related to the quantum compass model in two dimensions [24,38,39]. In the quantum model one considers a Hamiltonian on a two-dimensional square lattice where nearest horizontal neighbors couple the $x$ component of their spins and nearest vertical neighbors couple the $z$ component of their spins, so that the Hamiltonian is given by

$$H = -\Delta \sum_{i,j} (X_{i,j} X_{i+1,j} + Z_{i,j} Z_{i,j+1}), \tag{1}$$

where $P_{i,j}$ represents the Pauli operator $P$ acting on the qubit at location $(i,j)$. This model is interesting for a few reasons. The first is that the energy levels of this system can be best thought of as elements of a quantum error-correcting subsystem code. The second reason is that the model provides some amount of protection from quantum errors because errors are energetically unfavored.[2] Many other examples of systems which have energy protecting properties are also known: the most famous being Kitaev's toric code in two and four spatial dimensions [18,19,40]. The study of such systems is still in its infancy and one central question is whether there exist Hamiltonians with reasonable physical parameters (such as existing in three or fewer spatial dimensions and involving two-body interactions [41,42]) whose physics enacts quantum error correction on the system when the system is in contact with a thermal reservoir at sufficiently low temperature; such systems are called *self-correcting* quantum computers [24,27]. In this paper we will talk about quantum subsystem codes from the perspective of active error correction, where error syndromes are identified through carefully engineered measurements, but it should be understood that this formalism can equivalently be seen from the perspective of passive error correction, where errors are guarded against by carefully engineered interactions. That is, measurement operators in the active error correction picture are equivalent to interactions in the passive error correction picture.

Because we ultimately want to build a system implementing our measurements, physical considerations typically constrain our measurements to be *local*, which means that they can be expressed in the physical space as a tensor product of single-qubit Pauli operators—i.e, for each measurement operator $o$ we have that $o := \bigotimes_i P_i$ where $P_i$ is the Pauli operator $P$ acting on the $i$th qubit. An important question then is which sets of local measurements give rise to useful quantum error-correcting subsystem codes.

Approaches to answering this question typically involve applying theoretical analysis with varying degrees of cleverness. In this paper we present an alternative approach. In Sec. III, we present an algorithm which for every set of local measurement operators computes a quantum subsystem code that arises from the algebra of these operators.[3] Along the way we develop a formalism that allows us to prove not only that this algorithm is correct, but also that the code that it computes is *optimal* in the sense that there exists no other code arising from the same set of measurements for which the distance of any of the logical qubits has been increased. This property makes this algorithm useful for analyzing the properties of codes arising from measurements that are too overwhelming to analyze by hand.

We shall also show that an important property of this algorithm is that it terminates (relatively) quickly when the distance of the code is small, which allows it to be used not only to solve for individual codes, but also to search through entire classes of sets of measurements to see if any have high-distance qubits. Motivated by previous results demonstrating the utility of codes implemented using systems on a lattice, we undertake a systematic investigation of codes where the measurement operators are restricted to the two-body interactions arising from the edges of periodic lattices derived from the 11 regular tilings. In Sec. IV we discuss our approach for applying the algorithm to perform a systematic search for codes that can be implemented on these tilings, and we then present numerical results that we obtained. In Sec. V we present our conclusions.

### A. Notation

In this paper we adopt the following conventions for notation:

(a) *sets* are denoted by a symbol with a tilde, e.g. $\tilde{A}$;

(b) *sequences* are denoted by a symbol with an arrow, e.g. $\vec{A}$;

(c) *operators* and *integers* are denoted by using lower-case letters, e.g. $o$ and $i$;

(d) *collections* of *operators* and *pairs of operators* are denoted by using upper-case letters with either a tilde or an arrow above them, e.g., $\tilde{O}$ and $\vec{O}$;

(e) *collections* of *integers* are denoted by using lower-case letters with either a tilde or an arrow above them, e.g., $\tilde{k}$ and $\vec{k}$;

(f) and *collections* of *other kinds* of objects are typically denoted by capital letters in a fancy script.

## III. THEORY

### A. Construction of the subsystem code

*Remark.* This section describes by way of a constructive proof an algorithm that, given a set of measurement operators, computes a quantum code that can be implemented by these operators. For a listing of pseudocode that implements this algorithm, see Table I near the end of this section.

Although conceptually a subsystem code is an isomorphism $T$ such that $\mathscr{P} \approx^T \mathscr{S} \times \mathscr{G} \times \mathscr{Q}$—that is, an isomorphism between the physical space of qubits and the computational space of qubits in whose terms our computation is actually expressed—we do not need to actually construct this isomorphism in order to be able to use the code. Since all of our work will be done on the physical system anyway, it suffices to know the operators in the physical space $\mathscr{P}$ that are isomorphic to the qubit measurement operators of interest in the computational space $\mathscr{S} \times \mathscr{G} \times \mathscr{Q}$, and it is exactly the operators on $\mathscr{P}$ that the algorithm we present will compute.[4]

When one wants to define a qubit in terms of its measurement operators, it suffices to define two operators that anticommute with each other but which commute with all of the other measurement operators that have been defined, since this gives us the $X$ and $Z$ measurements on the qubit which are

[2] Unfortunately, in this particular system the protection vanishes as the size of the lattice goes to infinity [39], but for small lattice sizes there is some protection from errors due to the energy level structure of the system [38].

[3] We say that we compute "a" code rather than "the" code because there is almost never a unique solution, since among other transformations one can multiply every gauge and logical qubit operator by an element from the stabilizers and end up with an equivalent code.

[4] If one really wanted to, one could explicitly construct the isomorphism $\mathscr{T}$ from these operators by computing the unitary operator which simultaneously diagonalizes the maximal subset of commuting measurements from this set of operators on $\mathscr{P}$, but in practice this is not particularly useful.

sufficient to generate the full *Pauli* group (minus phases). Since working with such pairs of operators will be a common theme in this algorithm, we shall introduce the following definition in order to simplify the language used to describe them.

*Definition.* A pair of operators is a *conjugal pair in relation to the set* $\tilde{X}$ when each of the operators in the pair commutes with every operator in $\tilde{X}$ except for its *conjugal partner*—that is, the other operator in the conjugal pair—should its conjugal partner be a member of $\tilde{X}$.

Note that we have explicitly not required that the operators in the conjugal pair be members of $\tilde{X}$ in order to be a conjugal pair in relation to it. However, should both operators be members of $\tilde{X}$, then neither operator can belong to a different conjugal pair with respect to $\tilde{X}$, since in that case there would be an operator in $\tilde{X}$ (namely, its original conjugal partner) with which it anticommutes that was not its conjugal partner in the new pair, leading to a contradiction.

For convenience, we introduce the following additional definitions:

*Definition.* (1) $\tilde{\mathfrak{P}}$ is the group of Pauli operators—that is, the group of tensor products of the (unnormalized) Pauli matrices—acting on the physical space $\mathscr{P}$, *modulo phases*;

(2) $\tilde{\mathcal{P}}(\tilde{S})$ is the power set of $\tilde{S}$, i.e. the set of all subsets of $\tilde{S}$;

(3) and $\tilde{\mathcal{C}}_{\mathfrak{G}}(\tilde{S})$ is the centralizer of $\tilde{S}$, that is the subgroup of elements in $\mathfrak{G}$ which commute with $\tilde{S}$;

(4) the function $\tilde{\mathcal{G}} : \tilde{\mathcal{P}}(\tilde{\mathfrak{P}}) \to \tilde{\mathcal{P}}(\tilde{\mathfrak{P}})$ is defined such that $\tilde{\mathcal{G}}(\tilde{S})$ is the set of all possible products of operators in $\tilde{S}$—that is, it is the set *generated* by $\tilde{S}$.

We now introduce the main theorem of this section.

*Theorem 1.* Suppose that we are given a sequence of Pauli operators $\vec{O}$. Then there exist sets of Pauli operators $\tilde{S} \subseteq \tilde{\mathfrak{P}}$, $\tilde{G} \subseteq \tilde{\mathfrak{P}}$, and $\tilde{L} \subseteq \tilde{\mathfrak{P}}$ such that

(1) each of the operators in $\tilde{S} \cup \tilde{G} \cup \tilde{L}$ is independent from the rest—i.e., no operator in this (unioned) set can be written as a product of other operators in the set;

(2) each operator in $\tilde{L} \cup \tilde{G}$ is a member of a conjugal pair in relation to $\tilde{S} \cup \tilde{G} \cup \tilde{L}$;

(3) $\tilde{\mathcal{G}}(\tilde{S} \cup \tilde{G}) = \tilde{\mathcal{G}}(\{\vec{O}_i\})$;[5]

(4) and $\tilde{\mathcal{G}}(\tilde{S} \cup \tilde{G} \cup \tilde{L}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$.

*Remark.* This theorem follows, at least implicitly, from prior work on stabilizer codes [6], the definitions of stabilizer subsystem codes given by Poulin [14], and the constructive approach to finding such codes as exemplified in [24]. Because we wish to be constructive, however, we will present a full proof of this theorem and show how it gives rise to an algorithm for finding sets of Pauli operators which satisfy Theorem 1. To be explicit, we note that $\tilde{S}$ will be a set of stabilizers (or equivalently, generators for the stabilizer group), $\tilde{G}$ will be a set of gauge qubit operators, and $\tilde{L}$ will be a set of logical qubit operators (i.e., those on which the computation is performed).

The main work in the proof of this theorem will be performed by proving several related propositions. First we shall show how the set $\tilde{G}$ and a sequence $\vec{S}$ are constructed from the sequence of operators $\vec{O}$. Since we want our stabilizers to form an independent set of operators, we shall then show that through a Gaussian elimination procedure it is possible to extract a list of independent operators from a sequence $\vec{S}$ resulting in a set $\tilde{S}$. Finally, we shall show how using this same Gaussian elimination procedure we can transform a subset of the operators of $\tilde{S} \cup \tilde{G}$ into a form that makes it trivial to compute the logical qubit operators $\tilde{L}$.

*Proposition 1.* Suppose that we are given a sequence of Pauli operators $\vec{O} \subseteq \tilde{\mathfrak{P}}$. Then there exists a sequence of Pauli operators $\vec{S} \subseteq \tilde{\mathfrak{P}}$ and a set of Pauli operators $\tilde{G} \subseteq \tilde{\mathfrak{P}}$ such that

(1) all of the operators in $\vec{S}$ commute with each other and also all of the operators in $\tilde{G}$;

(2) each operator in $\tilde{G}$ is a member of a *conjugal pair* (see the first definition in Sec. III A) in relation to $\{\vec{S}_i\} \cup \tilde{G}$; and

(3) $\tilde{\mathcal{G}}(\{\vec{S}_i\} \cup \tilde{G}) = \tilde{\mathcal{G}}(\{\vec{O}_i\})$.

*Proof.* The proof is by induction. For the base case, note that if $\vec{O}$ is empty then $\vec{S} := \emptyset$ and $\tilde{G} := \emptyset$ trivially satisfy all properties.

Now assume that the proposition holds for a sequence of length $n-1$, and consider a sequence of operators $\vec{O}$ of length $n$. By the inductive hypothesis, we know that there is a sequence $\vec{S}'$ and a set $\tilde{G}'$ satisfying the properties above for the subsequence of $\vec{O}$ consisting of the first $n-1$ operators. Let $o := \vec{O}_n \cdot \prod_{g \in \tilde{G}, \{\vec{O}_n, g\} = 0} \text{conj}_{\tilde{G}}(g)$—that is, the product of $\vec{O}_n$ with the conjugal partner of every operator in $\tilde{G}$ with which $\vec{O}_n$ anticommutes. This definition guarantees that $o$ commutes with every operator in $\tilde{G}$; furthermore, we can obtain $\vec{O}_n$ back from $o$ since every operator in $\tilde{G}$ squares to the identity and thus $\vec{O}_n = o \cdot \prod_{g \in \tilde{G}, \{\vec{O}_n, g\} = 0} \text{conj}_{\tilde{G}}(o)$; therefore we conclude that $\tilde{\mathcal{G}}(\{\vec{S}_i'\} \cup \tilde{G}' \cup \{o\}) = \tilde{\mathcal{G}}(\{\vec{O}_i\})$.

If $o$ commutes with every operator in $\vec{S}'$, then set

$$\vec{S}_i := \begin{cases} \vec{S}_i', & i \leqslant n-1, \\ o, & i = n, \end{cases}$$

and $\tilde{G} := \tilde{G}'$, and we are done. Otherwise, let $s$ be some operator in $\vec{S}'$ that anticommutes with $o$, $\tilde{G} := \tilde{G}' \cup \{s, o\}$,[6] $\vec{S}_i'' := f(\vec{S}_i')$, and $\vec{S}$ be the subsequence of $\vec{S}''$ with the identity operators removed, where

$$f(s') := \begin{cases} s' \cdot s, & \{s', o\} = 0, \\ s' & \text{otherwise.} \end{cases}$$

Observe that by this definition, all of the operators in $\vec{S}$ commute with every operator in $\tilde{G}$, so property 1 is satisfied. Since the only difference between $\tilde{G}'$ and $\tilde{G}$ is the addition of $s$ and $o$, which form a conjugal pair with respect to $\{\vec{S}_i\} \cup \tilde{G}$, we conclude that property (1) is satisfied. Lastly, since $s \in \tilde{G}$, we can form any operator in $\vec{S}'$ with products of operators in

[5] Here we use the notation $\{\vec{O}_i\}$ to refer to the set of elements in the sequence $\vec{O}$.

[6] Observe that neither $o$ nor $s$ can be present in $\tilde{G}'$ since they commute with every operator in $\tilde{G}'$, so the new set $\tilde{G} := \tilde{G}' \cup \{s, o\}$ gives us a strictly larger set. This fact is irrelevant as far as the proof is concerned, but it has the important consequence that a computer code implementing the algorithm described by this proof can append $s$ and $o$ to a list of gauge operators and assume that this list continues to form a set (i.e., a sequence without duplicates) without having to explicitly check for this.

$\vec{S}$ and $\tilde{G}$, so therefore $\tilde{\mathcal{G}}(\{\vec{S}_i\} \cup \tilde{G}) = \tilde{\mathcal{G}}(\{\vec{S}'_i\} \cup G' \cup \{s,o\}) = \tilde{\mathcal{G}}(\{\vec{O}_i\})$, and so the final property is satisfied.

We conclude by noting that since all of the operators in $\vec{S}$ and $\tilde{G}$ were formed from products of operators in $\vec{O}$, which are Pauli operators (i.e., members of the group $\tilde{\mathfrak{P}}$), they are Pauli operators themselves. ■

*Remark.* A consequence of not requiring independence of the operators in $\vec{O}$ is that the operators $\vec{S}$ given by Proposition 1 are not necessarily independent. Happily, since all of these operators can be expressed as tensor products of Pauli operators, we can construct a set of independent operators by performing an analog of Gaussian elimination.

*Proposition 2.* Suppose that we have been given a sequence of Pauli operators which commute with each other, $\vec{R}$. Then there exists

(1) a sequence $\vec{S}$ of $n$ independent operators such that $\tilde{\mathcal{G}}(\{\vec{S}_i\}) = \tilde{\mathcal{G}}(\{\vec{R}_i\})$,

(2) a sequence of $n$ integers without duplicates in the inclusive range $1, \ldots, n$,

(3) and a map $p : \{1, \ldots, n\} \to \{0,1\}$ such that $\vec{S}_i$ is the only operator in $\vec{S}$ that anticommutes with $P_{k_i}^{[p(i)]}$, where $P_k^{[0]} := X_k$ and $P_k^{[1]} := Z_k$.

*Proof.* The proof is by induction. For the base case, we observe that if $\vec{R}$ is empty, then the trivial sequences $\vec{S} := \emptyset$ and $\vec{k} := \emptyset$ and the trivial function $p : \emptyset \to \emptyset$ satisfy the requirements.

Now suppose that we know the proposition holds for sequences of length $N-1$, and we are given a sequence $\vec{S}$ of length $N$. By our inductive hypothesis, we can apply the proposition to the first $N-1$ operators in $\vec{R}$ obtain sequences $\vec{S}'$ and $\vec{k}'$ of length $n-1$,[7] and a map $p' : \{1, \ldots, n-1\} \to \{0,1\}$ which all satisfy the respective properties of the theorem. Let

$$s := \vec{R}_N \cdot \prod_{i=1,\ldots,n-1,\ \{\vec{R}_N, P_{k'_i}^{[p(i)]}\}=0} \vec{S}'_i .$$

We know that $s$ commutes with every operator in $\vec{S}'$ because both $s$ and every operator in $\vec{S}'$ are equal to products of operators in $\vec{R}$, which all commute with each other. Furthermore, since $s$ is a product of $\vec{R}_N$ and a factor of $\vec{S}'_i$ for every $i$ such that $\vec{R}_N$ and $P_{k'_i}^{[p'(i)]}$ anticommute, and we know that $\vec{S}'_i$ is the only operator in $\vec{S}'$ that anticommutes with $P_{k'_i}^{[p'(i)]}$ for $i = 1, \ldots, n-1$, it is therefore the case that $s$ commutes with every member of the set $\{P_{k'_i}^{[p'(i)]}\}_{i=1,\ldots,n-1}$. Finally, since $s$ is a product of $\vec{R}_N$ and operators in $\vec{S}'$, we can obtain $\vec{R}_N$ entirely from products of operators in $\{\vec{S}'_i\} \cup \{s\}$, and so $\tilde{\mathcal{G}}(\{\vec{S}'_i\} \cup \{s\}) = \tilde{\mathcal{G}}(\{\vec{R}_i\})$.

If $s$ is the identity operator, then let $\vec{S} := \vec{S}'$ and $p := p$ and we are done. Otherwise, we shall now show that there must exist integers $j \in \{1, \ldots, N\} \backslash \{\vec{k}'_i\}$ and $l \in \{0,1\}$ such that $s$ anticommutes with $P_j^{[l]}$, by demonstrating that if this were not the case then $s$ would have to anticommute with some element in $\vec{S}'$, leading to a contradiction.

Assume that $s$ commutes with every operator in the set $\{P_j^{[l]} :\ j \in \{1, \ldots, N\} \backslash \{\vec{k}'_i\},\ l \in \{0,1\}\}$. Recalling that $s$ is a member of the Pauli group and thus a tensor product of single-particle Pauli spin matrices, and also that $s$ commutes with every member of the set $\{P_{\vec{k}'_i}^{[p'(i)]}\}_{i=1,\ldots,n-1}$, we see therefore that $s$ must be a product of elements from this set—that is, there is some subset $\emptyset \neq \tilde{F} \subseteq \{P_{\vec{k}'_i}^{[p'(i)]}\}_{i=1,\ldots,n-1}$ such that $s = \prod_{o \in \tilde{F}} o$. However, from our inductive hypothesis we know that for every operator $f \in \tilde{F}$ there is an operator $s' \in \vec{S}'$ that anticommutes with $f$ but commutes with the operators in $\tilde{F} \backslash \{f\}$. Since $s$ is therefore a product of a single operator that anticommutes with $s'$ and more operators that commute with $s'$, we conclude that $s$ and $s'$ anticommute, which contradicts our earlier conclusion that $s$ commutes with every operator in $\vec{S}'$.

Now that we have shown that there exist integers $j \in \{1, \ldots, N\} \backslash \{\vec{k}'_i\}$ and $l \in \{0,1\}$ such that $s$ anticommutes with $P_j^{[l]}$, in terms of these integers we define

$$\vec{S}_i := \begin{cases} \begin{cases} \vec{S}'_i \cdot s, & \{\vec{S}'_i, P_j^{[l]}\} = 0, \\ \vec{S}'_i & \text{otherwise}, \end{cases} & 1 \leqslant i \leqslant n-1, \\ S', & i = n, \end{cases}$$

$$\vec{k}_i := \begin{cases} \vec{k}'_i, & 1 \leqslant i \leqslant n-1, \\ j, & i = n, \end{cases} \quad \text{and}$$

$$p(i) := \begin{cases} p'(i), & 1 \leqslant i \leqslant n-1, \\ l, & i = n, \end{cases}$$

and we are done. ■

*Remark.* Proposition 2 is good for more than computing an independent set of generators from a commuting list of operators; it is also the key ingredient in computing the logical qubit operators.

*Proposition 3.* Suppose that we have been given the objects described in (1)–(3) of Proposition 2. Let $\tilde{S} := \{\vec{S}_i\}_i$. Then there exists a set of operators $\tilde{L}$ such that

(1) the operators in $\tilde{S} \cup \tilde{L}$ are independent;

(2) every operator in $\tilde{L}$ is a member of a conjugal pair with respect to $\tilde{S} \cup \tilde{L}$;

(3) $\tilde{\mathcal{G}}(\tilde{S} \cup \tilde{L}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$—that is, the set generated by $\tilde{S} \cup \tilde{L}$ is equal to the set of Pauli operators that commute with $\tilde{S}$.

*Proof.* Recalling that $n$ is the number of elements in $\vec{S}$ (and $\tilde{S}$), let $\vec{l}$ be some sequential ordering of $\{1, \ldots, N\} \backslash \{\vec{k}_i\}_i$, and then let $\tilde{L} := \{\vec{A}_i\}_i \cup \{\vec{B}_i\}_i$ where

$$\vec{A}_i := P_{\vec{l}_i}^{[1]} \cdot \prod_{\substack{j=1,\ldots,n \\ \{P_{l_i}^{[1]}, \vec{S}_j\}=0}} P_{\vec{k}_j}^{[s(j)]}, \quad \vec{B}_i := P_{\vec{l}_i}^{[0]} \cdot \prod_{\substack{j=1,\ldots,n \\ \{P_{l_i}^{[0]}, \vec{S}_j\}=0}} P_{\vec{k}_j}^{[s(j)]}.$$

To see that property (3) is satisfied, observe the following. First, the operators in $\tilde{L}$ are independent from the operators in $\tilde{S}$ since none of them is the identity operator and they all commute with every operator in $\{P_{\vec{k}_i}^{[s(i)]}\}_{i=1,\ldots,n}$. Second, they are independent from each other since for every $i = 1, \ldots, |\vec{l}|$ we have that $\vec{A}_i$ is the only operator that anticommutes with $P_{\vec{l}_i}^{[0]}$ and $\vec{B}_i$ is the only operator that anticommutes with $P_{\vec{l}_i}^{[1]}$.

[7] Note that $n \neq N$ in general, since some of the first $N-1$ operators might not have been independent.

Thus we conclude that all of the operators in $\tilde{S} \cup \tilde{L}$ are independent.

Next, to see that property (3) holds, observe that for every choice of operators $\vec{A}_i$ and $\vec{S}_j$ we have (by intentional construction) that $\vec{S}_j$ either anticommutes with two of the operators in the product forming $\vec{A}_i$ or with none at all, and so $[\vec{S}_i,\vec{A}_j] = 0$ for all $i = 1,\ldots,n$ and $j = 1,\ldots,|\vec{l}|$; by the same reasoning we see also that $[\vec{S}_i,\vec{B}_j] = 0$ for all $i = 1,\ldots,n$ and $j = 1,\ldots,|\vec{l}|$. Furthermore, each operator $\vec{A}_i$ commutes with every operator in $\tilde{L}$ except for its conjugal partner $\vec{B}_i$, since the only factor in $\vec{A}_i$ that could anticommute with a factor contained within another operator in $\tilde{L}$ is $P^{[1]}_{l_i}$, and $\vec{B}_i$ is the only operator in $\tilde{L}$ that contains a factor $P^{[0]}_{l_i}$ that anticommutes with $\vec{X}_{l_i}$; reversing this argument, we also see that $\vec{B}_i$ commutes with every operator in $\tilde{L}$ except for $\vec{A}_i$. Thus, every operator in $\tilde{L}$ is a member of a conjugal pair with respect to $\tilde{L} \cup \tilde{S}$.

Finally, to see that property (3) holds, observe that since the operators in $\tilde{S}$ commute they can therefore be simultaneously diagonalized, which means that there is an automorphism on $\tilde{\mathfrak{P}}$ that takes $\vec{S}_i \mapsto P^{[1]}_i$ for every $i = 1,\ldots,n$. The only operators that commute with every such $P^{[1]}_i$ are those which do not contain any factor of $P^{[0]}_i$ for $i = 1,\ldots,n$, and so $\tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\{P^{[0]}_i\}_{i=1,\ldots,n}) = \tilde{\mathcal{G}}(\{P^{[1]}_i\}_{i=1,\ldots,n} \cup \{P^{[l]}_i\}_{i=n+1,\ldots,N,\ l=0,1})$, which has $2N - n$ generators. Since the automorphism preserves the number of generators in the centralizer, we thus conclude that $\tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ has exactly $2N - n$ generators. Since $\tilde{S} \cup \tilde{L}$ contains independent operators which commute with every member of $\tilde{S}$, and furthermore $|\tilde{S} \cup \tilde{L}| = 2N - n$, we thus conclude that $\tilde{\mathcal{G}}(\tilde{S} \cup \tilde{L}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$. ■

With these building blocks in place, we now prove the main theorem:

*Proof of Theorem 1.* By Proposition 1, we know that there exists a list of operators $\vec{S}$ and a set of independent operators $\tilde{G}$ satisfying the properties that are listed there. By Proposition 2, we know that there is an independent set of operators $\tilde{S}$ that generate the same subgroup as $\vec{S}$.

Now let $\tilde{F}$ be a maximal subset of commuting operators in $\tilde{G}$—i.e., for each conjugal pair in $\tilde{G}$ take one of the two operators-and then let $\tilde{O} := \tilde{F} \cup \tilde{S}$. Since all of the operators in $\tilde{O}$ commute, we apply Proposition 2 again to conclude the existence of the objects listed there, and then we immediately apply Proposition 3 to show that a set $\tilde{M}$ exists with the properties listed there. We are not done yet, however, since there might be operators in $\tilde{G}$ with which operators in $\tilde{M}$ anticommute, so we let

$$\tilde{L} := \left\{ m \cdot \prod_{\substack{f \in \tilde{F} \\ \{M,\mathrm{conj}_{\tilde{G}}(f)\}=0}} f : \quad m \in \tilde{M} \right\},$$

where $\mathrm{conj}_{\tilde{G}}(F)$ is the conjugal partner of $F$ in the set $\tilde{G}$. This guarantees that the operators in $\tilde{L}$ commute with every operator in $\tilde{S} \cup \tilde{G}$, and so we are done. ■

*Remark.* A pseudocode representation of the algorithm described by Theorem 1 is given in Table I.

### B. Optimization of the logical qubits

*Remark.* A pseudocode representation of the algorithm that will be described in this section is presented in Table III.

TABLE I. Algorithm which computes the subsystem code generated by a given list of measurement operators $\vec{O}$. The subroutine GAUSSIAN-ELIMINATION is listed in Table II.

```
1   $\vec{S} \leftarrow []$
2   $\vec{G} \leftarrow []$
3   for $o \leftarrow \vec{O}$
4       do
5           for $(g_X, g_Z) \leftarrow \vec{G}$
6               do
7                   if anti$(o, g_X)$ then $o \leftarrow o \cdot g_Z$
8                   if anti$(o, g_Z)$ then $o \leftarrow o \cdot g_X$
9           if $o$ is identity then goto 3
10          for $s \leftarrow \vec{S}$
11              do
12                  if anti$(o, s)$ then goto 14
13          goto 3
14          $\vec{G} \leftarrow \vec{G} \cup [(o, s)]$
15          $i \leftarrow 1$
16          for $s' \leftarrow \vec{S}$
17              do
18                  if $s' = s$ then goto 16
19                  if anti$(s', o)$
20                    then
21                          $\vec{S}[i] \leftarrow s' \cdot s$
22                    else
23                          $\vec{S}[i] \leftarrow s$
24                  $i \leftarrow i + 1$
25          delete $\vec{S}[i \ldots |\vec{S}|]$
26  $\vec{I} \leftarrow []$
27  $\vec{P} \leftarrow []$
28  call GAUSSIAN-ELIMINATION$(\vec{S}, 1, \vec{I}, \vec{P})$ (Table II)
29  $\vec{T} \leftarrow \vec{S} \cup [g_X | (g_X, g_Z) \in \vec{G}]$
30  call GAUSSIAN-ELIMINATION$(\vec{T}, |\vec{S}| + 1, \vec{I}, \vec{P})$ (Table II)
31  $\vec{L} \leftarrow []$
32  for $i \leftarrow 1$ to number of physical qubits
33      do
34          if $i \in \vec{I}$ then goto 32
35          $l_X \leftarrow X_i$
36          $l_Z \leftarrow Z_i$
37          for $(j, p, t) \leftarrow (\vec{I}, \vec{P}, \vec{T})$
38              do
39                  if $p = 0$
40                    then
41                          if anti$(t, X_j)$ then $l_X \leftarrow l_X \cdot Z_j$
42                          if anti$(t, Z_j)$ then $l_Z \leftarrow l_Z \cdot Z_j$
43                    else
44                          if anti$(t, X_j)$ then $l_X \leftarrow l_X \cdot X_j$
45                          if anti$(t, Z_j)$ then $l_Z \leftarrow l_Z \cdot X_j$
46          for $(g_X, g_Z) \in \vec{G}$
47              do
48                  if anti$(l_X, g_Z)$ then $l_X \leftarrow l_X \cdot g_X$
49                  if anti$(l_Z, g_Z)$ then $l_Z \leftarrow l_Z \cdot g_X$
50  return $(\vec{S}, \vec{G}, \vec{L})$
```

In general there are multiple sets of operators that satisfy the properties of Theorem 1, as is illustrated by the following lemma:

*Lemma 1.* Given conjugal pairs $Q := (a,b)$ and $R := (c,d)$ in relation to some set $\tilde{X}$ such that either $a \neq c$ or $b \neq d$, we have that

(1) the pairs $Q' := (a \cdot c, b)$ and $R' := (c, d \cdot b)$ are conjugal pairs with respect to $\tilde{X} \backslash \{Q, R\} \cup \{Q', R'\}$; and

(2) $\tilde{\mathcal{G}}(\{a,b,c,d\}) = \tilde{\mathcal{G}}(\{a \cdot c, b, c, d \cdot b\})$.

TABLE II. Subroutine which performs a procedure analogous to Gaussian elimination on $\vec{S}$ to distill a set of independent operators from a possible dependent set of operators. This subroutine is called by the main subsystem code algorithm in Table I.

```
1   while $i < |\vec{S}|$
2       do
3           $s \leftarrow \vec{S}[i]$
4           for $j \leftarrow 0$ to $i - 1$
5               do
6                   $(n, z) \leftarrow (\vec{I}[j], \vec{P}[j])$
7                   if $z = 0$
8                     then
9                           if anti($s, X_n$)
10                            then $s \leftarrow s \cdot \vec{S}[j]$
11                    else
12                          if anti($s$‹ $Z_n$)
13                            then $s \leftarrow s \cdot \vec{S}[j]$
14          if $s$ is identity
15            then
16                  delete $\vec{S}[i]$
17                  goto 1
18          for $n \leftarrow 0$ to number of physical qubits
19              do
20                  if $n \in \vec{I}$ then goto 18
21                  if anti($s, X_n$)
22                    then
23                          $z \leftarrow 0$
24                          goto 29
25                  if anti($s, Z_n$)
26                    then
27                          $z \leftarrow 1$
28                          goto 29
29          if $z = 0$
30            then
31                  for $j \leftarrow 0$ to $i - 1$
32                      do
33                          if anti($\vec{S}[j], X_n$)
34                            then $\vec{S}[j] \leftarrow \vec{S}[j] \cdot s$
35            else
36                  for $j \leftarrow 0$ to $i - 1$
37                      do
38                          if anti($\vec{S}[j], Z_n$)
39                            then $\vec{S}[j] \leftarrow \vec{S}[j] \cdot s$
40          append $n$ to $\vec{I}$
41          append $z$ to $\vec{P}$
42          $\vec{S}[i] \leftarrow s$
43          $i \leftarrow i + 1$
```

*Proof.* (1) Since $[a,c] = [a,d] = [b,c] = [b,d] = \{a,b\} = \{c,d\} = 0$, we see therefore that $[a \cdot c, c] = [a \cdot c, d \cdot b] = [b,c] = [b, d \cdot b] = \{a \cdot c, b\} = \{c, d \cdot b\} = 0$. Furthermore, since $a$, $b$, $c$, and $d$ commute with every operator in $\tilde{X} \backslash \{Q, R\}$, so do $a \cdot c$ and $d \cdot b$.

(2) Since $b$ and $c$ are Pauli operators and thus square to the identity, we have that $a \cdot c \cdot c = a$ and $d \cdot b \cdot b = d$, and so $\tilde{\mathcal{G}}(\{a,b,c,d\}) = \tilde{\mathcal{G}}(\{a \cdot c, b, c, d \cdot b\})$. ■

As a result of this lemma, we see that we can take pairs of arbitrary conjugal pairs from sets $\tilde{G}$ and $\tilde{L}$ of Theorem 1 and replace them with different pairs per the recipe in Lemma 1 such that the properties of the theorem still hold. So given that these sets are not unique, the natural question arises: What is the best choice of $\tilde{G}$ and $\tilde{L}$? To answer this, we observe that another criterion we would like our code to satisfy is that it be as robust to errors as possible; in particular, we seek to maximize the difficulty of *undetectable errors*, which are defined as follows:

TABLE III. Algorithm which optimizes the logical qubits for a given subsystem code.

```
1   $N \leftarrow |\vec{L}|$
2   $k \leftarrow 0$
3   $\vec{m} \leftarrow$ [TRUE] $* |\vec{L}|$
4   nested function QUERY-FUNCTION($o$)
5       do
6           for $i \leftarrow 0$ to $k$, $(l_X, l_Z) \leftarrow \vec{L}[i]$
7               do
8                   if $\vec{m}[i]$ and anti($o, l_Z$)
9                     then
10                          return (TRUE, (1,$i$))
11          for $i \leftarrow k + 1$ to $N$, $(l_X, l_Z) \leftarrow \vec{L}[i]$
12              do
13                  if anti($o, l_X$) or anti($o, l_Z$)
14                    then
15                          return (TRUE, (2, $i$))
16          return (FALSE, UNDEFINED)
17  $\vec{O} \leftarrow$ copy($\vec{S}$)
18  for $(g_X, g_Z) \leftarrow \vec{G}$
19      do
20          append $g_X$ and $g_Z$ to $\vec{O}$
21  for $(l_X, l_Z) \leftarrow \vec{L}$
22      do
23          append $l_X$ and $l_Z$ to $\vec{O}$
24  $\vec{P} \leftarrow$ COMPUTE-PSEUDOGENERATORS($\vec{O}$) (Table VI)
25  while $k < N$
26      do
27          $(e, (c, l)) \leftarrow$ FIND-WEIGHT-MINIMIZER
                        (QUERY-FUNCTION, $\vec{P}$) (Table V)
28          $(q_X, q_Z) \leftarrow \vec{L}[l]$
29          if $c = 1$
30            then
31                  $\vec{m}[l] \leftarrow$ FALSE
32                  for $i \leftarrow l + 1$ to $k$, $(l_X, l_Z) \leftarrow \vec{L}[i]$
33                      do
34                          if $m[i]$ and $\{e, l_Z\}$
35                            then
36                                  $q_X \leftarrow q_X \cdot l_X$
37                                  $\vec{L}[i] \leftarrow (l_X, l_Z \cdot q_Z)$
38                  call FIX-LOGICAL-QUBITS
                        ($\vec{L}, k, e, q_Z, q_X$) (Table IV)
39                  $L[l] \leftarrow (q_X, q_Z)$
40            else
41                  if commute($e, q_X$)
42                    then swap $q_X$ and $q_Z$
43                  $(q_X, q_Z) \leftarrow \vec{L}[l]$
44                  $\vec{L}[l] \leftarrow \vec{L}[k]$
45                  $k \leftarrow k + 1$
46                  call FIX-LOGICAL-QUBITS
                        ($\vec{L}, k, e, q_X, q_Z$) (Table IV)
47                  $L[k] \leftarrow (q_X, q_Z)$
```

TABLE IV. Algorithm which "fixes" a subset of the logical qubits so that they are robust to a given error.

```
for $i \leftarrow k$ to $|\vec{L}|$, $(l_X, l_Z) \leftarrow \vec{L}[i]$
    do
        if $\text{anti}(e, l_Z)$ and $\text{anti}(e, l_X)$
          then
                $q_A \leftarrow q_A \cdot l_X \cdot l_Z$
                $\vec{L}[i] \leftarrow (l_X \cdot q_B, l_Z \cdot q_B)$
        elseif $\text{anti}(e, l_Z)$
          then
                $q_A \leftarrow q_A \cdot l_X$
                $\vec{L}[i] \leftarrow (l_X, l_Z \cdot q_B)$
        elseif $\text{anti}(e, l_X)$
          then
                $q_A \leftarrow q_A \cdot l_Z$
                $\vec{L}[i] \leftarrow (l_X \cdot q_B, l_Z)$
if $\text{ANTI}(e, q_A)$
    then $q_A \leftarrow q_A \cdot q_B$
return $(q_A, q_B)$
```

*Definition.* Given a set $\tilde{S} \subseteq \tilde{\mathfrak{P}}$ and operators $l \in \tilde{\mathfrak{P}}$ and $e \in \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ which anticommute (i.e., $\{l,e\} = 0$), we say that $e$ is an *undetectable error with respect to $\tilde{S}$ acting on $l$*.

We assume that the "difficulty" of an interaction between our physical system and its environment is related to the number of physical qubits in our system that are participating in the interaction. Thus, the natural metric for measuring the relative difficulty of an error is given by its weight, which recall is defined as follows:

*Definition.* Given an operator $p \in \tilde{\mathfrak{P}}$—which recall means that $p$ is the tensor product of single-qubit Pauli unnormalized spin matrices—the *weight* of $p$ is the number of single-qubit operators in the product which are nontrivial (i.e., not the identity). So, for example, the weight of $I \otimes I \otimes I$ is 0, the weight of $I \otimes Z \otimes I \otimes X$ is 2, and the weight of $Z \otimes X \otimes Y$ is 3.

For convenience, we introduce the following additional notation:

*Definition.* (a) the function $\Pi : \tilde{\mathcal{P}}(\tilde{\mathfrak{P}}) \to \tilde{\mathfrak{P}}$ is defined such that $\Pi(\tilde{X}) := \prod_{x \in \tilde{X}} x$—that is, it is the product of the operators in $\tilde{X}$;

(b) assuming we have a set of independent operators, $\tilde{Q}$, the function $G_{\tilde{Q}} : \mathcal{G}(Q) \to \mathcal{P}(Q)$ is defined (uniquely) such that for every $q \in \tilde{Q}$ we have that $q = \prod_{o \in \tilde{G}_{\tilde{Q}}(q)} o$;

(c) the function $w : \tilde{\mathfrak{P}} \to \mathcal{N}$ is defined such that $w(o)$ gives the weight of $o$;

(d) the function $e_{\tilde{S}} : \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}) \to (\tilde{\mathcal{P}} \circ \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}})(\tilde{S})$ is defined such that $\tilde{e}_{\tilde{S}}(l)$ is the set of minimizers of $w$ over the set $\{o : o \in \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}), \{o,l\} = 0\}$—that is, it gives the undetectable errors with respect to $\tilde{S}$ acting on $l$ that are of minimum weight;

(e) the function $\omega_{\tilde{S}} : \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}) \to \mathcal{N}$ is defined such that $\omega_{\tilde{S}} := w(o)$ for an arbitrarily chosen $o \in \circ\tilde{e}_{\tilde{S}}$—note that the function is well defined since all operators in the set $\tilde{e}_{\tilde{S}}$ have the same weight;

(f) the function $m_{\tilde{S}} : \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}) \times \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}) \to \mathcal{N}$ is defined such that $m_{\tilde{S}}(l,l') := \min\{\omega_{\tilde{S}}(l), \omega_{\tilde{S}}(l')\}$—that is, it gives the smaller of the weights of the smallest weight errors acting respectively on $\tilde{L}$ and $\tilde{L}'$;

(g) the function $\vec{M}_{\tilde{S}}$ is defined such that $\vec{M}_{\tilde{S}}(\vec{P})$ is the sequence of $|\vec{P}|$ integers such that $\vec{M}_{\tilde{S}}(\vec{P})_i := m_{\tilde{S}}(\vec{P}_i)$;

(h) the functions $p_1$ and $p_2$ are defined such that, given $(a,b) := x$, we have that $p_1(x) := a$ and $p_2(x) := b$;

(i) the function $\tilde{U} : \tilde{\mathcal{P}}(\tilde{\mathfrak{P}} \times \tilde{\mathfrak{P}}) \to \tilde{\mathcal{P}}(\tilde{\mathfrak{P}})$ is defined (for convenience) such that $\tilde{U}(\tilde{X}) := \bigcup_{x \in \tilde{X}} \{p_1(x), p_2(x)\}$—that is, it "unpacks" a set of pairs of operators into a set of operators; in an abuse of notation, we shall also allow $\tilde{U}$ to apply to sequences, so that $\tilde{U}(\vec{P}) := \tilde{U}(\{\vec{P}_i\}_i)$, and to individual pairs, so that if $X$ is a single pair then $\tilde{U}(X) := \tilde{U}(\{X\})$;

(j) finally, a *choice of qubits stabilized by* $\tilde{S}, \vec{P}$, is a sequence of pairs of operators from the Pauli group such that

(1) no operator in $\tilde{U}(\vec{P})$ appears in more than one pair in $\vec{P}$;

(2) $\tilde{U}(\vec{P}) \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$;

(3) every pair in $\vec{P}$ is a conjugal pair with respect to $\tilde{S} \cup \tilde{U}(\vec{P})$;

(4) $(\omega_{\tilde{S}} \circ p_1)(\vec{P}_i) = m_{\tilde{S}}(\vec{P}_i)$; and

(5) $\vec{M}_{\tilde{S}}(\vec{P})$ is an ordered sequence.

Given the notation above, we now precisely define what we mean by the "best choice" of logical qubits.

*Definition.* An *optimal choice of qubits stabilized by* $\tilde{S}$ is any choice of qubits, $\vec{P}$ stabilized by $\tilde{S}$, such that given any other choice of qubits, $\vec{P}'$, that is also stabilized by $\tilde{S}$ and which satisfies $(\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}')$, we have that $\vec{M}(\vec{P})_i \geqslant \vec{M}(\vec{P}')_i$ for all $1 \leqslant i \leqslant |\vec{P}| = |\vec{P}'|$.[8]

We now present an algorithm for computing the optimal choice of logical qubits from a set of input qubits. The key insight upon which the algorithm is built is that undetectable errors acting on the space of logical qubits can never be eliminated entirely, so there will always be *some* operator on which they act. Thus, the goal of the optimization procedure is not to eliminate errors, but rather to *contain* them, so that they act on as few operators as possible.

The optimization algorithm works by starting with an empty (and therefore automatically optimal) choice of qubits and a set of "unoptimized" qubits, and making progress by gradually moving qubits from the unoptimized set into the choice in such a way that preserves the optimality of the choice. The trick is that we want to delay as long as possible moving a qubit into the choice, until we have had every chance to improve it. Thus, we additionally keep track of a subset of pairs in the choice whose second members have yet to be used to contain an error, and then use them as much as possible to fix errors. That is, at every step in the algorithm, we scan for the minimal weight undetectable error acting on the set of operators consisting of both the second member of the pairs in this subset and all of the operators in the unoptimized set of qubits. If the minimal weight error acts on an operator in the first category, then we remove the pair from the subset and use this operator to fix this error wherever it occurs in both the second members of pairs in the subset and the unoptimized qubits. Otherwise, we pull out a qubit from the unoptimized set on which the error

[8] Note that since $\vec{P}$ and $\vec{P}'$ are sequences of conjugal pairs without duplicates they are therefore independent, and so if $(\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}')$ then we know automatically that $|\vec{P}| = |\vec{P}'|$.

acts, use the first member in the pair to fix the error in the qubits remaining in the unoptimized subset, add the pair to the subset of qubits whose second members have yet to be used to contain an error, and then add it to the end of the choice. At this point our choice turns out to still be optimal because if there had been a way to make the qubit we just added any better by recombining it with other qubits in the choice then we would have already done so by now.

This procedure is presented formally by means of the following inductive definition.

*Definition.* Let the function $\vec{\mathcal{O}}$ be a map from a tuple of the form tuple $(\tilde{S},\tilde{L})$ to a sequence of tuples each of the form $(\tilde{Q},\vec{P},\vec{s})$, where

(a) $\tilde{S}$ is a set of commuting Pauli operators;

(b) $\tilde{L}$ is a set of Pauli operators that are conjugal in relation to $\tilde{L}\cup\tilde{S}$;

(c) $\tilde{Q}$ is a set of pairs of Pauli operators;

(d) $\vec{P}$ is a sequence of pairs of Pauli operators; and

(e) $\vec{s}$ is a sequence of integers from $\{0,1\}$ with the same length as $\vec{P}$.

The sequence is defined inductively. For convenience, we let the first index of this sequence be zero, and define $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_0 = (\tilde{L}',\vec{\emptyset},\vec{\emptyset})$, where $\tilde{L}'$ is the set of pairs such that $\tilde{U}(\tilde{L}')$ and no operator appears in more than one pair in $\tilde{L}'$, and $\vec{\emptyset}$ is the empty sequence. Now assume that $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_i$ is defined and that $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_i = (\tilde{Q},\vec{P},\vec{s})$. If $\tilde{Q}$ is the empty set, then $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_i$ is the last element of the sequence so that $|\vec{\mathcal{O}}(\tilde{S},\tilde{L})| = i+1$. Otherwise, $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_{i+1} := (\tilde{Q}',\vec{P}',\vec{s}')$, where $\tilde{Q}'$, $\vec{P}'$, and $\vec{s}'$ are defined as follows.

Let $\tilde{R} := \{p_2(\vec{P}_i) : i \in \{1,\ldots,|\vec{P}|\},\vec{s}' = 1\}$ and $\tilde{X} := \tilde{U}(\tilde{Q})\cup\tilde{R}$. Note that, since $\tilde{Q}\neq\emptyset$, therefore $\tilde{X}\neq\emptyset$. Let $h$ be the minimal weight error[9] with respect to $\tilde{S}$ acting on any operator in $\tilde{X}$. There are two cases: either $h$ acts on some operator in $\tilde{R}$, or it does not and so must act on some operator in $\tilde{U}(\tilde{Q})$. The definitions of $\tilde{Q}'$, $\vec{P}'$, and $\vec{a}'$ depend on which of these cases holds.

### *1. Case (1): h acts on some operator in $\tilde{R}$*

Let $k$ be the smallest index such that $h$ acts on $p_2(\vec{P}_k)$, and $o := p_2(\vec{P}_k)$. Define

$$f(x) := \begin{cases} x, & [h,x]=0,\\ x\cdot o, & \{h,x\}=0,\end{cases}$$

and

$$g(a,b) := \begin{cases} I, & [h,a]=0,\\ b, & \{h,b\}=0.\end{cases}$$

Let $s' := p_1(\vec{P}_k)\cdot\alpha\cdot\beta$ where

$$\alpha := \Pi(\{g(p_2(\vec{P}_i),p_1(\vec{P}_i))\} : i\in\{k+1,\ldots,|\vec{P}|\},\vec{a}_i=\vec{0})$$

and

$$\beta := \Pi(\{g(a,b)\cdot g(b,a) : (a,b)\in\tilde{Q}\}).$$

Then we define

$$\tilde{Q}' := \{(f(a),f(b)) : (a,b)\in\tilde{Q}\},$$

$$\vec{P}_i' := \begin{cases} \vec{P}_i, & i<k,\\ (a',o), & i=k,\\ \vec{P}_i, & i>k,\quad \vec{a}_i=\vec{0},\\ \vec{P}_i, & i>k,\quad [h,p_2(\vec{P}_i)]=0,\\ (p_1(\vec{P}_i),p_2(\vec{P}_i)\cdot o) & \text{otherwise},\end{cases}$$

$$\vec{s}_i' := \begin{cases} \vec{s}_i, & i\neq k,\\ 0, & i=k,\end{cases}$$

### *2. Case (2): h does not act on some operator in $\tilde{R}$*

Let $q\in\tilde{Q}$ be a pair such that $h$ acts on one of its members, and without loss of generality assume that $h$ acts on the first member since otherwise we can swap the members of the pair. Let $o := p_1(q)$. Define

$$f(x) := \begin{cases} x, & [h,x]=0,\\ x\cdot o, & \{h,x\}=0,\end{cases}$$

and

$$g(a,b) := \begin{cases} I, & [h,a]=0,\\ b, & \{h,b\}=0.\end{cases}$$

Let

$$b'' := p_2(q)\cdot\Pi(\{g(a,b)\cdot g(b,a) : (a,b)\in\tilde{Q}\backslash\{q\}\})$$

and

$$b' := \begin{cases} b'', & [h,b'']=0,\\ b''\cdot o, & \{h,b''\}=0.\end{cases}$$

Then we define

$$\tilde{Q}' := \{(f(a),f(b)) : (a,b)\in\tilde{Q}\backslash\{q\}\},$$

$$\vec{P}_i' := \begin{cases} \vec{P}_i, & i\leqslant|\vec{P}|,\\ (o,b'), & i=|\vec{P}|+1,\end{cases}$$

$$\vec{s}_i' := \begin{cases} \vec{s}_i, & i\leqslant|\vec{s}|,\\ 1, & i=|\vec{s}|+1.\end{cases}$$

Table III contains a listing of pseudocode that uses the above algorithm to compute the optimal choice of qubits. For the sake of completeness, it includes additional steps that pertain to the details of how the minimal weight operator is computed, which will be discussed in more detail in Sec. III B 5.

In addition to proving that the above algorithm successfully constructs an optimal choice of qubits, we shall also provide a bound on its running time. In order to do this, we first need to precisely define what we mean by the running time for the purposes of this section.

*Definition.* We say that a computation can be performed *in time* $x$ if the computation requires taking $x$ products of Pauli operators.

Of course, the number of products of Pauli operators is not the only metric that could serve as the gauge for the running

[9] An observant reader may have noticed that we do not specify exactly how one goes about computing $h$. This was an intentional omission since the details are quite technical and fortunately they are irrelevant for proving that this algorithm works correctly as long as we can assume that $h$ *can* be computed. Thus, the discussion of how to compute $h$ will be deferred until Sec. III B 5 when we analyze bounds on the running time of the algorithm.

time, but it suffices for our purposes. We now present the main result of this section.

*Theorem 2.* Suppose we are given

(a) a set of commuting Pauli operators, $\tilde{S}$, acting on $N$ physical qubits;

(b) a set of pairs, $\tilde{L} \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$, conjugal with respect to $\tilde{U}(\tilde{Q}) \cup \tilde{S}$;

(c) and a set of Pauli operators $\tilde{C}$ such that $\tilde{\mathcal{G}}(\tilde{C}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$;

then $\vec{\mathcal{O}}(\tilde{S},\tilde{L})$ is a sequence of finite length, and if $(\tilde{Q},\vec{P},\vec{s})$ is the last element in the sequence then $\vec{P}$ is an optimal choice of qubits such that $\tilde{\mathcal{G}}(\vec{P}) = \tilde{\mathcal{G}}(\tilde{L})$, and furthermore it can be computed in a time that is in the set $O(|\tilde{C}|^2 + |\vec{L}|^2 d3^d \binom{N}{d})$,[10] where $d := \vec{M}(\vec{P})_{|\vec{P}|}$.[11]

The proof of this theorem is rather technical and will be split into several sections. First we shall prove the existence of a condition that suffices to prove that a choice of logical qubits is optimal. Second we shall prove that the algorithm above constructs a choice satisfying this condition. Third we shall prove that the running time of the algorithm has the claimed bound. Finally we shall tie these results together to prove the theorem above.

### 3. *Optimality condition*

How do we know that a choice of qubits is optimal? Intuitively, it should be sufficient to prove that a choice of qubits is optimal if we can show that there is no way that we can recombine qubits in the choice to form one or more qubits that are more robust than their component factors—that is, there is no way that any qubit can be "improved" by its involvement in such a product. This condition is stated formally in the following definition of *unimprovable sets*:

*Definition.* An *unimprovable set with respect to* $\tilde{S}$ is a set of Pauli operators $\tilde{O}$ such that for any subset $\tilde{X} \subseteq \tilde{O}$ we have that $(\omega_{\tilde{S}} \circ \prod)(\tilde{X}) = \min_{x \in \tilde{X}} \omega_{\tilde{S}}(x)$. We say that an unimprovable set $\tilde{O}$ *extends to* $\vec{Q}$ if for all subsets $\tilde{X} \subseteq \tilde{O} \cup \tilde{Q}$ such that $\tilde{X} \cap \tilde{O} \neq \emptyset$ we have that $(\omega_{\tilde{S}} \circ \prod)(\tilde{X}) \leqslant \min_{x \in \tilde{X} \cap \tilde{O}} \omega_{\tilde{S}}(x)$.

The following theorem is the main result of this section and proves that this condition is indeed sufficient to show that a choice of logical qubits is optimal.

*Theorem 3.* If $\vec{P}$ is a choice of $N$ logical qubits stabilized by $\tilde{S}$ such that $\{p_1(\vec{P}_i)\}_i$ is an unimprovable set with respect to $\tilde{S}$ that extends to $\tilde{U}(\vec{P})$, then $\vec{P}$ is an optimal choice of qubits.

*Remark.* The intuition behind the proof of this theorem is that because the set of first members of pairs is unimprovable and extends to the set of all members of pairs, we know that no qubit can be "improved" by recombining it with one or more other qubits. Thus, the only way one could construct a better choice would by forming $n + k$ independent qubits from products of $n$ qubits (where $k > 0$), which intuitively should be impossible. Thus, we conclude that it is not possible for there to be a choice of qubits generated by the same qubits in this choice that is "better" than this choice.

To assist us in proving this theorem, we shall first prove a number of useful lemmas and propositions. We start with a simple lemma that proves that taking a product of operators results in an operator that is no "worse" (with respect to its robustness to errors) than the worst operator in the product.

*Lemma 2.* For any set of operators $\tilde{O}$, we have that $(\omega_{\tilde{S}} \circ \prod)(\tilde{O}) \geqslant \min_{o \in \tilde{O}} \omega_{\tilde{S}}(o)$.

*Proof of Lemma 2.* Any undetectable error with respect to $\tilde{S}$ acting on $(\omega_{\tilde{S}} \circ \prod)(\tilde{O})$ must also act on at least one of the operators in $\tilde{O}$ since otherwise it cannot anticommute with the product. ■

*Remark.* In general, taking products of operators might result in an operator that is better than the worst operator in the product because errors will cancel each other out—i.e., if two operators in the product anticommute with an error then their product commutes with the error. Thus, it is useful to state a condition under which we can be certain that this will not happen, so that the product is exactly as bad as the worst operator, which we do in the following lemma.

*Lemma 3.* Suppose we are given two operators $a,b \in \tilde{\mathfrak{P}}$ such that $\omega_{\tilde{S}}(a) < \omega_{\tilde{S}}(b)$; then $\omega_{\tilde{S}}(a \cdot b) = \omega_{\tilde{S}}(a)$.

*Proof of Lemma 3.* Since $\omega_{\tilde{S}}(a) < \omega_{\tilde{S}}(b)$, there must be an undetectable error with respect to $\tilde{S}$ that acts on $a$ but not on $b$; thus, it must anticommute with and hence act on the product $a \cdot b$, so that $\omega_{\tilde{S}}(a \cdot b) \leqslant \omega_{\tilde{S}}(a)$. Since $\omega_{\tilde{S}}(a \cdot b) \geqslant \omega_{\tilde{S}}(a)$ by Lemma 3, we conclude that $\omega_{\tilde{S}}(a \cdot b) = \omega_{\tilde{S}}(a)$. ■

*Remark.* Intuitively we should expect that it is not possible to take $n$ qubits and recombine them to form $n + k$ independent qubits where $k > 0$. To state this intuition in other terms, suppose we are given a set of conjugal pairs $\tilde{C}$ that are generated from some other set of conjugal pairs $\tilde{D}$. We know that every pair in $\tilde{C}$ must have a member that includes a factor that is a first member of a pair in $\tilde{D}$ (since otherwise the members of the pair cannot anticommute), so let $\tilde{A}$ be the set of first members of pairs in $\tilde{D}$. Our intuition then tells us that $|\tilde{C}| \leqslant |\tilde{A}| = |\tilde{D}|$. The following proposition states this fact formally:

*Proposition 4.* Suppose we are given

(1) sets of independent Pauli operators $\tilde{Q}$ and $\tilde{S}$;

(2) a nonempty set of conjugal pairs, $\tilde{C}$, with respect to $\tilde{U}(\tilde{C}) \cup \tilde{S}$, such that $U(\tilde{C}) \subseteq \tilde{\mathcal{G}}(\tilde{Q})$; and

(3) a set $\tilde{A}$ of independent Pauli operators with the property that for any conjugal pair $X := (a,b)$ such that $\{a,b\} \in \tilde{\mathcal{G}}(\tilde{C})$, we must have that $\tilde{G}_{\tilde{Q}}(X) \cap \tilde{A} \neq \emptyset$.

Then $|\tilde{C}| \leqslant |\tilde{A}|$.

*Remark.* The basic idea behind the proof of this proposition is that an analog of Gaussian elimination can be used on the conjugal pairs to eliminate the presence of members of $\tilde{A}$ from them; when we are done with this process, we can see that unless $|\tilde{C}| \leqslant |\tilde{A}|$ we will have eliminated *all* members of $\tilde{A}$ from some of the qubits, which contradicts the assumptions of this proposition.

The formal proof is somewhat technical and so we first introduce several lemmas. First we prove a small helper lemma that shows that it is possible to take a conjugal pair in which a given generator appears and rearrange it so that the generator appears only in the first member of the pair.

[10] A function $f$ is said to be in the set $O(g)$ if $f$ is asymptotically bounded by some fixed constant times $g$; formally $f \in O(g)$ if and only if there exist constants $c$ and $x_0$ such that $f(x) < cg(x)$ for all $x > x_0$.

[11] Recall that $\vec{M}(\vec{P})_{|\vec{P}|}$ is the distance of the best qubit in the (optimized) code.

*Lemma 4.* Let $A := (a,b)$ with $\{a,b\} \subseteq \tilde{\mathcal{G}}(\tilde{Q})$ be a conjugal pair with respect to some set $\tilde{S}$, and $o$ be some Pauli operator such that $o \in \tilde{G}_{\tilde{Q}}(A)$. Then there exists a pair $B := (c,d)$ such that

(1) $\{c,d\} \subseteq \tilde{\mathcal{G}}(\tilde{Q})$;

(2) $o \in \tilde{G}_{\tilde{Q}}(c)$;

(3) $o \notin \tilde{G}_{\tilde{Q}}(d)$;

(4) $(c,d)$ is a conjugal pair with respect to $(\tilde{S}\backslash\{a,b\}) \cup \{c,d\}$; and

(5) $(\tilde{\mathcal{G}} \circ \tilde{U})(B) = (\tilde{\mathcal{G}} \circ \tilde{U})(A)$.

*Proof.* Let

$$(c,d) := \begin{cases} (a,b), & o \in \tilde{G}_{\tilde{Q}}(a), \quad o \notin \tilde{G}_{\tilde{Q}}(b), \\ (b,a), & o \notin \tilde{G}_{\tilde{Q}}(a), \quad o \in \tilde{G}_{\tilde{Q}}(b), \\ (a,b \cdot a), & o \in \tilde{G}_{\tilde{Q}}(a), \quad o \in \tilde{G}_{\tilde{Q}}(b). \end{cases}$$

Note that in any of the above cases, properties (1)–(3) are satisfied by construction, property (4) is satisfied because $c$ and $d$ are products of $a$ and $b$ which commute with every element in $\tilde{S}\backslash\{a,b\}$ and $\{c,d\} = 0$, and finally property (5) is satisfied because $\{a,b\} \subseteq \tilde{\mathcal{G}}\,(\{c,d\})$ and $\{c,d\} \subseteq \tilde{\mathcal{G}}\,(\{a,b\})$. ■

*Remark.* This next lemma contains the heart of this Proposition 4 by introducing an analog to a directed Gaussian elimination procedure. Specifically, it shows that if we have a generator $a \in \tilde{A}$ that appears in one or more conjugal pairs, then we can take products of the conjugal pairs to eliminate it from appearing anywhere except in the first member of a single pair.

*Lemma 5.* In the context of Proposition 4, suppose we are given an element $a \in \tilde{A}$ with the property that there exists a pair $Y'' \in \tilde{C}$ such that $a \in \tilde{G}(Y'')$. Then there exists a conjugal pair $Y$ and set of conjugal pairs $\tilde{D}$, all with respect to $\tilde{U}(\{Y\}) \cup \tilde{D} \cup \tilde{S}$, such that

(1)$|\tilde{D}| = |\tilde{C}| - 1$;

(2) $(\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\} \cup \tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$;

(3) $a \in (\tilde{G}_{\tilde{Q}} \circ p_1)(Y)$ but $a \notin (\tilde{G}_{\tilde{Q}} \circ p_2)(Y)$;

(4) $a \notin \bigcup_{D \in \tilde{D}} \tilde{G}_{\tilde{Q}}(D)$; and

(5) for every conjugal pair $O \in \tilde{D}$, we have that $\tilde{G}_{\tilde{Q}}(O) \cap \tilde{A}\backslash\{a\} \neq \emptyset$.

*Proof.* The proof is by induction on the size of $\tilde{C}$. If $\tilde{C} = \{Y''\}$, then apply Lemma 4, letting $o := a$, $A := Y''$, and $Y : = B$. We see that we have a pair $Y$ which is conjugal with respect to $\{Y\} \cup \tilde{S}$ and also such that $(\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{Y''\})$. Let $\tilde{D} := \emptyset$, and we see that the remaining properties hold trivially, so we are done.

Now let us assume that this lemma has been proven for the case where $|\tilde{C}| = n - 1$, and we are given a set $\tilde{C}$ with $n$ elements. Take any $X'' \in \tilde{C}\backslash\{Y''\}$, and apply the lemma to $\tilde{C}\backslash\{X''\}$, $\tilde{A}$, $a$, and $Y''$ to obtain the objects $Y'$ and $\tilde{D}'$ described in this lemma without the primes. If $a \notin \tilde{G}(X'')$, then by the assumptions of the lemma we know that $\tilde{G}(X'') \cap \tilde{A}\backslash\{a\} \neq \emptyset$, so let $Y := Y'$ and $\tilde{D} := \tilde{D}' \cup \{X''\}$, and we are done.

Otherwise, apply Lemma 4, setting $A := X''$, $o := a$, and $X' := B$, and let $X := (p_1(X') \cdot p_1(Y'), p_2(X'))$ and $Y := (p_1(Y'), p_2(Y') \cdot p_2(X'))$. Note that $X'$ and $Y'$ are conjugal pairs with respect to $\tilde{U}(\{X',Y'\} \cup \tilde{D}) \cup \tilde{S}$ and $\{X',Y'\} \cap (\tilde{D} \cup \tilde{S}) = \emptyset$, and so by Lemma 1 we conclude that $X$ and $Y$ are conjugal pairs with respect to $\tilde{U}(\{X,Y\} \cup \tilde{D}) \cup \tilde{S}$, and also that $(\tilde{\mathcal{G}} \circ \tilde{U})(\{X,Y\}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{X',Y'\})$; since $X'$ was obtained from applying Lemma 4 to $X''$ and $a$, we furthermore conclude that $(\tilde{\mathcal{G}} \circ \tilde{U})(\{X,Y\}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{X'',Y'\})$. Since $X'$ was obtained as a result of Lemma 4, we know that $a \in (\tilde{G} \circ p_1)(X')$ but $a \notin (\tilde{G}_{\tilde{Q}} \circ p_2)(X')$, and we also know from the earlier recursive application of this lemma that $a \in (\tilde{G} \circ p_1)(Y')$ but $a \notin (\tilde{G}_{\tilde{Q}} \circ p_2)(Y')$. Thus, we observe that by construction, $a \in (\tilde{G}_{\tilde{Q}} \circ p_1)(Y)$, and $a \notin ((\tilde{G}_{\tilde{Q}} \circ p_2)(Y) \cup \tilde{G}_{\tilde{Q}}(X))$.

Let $\tilde{D} := \{X\} \cup \tilde{D}'$, and observe that $|\tilde{D}| = |\tilde{D}'| + 1 = |\tilde{C}\backslash\{X''\}| - 1 + 1 = |\tilde{C}| - 1$, and also that $(\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\} \cup \tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{X,Y\} \cup \tilde{D}') = (\tilde{\mathcal{G}} \circ \tilde{U})(\{X''\} \cup (\{Y'\} \cup \tilde{D}')) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{X''\} \cup (\tilde{C}'\backslash\{X''\})) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$. Furthermore, by the earlier recursive application of this lemma we know that $a \notin \tilde{G}_{\tilde{Q}}(O)$ for every $O \in \tilde{D}'$, so since we have also established that $a \notin \tilde{G}_{\tilde{Q}}(X)$, we conclude that $a \notin \tilde{G}_{\tilde{Q}}(O)$ for every $O \in \tilde{D}$; since we also know that every such $O$ must also satisfy $\tilde{G}_{\tilde{Q}}(O) \cap \tilde{A} \neq \emptyset$, we conclude that every such $O$ satisfies $\tilde{G}_{\tilde{Q}}(O) \cap A\backslash\{a\} \neq \emptyset$. ■

*Remark.* This next lemma provides the small but important result that we can always find a generator $a$ that appears somewhere in the conjugal pairs; this has the consequence that we can now perform *undirected* Gaussian elimination (in contrast to the *directed* Gaussian elimination procedure described in the previous lemma) by picking an arbitrary generator to eliminate rather than specifying a particular generator up front.

*Lemma 6.* In the context of Proposition 4, there exists a Pauli operator $a$ satisfying the assumption of Lemma 5.

*Proof.* Take any pair $Y' \in \tilde{C}$. By the assumptions of Proposition 4, we know that $\tilde{G}(Y') \cap \tilde{A} \neq \emptyset$, which implies that there exists an element $a \in A$ such that either $a \in (\tilde{G}_{\tilde{Q}} \circ p_1)(Y')$ or $a \in (\tilde{G}_{\tilde{Q}} \circ p_2)(Y')$. The existence of $Y$ and $\tilde{D}$ then follows immediately from the application of Lemma 5. ■

*Remark.* This final lemma (inside the proof of Proposition 4) shows using Gaussian elimination that there must be a number of generators from $\tilde{A}$ present in the pairs in $\tilde{C}$ that is equal to the size of $\tilde{C}$, since otherwise we could recombine the pairs in $\tilde{C}$ to obtain a pair that includes no generator from $\tilde{A}$, contradicting the assumptions of Proposition 4.

*Lemma 7.* In the context of Proposition 4, there exists a set of conjugal pairs $\tilde{X}$ with respect to $\tilde{X} \cup \tilde{S}$, and a subset of operators $\tilde{O} \subseteq \tilde{A}$ such that

(1) $|\tilde{X}| = |\tilde{O}| = |\tilde{C}|$;

(2) $(\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{X}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$; and

(3) for every $o \in \tilde{O}$, there is a conjugal pair $Y \in \tilde{X}$ such that $o \in \tilde{G}_{\tilde{Q}}(Y)$.

*Proof.* The proof is by induction. If $\tilde{C}$ is empty, then the empty sets trivially satisfy this lemma.

Now suppose that we have proven this lemma for $|\tilde{C}| = N - 1$, and assume we have been given sets $\tilde{C}$ and $\tilde{A}$ such that $|\tilde{C}| = N$. Applying Lemma 6 to $\tilde{C}$ and $\tilde{A}$, we obtain the conjugal pair $Y$, the set of conjugal pairs $\tilde{D}$, and the element $a$ described in the conclusions of that lemma. Applying this lemma recursively to the respective sets $\tilde{D}$ and $\tilde{A}\backslash\{a\}$, we obtain the sets $\tilde{X}'$ and $\tilde{O}'$ described (without the primes) in this lemma; let $\tilde{X} := \tilde{X}' \cup \{Y\}$ and $\tilde{O} := \tilde{O}' \cup \{a\}$. Note that $\tilde{X}$ is a set of conjugal pairs with respect to $\tilde{X} \cup \tilde{S}$ since $\tilde{X}'$ is a set of conjugal pairs with respect to $\tilde{X}' \cup \tilde{S}$, and we know that the operators in $Y$ commute with every operator in every pair in $\tilde{X}'$ since they commute with every operator in $(\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{X}')$.

First, observe that $|\tilde{O}| = |\tilde{O}'| + 1$ since $a \notin \tilde{O}'$. Furthermore, $a \notin \bigcup_{x \in \tilde{U}(\tilde{X}')} \tilde{G}(x)$ since $a \notin \bigcup_{x \in \tilde{U}(\tilde{D})} \tilde{G}_{\tilde{Q}}(x)$ by Lemma 6 and $(\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{X}')$ by recursive application of this corollary. Thus, $|\tilde{X}| = |\tilde{X}'| + 1$ since $Y \notin X'$ as $a \in \tilde{G}(Y)$, and $|\tilde{X}| = |\tilde{O}| = |\tilde{D}| + 1 = |\tilde{C}| - 1 + 1 = |\tilde{C}|$.

Second, observe that since $(\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{X}') = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$ by recursive application of this lemma and $(\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\} \cup \tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$ by Lemma 6, we conclude that $(\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{X}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\} \cup \tilde{X}') = (\tilde{\mathcal{G}} \circ \tilde{U})(\{Y\} \cup \tilde{D}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{C})$.

Finally, observe that for every $o \in \tilde{O}$ we have that either $o = a$, in which case $Y \in \tilde{X}$ and $o \in \tilde{G}_{\tilde{Q}}(Y)$, or $o \in \tilde{A}\backslash\{a\}$, in which case by recursive application of this lemma we know that there is an operator $Z \in \tilde{X}' \subseteq \tilde{X}$ such that $o \in \tilde{G}_{\tilde{Q}}(Z)$. ■

*Remark.* With the preceding lemmas having performed the heavy lifting, the proof of Proposition 4 is quite simple.

*Proof of Proposition 4.* The proof is by contradiction. By Lemma 7, there would have to exist a subset $\tilde{O} \subseteq \tilde{A}$ such that $|\tilde{C}| = |\tilde{O}| > |\tilde{A}|$, which is impossible. ■

*Remark.* With the preceding lemmas and propositions, we now have all of the tools that we need to prove Theorem 3. Again, the idea behind this proof is that because the first members of pairs in the choice are contained in an unimprovable set, one cannot take products of the qubits in the choice in order to improve them; thus, the only way one could construct a better choice would be by forming $n + k$ independent qubits from products of $n$ qubits (where $k > 0$), which is disallowed by the result of Proposition 4. Hence, there can be no better choice.

*Proof of Theorem 3.* The proof is by contradiction. Let $\vec{P}'$ be some choice of qubits stabilized by $\tilde{S}$ such that $(\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}) = (\tilde{\mathcal{G}} \circ \tilde{U})(\vec{P}')$ (which automatically implies that $|\vec{P}| = |\vec{P}'|$) and there exists some integer $k$ such that $M_{\tilde{S}}(\vec{P}')_k > M_{\tilde{S}}(\vec{P})_k$; in particular, let $k$ be the smallest such integer, and let $\tilde{C} := \{\vec{P}_i' : i \geqslant k\}$. Let $l$ be the smallest integer such that $\vec{M}_{\tilde{S}}(\vec{P})_l \geqslant \vec{M}_{\tilde{S}}(\vec{P}')_k$ or $|\vec{P}| + 1$ if there is no such integer, and let $\tilde{A} := \{p_1(\vec{P}_i) : i \geqslant l\}$; note that since $\vec{M}_{\tilde{S}}(\vec{P}')_k > \vec{M}_{\tilde{S}}(\vec{P})_k$ we must have $l > k$, and hence $|\tilde{C}| > |\tilde{A}|$.

Take any conjugal pair $O := (a,b)$ such that $\{a,b\} \in \tilde{\mathcal{G}}(\tilde{C})$. Since $a$ and $b$ anticommute, it must be the case that $\{p_1(\vec{P}_i)\}_i \cap \tilde{G}_{\tilde{U}(\vec{P})}(O) \neq \emptyset$, because if every operator in $\tilde{G}_{\tilde{U}(\vec{P})}(O)$ were the second member of a pair in $\vec{P}$ then $a$ and $b$ would commute. Let $c$ be a choice of $a$ or $b$ such that $\{p_1(\vec{P}_i)\}_i \cap \tilde{G}_{\tilde{U}(\vec{P})}(c) \neq \emptyset$. By Lemma 2 we know that $\vec{M}_{\tilde{S}}(\vec{P}')_k \leqslant \omega_{\tilde{S}}(c)$ since $c \in \tilde{\mathcal{G}}(\tilde{C})$. By the assumption of this theorem that $\{p_1(\vec{P}_i)\}_i$ is an unimprovable set that extends to $\tilde{U}(\vec{P})$, we know that $\vec{M}_{\tilde{S}}(\vec{P}')_k \leqslant \omega_{\tilde{S}}(c) \leqslant \min\{\omega_{\tilde{S}}(x) : x \in \{p_1(\vec{P}_i)\}_i \cap \tilde{G}_{\tilde{U}(\vec{P})}(c)\}$. From these bounds we conclude that $\{p_1(\vec{P}_i)\}_i \cap \tilde{G}_{\tilde{U}(\vec{P})}(c) \subseteq \tilde{A}$, and since $\{p_1(\vec{P}_i)\}_i \cap \tilde{G}_{\tilde{U}(\vec{P})}(c) \neq \emptyset$ we see therefore that $\tilde{G}_{\tilde{U}(\vec{P})}(c) \cap \tilde{A} \neq \emptyset$ and so $\tilde{G}_{\tilde{U}(\vec{P})}(O) \cap \tilde{A} \neq \emptyset$.

We have now demonstrated that for every pair $O: = (a,b)$ such that $\{a,b\} \in \tilde{\tilde{\mathcal{G}}}(\tilde{C})$, we must have $\tilde{G}(O) \cap \tilde{A} \neq \emptyset$. Observe that this means that sets $\tilde{C}$ and $\tilde{A}$ match the descriptions in Proposition 4 (letting set $\tilde{Q} := \{\tilde{P}_i\}_i$), and thus we see that it is impossible for $|\tilde{C}| > |\tilde{A}|$, and so we have a contradiction. We thus conclude that no such choice $\vec{P}'$ can exist. ■

### *4. Correctness of the algorithm*

We now prove that this algorithm is correct—that is, that it terminates and outputs an optimal choice of logical qubits. We do so by proving the following theorem, which is the main result of this section.

*Theorem 4.* Given a set of commuting Pauli operators $\tilde{S}$ and a set of pairs $\tilde{L} \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ conjugal in relation to $\tilde{U}(\tilde{L}) \cup \tilde{S}$, the sequence $\vec{\mathcal{O}}(\tilde{S},\tilde{L})$ is finite and if $(\tilde{Q},\vec{P},\vec{s})$ is the last element then $\tilde{Q} = \emptyset$ and $\vec{P}$ is an optimal choice of logical qubits stabilized by $\tilde{S}$ such that $\tilde{\mathcal{G}}(\vec{P}) = \tilde{\mathcal{G}}(\tilde{L})$.

*Remark.* Before proving this theorem, we shall first prove several related lemmas and propositions.

Our ultimate goal is to expand the unimprovable set so that it includes at least the first member of every conjugal pair in the set of logical qubit operators, since this means that we have satisfied the optimality condition. Thus, we want to be able to add operators to this set while preserving the property of being an unimprovable set.

The following lemma shows that if we have an operator $o$ in a set $\tilde{X}$ to which some unimprovable set extends, then if the smallest weight undetectable error acting on $o$ acts on no other operator in $\tilde{X}$ we may move $o$ to the unimprovable set to obtain a new unimprovable set that extends to $\tilde{X}/\{o\}$. The intuition here is that because said error acts only on $o$, it cannot be canceled by multiplying $o$ by other operators, and so it is an "unimprovable" operator that can be included in our unimprovable set.

*Lemma 8.* If $\tilde{O}$ is an unimprovable set with respect to $\tilde{S}$ that extends to $\tilde{X} := \{o\} \cup \tilde{X}'$, and there exists an undetectable error $h$ of weight $\omega_{\tilde{S}}(o)$ that acts on $o$ but not on any operator in $\tilde{X}'$, then $\tilde{O}' := \tilde{O} \cup \{o\}$ is an unimprovable set with respect to $\tilde{S}$ that extends to $\tilde{X}'$.

*Proof.* Take any subset $\tilde{R} \subseteq \tilde{O}' \cup \tilde{X}'$ such that $\tilde{R} \cap \tilde{O}' \neq \emptyset$. We need to show that $\omega_{\tilde{S}}\,(r) \leqslant \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a)$ where $r := \Pi(\tilde{R})$.

First consider the case where $o \notin \tilde{R}$; in this case we have that $\tilde{R} \subseteq \tilde{O} \cup \tilde{X}$ such that $\tilde{R} \cap \tilde{O} = \tilde{R} \cap \tilde{O}' \neq \emptyset$, and so since $\tilde{O}$ extends to $\tilde{X}$ we conclude that $\omega_{\tilde{S}}\,(r) \leqslant \min_{a \in \tilde{R} \cap \tilde{O}} \omega_{\tilde{S}}(a) = \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a)$.

Now consider the case where $\tilde{R} \cap \tilde{O}' = \{o\}$. In this case, by the assumptions of this lemma, we know $h$ acts on $o$ but not on any operator in $\tilde{X}'$ which implies that $h$ acts on $r$ and so $\omega_{\tilde{S}}(r) \leqslant w(h) = \omega_{\tilde{S}}(o) = \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a)$.

Finally we consider the remaining case where $\{o\} \subset \tilde{R} \cap \tilde{O}'$. Let $\tilde{Z} := \tilde{R}\backslash\{o\} \neq \emptyset$. Since $\tilde{O}$ extends to $\tilde{X}$ and $\tilde{R} \cap \tilde{O} = \tilde{Z} \cap \tilde{O} \neq \emptyset$, we know that $\omega_{\tilde{S}}(x) \leqslant \min_{a \in \tilde{Z} \cap \tilde{O}} \omega_{\tilde{S}}(y) =: d$. If $d \leqslant \omega_{\tilde{S}}(o)$, then $\omega_{\tilde{S}}(r) \leqslant d = \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a)$, and we are done. Otherwise, since $d > \omega_{\tilde{S}}(o)$ we know that $h$ acts on $r$ since there can be no other operator in $\tilde{Z}$ that anticommutes with $h$, and so $\omega_{\tilde{S}}(r) \leqslant w(h) \leqslant \omega_{\tilde{S}}(o) = \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}$ $(a)$. ■

*Remark.* In Case (1) of the algorithm we take an element that is a member of an unimprovable set and replace it with the product of this element times some elements in the set to which the unimprovable set extends. We want to show that this preserves the unimprovability of the set, and this is done in the following lemma.

*Lemma 9.* Suppose we are given an unimprovable set $\tilde{O}$ with respect to $\tilde{S}$ that extends to $\tilde{X}$. Let $o$ be any element in $\tilde{O}$, and $\tilde{Z} \subseteq \tilde{O} \cup \tilde{X}$ such that $o \in \tilde{Z}$. Let $o' := \Pi(\tilde{Z})$ and $\tilde{O}' := (\tilde{O} \backslash \{o\}) \cup \{o'\}$. If $\omega_{\tilde{S}}(o') = \omega_{\tilde{S}}(o)$, then $\tilde{O}'$ is also an unimprovable set with respect to $\tilde{S}$ that extends to $\tilde{X}$.

*Proof.* Take any subset of elements $\tilde{R} \subseteq \tilde{O}' \cup \tilde{X}$ such that $\tilde{R} \cap \tilde{O}' \neq \emptyset$, and let $x := \Pi(\tilde{R})$. We need to show that $\omega_{\tilde{S}}(x) \leqslant \min_{a \in \tilde{R} \cap \tilde{O}} \omega_{\tilde{S}}(a)$. If $o' \notin \tilde{R}$, then this follows immediately from the fact that $\tilde{R} \subseteq \tilde{O} \cup \tilde{X}$ and $\tilde{R} \cap \tilde{O} \neq \emptyset$ and $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$, so assume that $o' \in \tilde{R}$. Since $o' = \Pi(\tilde{Z})$ and $\tilde{Z} \subseteq \tilde{O} \cup \tilde{X}$, we conclude that the set $\tilde{T} \subseteq \tilde{O} \cup \tilde{X}$ which is the symmetric difference of $\tilde{Z}$ and $\tilde{R}$ satisfies the property that $x = \Pi(\tilde{T})$. Note that $o \in \tilde{T}$ since $o \in \tilde{Z}$ and $o \notin \tilde{O}'$ and so $o \notin \tilde{R}$. Thus, $\tilde{T} \cap \tilde{O} \neq \emptyset$, and so $\omega_{\tilde{S}}(x) \leqslant \min_{a \in \tilde{T} \cap \tilde{O}} \omega_{\tilde{S}}(a)$ since $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$. Thus, for us to show that $\omega_{\tilde{S}}(x) \leqslant \min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a)$, it suffices for us to show that $\min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a) = \min_{a \in \tilde{T} \cap \tilde{O}} \omega_{\tilde{S}}(a)$.

First note that since $\omega_{\tilde{S}}(o') = \omega_{\tilde{S}}(o)$, there is no element $z \in \tilde{Z} \cap \tilde{O}$ such that $\omega_{\tilde{S}}(z) < \omega_{\tilde{S}}(o)$. Thus, any operator $t \in \tilde{T} \cap \tilde{O}$ such that $\omega_{\tilde{S}}(t) < \omega_{\tilde{S}}(o)$ must also appear in $\tilde{R} \cap \tilde{O}'$, and vice versa; put another way, any operator that is less robust to errors than $o$ must be present in both $\tilde{T} \cap \tilde{O}$ and $\tilde{R} \cap \tilde{O}'$ together or in neither. Thus, if at least one such operator exists, then we conclude that $\min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a) = \min_{a \in \tilde{T} \cap \tilde{O}} \omega_{\tilde{S}}(a)$ since in this case any minimizer of $\omega_{\tilde{S}}$ must be shared between the two sets. If no such operator exists, then since $o \in \tilde{T} \cap \tilde{O}$ and $o' \in \tilde{R} \cap \tilde{O}'$ and there is no other operator present in either set with a smaller minimum weight undetectable error, we conclude that $o$ is the minimizer of $\omega_{\tilde{S}}$ over $\tilde{T} \cap \tilde{O}$ and $o'$ is the minimizer over $\tilde{R} \cap \tilde{O}'$, and since $\omega_{\tilde{S}}(o') = \omega_{\tilde{S}}(o)$ we have that $\min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a) = \omega_{\tilde{S}}(o) = \min_{a \in \tilde{T} \cap \tilde{O}} \omega_{\tilde{S}}(a)$.

Thus we have shown that $\min_{a \in \tilde{R} \cap \tilde{O}'} \omega_{\tilde{S}}(a) = \min_{a \in \tilde{T} \cap \tilde{O}} \omega_{\tilde{S}}(a)$, and since our choice of $\tilde{R}$ was arbitrary we conclude that $\tilde{O}'$ is an unimprovable set that extends to $\tilde{X}$. ■

*Remark.* In both cases of the algorithm we replace a set to which an unimprovable set extends with a new set that is a product of elements in the old set. We want to show that the new set is also an extension of the unimprovable set, and this is proved by the following lemma.

*Lemma.* If $\tilde{O}$ is an unimprovable set with respect to $\tilde{S}$ that extends to $\tilde{X}$, and $\tilde{X}'$ is a set such that $\tilde{X}' \subseteq \tilde{\mathcal{G}}(\tilde{X})$, then $\tilde{O}$ also extends to $\tilde{X}'$. ■

*Proof.* Take any subset $\tilde{R}' \subseteq \tilde{O} \cup \tilde{X}'$ such that $\tilde{A} := \tilde{R}' \cap \tilde{O} \neq \emptyset$. We need to show that $\omega_{\tilde{S}}(x) \leqslant \min_{y \in \tilde{A}} \omega_{\tilde{S}}(y)$, where $x := \Pi(\tilde{R}')$. Note that since $\tilde{X}' \subseteq \tilde{\mathcal{G}}(\tilde{X})$, there exists a set $\tilde{B} \subseteq \tilde{X}$ such that $x = \Pi(\tilde{A} \cup \tilde{B})$, and so since $\tilde{O}$ extends to $\tilde{X}$ we conclude that $\omega_{\tilde{S}}(x) \leqslant \min_{y \in \tilde{A}} \omega_{\tilde{S}}(y)$. Since our choice of $\tilde{R}'$ was arbitrary, we conclude that that $\tilde{O}$ extends to $\tilde{X}'$.

*Remark.* Most of the heavy lifting in this section is performed in the following proposition, which uses induction to prove a number of properties about the output of the algorithm at every step.

*Proposition 5.* Given a set of Pauli operators $\tilde{S}$ and a set of pairs $\tilde{L} \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ conjugal in relation to $\tilde{U}(\tilde{L}) \cup \tilde{S}$, for every $(\tilde{Q},\vec{P},\vec{s}) \in \vec{\mathcal{O}}(\tilde{S},\tilde{L})$ we have that

(1) $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P})) = \tilde{\mathcal{G}}(\tilde{L})$;

(2) $\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$;

(3) $2(|\tilde{Q}| + |\vec{P}|) = |\tilde{L}|$;

(4)$|\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P})| = |\tilde{L}|$;

(5) $\tilde{U}(\tilde{Q}) \cap \tilde{U}(\vec{P}) = \emptyset$, and no operator appears in more than one pair in either $\tilde{Q}$ or $\vec{P}$;

(6) $\tilde{O}$ is an unimprovable set of operators that extends to $\tilde{X}$;

(7) $\max_{o \in \tilde{O}} \omega_{\tilde{S}}(o) \leqslant \min_{x \in \tilde{X}}(x)$;

(8) $(\omega_{\tilde{S}} \circ p_1)(q) = m_{\tilde{S}}(q)$ for all $q \in \vec{P}$;

(9) $\vec{M}(\vec{P})$ is ordered;

(10) $\vec{P}$ is a choice of qubits stabilized by $\tilde{S}$;

(11) $\vec{P}$ is an optimal choice of qubits;

where $\tilde{X} := \tilde{U}(\tilde{Q}) \cup \{p_2(\vec{P}_i) : 1 \leqslant i \leqslant |\vec{P}|, \vec{s}_i = 1\}$ if $\vec{P}$ is nonempty and $\tilde{X} := \tilde{U}(\tilde{Q})$ otherwise, and $\tilde{O} := \tilde{U}(\vec{P}) \backslash \tilde{X}$.

*Proof.* The proof is by induction. It is easy to see that these properties hold for $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_0 = (\tilde{L}',\vec{\emptyset},\vec{\emptyset})$, so now assume that they hold for $(\tilde{Q},\vec{P},\vec{s}) := \vec{\mathcal{O}}(\tilde{S},\tilde{L})_i$, and let $(\tilde{Q}',\vec{P}',\vec{s}') := \vec{\mathcal{O}}(\tilde{S},\tilde{L})_{i+1}$. For convenience, define $\tilde{X} := \tilde{U}(\tilde{Q}) \cup \{p_2(\vec{P}_i) : 1 \leqslant i \leqslant |\vec{P}|, \vec{s}_i = 1\}$ [or $\tilde{X} := \tilde{U}(\tilde{Q})$ if $\vec{P}$ is empty], $\tilde{X}' := \tilde{U}(\tilde{Q}') \cup \{p_2(\vec{P}_i') : 1 \leqslant i \leqslant |\vec{P}'|, \vec{s}_i' = 1\}$, $\tilde{O} := \tilde{U}(\vec{P}) \backslash \tilde{X}$, and $\tilde{O}' := \tilde{U}(\vec{P}') \backslash \tilde{X}'$.

We now prove each of the conclusions above; note that in each conclusion we may assume that the conclusions prior to it have already been established, so we do so implicitly.

Also, when we say that we are assuming we are in "Case (1)" or "Case (2)," we mean that we are assuming that $(\tilde{Q}',\vec{P}',\vec{s}')$ followed from respectively Case (1) or Case (2) in the definition of $\vec{\mathcal{O}}$.

(1) Examination of the definition reveals that $\tilde{Q}'$ and $\vec{P}'$ are constructed entirely from products of elements in $\tilde{Q}$ and $\vec{P}$ so that $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')) \subseteq \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}))$.

If $\vec{\mathcal{O}}(\tilde{S},\tilde{L})_{i+1}$ was defined using Case (1) then let $(b,a) := \vec{P}_k$ and $(b',a') := \vec{P}_k'$ where $k$ is the integer described in Case (1); otherwise let $(a,b) := q$ be the pair selected from $\tilde{Q}$ in the definition and $(a',b')$ be the last element of $\vec{P}'$. Note that in either case, $a = a'$.

In both cases, observe that for every operator $o \in \tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \backslash \{b\}$ we have that either $o$ or $o \cdot a$ is contained in $\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')$, and since $a$ is also contained in this set we see immediately that any operator in $\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \backslash \{b\}$ can be obtained from products of elements in $\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')$ [i.e., from an operator in $\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')$ times possibly $a$]. Thus we conclude that $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \backslash \{b\}) \subseteq \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}'))$. Since $b'$ is the product of $b$ with elements in $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \backslash \{b\})$, and $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \backslash \{b\}) \subseteq \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}'))$, we conclude that $b \in \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{(}\vec{P}'))$ and so $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P})) \subseteq \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}'))$.

Thus we have proven that $\tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')) = \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P})) = \tilde{\mathcal{G}}(\tilde{L})$, and so we are done.

(2) This follows from the fact that $\tilde{L} \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S}) \Rightarrow \tilde{\mathcal{G}}(\tilde{L}) \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ and $\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P}) \subseteq \tilde{\mathcal{G}}(\tilde{U}(\tilde{Q}) \cup \tilde{U}(\vec{P})) = \tilde{\mathcal{G}}(\tilde{L})$ (which we just proved).

(3) By construction, either $|\tilde{Q}'| = |\tilde{Q}|$ and $|\vec{P}'| = |\vec{P}$ [in Case (1)] or $|\tilde{Q}'| = |\tilde{Q}| - 1$ and $|\vec{P}'| = |\vec{P}| + 1$ [in Case (2)]. In either case we have that $2(|\tilde{Q}'| + |\vec{P}'|) = 2(|\tilde{Q}| + |\vec{P}|) = |\tilde{L}|$.

(4) Since the elements in $\tilde{L}$ are members of conjugal pairs, they are therefore independent, and so we see that we need at least $|\tilde{L}|$ operators to generate $\tilde{\mathcal{G}}(\tilde{L})$. Thus we need $|\tilde{L}| \leqslant |\tilde{U}(\tilde{Q}') \cap \tilde{U}(\vec{P}')| \leqslant 2(|\tilde{Q}'| + |\vec{P}'|) = |\vec{L}|$, where the second inequality comes from the fact that a pair can unpack to at most two operators, and the last equality comes from the previous conclusion. We thus conclude that $|\tilde{U}(\tilde{Q}') \cap \tilde{U}(\vec{P}')| = |\vec{L}|$.

(5) By combining the previous two conclusions we see that $|\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')| = 2(|\tilde{Q}'| + |\vec{P}'|)$; if this conclusion were false (i.e., an operator were repeated somewhere) then we would have that $|\tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')| < 2(|\tilde{Q}'| + |\vec{P}'|)$, which contradicts our earlier results.

(6) First assume that we are in Case (1). Let $k$ be the integer described in this case, $(a,b) := \vec{P}_k$, and $(a',b') := \vec{P}'_k$. Note that $a \in \tilde{O}$, and by construction $a' := \Pi(\tilde{A}')$ where $\tilde{A}' \subseteq \tilde{O} \cup \tilde{X}$ and $\tilde{A}' \cap \tilde{O} = \{a\}$, and so since by the inductive hypothesis we know that $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$ we know that $\omega_{\tilde{S}}(a') \leqslant \omega_{\tilde{S}}(a)$. Since by the inductive hypothesis we also know that $\max_{o\in\tilde{O}}\omega_{\tilde{S}}(o) \leqslant \min_{x\in\tilde{X}}(x)$, by Lemma 2 we conclude that $\omega_{\tilde{S}}(a') = \omega_{\tilde{S}}(a)$. Lemma 9 thus applies to our situation and allows us to conclude that $\tilde{O}'' := (\tilde{O}\backslash\{a\}) \cup \{a'\}$ is an unimprovable set that extends to $\tilde{X}$. Furthermore, since by construction $\tilde{X}' \subseteq \tilde{\mathcal{G}}(\tilde{X})$ and $b \in \tilde{X}$, Lemma 10 allows us to conclude that $\tilde{O}''$ extends to $\{b\} \cup \tilde{X}'$. By construction, there is an error of minimal weight that acts on $b$ but not on any other operator in $\tilde{X}'$, which means that by Lemma 8 we conclude that $\tilde{O}'' \cup \{b\} \equiv \tilde{O}'$ extends to $\tilde{X}'$.

Now assume that we are in Case (2). Note that the only change from $\tilde{O}$ to $\tilde{O}'$ is the addition of a single element $o$ that has an error that acts only on it but not on any other operator in $\tilde{X}'$. Note that since $\{o\} \cup \tilde{X}' \subseteq \tilde{\mathcal{G}}(\tilde{X})$ we conclude from Lemma 10 that $\tilde{O}$ extends to $\{o\} \cup \tilde{X}'$, and from Lemma 8 we conclude that $\tilde{O}'$ extends to $\tilde{X}'$.

(7) First observe that since $\tilde{X}' \subseteq \tilde{\mathcal{G}}(\tilde{X})$, we conclude from Lemma 2 that $\min_{x\in\tilde{X}}\omega_{\tilde{S}}(x) \leqslant \min_{x'\in\tilde{X}'}\omega_{\tilde{S}}(x')$.

The difference between $\tilde{O}$ and $\tilde{O}'$ is the addition of a minimizer of $\omega_{\tilde{S}}$ over $\tilde{X}$, $o$, and possibly also the replacement of a single element. Since $\max_{a\in\tilde{O}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}'}\omega_{\tilde{S}}(a)$, we conclude that since $\omega_{\tilde{S}}(o) = \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a)$ therefore $\max_{a\in\tilde{O}\cup\{o\}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}'}\omega_{\tilde{S}}(a)$. If $\tilde{O}' = \tilde{O} \cup \{o\}$ then we are done. Otherwise, we are in Case (1) which means that we have also replaced an element in $\tilde{O}$; however, the operator we have replaced it with is the product of an operator from $\tilde{O}$ and operators from $\tilde{X}$, and since $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$ we conclude that the replacement can be no better than the operator it is replacing. Thus, $\max_{a\in\tilde{O}'}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}'}\omega_{\tilde{S}}(a)$.

(8) Since $\max_{a\in\tilde{O}'}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}'}\omega_{\tilde{S}}(a)$, we immediately conclude that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_i) = m_{\tilde{S}}(\vec{P}'_i)$ when $\vec{a}'_i = 1$. By the inductive hypothesis, we know that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_i) = m_{\tilde{S}}(\vec{P}'_i)$ where $\vec{P}'_i = \vec{P}_i$; furthermore, in both cases the pairs at the locations where $\vec{a}_i = \vec{0}$ are unchanged from $\vec{P}$ to $\vec{P}'$, and in each case this turns out to leave just a single location that we still need to examine.

In Case (1), this location is the index $k$ described in that case, where $\vec{a}_k = \vec{1}$ and $\vec{a}'_k = \vec{0}$. Since $p_1(\vec{P}'_k)$ is the product of a single element of $\tilde{O}$ and elements from $\tilde{X}$, we conclude from the fact that $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$ that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_k) \leqslant (\omega_{\tilde{S}} \circ p_1)(\vec{P}_k)$. By the inductive hypothesis we know that $\max_{a\in\tilde{O}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a)$. Because $p_2(\vec{P}'_k)$ is a product of elements from $\tilde{X}$ we conclude from Lemma 2 that $\min_{a\in\tilde{X}}\omega_{\tilde{S}}(a) \leqslant (\omega_{\tilde{S}} \circ p_2)(\vec{P}'_k)$. Since $p_1(\vec{P}) \in \tilde{O}$, we conclude that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}_k) \leqslant \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a)$. Combining all of these inequalities we reach the conclusion that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_k) \leqslant (\omega_{\tilde{S}} \circ p_2)(\vec{P}'_k)$ and hence $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_k) = m_{\tilde{S}}(\vec{P}'_k)$.

In Case (2), this location is the end of the sequence $\vec{P}'$, but since the addition to the sequences is a pair of operators from $\tilde{X}$ such that the first member is a minimizer of $\omega_{\tilde{S}}$ over $\tilde{X}$ we conclude that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_{|\vec{P}'|}) = m_{\tilde{S}}(\vec{P}'_{|\vec{P}'|})$.

(9) By the inductive hypothesis we have that $\vec{M}_{\tilde{S}}(\vec{P})_i = (\omega_{\tilde{S}} \circ \omega_{\tilde{S}})(\vec{P}_i)$, and we have just shown that $\vec{M}_{\tilde{S}}(\vec{P}')_i = (\omega_{\tilde{S}} \circ \omega_{\tilde{S}})(\vec{P}'_i)$. By the inductive hypothesis we know that $\vec{M}_{\tilde{S}}(\vec{P})$ is ordered, and so to prove that $\vec{M}_{\tilde{S}}(\vec{P}')$ is ordered we need only check the places in the sequence where $p_1(\vec{P}_i) \neq p_1(\vec{P}'_i)$. In both cases there is exactly one location where the first member of a pair is modified from $\vec{P}$ to $\vec{P}'$.

In Case (1), this is the index $k$ defined in that case, at which the first member was replaced with a product of that first member with elements in $\tilde{X}$. Since this member is in $\tilde{O}$, and since $\max_{a\in\tilde{O}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a)$ (by the inductive hypothesis), we conclude from the fact that $\tilde{O}$ is an unimprovable set that extends to $\tilde{X}$ that $(\omega_{\tilde{S}} \circ p_1)(\vec{P}'_k) = (\omega_{\tilde{S}} \circ p_1)(\vec{P}_k)$, and so we conclude that $\vec{M}_{\tilde{S}}(\vec{P}')$ is ordered.

In Case (2), this is the end of the sequence $\vec{P}'$ where a pair was appended to $\vec{P}$. Since the pair contains elements from $\tilde{X}$, and $\max_{a\in\tilde{O}}\omega_{\tilde{S}}(a) \leqslant \min_{a\in\tilde{X}}\omega_{\tilde{S}}(a)$, we conclude that $\vec{M}_{\tilde{S}}(\vec{P}')_{|\vec{P}'|} \geqslant \vec{M}_{\tilde{S}}(\vec{P})_i$ for $i < |\vec{P}'|$, and so we conclude that $\vec{M}_{\tilde{S}}(\vec{P}')$ is ordered.

(10) The fact that $\vec{P}$ is a choice of logical qubits stabilized by $\tilde{S}$ follows directly from the previous conclusions.

(11) From the definition of an unimprovable set it is easy to see that since $\tilde{O}'$ is an unimprovable set that extends to $\tilde{X}'$, it also extends to $\tilde{X}' \cup \tilde{O}' = \tilde{U}(\tilde{Q}') \cup \tilde{U}(\vec{P}')$. Since $\{p_1(q) : q \in \vec{P}'\} \subseteq \tilde{O}'$ and $\tilde{U}(\vec{P}') \subseteq \tilde{X}' \cup \tilde{O}'$, it is also easy to see from the definition that $\{p_1(q) : q \in \vec{P}'\}$ is an unimprovable set that extends to $\tilde{U}(\vec{P}')$—that is, taking subsets does not affect the property of unimprovability. Thus, we conclude from Theorem 3 that $\vec{P}'$ is therefore an optimal choice of qubits. ■

*Remark.* Now that the heavy lifting has been done by the preceding proposition, the proof of Theorem 4 is relatively simple.

*Proof of Theorem 4.* At every step in the algorithm, we either change an entry in $\vec{s}$ from 1 to 0 or remove an element from $\tilde{Q}$. Since $\vec{s}$ is of finite length, as long as $\tilde{Q}$ is nonempty there will be a step at which another element is removed from it. Thus, there is an index $k$ such that if $\vec{O}(\tilde{S},\tilde{L})_k = (\tilde{Q},\vec{P},\vec{s})$ then $\tilde{Q} = \emptyset$, and by definition this is the last element of the sequence. By Proposition 5 we know that $\vec{P}$ is optimal and also that $\tilde{\mathcal{G}}(\vec{P}) = \tilde{\mathcal{G}}(\vec{L})$ (since $\tilde{Q}$ is empty), and so we are done. ■

#### *5. Running time of the algorithm*

In this section we analyze the running time of the optimization algorithm; the result is presented in the following theorem.

*Theorem 5.* Suppose we are given

(a) a set of commuting Pauli operators, $\tilde{S}$, acting on $N$ physical qubits;

(b) a set of pairs, $\tilde{L} \subseteq \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$, conjugal with respect to $\tilde{U}(\tilde{Q}) \cup \tilde{S}$;

(c) and a set of Pauli operators $\tilde{C}$ such that $\tilde{\mathcal{G}}(\tilde{C}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$; then the time needed to compute the sequence $\vec{\mathcal{O}}(\tilde{S},\tilde{L})$ is in the set $O(|\tilde{C}|^2 + |\tilde{L}|(|\tilde{L}| + d)3^d\binom{N}{d})$, where $d := \vec{M}(\vec{P})_{|\vec{P}|}$ and $(\tilde{Q},\vec{P},\vec{S})$ is the last element in the sequence (i.e., $\vec{P}$ is the desired optimal choice of qubits).

*Remark.* Before proving this theorem, we shall first prove a number of related lemmas and propositions.

The most complicated part of analyzing the running time of the optimization algorithm is analyzing the time needed to find the minimum weight undetectable error. In fact, the procedure for doing this was not even described explicitly in the algorithm, so we shall now explain how we do it.

The algorithm we employ is based on the Brouwer-Zimmermann search algorithm, which searches for the minimum weight binary string satisfying some property, given a set of binary string generators endowed with a multiplication operation defined to be the exclusive-or operation. The Brouwer-Zimmermann algorithm works by using a Gaussian elimination analog to express the generators in reduced row-echelon form; it then performs its search by examining all products of $r$ generators for increasing $r$. When all of the products of $r$ generators have been enumerated, one knows that the set of strings that has yet to be enumerated has weight $r + 1$ or greater, since the row-echelon form means that every string has a column for which it is the only string with a 1 in that column, and so a product of $k$ strings must have a weight of at least $k$. Thus, as the search proceeds, there is a growing lower bound on the weight of the binary string, and the search halts when a string has been found that matches this bound.

This algorithm cannot be immediately applied to the current problem because we are not working with binary strings, and in particular Pauli operators have a more complicated multiplication operation than binary strings. Fortunately, in [43] White and Grassl showed that the Brouwer-Zimmermann enumeration can be generalized.

The key difference between binary strings and Pauli operators is that binary strings only have two possible values in a given column, whereas Pauli operators have four. Thus, whereas we only need one element to generate all of the possible values in a given column for a binary string, we need *two* elements to generate all of the possible values in a given column for a Pauli operator. Thus, rather than working with generators, we instead work with a generalization that White and Grassl call *pseudogenerators*, which we shall define here as follows.

*Definition.* A *pseudogenerator* is a set of either one or two Pauli operators. In an abuse of notation, we extend the functions $\tilde{U}$ and $\tilde{\mathcal{G}}$ to be respectively $\tilde{G} \mapsto \bigcup_{\tilde{g}\in\tilde{G}} \tilde{g}$ and $\tilde{G} \mapsto (\tilde{\mathcal{G}} \circ \tilde{U})(\tilde{G})$ when applied to a set of pseudogenerators.

*Remark.* (The preceding definition does not follow that of White and Grassl exactly; it has been specialized to our situation for the sake of simplicity.)

Unlike "normal" generators—i.e., Pauli operators—a product of generators is not a Pauli operator but rather a set of Pauli operators, which we define as follows.

*Definition.* Suppose we are given a set of $r$ pseudogenerators $\tilde{G}$. Let $\tilde{X} := \{\tilde{\mathcal{G}}(\tilde{g})\backslash\{I\} : \vec{g} \in \tilde{G}\}$, $\tilde{Y}$ be the $r$-ary Cartesian product of the $r$ sets contained in $\tilde{X}$, and $\tilde{Z}$ be the set consisting of the normal quantum operator product of the $r$ operators in each $r$-tuple in $\tilde{Y}$. Then $\tilde{Z}$ is defined to be the *pseudoproduct* of the pseudogenerators in $\tilde{G}$. For convenience, we define a function $\tilde{\Pi}$ such that $\tilde{\Pi}(\tilde{G})$ is the pseudoproduct of the pseudogenerators in $\tilde{G}$.

*Remark.* The following lemma places a bound on the size of the pseudoproduct.

*Lemma 11.* The pseudoproduct of $r$ pseudogenerators contains at most $3^r$ operators.

*Proof.* Every set contained in the set $\tilde{X}$ that was described in the definition of pseudogenerator (which appears just before this Lemma) has a cardinality of either 1 or 3, and the size of $\tilde{Y}$ is equal to the product of the sizes of all the sets in $\tilde{X}$; since $|\tilde{X}| = r$, we therefore conclude that the cardinality of $\tilde{Y}$ and hence the number of operators in the pseudoproduct is at most $3^r$. ■

*Corollary 1 (to Lemma 11).* Given a set of $r$ pseudogenerators $\tilde{G}$ then the set $\tilde{O} := \{f(o) : o \in \tilde{\mathcal{G}}(\tilde{G})\}$ can be computed in time $O((T + r)3^r)$, where the time needed to compute $f$ is $O(T)$.

*Proof.* From Lemma 11 we know that there are at most $3^r$ operators in the pseudoproduct, so $|\tilde{O}| \leqslant 3^r$. Furthermore, for every element in the set we first need to compute the corresponding operator in the pseudoproduct, which requires $r$ time since it is the product of $r$ operators, and then we need to compute $f$, which by assumption requires a time of $O(T)$. ■

*Remark.* In order to be able to place a lower bound on binary strings that have yet to be examined in the Brouwer-Zimmermann enumeration, we need the generators of the binary strings over which we are searching to have the property that each generator has a column such that it is the only generator with a 1 in that column, so that products of $r$ generators must have at least weight $r$. Because we want to similarly place a bound on unexamined products of pseudogenerators, we generalize this property with the following definition.

*Definition.* A set of pseudogenerators $\tilde{G}$ is said to be *disjoint* if for every $\tilde{g} \in \tilde{G}$ there exists some physical qubit $k$ such that either $X_k$ or $Z_k$ (or both) anticommutes with every operator in $\tilde{\mathcal{G}}(\tilde{g})\backslash\{I\}$, but both $X_k$ and $Z_k$ commute with every operator in $\bigcup_{\tilde{g}'\in\tilde{G}\backslash\{\tilde{g}\}} \tilde{\mathcal{G}}(\tilde{g}')$.

*Remark.* With the following lemma, we show that the property of disjointness is exactly what we need to obtain the bounds that we want.

*Lemma 12.* All of the operators in the pseudoproduct of any $r$ (distinct) pseudogenerators chosen from a disjoint set of pseudogenerators have weight of at least $r$.

*Proof.* Every operator in the pseudoproduct is the product of $r$ factors, each of which is associated with some distinct physical qubit $k$ such that it anticommutes with either $X_k$ or

$Z_k$ (or both) but every other factor commutes with both $X_k$ and $Z_k$; thus, the product must anticommute with at least $r$ single-qubit operators acting on distinct physical qubits, and so it must have a weight of at least $r$.

*Remark.* Now that we have the concept of a disjoint set of pseudogenerators and a result showing that an operator in a pseudoproduct of $r$ of them must have a weight of at least $r$, we present in the following lemma an algorithm for searching through the space spanned by the pseudogenerators for an operator satisfying a given property.

*Lemma 13.* Given a set of pseudogenerators $\tilde{G}$ acting on $N$ physical qubits and a test function $f : \tilde{\mathfrak{P}} \to \{0,1\}$ such that $f^{-1}(1) \cap \tilde{\mathcal{G}}(\tilde{G}) \neq \emptyset$, then a solution $o$ such that $f(o) = 1$ and $\omega_{\tilde{S}}(o) = \min_{o' \in \tilde{\mathcal{G}}(\tilde{G}), f(o')=1} w(o')$ can be computed in time $O((T + d)3^d \binom{|\tilde{G}|}{r})$ where $d := \min(w(o), |\tilde{G}|)$ and $T$ is the time needed to compute $f$.

*Remark.* This lemma follows directly from the results in [43], though the proof is included here both for completeness and also to show specifically how the results specialize to our case. A pseudocode representation of the algorithm can be found in Table V.

*Proof.* Define $\tilde{C}_r$ to be the set of all operators such that if $o \in \tilde{C}_r$ then there is some subset of exactly $r$ pseudogenerators from $\tilde{G}$ such that $o$ is contained in their pseudoproduct. Note that $\cup_r \tilde{C}_r = \tilde{\mathcal{G}}(\tilde{G})$, so for every operator $o$ in the search space there is an integer $r$ such that $o \in \tilde{C}_r$. Corollary 1 shows that we can evaluate $f$ on every element of the pseudoproduct of $r$ pseudogenerators in time $O((T + r)3^r)$, so since there are $\binom{|\tilde{G}|}{r}$ ways to choose $r$ pseudogenerators from $\tilde{G}$ we conclude that we can search $\tilde{C}_r$ for a solution to $f(o) = 1$ in time $O((T + r)3^r \binom{|\tilde{G}|}{r})$. From Lemma 12 we conclude that $w(o) \geqslant r$ for every $o \in \tilde{C}_r$. By extension this means that $w(o) \geqslant r$ for every $o \in \bigcup_{r'=r}^{|\tilde{G}|} \tilde{C}_{r'}$, and therefore that if $o \notin \bigcup_{r'=0}^{r-1} \tilde{C}_{r'}$ and $o \in \tilde{\mathcal{G}}(\tilde{G})$ then $w(o) \geqslant r$. Thus, if there exists an $r$ such that $r = \min_{o \in \bigcup_{r'=0}^{r-1} \tilde{C}_{r'}, f(o)=1} w(o)$ then we know that $r = \min_{o \in \tilde{\mathcal{G}}(\tilde{O}), f(o)=1} w(o)$—that is, $r$ is *exactly* the weight of the minimum weight solution to $f(o) = 1$, since any operator in the search space that *is not* contained in $\bigcup_{r'=0}^{r-1} \tilde{C}_{r'}$ must have a weight of at least $r$. Put another way, after having enumerated all of the elements in $\bigcup_{r'=0}^{r-1} \tilde{C}_{r'}$ we can check to see whether the smallest solution to $f$ we have seen so far (if any) has weight less than or equal to $r$, and if so we are done since we have found the minimal weight solution.

TABLE V. Algorithm which finds the minimal weight operator in a given generating set that satisfies a given predicate. For the sake of convenience, we also allow the query function to return auxiliary information that is returned to the caller along with the minimal weight operator.

```
1   $r \leftarrow 1$
2   $m \leftarrow \infty$
3   while $m > r$ and $r \le |\vec{G}|$
4       do
5           for each $\vec{H} \subseteq \vec{G}$ such that $|\vec{H}| = r$,
6           and each $o$ in the pseudoproduct of $\vec{H}$
7               do
8                   if weight$(o) < m$
9                       then
10                          $(q, u) \leftarrow f(o)$
11                          if $q$ is TRUE
12                              then
13                                  $m \leftarrow$ weight$(o)$
14                                  $\alpha \leftarrow (o, u)$
15                                  if $m = r$
16                                      then
17                                          goto 19
18          $r \leftarrow r + 1$
19  return $\alpha$
```

Now consider the procedure of searching through each $\tilde{C}_r$ starting with $r = 0$. We know that we will eventually find at least one solution to $f(o) = 1$, since in this proposition we have assumed that such an operator exists in the search space [by the assumption that $f^{-1}(1) \cap \tilde{\mathcal{G}}(\tilde{G}) \neq \emptyset$]. Furthermore, employing this procedure we will find the minimal solution $o$ no *later* than after we have searched through $\tilde{C}_r$ for $r = 0, \ldots, w(o)$, since at that point all of the unexamined operators have a weight greater than $w(o)$. Thus, we conclude that we shall find the minimal weight solution after having searched at most all of the elements in $\bigcup_{r=0}^{\min(w(o),|\tilde{G}|)} C_r$, which we can do in time

$$O\left(\sum_{r=0}^{\min(w(o),|\tilde{G}|)} (T + r)3^r \binom{|\tilde{G}|}{r}\right) \subseteq O\left((T + d)3^d \binom{N}{d}\right),$$

where $d := \min(w(o), |\tilde{G}|)$. ■

*Remark.* The preceding lemma is rather general, so we shall show how it specializes to our case. First, however, we use the following three lemmas to prove that our search space is generated by exactly $N$ pseudogenerators.

*Lemma 14.* Given a set of disjoint pseudogenerators, $\tilde{G}$, the largest subset $\tilde{X} \subseteq \tilde{U}(\tilde{G})$ such that $\tilde{X}$ commutes has size $|\tilde{X}| \leqslant |\tilde{G}|$.

*Proof.* If $\tilde{X}$ contained more that $\tilde{G}$ operators then by the pigeon hole principle there would have to be at least two operators from the same pseudogenerator, and thus which did not commute. ■

*Lemma 15.* A set of disjoint pseudogenerators $\tilde{G}$ acting on $N$ qubits satisfies $|\tilde{G}| \leqslant N$.

*Proof.* This follows directly from the definition and the pigeon hole principle. ■

*Lemma 16.* For any set of commuting operators $\tilde{S}$ acting on $N$ physical qubits, if $\tilde{G}$ is a set of disjoint pseudogenerators satisfying $\tilde{\mathcal{G}}(\tilde{G}) = \tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ then $|\tilde{G}| = N$.

*Proof.* Since $\tilde{\mathcal{C}}_{\tilde{\mathfrak{P}}}(\tilde{S})$ contains a subset of $N$ independent commuting operators, so must $\tilde{\mathcal{G}}(\tilde{G})$ and therefore $\tilde{U}(\tilde{G})$; thus, Lemma 14 implies that $|\tilde{G}| \geqslant N$, and combining this bound with that given by Lemma 15 we see that $|\tilde{G}| = N$. ■

*Remark.* We now prove a lemma which shows how the search specializes to the case of our qubit optimization algorithm.

*Lemma 17.* Given

(a) a set of Pauli operators $\tilde{S}$ acting on $N$ physical qubits,

(b) a set of disjoint pseudo-generator $\tilde{G}$ such that $\tilde{\mathcal{G}}(\tilde{G}) = \tilde{\mathcal{C}}(\tilde{S})$, and

(c) a nonempty set of Pauli operators $\tilde{Q}$ such that $\tilde{Q} \cap \tilde{S} = \emptyset$ and every operator in $\tilde{Q}$ is a member of a conjugal pair in relation to $\tilde{Q} \cup \tilde{S}$,
then a minimal weight undetectable error acting on any operator in $\tilde{Q}$ can be found in time $O((|\tilde{Q}| + d)3^d\binom{N}{d})$ where $d := \min(w(o),N)$.

*Proof.* First observe that by Lemma 16 we know that $|\tilde{G}| = N$.

Define the function $f : \tilde{\mathcal{G}}(\tilde{G}) \to \{0,1\}$ by

$$f(o) := \begin{cases} 1 & \exists\, q \in \tilde{Q} \text{ such that } \{o,q\} = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that solutions to $f$ are undetectable errors acting on $\tilde{Q}$, and also that this function can be computed in time $O(|\tilde{Q}|)$ by checking the commutator for each element in $\tilde{Q}$. Furthermore note that for every operator in $\tilde{Q}$ there is another operator in $\tilde{Q}$ which anticommutes with it, and also that $\tilde{Q} \subset \tilde{\mathcal{C}}_{\mathfrak{P}}(\tilde{S})$. Thus, since $\tilde{Q}$ is nonempty, there is at least one operator $o \in \tilde{\mathcal{G}}(\tilde{G})$ such that $f(o) = 1$. Thus, by Lemma 13, we know that we can compute a minimal weight solution to $f$ in time $O((|\tilde{Q}| + d)3^d\binom{N}{d})$ where $d := \min(w(o),N)$. ■

*Remark.* In order to make use of the preceding result, we need to have a set of disjoint pseudogenerators whose pseudoproduct covers our search space. However, we usually start instead with a set of ordinary Pauli operators that generate this space. Thus, we shall now show that the former can be computed from the latter—i.e., that given a set of Pauli operators, we can compute a set of disjoint pseudogenerators that spans the same space. First we present a lemma that provides a criterion sufficient to show that a set of pseudogenerators is distinct.

*Lemma 18.* Given a set of pseudogenerators, $\tilde{G}$, if there exists a map $p : \tilde{U}(\tilde{G}) \to (\{X_k\}_k \cup \{Z_k\}_k)$ such that

(1) for every $o \in \tilde{U}(\tilde{G})$, $o$ is the unique operator in $\tilde{\mathcal{G}}(\tilde{G})$ that anticommutes with $p(o)$ and

(2) for every $\tilde{g} \in \tilde{G}$, the operators in $\tilde{g}$ are both mapped by $p$ to single-qubit operators acting on the same physical qubit $k$, and they are the only such operators in $\tilde{\mathcal{G}}(\tilde{G})$ that are mapped by $p$ to operators acting on $k$, then $\tilde{G}$ is disjoint.

*Proof.* For every $\tilde{g} \in \tilde{G}$, we conclude from property (2) of $p$ that there is some physical qubit $k$ such that every operator in $\tilde{g}$ is mapped by $p$ to either $X_k$ or $Z_k$, and hence by property (1) this means that every operator in $\tilde{\mathcal{G}}(g)$ anticommutes with either $X_k$ or $Z_k$. Since by property (1) we know that the choice of $X_k$ or $Z_k$ is different for each operator in $\tilde{\mathcal{G}}(\tilde{g})$, we conclude that if there is more than one operator in $\tilde{g}$ then the product anticommutes with *both* $X_k$ and $Z_k$. Finally, by property (2) we know that every operator in $\tilde{g}'$ for $\tilde{g}' \in \tilde{\tilde{G}}\backslash\{g\}$ commutes with $X_k$ and $Z_k$. ■

*Remark.* We now show that any set of operators that we are using to generate a search space can be expressed equivalently as a set of disjoint pseudogenerators. A pseudocode representation of the algorithm described in the following Lemma is shown in Table VI.

*Lemma 19.* Given any set of Pauli operators, $\tilde{O}$, there exists a set $\tilde{G}$ of pseudogenerators such that

(1) $|\tilde{G}| \leqslant |\tilde{O}|$,

(2) $\tilde{\mathcal{G}}(\tilde{O}) = \tilde{\mathcal{G}}(\tilde{G})$,

(3) the map $p$ described in Lemma 18 exists for $\tilde{G}$, and $\tilde{G}$ can be computed in time $O(|\tilde{O}|^2)$.

TABLE VI. Algorithm which computes a set of disjoint pseudogenerators that generates the input set of operators.

```
1   $\vec{p} \leftarrow []$
2   $i \leftarrow 0$
3   while $i < |\vec{O}|$
4       do
5           $o \leftarrow \vec{O}[i]$
6           for $j \leftarrow 0$ to $i - 1$
7               do
8                   $(n,z) \leftarrow \vec{p}[j]$
9                   if $z = 0$
10                      then
11                          if anti$(o, X_n)$
12                              then $o \leftarrow o \cdot \vec{O}[j]$
13                      else
14                          if anti$(o, Z_n)$
15                              then $o \leftarrow o \cdot \vec{O}[j]$
16          if $o$ is identity
17             then
18                 delete $\vec{O}[i]$
19                 goto 3
20          for $n \leftarrow 0$ to number of physical qubits
21              do
22                  if anti$(o, X_n)$
23                      then
24                          $z \leftarrow 0$
25                          goto 30
26                  elseif anti$(o, Z_k)$
27                      then
28                          $z \leftarrow 1$
29                          goto 30
30          if $z = 0$
31             then
32                 for $j \leftarrow 0$ to $i - 1$
33                     do
34                         if anti$(\vec{O}[k], X_n)$
35                             then $\vec{O}[j] \leftarrow \vec{O}[j] \cdot o$
36             else
37                 for $j \leftarrow 0$ to $i - 1$
38                     do
39                         if anti$(\vec{O}[j], Z_n)$
40                             then $\vec{O}[j] \leftarrow \vec{O}[j] \cdot o$
41          append $(n, z)$ to $\vec{p}$
42          $\vec{O}[i] \leftarrow o$
43          $i \leftarrow i + 1$
44  $\vec{G} \leftarrow []$
45  for $n \leftarrow 0$ to number of physical qubits
46      do
47          $\vec{g} \leftarrow []$
48          for $i \leftarrow 0$ to $|\vec{O}| - 1$
49              do
50                  $(n', \_) \leftarrow \vec{p}[i]$
51                  if $n = n'$
52                      then append $\vec{O}[i]$ to $\vec{g}$
53          if $\vec{g} \neq []$
54             then append $g$ to $\vec{G}$
55  return $\vec{G}$
```

*Remark.* The structure of this proof bears some similarities to Proposition 2. In contrast with Proposition 2, however, in the setting of this lemma we are working with operators that in general will not commute.

*Proof.* The proof is by induction. For the base case, we observe that if $\tilde{O}$ is empty, then the trivial set $\tilde{G} := \emptyset$ and the trivial function $p$ whose domain is the empty set satisfy the requirements.

Now assume that this lemma has been proven for sets of cardinality $n-1$, and suppose we are given a (nonempty) set $\tilde{O}$ of cardinality $n$. Take any operator $o \in \tilde{O}$. By recursive application of this lemma, we know that we can construct the set $\tilde{G}' := \tilde{G}$ and the function $p' := p$ described in this lemma given $\tilde{O} := \tilde{O}\backslash\{o\}$ in time $O((n-1)^2) = O(n^2)$.

Let

$$o' := o \cdot \prod_{\substack{x\in\tilde{U}(\tilde{G}'),\\ \{o,p(x)\}=0}} x.$$

Note that for every $x \in \tilde{U}(\tilde{G}')$, it must be that $o'$ commutes with $p'(x)$, since $o'$ is formed from a product that has either two factors that anticommute with $p'(x)$ [namely, $o$ and $p'(x)$] or no operators that anticommute with $p'(x)$. If $o'$ is the identity operator, then let $\tilde{G} := \tilde{G}'$ and $p := p'$ and we are done. Otherwise, there must be some operator $z \in (\{X_k\}_k \cup \{Z_k\}_k) / \{p'(x) : x \in \tilde{U}(\tilde{G}')\}$ that anticommutes with $o$. Define the function $f$ by

$$f(x) := \begin{cases} x \cdot o', & \{x,z\} = 0, \\ x & \text{otherwise}, \end{cases}$$

and let $\tilde{G}'' := \{\{f(x) : x \in \tilde{g}\} : \tilde{g} \in \tilde{G}'\}$ and $p'' := p' \circ f^{-1}$. Note that $o'$ must be independent of the operators in $\tilde{U}(\tilde{G}')$, because $o'$ is not the identity and the product of $o'$ with any subset of operators $\tilde{A} \subset \tilde{U}(\tilde{G}')$ cannot be the identity since it must anticommute with $p'(a)$ for every $a \in \tilde{A}$. Thus, $f$ is a bijective map from $\tilde{U}(\tilde{G}')$ to $\tilde{U}(\tilde{G}'')$ and hence is invertible, and so we conclude that $p''$ is well defined. Since, as previously discussed, $o'$ commutes with $p(y)$ for every $y \in \tilde{U}(\tilde{G}')$, we conclude that multiplication by $o'$ does not change whether any operator $x \in \tilde{U}(\tilde{G}')$ commutes or anticommutes with $p(y)$ for any $y \in \tilde{U}(\tilde{G}')$, and so we conclude that the properties listed in Lemma 18 that $p'$ has in relation to $\tilde{G}'$ (from the inductive hypothesis) are preserved in the transformation by $f$ so that $p''$ also has the same properties in relation to $\tilde{G}''$. Furthermore, since every operator $x \in \tilde{U}(\vec{G}')$ was multiplied by a factor of $o'$ if and only if it anticommutes with $z$, we conclude that $f(z)$ must commute with $z$, and thus every operator in $\tilde{U}(\vec{G}'')$ must commute with $z$.

There are two cases to consider: either there is no operator $x \in \tilde{U}(\tilde{G}'')$ such that $p''(x)$ acts on the same qubit as $z$, or there is exactly one, since if there were more than two then it would violate the properties of $p''$, and if there were exactly two then by construction $o'$ would commute with $z$, leading to a contradiction. In the first case, let $\tilde{G} := \tilde{G}'' \cup \{\{o'\}\}$. In the second case, let $\tilde{G} := (\tilde{G}''\backslash\{\{x\}\}) \cup \{\{x,o'\}\}$, where $x$ is the single operator in $\tilde{U}(\tilde{G}'')$ such that $p(x)$ acts on the same qubit as $z$. In either case, define

$$p(x) := \begin{cases} p''(x), & x \in \tilde{U}(\tilde{G}''), \\ z, & x = o', \end{cases}$$

observing that it is well defined since $\tilde{U}(\tilde{G}) = \tilde{U}(\tilde{G}'') \cup \{o'\}$.

To prove conclusion (1), we note that $\tilde{G}$ has at most one more element than $\tilde{G}'$ and $\tilde{O}$ always has one more element than $\tilde{O}\backslash\{o\}$, so conclusion (1) follows from this fact combined with the inductive hypothesis.

To prove conclusion (2), we note that since $o'$ (and hence $o$) is independent with respect to $\tilde{U}(\tilde{G}')$, then because of how $\tilde{G}$ was constructed and the inductive hypothesis we have that $\tilde{\mathcal{G}}(\tilde{G}) = \tilde{\mathcal{G}}(\tilde{U}(\tilde{G}'') \cup \{o'\}) = \tilde{\mathcal{G}}(\tilde{U}(\tilde{G}') \cup \{o\}) = \tilde{\mathcal{G}}((\tilde{O}\backslash\{o\}) \cup \{o\}) = \tilde{\mathcal{G}}(\tilde{O})$.

To prove conclusion (3), we need to show that $p$ satisfies the properties listed in Lemma 18. To prove the first property, we note that for every $x \in \tilde{U}(\tilde{G})$ we have that either $x \in \tilde{U}(\tilde{G}'')$, in which case we have already shown that it is the unique operator that commutes with $p(x)$ as this is true for the operators in $\tilde{U}(\tilde{G}'')$ as well as for $o'$ (by construction), or $x = o'$, in which case this is still true since by construction $o'$ is the only operator in $\tilde{U}(\tilde{G})$ that anticommutes with $z$. To prove the second property, we note that due to the inductive hypothesis we need only consider the single change from $\tilde{G}''$ to $\tilde{G}$, which consisted of either adding or replacing a pseudogenerator; in the first case (adding a generator), observe that we showed earlier that no operator $y \in \tilde{G}''$ is such that $p''(y)$ acts on the same qubit as $z$, and in the second case (replacing a generator), note that we added $o'$ to the only generator in $\tilde{G}''$ containing an operator $y$ such that $p''(y)$ acts on the same qubit as $z$; in either case, we see that the second property holds for $\tilde{G}$.

Finally, we consider the running time. In addition to the $O(n^2)$ time required to construct $\tilde{G}'$, we required an additional $O(n)$ multiplication operations to construct $o'$ and $\tilde{G}''$; hence the total running time is $O(n^2)$. ■

*Corollary 2.* Given any set of Pauli operators, $\tilde{O}$, there exists a disjoint set of pseudogenerators $\tilde{G}$ such that $|\tilde{G}| \leqslant |\tilde{O}|$, $\tilde{\mathcal{G}}(\tilde{O}) = \tilde{\mathcal{G}}(\tilde{G})$, and $\tilde{G}$ can be computed in time $O(|\tilde{O}|^2)$.

*Proof.* This follows immediately from Lemmas 18 and 19. ■

*Remark.* We now have the tools that we need to analyze the running time of the algorithm.

*Proof of Theorem 5.* First observe that from Lemma 18 we conclude that we can compute a set of disjoint pseudogenerators $\tilde{G}$ such that $\tilde{\mathcal{G}}(\tilde{G}) = \tilde{\mathcal{G}}(\tilde{C}) = \tilde{\mathcal{C}}_{\mathfrak{P}}(\tilde{S})$ in time $O(|\tilde{C}|^2)$. We will assume that this set of pseudogenerators is implicitly available to us throughout the algorithm so that we do not need to compute it more than once.

At each step of the algorithm, we first need to find an operator $o$ that has an undetectable error of minimal weight

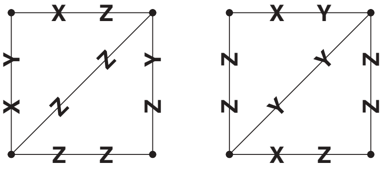


FIG. 1. Illustration of two possible labelings of a graph, which correspond to two possible choices of two-body measurements. In both graphs we see that there are four vertices and five edges, which indicates that our system is constrained to have four qubits and five two-body measurement operators. The edges (without the labels) indicate the pairs of qubits on which the two-body measurement operators are constrained to act within our system. Within these constraints, we see in this figure two possible choices of measurement operators as specified by the two labelings of the edges of the graph.

TABLE VII. The number of vertex equivalence classes, the number of rays at each vertex, and the total number of labelings for each of the 11 convex vertex-uniform tilings. By our scheme the number of labelings is equal to $(1+\frac{3^{n-1}-1}{2})^m$, where $m$ is the number of vertex equivalence classes and $n$ is the number of rays at each vertex.

| Archimedean tiling | No. of classes | No. of rays | No. of labelings |
|---|---|---|---|
| Quadrille | 1 | 4 | 14 |
| Truncated quadrille | 4 | 3 | 625 |
| Snub quadrille | 4 | 5 | 282 576 1 |
| Isosnub quadrille | 2 | 5 | 1681 |
| Hextille | 2 | 3 | 25 |
| Truncated hextille | 6 | 3 | 156 25 |
| Snub hextille | 6 | 5 | 475 010 424 1 |
| Deltille | 1 | 6 | 122 |
| Hexadeltille | 3 | 4 | 2744 |
| Truncated hexadeltille | 12 | 3 | 244 140 625 |
| Rhombihexadeltille | 6 | 4 | 752 953 6 |

TABLE IX. The wallpaper symmetry groups (using crystallographic notation) for each of the 11 convex vertex-uniform tilings, along with the particular group that we chose for our search. For two of the tilings no symmetry group was chosen because we decided not to search the tiling.

| Archimedean tiling | Symmetries | Chosen |
|---|---|---|
| Quadrille | $p4m$ | $p4m$ |
| Truncated quadrille | $p4m$ | $p4m$ |
| Snub quadrille | $p4g$, $p4$, and $pg$ | $p4$ |
| Isosnub quadrille | $cmm$ | $cmm$ |
| Hextille | $p6m$ | $p6m$ |
| Truncated hextille | $p6m$ and $p3m1$ | $p6m$ |
| Snub hextille | $p6$ | N/A |
| Deltille | $p6m$ and $p3m1$ | $p6m$ |
| Hexadeltille | $p6m$ and $p3m1$ | $p6m$ |
| Truncated hexadeltille | $p6m$ | N/A |
| Rhombihexadeltille | $p6m$ | $p6m$ |

inside a set which we know from Proposition 5 has at most $|\tilde{L}|$ elements. By Lemma 17, we conclude that this operator can be found in time $O((|\tilde{L}|+d)3^d\binom{N}{d})$ where $d := \min(w(o),N) = w(o)$ (since the weight of any operator cannot be greater than $N$). After this has been found, examination of the algorithm reveals that the computation performed afterward takes a running time in $O(|\tilde{Q}|+|\vec{P}|) = O(|\tilde{L}|)$, where the equality comes from Proposition 5. Thus, the total time needed for each step is in $O((|\tilde{L}|+d)3^d\binom{N}{d}+|\tilde{L}|) = O((|\tilde{L}|+d)3^d\binom{N}{d}))$.

Let $(\tilde{Q}',\vec{P}',\vec{s}')$ be the second to last element of $\vec{\mathcal{O}}(\tilde{S},\tilde{L})$ and $(\tilde{Q},\vec{P},\vec{s})$ the last element. In the final step of the algorithm, we move the last remaining pair in $\tilde{Q}'=\{q\}$ over to $\vec{P}'$, which means that the operator $o$ with the minimal weight is a member of $q$. From Proposition 5, we know that $\omega_{\tilde{S}}(o) \geqslant m_{\tilde{S}}(\vec{P}_i')$ for any $i$. Thus, at each step of the algorithm before this one we know that we spent a time in $O((|\tilde{L}|+d)3^d\binom{N}{d})$ where $d := w(o)$—i.e., a time no greater than the time spent on the last step. Since the algorithm requires at most $|\tilde{L}|$ steps we conclude that the total running time, including that needed to compute the set of pseudogenerators, is in $O(|\tilde{G}|^2 + |\tilde{L}|(|\tilde{L}|+d)3^d\binom{N}{d}))$. Since $\omega_{\tilde{S}}(o) = \vec{M}(\vec{P})_{|\vec{P}|}$, we conclude that $d \equiv \vec{M}(\vec{P})_{|\vec{P}|}$ (since there can be at most $N$ qubits in the choice), and so we are done.

## IV. PRACTICE

### A. Methodology

In the previous section we presented an algorithm that computes the optimal subsystem code that can be implemented using a given set of measurements. The procedure for optimizing the code requires an exponential amount of time, but fortunately the power of the exponential is a function of the distance of the best qubit in the code. Because of this property, this algorithm can be effectively applied to search over a set of choices of measurement operators to see if there is any good choice for implementing a code, since it can (relatively) quickly skip over the bad choices of measurements.

In this section, we shall present an example of applying this algorithm to search for codes on quantum systems with the structure of a graph. That is, we assume that we have a

TABLE VIII. The number of labelings for each tiling that were not redundant under rotational symmetry transformations. This number was obtained by placing an ordering on the labelings and counting the number of labelings such that no symmetry transformation obtained a labeling less than the current labeling. Also listed for the sake of comparison are the total numbers of labelings from Table VII.

| Archimedean tiling | No. nonredundant | No. total |
|---|---|---|
| Quadrille | 10 | 14 |
| Truncated quadrille | 155 | 625 |
| Snub quadrille | 706 881 | 282 576 1 |
| Isosnub quadrille | 743 | 1681 |
| Hextille | 11 | 25 |
| Truncated hextille | 2392 | 156 25 |
| Deltille | 58 | 122 |
| Hexadeltille | 594 | 2744 |
| Rhombihexadeltille | 904 741 | 752 953 6 |

TABLE X. The maximum radius lattice that was completely scanned (i.e., such that every possible labeling was examined by the algorithm) for each tiling. To give a sense of the size of the lattices involved, we also list the number of physical qubits (corresponding to vertices) for the lattice with the maximum radius.

| Tiling | Maximum radius | No. of qubits |
|---|---|---|
| Quadrille | 4 | 64 |
| Truncated quadrille | 6 | 576 |
| Snub quadrille | 5 | 200 |
| Isosnub quadrille | 8 | 768 |
| Hextille | 10 | 600 |
| Truncated hextille | 5 | 600 |
| Deltille | 8 | 256 |
| Hexadeltille | 3 | 108 |
| Rhombihexadeltille | 3 | 162 |

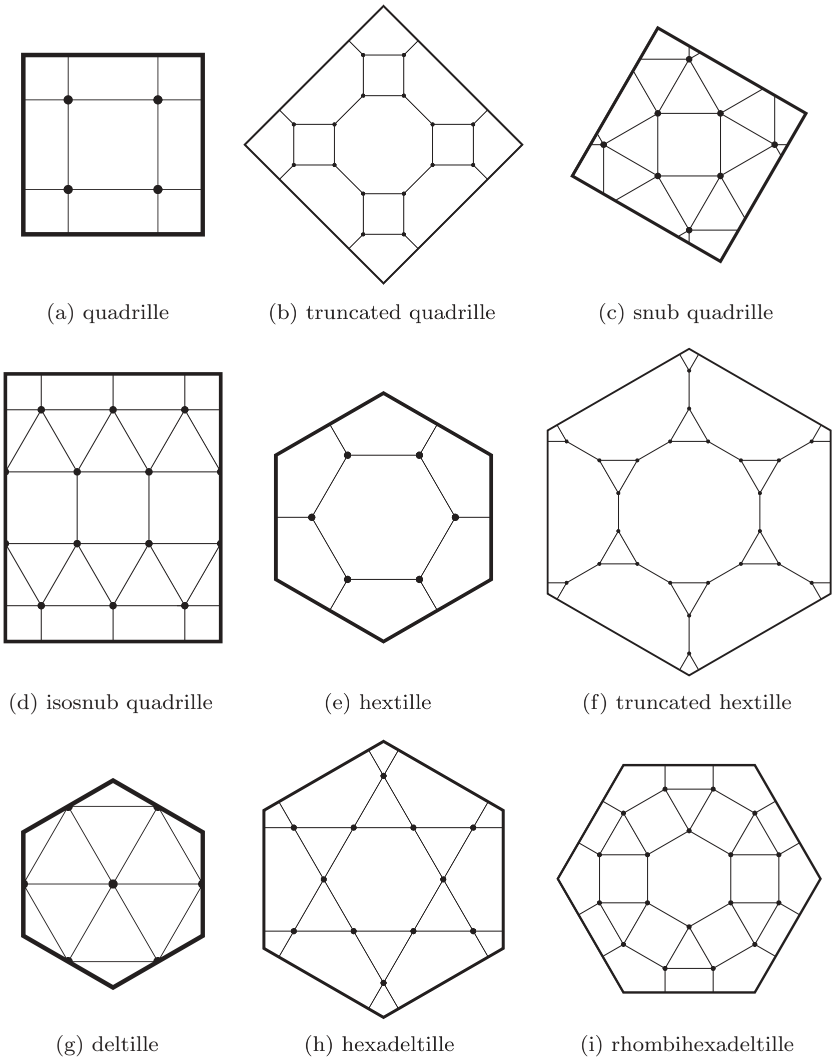


FIG. 2. Illustration of how we placed the centers and boundaries for each of the tilings that we scanned. The boundaries are periodic, so that edges that pass through one side of the boundary wrap around to the opposite side. Edges and vertices *on* a boundary are merged with the corresponding edges and vertices on the opposite edge. Note that under this scheme there can be no vertices on a corner. For most tilings this will never happen, but it turns out that in the case of the deltille tiling there are vertices on the corners when the radius (smallest distance from the center to the boundary) is three times the radius of the unit cell; we thus ignore deltille tilings of these sizes.

system of qubits, two-body Pauli measurement operators, and a graph such that there is a bijection between the qubits and vertices and between the edges and measurement operators, and also such that each measurement acts only on the two qubits corresponding to the vertices adjacent to its associated edge. Specifying a particular graph constrains the number of qubits and the types of measurement operators, but it still allows a great deal of freedom in the choice of the measurement operator at each edge. In Fig. 1 we illustrate an example of a graph with two possible such choices of measurement operator labelings; note that for the sake of generality we do not impose the constraint that the two operators in the two-body measurement be identical.

For reasons that will become clear, it turns out to be useful to specify choices of measurement operators in terms of ray labelings rather than edge labelings since the former are associated with vertices. Define a *ray* of a graph to be a pair consisting of a vertex and an edge adjacent to the vertex; note that every ray can be uniquely associated with an edge, and every edge can be associated uniquely with a ray for each of its two incident vertices. Thus, we can define a particular choice of measurement operators by labeling each ray in the graph with a single-qubit Pauli operator acting on the qubit of the incident vertex, and then letting the measurement operator associated with each edge be equal to the product of the single-qubit operators in the edge's two rays.

There is a natural symmetry of quantum codes that can be factored out to reduce the search space: the relevant properties of the code are invariant under single-qubit rotations. That is, transformations such as swapping the $X$ and $Z$ operators at the location of a single physical qubit in every stabilizer, gauge qubit, and logical qubit operator do not affect the code. Thus, when labeling the rays of a vertex, exactly which ray is labeled $X$, $Y$, and $Z$ is not important; what matters is which rays commute and which rays anticommute. We see therefore that we need only search over the possible ways to divide the rays into three indistinguishable groups, so that a vertex with $n$ rays only has $1+\frac{3^{n-1}-1}{2}$ relevant labelings that need to be examined.

The specific graphs we shall examine in this section are lattices generated by nine of the eleven convex vertex-uniform (also known as the "Archimedean") tilings of the plane—that is, those tilings with the property that every face is convex and every vertex has the same sequence of faces [44].[12]

[12] In particular, see Theorem 2.1.3 on p. 59 of Ref. [44] for the proof that there are exactly 11 tilings with this property.

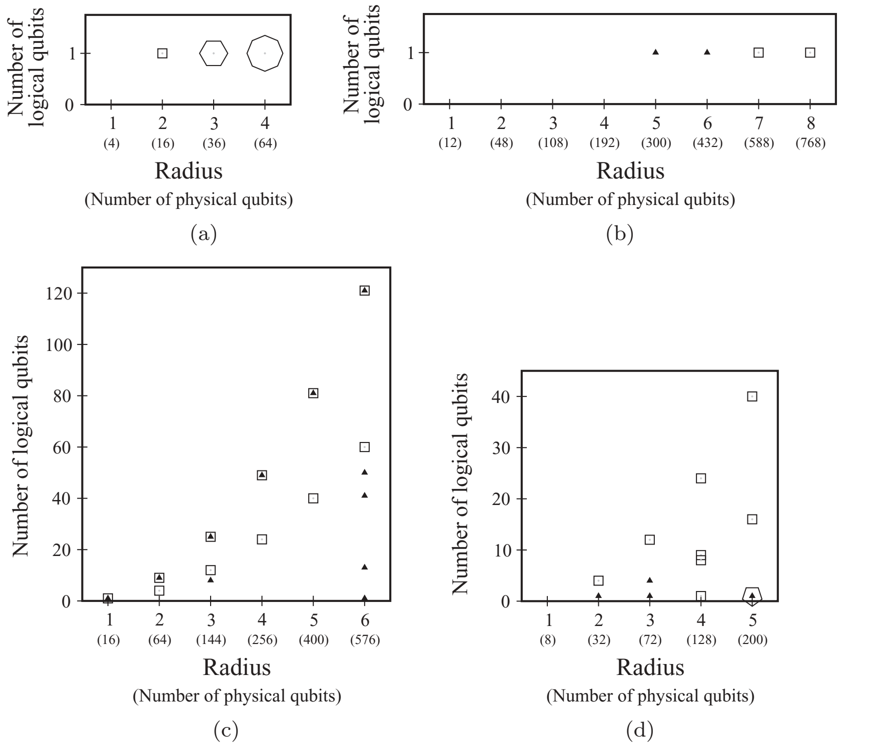


FIG. 3. Plots of the results from scanning the (a) quadrille, (b) isosnub quadrille, (c) truncated quadrille, and (d) snub quadrille tilings. Every polygon in the plot corresponds to a code that was found with a distance equal to the number of sides of the polygon, so that triangles indicate distance-3 codes, squares indicate distance-4 codes, etc. The position along the $x$ axis indicates the radius of the lattice where the code was found, where the radius is an integer defined to be the length of the lattice divided by the length of the smallest possible periodic lattice for the tiling; it also indicates the number of physical qubits in the lattice where the code was found, which appears on the $x$ axis just under the value of the radius. The position along the $y$ axis indicates the number of logical qubits with that distance in the code. Note that in cases where multiple codes were found for the same radius and with the same number of logical qubits, multiple polygons are drawn, so that, for example, in plot (c) we see several cases in which a distance-3 code (triangle) and a distance-4 code (square) were found that were in lattices with the same radius and also had the same number of logical qubits.

Since these tilings have many translational symmetries, we intentionally narrow our search to the set of labelings that share the translational symmetries of the lattice.[13] Since the ray labelings must be preserved under these symmetries, we can partition the rays of the graph into equivalence classes such that two rays are equivalent if and only if they are related by a translation symmetry; thus we see that our narrowed search space is equivalent to the space of possible labelings of each *class* of rays in the lattice examined. Since there is a symmetry that can be factored out at each vertex (as discussed previously), we note that we can likewise partition the vertices into equivalence classes of vertices related by translation symmetries. If there are $m$ vertex equivalence classes, and every vertex has $n$ rays, then our search space consists of $(1+\frac{3^{n-1}-1}{2})^m$ total possible labelings. In Table VII, we list the 11 convex vertex-uniform tilings with the number of vertex equivalence classes, the number of rays at each vertex, and the total number of labelings. (Two of the eleven tilings, "truncated hexadeltille" and "snub hextille," had such a large number of possible labelings that we decided to exclude them from our search.)

Note that we could furthermore refine our search to consist of those codes which also share the *rotational* symmetries of the lattice. We explicitly avoid making this refinement because the existence of such codes as the quantum compass model code [24] indicates that there are good codes on lattices that require breaking the rotational symmetry of the lattice. However, we can use the rotational symmetries in a different way to reduce the search space as follows. Partition the labelings into equivalence classes such that two labelings are in the same class if and only if there is a rotational symmetry that relates them, and observe that all of the labelings in each class will give rise to quantum codes with identical properties. Thus, we can reduce our search space to ignore redundant labelings by only examining one labeling in each equivalence class.

Our search algorithm thus works in the following manner. We start by putting a total ordering on all of the lattice labelings (after having factored out the symmetry at each vertex). We enumerate these labelings in order. For each labeling, we generate new labelings by applying each rotational symmetry to the current labeling. If any of these new labelings is less than the current labeling under our ordering, then we skip the current labeling because we know that we have already previously examined an equivalent labeling. Although this algorithm proceeds serially through the search space, it can be parallelized by making use of $n$ walkers, each of which starts at a different labeling (from 0 to $n-1$) and which proceed by examining the current labeling and then skipping directly to the $n$th labeling after the current one. In Table VIII we list the number of nonredundant labelings for each tiling.

In order to preserve the rotational symmetries of the tiling, it is important that the lattice be constructed such that the center of the lattice is at a point of rotational symmetry. There is not a single unique center point that preserves all of the rotational symmetries of a given tiling, and furthermore for many tilings there are multiple rotational symmetry groups (known as "wallpaper" symmetry groups), each of which has a different set of center points. We thus chose the center of our lattice by picking the largest of the wallpaper groups present in the tiling and choosing the center to give rise to the rotational

[13] This is not to claim that there are no interesting codes that break these translational symmetries; however, the investigation of such codes is outside the scope of this particular study.

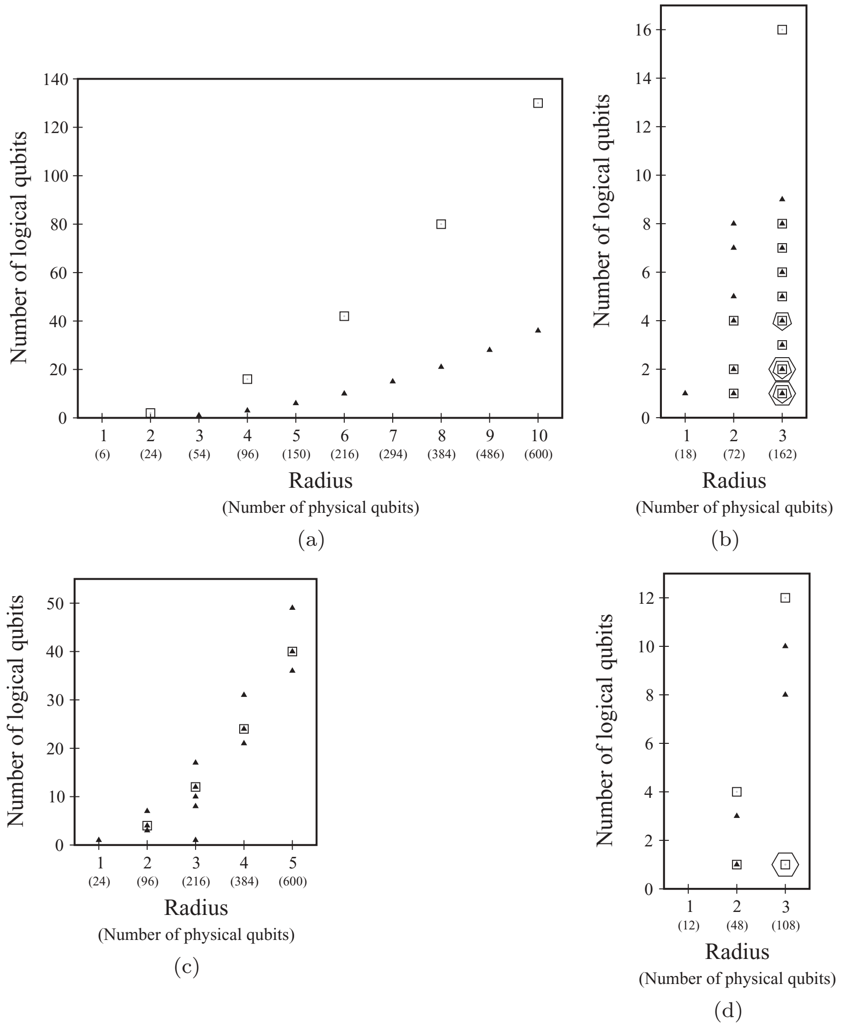


FIG. 4. Plots of the results from scanning the (a) hextille, (b) rhombihexadeltille, (c) truncated hextille, and (d) hexadeltille tilings. See the caption of Fig. 3 for an explanation of how to interpret these plots.

symmetries in that group.[14] In Table IX we list the wallpaper symmetries for each of the 11 convex vertex-uniform tilings along with (where applicable) the particular symmetry group that we chose to utilize.

As is usually the case in physical systems, it is important to pay careful attention to the boundary conditions of the lattice. In order to minimize boundary effects, we decided to put periodic boundary conditions on our lattices; care had to be taken to impose the periodic boundary conditions in such a way as to preserve the rotational symmetry group. For example, a boundary that only wraps from left to right and from top to bottom breaks some of the rotational symmetries for hexagonal tilings. In Fig. 2 we illustrate how we placed the centers and the boundaries of the tilings.

Due to limits on our computational resources, we were limited in the size of the lattices that we could search with the algorithm. We describe the size of the lattices using a quantity we call the "radius," which is an integral quantity equal to the length of the lattice divided by the length of the smallest lattice defined for that tiling; the unit radius lattices are those illustrated in figure. In Table X we show the maximum radius lattice that was completely scanned (i.e., such that every possible labeling was examined by the algorithm) for each tiling.

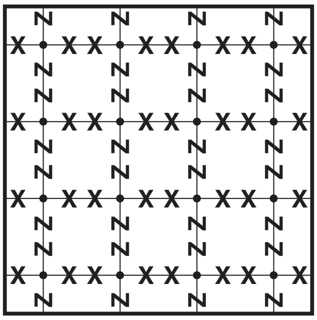


FIG. 5. Illustration of the single labeling of the quadrille tiling (on a radius-2 lattice) that results in a useful quantum code.

[14] Note that this approach does not mean that we have eliminated redundant labelings resulting from *all* of the symmetries in the lattice. For example, we have not eliminated labelings which are equivalent under rotations around a different point, nor which are equivalent under a glide-reflection symmetry. It is certainly possible to eliminate these labelings, but we choose not to do so in this project for the sake of simplicity. An implication of this is that for many codes we expect to see multiple labelings giving rise to them that are equivalent under symmetry transformations but not eliminated by our approach.

Since each labeling of every lattice results in a quantum code, we had to provide some criterion for our search algorithm to decide whether a code was interesting enough to log. We set our criterion relatively low: a code was deemed to be interesting if there was at least one logical qubit with distance 3, that is, if there was at least one logical qubit such that a single arbitrary error on that qubit can be corrected. This was done under the reasoning that as long as some of the logical qubits in a code are sufficiently useful to us to make implementing the code worthwhile, then we should not be troubled by the fact that there might be other logical qubits that are not useful because we can always ignore them (or, equivalently, classify them as gauge qubits).

### B. Results

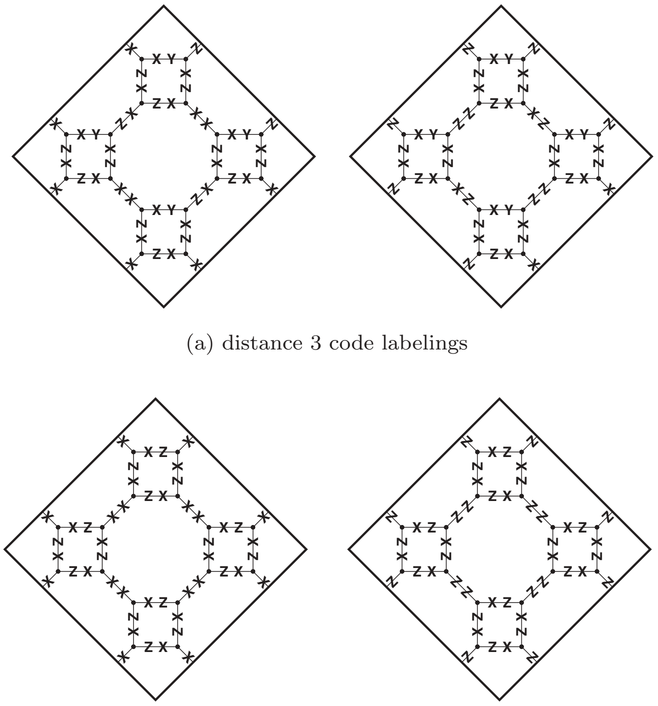
In the previous section we described the search space to which we applied the algorithm in order to computationally find possible codes that can be implemented using systems with two-body interactions and a lattice structure following nine of the eleven convex vertex-uniform tilings. In this section we present the results of this search. The codes that we found are shown in the plots appearing in Figs. 3 and 4. No plot appears for the deltille tiling because no codes were found for that tiling. It is worth emphasizing that these codes indicated in these figures are *all* of the (useful) codes that exist for the scanned lattices of that tiling given our constraints, since we scanned every possible labeling that was not redundant under a rotational symmetry transformation about the center.

Observe that two kinds of trend appear frequently in the results: codes that grow in distance but remain constant in the number of logical qubits as the radius increase, and codes that remain constant in distance but grow in the number of logical qubits as the radius increases. The former trend appears in the quadrille, snub quadrille, isosnub quadrille, hexadeltille, and rhombihexadeltille tilings.[15] The latter trend appears in the truncated quadrille, snub quadrille, hextille, truncated hextille, hexadeltille, and rhombihexadeltille tilings. In many of the tilings there are also codes that were found that do not seem to belong to an obvious trend.

In the follow sections we will focus on some specifics of the results for each of the tilings.

#### *1. Quadrille*

For the quadrille lattice, we saw only one labeling, illustrated in Fig. 5, that resulted in an interesting code. This labeling corresponds to the compass model code, and the algorithm correctly found that the distance of the code grows linearly with the radius of the lattice and is exactly equal to the square root of the number of qubits in the lattice. This result is not terribly surprising, but it is good to see that our search technique employing the algorithm can correctly duplicate known results.



(a) distance 3 code labelings

(b) distance 4 code labelings

FIG. 6. Illustration of four labelings of the truncated quadrille tiling (on a radius-1 lattice) that result in a quantum code with distance 3 (4) [in respectively (a) and (b)], and a number of logical qubits proportional to the square of the radius of the labeling.

#### *2. Truncated quadrille*

There are three kinds of code that appear in this tiling where the number of qubits increases with the radius: two where the distance is fixed at 4, and one where the distance is fixed at 3. For the best two of these three kinds of code, the number of logical qubits ($l$) is related to the radius ($r$) by $l = (2r - 1)^2$. Since the number of physical qubits ($n$) is given by $n = (4r)^2$, the number of logical qubits per physical qubit is thus given by $\frac{l}{n} = (\frac{1}{2} - \frac{1}{r})^2$, a quantity which converges to $\frac{1}{4}$ as $r \to \infty$. There were four labelings with this property that we saw in

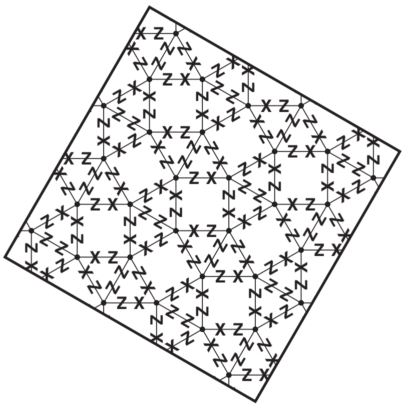


FIG. 7. Illustration of a labeling of the snub quadrille tiling (on a radius-2 lattice) that results with distance 4 and a number of distance logical qubits proportional to the square of the radius of the labeling.

[15] Note that where the former trend was present, the maximum radius that we scanned was often quite limited; this is due to the exponential explosion in the cost of finding the optimal code as a function of the distance of the code.

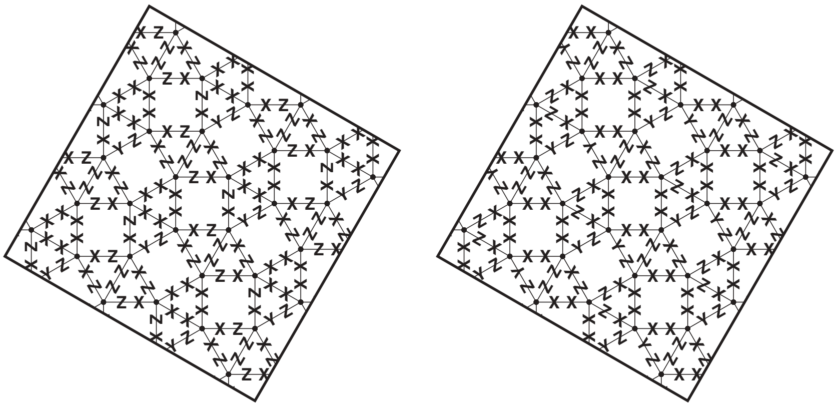


FIG. 8. Illustration of two of the labelings of the snub quadrille tiling (on a radius-2 lattice) that result in a quantum code with a single qubit whose distance grows with the radius of the lattice.

our search: two with distance-3 qubits [illustrated in Fig. 6(a)], and two with distance-4 qubits [illustrated in Fig. 6(b)].

### *3. Snub quadrille*

There are two kinds of interesting code found in this filing. First, we saw exactly one labeling that has the property that the distance is 4 and the number of logical qubits ($l$) is given by $l = 2r(r-1)$, where $r$ is the radius of the lattice. Since the number of physical qubits ($n$) is given by $n = 8r^2$, this means that the number of logical qubits per physical qubit is given by $\frac{l}{n} = \frac{2r(r-1)}{8r^2} = \frac{1}{4}(1-\frac{1}{r}) \rightarrow \frac{1}{4}$ as $r \rightarrow \infty$. This labeling is illustrated in Fig. 7.

Second, more usefully, we saw 12 labelings which result in a code that has one qubit whose distance grows with the size of the lattice. Two of these labelings are illustrated in Fig. 8.

### *4. Isosnub quadrille*

We saw only two labelings of the isosnub lattice that result in useful codes, both of which only have a single qubit that seems (assuming that the trend seen in Fig. 4 can be extrapolated) to have a distance that grows with the radius of the lattice. These two labelings are illustrated in Fig. 9.

### *5. Deltille*

We scanned this tiling up to a radius of 8; no interesting codes were found in any of the 122 labelings.

### *6. Hextille*

In this tiling we saw four labelings which resulted in two kinds of interesting code: two of the labelings [illustrated in Fig. 10(a)] resulted in codes of distance 3 that were present for every value of the radius, and two of the labelings [illustrated in Fig. 10(b)] resulted in codes of distance 4 that were only present for *even* values of the radius. The former resulted in codes which had a number of logical qubits ($l_3$) given by $l_3 = \frac{(r-1)(r-2)}{2}$, where $r$ is the radius, and the latter resulted in codes which had a number of logical qubits ($l_4$) given by $l_4 = r(r+3)$. Since the number of qubits ($n$) is given by $n = 6r^2$, we have that the number of logical qubits per physical qubit for the distance-3 and distance-4 codes were given respectively by $d_3 = \frac{l_3}{n} = \frac{1}{12}(1-\frac{1}{r})(1-\frac{2}{r})$ and $d_4 = \frac{l_4}{n} = \frac{1}{6}(1+\frac{3}{r})$; as $r \rightarrow \infty$, we have that $d_3 \rightarrow \frac{1}{12}$ and $d_4 \rightarrow \frac{1}{6}$.

It is interesting to observe that there is no distance versus qubit count trade-off in this tiling. As long as the radius is

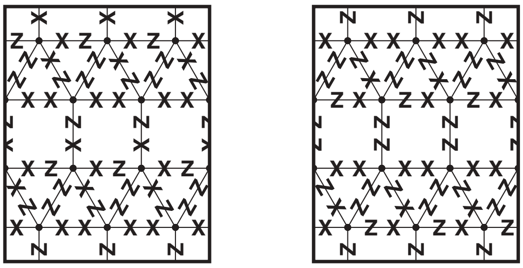
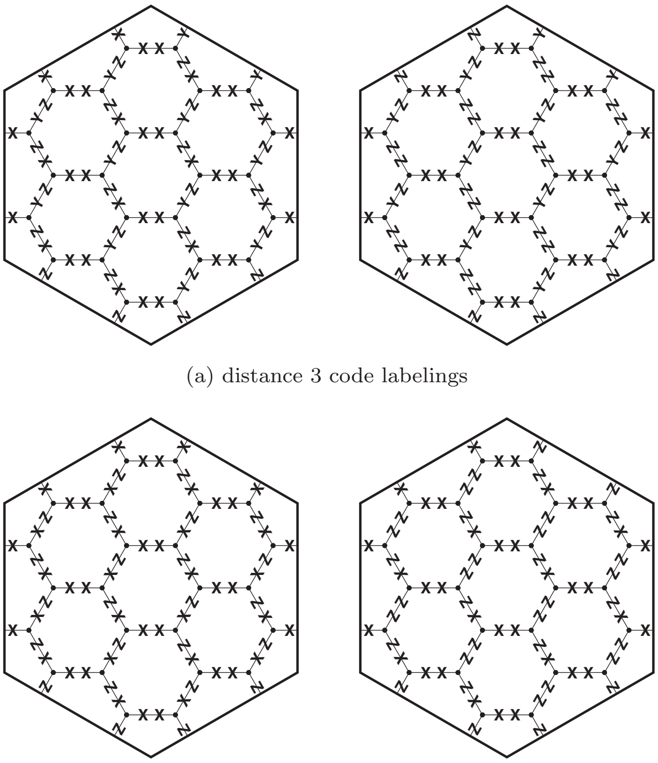


FIG. 9. Illustration of two of the labelings of the isosnub quadrille tiling (on a radius-1 lattice) that result in a quantum code with a single qubit whose distance grows with the radius of the lattice.



(a) distance 3 code labelings

(b) distance 4 code labelings

FIG. 10. Illustration of four labelings of the hextille tiling (on a radius-2 lattice) that result in a quantum code with distance 3 (4) [in respectively (a) and (b)], and a number of distance logical qubits proportional to the square of the radius of the labeling.

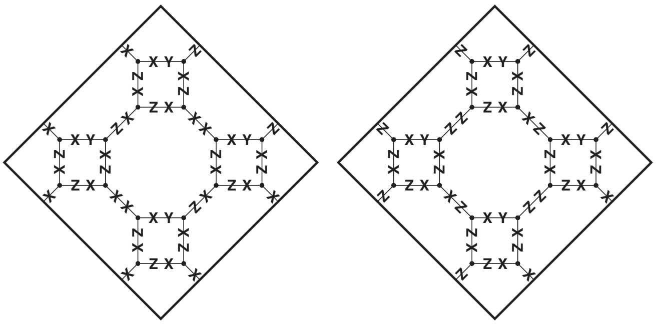


FIG. 11. Illustration of two labelings of the truncated hextille tiling (on a radius-1 lattice) that result in a quantum code with distance 3 and a number of distance logical qubits proportional to the square of the radius of the labeling.

even, the distance-4 code is superior in *both* distance *and* logical qubit count to the distance-3 code.

### 7. *Truncated hextille*

In the truncated hextille there are four kinds of code where the number of qubits increases with the radius: three with the distance fixed at 3, and one with the distance fixed at 4.

The best of the distance-3 codes has the number of logical qubits ($l_3$) given by $l_3 = 2r^2 - 1$, where $r$ is the radius of the code. Since the number of physical qubits ($n$) is given by $n = 24r^2$, the number of logical qubits per physical qubit is thus given by $\frac{l_3}{n} = \frac{1}{12}(1 - \frac{1}{24r})^2$, a quantity which converges to $\frac{1}{12}$ as $r \to \infty$. The two labelings we saw which give rise to this code are illustrated in Fig. 11. The best of the distance-4 codes has the number of logical qubits ($l_4$) given by $l_4 = 2r(r-1)$, and the number of logical qubits per physical qubit is thus given by $\frac{l_4}{n} = \frac{1}{12}(1 - \frac{1}{r})$, a quantity which converges to $\frac{1}{12}$ as $r \to \infty$. We see from this analysis that although the best distance-4 code contains fewer logical qubits than the best distance-3 code, they both converge to the same number of logical qubits per physical qubit in the large-radius limit. One of the nine labelings we saw which give rise to this distance-4 code is illustrated in Fig. 12.

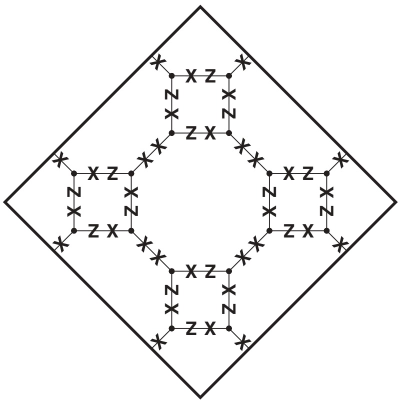


FIG. 12. Illustration of two labelings of the truncated hextille tiling (on a radius-1 lattice) that result in a quantum code with distance 4 and a number of distance logical qubits proportional to the square of the radius of the labeling.

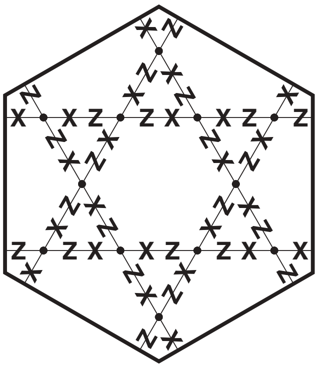


FIG. 13. Illustration of one of the labelings of the hexadeltille tiling (on a radius-1 lattice) that results in quantum code whose distance grows with the size of the tiling.

### 8. *Hexadeltille*

There are many codes that appear in the hexadeltille tiling, but it is difficult to draw conclusions about trends due to the limit on the size of the lattices that were scanned. The good news, though, is that the reason why scanning larger radii was difficult is because there is a code in this tiling with a qubit whose distance grows with the radius of the lattice. One of the nine labelings that we saw with this property is illustrated in Fig. 13.

### 9. *Rhombihexadeltille*

This tiling is interesting because it had many more labelings that resulted in codes than all of the other tilings combined; specifically, for the rhombihexadeltille tiling we saw 48 807 labelings that resulted in useful codes, whereas for all of the other tilings combined we saw only 421 labelings that resulted in useful codes. This is even more remarkable considering that the largest lattice we were able to scan for the rhombihexadeltille tiling was smaller than that for most of the other tilings.

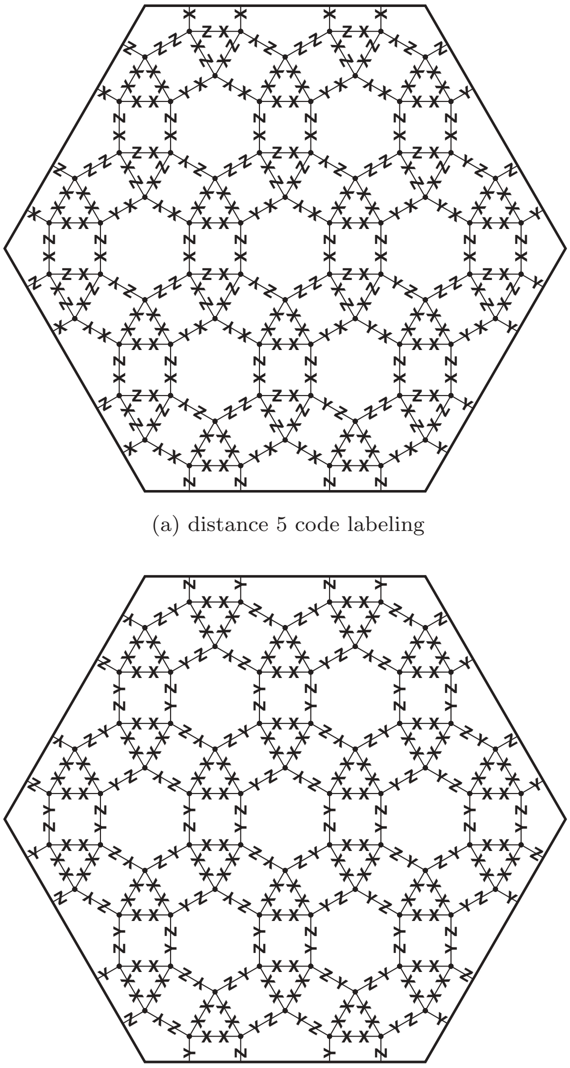
As can be seen in Fig. 4, this tiling is also interesting because it features so many different kinds of codes, including both codes that seem to grow in the number of logical qubits with radius and codes that grow in distance with size. It is the only tiling that features a lattice that contains a labeling resulting in a code for every distance up to 6.

The rhombihexadeltille tiling is the only tiling we have seen which has code both with a distance greater than 4 and with multiple qubits; we saw six labelings which resulted in codes with distance 6 and two qubits, and four labelings which resulted in codes with distance 5 and four qubits. In Fig. 14 we show an example of each of these labelings.

Two of the labelings resulted in codes with the highest number of qubits—16 logical qubits at distance 4 for a lattice of radius 3. These two labelings are illustrated in Fig. 15.

## C. Discussion

There are few surprises in our results. For example, the best code that we found that maximized the logical qubit distance per physical qubit was the compass model code in the



(a) distance 5 code labeling

(b) distance 6 code labeling

FIG. 14. Illustration of labelings of the rhombihexadeltille tiling (on a radius-2 lattice) that result in a quantum code with distance 5 and four qubits (a) and a quantum code with distance 6 and two qubits (b) when applied to a radius-3 lattice.

quadrille tiling, which is already well known. Furthermore, all of the codes obeyed the upper bounds $kd \in O(n)$ and $d^2 \in O(n)$—where $k$ is the number of logical qubits in the code, $d$ is the distance of the code, and $n$ is the number of physical qubits implementing the code—that were derived in [30] for codes having spatially local generators.

Some of the observed differences between the tilings are an artifact of the search space. For example, every code found on the hextille tiling could also be implemented on the deltille tiling, but although we found two kinds of code for the hextille tiling we found no codes for the deltille tilings. This is because our search space included no way for the deltille tilings to "knock out" the middle qubits in each hexagonal tiling, and furthermore the hextille tiling search space included two classes of vertices which could have independent labelings, whereas the deltille tiling search space had only one class of vertices.

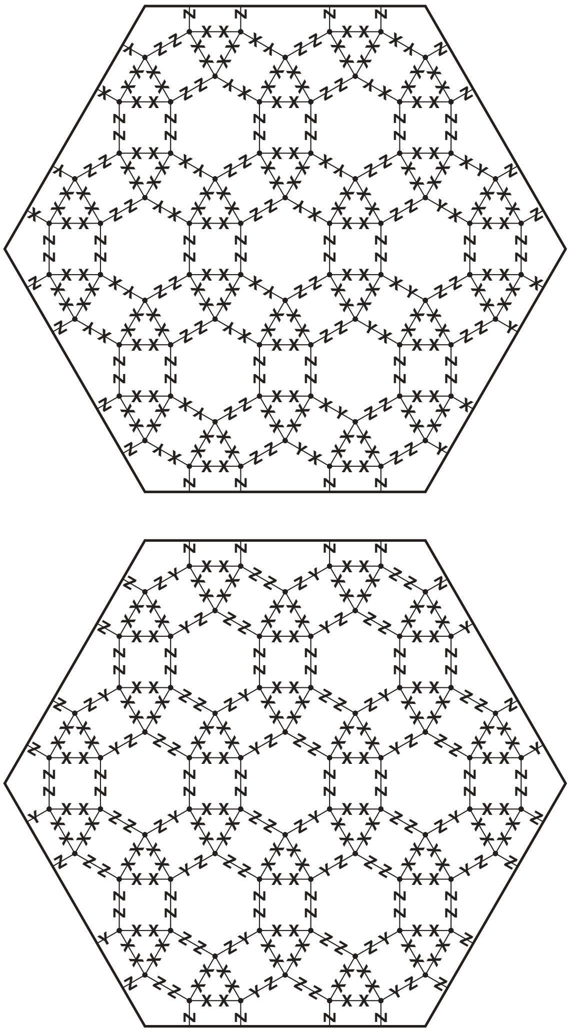


FIG. 15. Illustration of two labelings of the rhombihexadeltille tiling (on a radius-2 lattice) that result in a quantum code with distance 4 and 16 qubits when applied to a radius-3 lattice.

Although it is not clear how many of the codes we found will have practical applications, the success of this search demonstrates the feasibility of using brute-force computation to find useful codes within a constrained search space.

## V. CONCLUSION

In this paper we have presented an algorithm for computing the optimal quantum subsystem code that can be implemented using a given set of measurements. We have shown that although this algorithm requires exponential time in the worst case, this exponential is a function of the code distance, and so the algorithm terminates (relatively) quickly when the optimal code has low distance. Because of this, the algorithm can be used to perform a brute-force search through a space of possible measurements in order to see which give rise to "useful" (high-distance) codes. We demonstrated the feasibility of this approach by applying the algorithm to search for codes implemented on systems with lattice structures corresponding to nine of the eleven convex vertex-uniform tilings, and on all but one of these nine tilings we found useful codes.

This algorithm should prove helpful in two kinds of ways in particular. First, it can be applied in an exploratory setting to do the tedious work of computing the code resulting from a set of measurements so that the researcher can experiment with new ideas for choices of measurement to see how well they work. Second, it can be applied to hone a "rough" idea for how a code might be implemented (such as a particular lattice configuration) into a concrete idea by scanning through the possible choices of the degrees of freedom to see if any result in useful codes; of course, cleverness can often come up with an answer more quickly than a computationally intensive search, but it is good to have the alternative of brute-force computation to fall back on when brute-force cleverness fails.

## ACKNOWLEDGMENTS

GMC is supported by the DOE through the Computational Science Graduate Fellowship program (administered by the Krell Institute) and also by the DARPA-MTO Quest program through a grant from AFOSR. D.B. is supported by the NSF under Grants No. 0803478, 0829937, and 0916400, by the DARPA-MTO QuEST program through a grant from AFOSR, and by IAPRA under the MCQO program.

[1] P. W. Shor, Phys. Rev. A **52**, R2493 (1995).

[2] A. M. Steane, Phys. Rev. Lett. **77**, 793 (1996).

[3] A. M. Steane, Phys. Rev. A **54**, 4741 (1996).

[4] A. M. Steane, Proc. R. Soc. London, Ser. A **452**, 2551 (1996).

[5] E. Knill and R. Laflamme, Phys. Rev. A **55**, 900 (1997).

[6] D. Gottesman, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1997.

[7] P. W. Shor, in *Proceedings of the 37th Symposium on the Foundations of Computer Science* (IEEE Press, Los Alamitos, CA, 1996), pp. 56–65.

[8] D. Aharonov and M. Ben-Or, in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 1997), pp. 176–188.

[9] E. Knill, R. Laflamme, and W. H. Zurek, Science **279**, 342 (1998).

[10] E. Knill, R. Laflamme, and W. H. Zurek, Proc. R. Soc. London, Ser. A **454**, 365 (1998).

[11] J. Preskill, *Introduction to Quantum Computation and Information* (World Scientific Publishing Co., Singapore, 1998), pp. 213–269.

[12] P. Aliferis, D. Gottesman, and J. Preskill, Quantum Inf. Comput. **6**, 97 (2006).

[13] A. W. Cross, D. P. DiVincenzo, and B. M. Terhal, e-print arXiv:0711.1556.

[14] D. Poulin, Phys. Rev. Lett. **95**, 230504 (2005).

[15] D. Kribs, R. Laflamme, and D. Poulin, Phys. Rev. Lett. **94**, 180501 (2005).

[16] D. Kribs, R. Laflamme, D. Poulin, and M. Lesosky, Quantum Inf. Comput. **6**, 382 (2005).

[17] D. Kribs and R. W. Spekkens (unpublished).

[18] A. Kitaev, Ann. Phys. **303**, 2 (2003).

[19] A. Kitaev, Ann. Phys. **303**, 2 (2003).

[20] J. P. Barnes and W. S. Warren, Phys. Rev. Lett. **85**, 856 (2000).

[21] D. Bacon, K. R. Brown, and K. B. Whaley, Phys. Rev. Lett. **87**, 247902 (2001).

[22] S. P. Jordan, E. Farhi, and P. W. Shor, Phys. Rev. A **74**, 052322 (2006).

[23] Y. S. Weinstein and C. S. Hellberg, Phys. Rev. A **72**, 022319 (2005).

[24] D. Bacon, Phys. Rev. A **73**, 012340 (2006).

[25] D. Bacon, Phys. Rev. A **78**, 042324 (2008).

[26] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, Rev. Mod. Phys. **80**, 1083 (2008).

[27] H. Bombin, R. Chhajlany, M. Horodecki, and M. Martin-Delgado, e-print arXiv:0907.5228.

[28] S. Chesi, D. Loss, S. Bravyi, and B. M. Terhal, New J. Phys. **12**, 025013 (2010).

[29] D. Bacon and A. Casaccino, in *Proceedings of the 44th Annual Alerton Conference,* 2006, e-print arXiv:quant-ph/0610088.

[30] S. Bravyi, e-print arXiv:1008.1028.

[31] H. Bombin, Phys. Rev. A **81**, 032301 (2010).

[32] P. W. Shor, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA, 1994), pp. 124–134.

[33] D. Gottesman, Phys. Rev. A **54**, 1862 (1996).

[34] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, Phys. Rev. Lett. **78**, 405 (1997).

[35] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, IEEE Trans. Inf. Theory **44**, 1369 (1998).

[36] M. Nielsen and D. Poulin (unpublished).

[37] P. Aliferis and A. W. Cross, Phys. Rev. Lett. **98**, 220502 (2007).

[38] D. Bacon, Ph.D. thesis, University of Calfornia at Berkeley, Berkeley, CA, 2001.

[39] J. Dorier, F. Becca, and F. Mila, Phys. Rev. B **72**, 024448 (2005).

[40] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, J. Math. Phys. **43**, 4452 (2002).

[41] S. Bravyi and B. Terhal, New J. Phys. **11**, 043029 (2009).

[42] S. Bravyi, D. Poulin, and B. Terhal, Phys. Rev. Lett. **104**, 050503 (2010).

[43] G. White and M. Grassl, in *Proceedings of IEEE International Symposium on Information Theory, Seattle, 2006* (IEEE Press, Los Alamitos, CA, 2006).

[44] B. Grünbaum and G. C. Shephard, *Tiling and Patterns* (W. H. Freeman and Company, New York, 1987).